INFORMATION TECHNOLOGY

37  An introduction to Aditi deductive database system
    J VAGHANI, K RAMAMOHANARAO, DB KEMP, Z SOMOGYI,
    PJ STUCKEY

53  TQL: a nested relational query language
    JA THOM, AJ KENT, R SACKS-DAVIS

66  Action programmes for teaching and researching in information
    systems
    DE AVISON

73  A direct search algorithm for global optimisation of
    multivariate functions
    KK BENKE, DR SKINNER

SPECIAL FEATURES

72  Letter to the editor

85  Call for papers — Special issue algorithms

86  Book reviews
The Australian Computer Journal is an official publication of the Australian Computer Society Incorporated.

CONTRIBUTIONS
All material for publication should be sent to: Mr Rob Cook, Editor Australian Computer Journal Centre for Information Technology Research University of Queensland, 4072.

Prospective authors may wish to consult manuscript preparation guidelines published in the February 1991 issue. The paragraphs below briefly summarise the essential details.

Types of Material: Four regular categories of material are published: Papers, Short Communications, Letters to the Editor and Book Reviews.

Generally speaking, a paper will discuss significant new results of computing research and development, or provide a comprehensive summary of existing computing knowledge with the aim of broadening the outlook of Journal readers, or describe important computing experience or insight.

Short Communications are concise discussions of computing research or application. A letter to the Editor will briefly comment on material previously appearing in the Journal, or discuss a computing topic of current interest. The term 'computing' is interpreted broadly to include descriptions of computer hardware and software, and major computer applications such as information systems, expert systems, computer networks and office automation.

Refereeing: Papers and Short Communications are accepted if recommended by anonymous referees. Letters are published at the discretion of the Editor, and Book Reviews are written at the Editor’s invitation upon receipt of review copies of published books. All accepted contributions may be subject to minor modifications to ensure uniformity of style. Referees may suggest major revisions to be performed by the author.

Proofs and Reprints: Page proofs of Papers and Short Communications are sent to the authors for correction prior to publication. Fifty copies of reprints will be supplied to authors without charge. Reprints of individual papers may be purchased from Associated Business Publications, PO Box 440, Broadway, NSW 2007. Microfilm reprints are available from University Microfilms International, Ann Arbor/London.

Format: Papers, Short Communications and Book Reviews should be typed in double spacing on A4 size paper, with 2.5 cm margins on all four sides. The original, plus two clear bond paper copies, should be submitted. References should be cited in standard journal form, and generally diagrams should be ink drawn on tracing paper or board with stencil or Letraset lettering.

Papers and Short Communications should have a brief Abstract, Key word list and CR categories on the leading page, with authors’ affiliation as a footnote. The authors of an accepted paper will be asked to supply a brief biographical note for publication with the paper.

Copyright © 1991 Australian Computer Society Inc.

Production Management:
Associated Business Publications Pty Ltd, Room 104, 3 Small Street, Ultimo NSW 2007 (PO Box 440, Broadway, NSW 2007). Telephone 212 2780, 212 3780.

All advertising enquiries should be referred to the above address.

SUBSCRIPTIONS
The annual subscription is $40.00. All subscriptions to the Journal are payable in advance and should be sent (in Australian currency) to the Australian Computer Society Inc PO Box 319, Darlinghurst, NSW 2010

JA GODDARD Immediate Past President
RG HEINRICH National Treasurer
AW GOLDSWORTHY Executive Director & Chief Executive Officer
PO Box 319, Darlinghurst, NSW, 2010
Telephone (02) 211 5855

Publication Board
JM HUGHES Governor

AUSTRALIAN COMPUTER JOURNAL
Subscription/Change of Address Form
Photocopies of this form are acceptable.

NAME

CURRENT ADDRESS

□ Please enrol me as a subscriber for 1991. I enclose a cheque or bank draft for $40.00.

□ Please record my new address as shown above. I attach the mailing label for the last received issue.

*ATTACH LABEL HERE

Send all correspondence regarding subscriptions to the Circulation Manager, PO Box 319, Darlinghurst, NSW, 2010, Australia.

Printed by MAPS Litho Pty Ltd, Unit 3, 8 Aquatic Drive, Frenchs Forest, NSW, 2086.
An introduction to the Aditi deductive database system

Jayen Vaghani, Kotagiri Ramamohanarao, David B. Kemp, Zoltan Somogyi, Peter J. Stuckey
Key Centre for Knowledge Based Systems
Department of Computer Science
University of Melbourne, Parkville Vic 3052 Australia

1. INTRODUCTION

The field of deductive databases grew out of research in the late seventies and early eighties on the use of logic to manipulate large amounts of data. Deductive databases have several advantages over relational ones: they can naturally describe recursive relationships between entities, they allow tables to contain non-atomic data, and they can be integrated relatively easily into procedural logic programming languages such as Prolog for use in applications such as expert systems. This expressive power allows users to pose more complex queries, which in turn require new optimisation techniques.

The ultimate goal of the Aditi project at the University of Melbourne is to prove that deductive database technology can compete with relational database technology by building a deductive database system that offers facilities superior to those in current relational systems while maintaining competitive performance. Our approach is to exploit the results of existing research in deductive query evaluation and relational database technology to produce a system that uses logic to manipulate the large amounts of data found in traditional database systems.

We started work on Aditi in the second quarter of 1988. A basic version of the system has been operational since July 1989. We are continuously enhancing the system, adding functionality and increasing performance, but we are also using it as a research tool: several techniques employed by the current version of Aditi were first implemented and evaluated using previous versions. Other aspects of Aditi build on original research done previously by members of the Aditi team (Balbin, Port and Ramamohanarao, 1990; Balbin and Ramamohanarao, 1987; Kemp, Ramamohanarao, Balbin and Meenakshi, 1989; Kemp, Ramamohanarao and Somogyi, 1990; Ramamohanarao and Shepherd, 1986; Ramamohanarao et al., 1988).

In this paper we provide an overview of the structure and operation of Aditi, and we discuss some of the design issues that we tackled during the implementation of the system. The structure of the paper is as follows. Section 2 introduces the reader to deductive databases in general and defines some terms used in the rest of the paper. Section 3 gives the structure of Aditi and describes each of its major components. Section 4 talks about the front-end of Aditi: the languages used in Aditi and the transformations between them. Section 5 talks about the performance of the current Aditi prototype. Section 6 surveys other deductive database systems while section 7 outlines some future directions for our project.

2. DEDUCTIVE DATABASES

2.1 Logic as a database language

The idea of using predicate logic as a database language

1Aditi is the name of an Indian goddess; she is "the personification of the infinite" and "mother of the gods".

Copyright © 1991, Australian Computer Society Inc. General permission to republish, but not for profit, all or part of this material is granted, provided that the ACSI's copyright notice is given and that reference is made to the publication, its date of issue, and to the fact that reprinting privileges were granted by permission of the Australian Computer Society Inc.
has been around for more than a decade (Gallaire and Minker, 1978). It has several advantages:

- One can naturally view in logical terms virtually all database concepts of interest to users: queries, views and integrity constraints as well as the data in the database itself. This allows the database to present a single unified interface to its users.

- Deductive databases provide more expressive power than most relational databases. In relational terms, they can naturally represent non-first-normal-form relations, and they allow the definition of a view to depend on itself (this comes in particularly handy when dealing with transitive closure problems).

- Due to the similarity of their basic concepts, it is a relatively simple matter to embed a programming interface to a deductive database in a general-purpose logic programming language such as Prolog, thus simplifying the process of writing application programs. Prolog suits many of the problem domains that need the extra expressive power of deductive databases. As for applications that are not suited to Prolog, it is no harder to access a deductive database from a C or Cobol program than it is to access a relational database from those languages.

In the rest of this subsection we will show how deductive databases use logic as a database language. First, here is how they use logic to represent data:

```prolog
edge(a, b).
edge(a, c).
edge(b, c).
edge(b, d).
```

These four facts say that there are edges from a to b, from a to c, from b to c and from b to d. The equivalent SQL statements are:

```sql
create table edge (source char(20) not null, sink char(20) not null);
insert into edge values ('a', 'b');
insert into edge values ('a', 'c');
insert into edge values ('b', 'c');
insert into edge values ('b', 'd');
```

Unlike relational database systems, deductive databases allow tuples ("facts") to contain structured data. For example, one relation can state that unit 101 is taught by John Doe and is attended by three students with student numbers 890001, 890002 and 890003:

```prolog
course(101, john_doe,
[890001, 890002, 890003]).
```

The same data requires at least two relations in SQL, one linking units to lecturers and one linking units to students. This design is not at all ideal: with a large class, the name of the unit would be replicated many times, and information about the unit is scattered in several relations and several disk blocks. The deductive database design avoids both these problems. (There are some so-called post-relational database systems that can also store non-first-normal-form or NFNF databases; see e.g. Roth, Korth and Batary, 1987; Sacks-Davis, Wen, Kent and Ramamohanarao, 1990.)

Given that relational systems cannot handle this example, we continue with the previous one. In a deductive database, a query on the relation `edge` and its answer look like the following:

```prolog
?- edge(a, Y).
Y = [b, c].
```

The query asks for the names of all nodes Y that are at the ends of edges from a; the answer is the set containing b and c. The equivalent SQL query is:

```sql
select sink from edge where source = 'a';
sink
_____
'b'
'c'
```

To define a new relation as a view on other relations:

```prolog
twoedges(X, Y) :- edge(X, Z), edge(Z, Y).
```

This derivation rule defines a new relation called `two edges`. It states that nodes X and Y are separated by two edges if there is an edge from X to a third node Z, and an edge from Z to Y. (The symbol `\>', which looks somewhat like a left arrow, should be viewed as "is implied by". Derivation rules are so named because they can derive new facts from old ones.) The equivalent SQL view definition is:

```sql
create view twoedges(e1.source, e2.sink) as
select e1.source, e2.sink
from edge e1, edge e2
where e1.sink = e2.source;
```

Deductive databases allow database designers to generalise this notion. Using logic, one can define a relation `path` such that a tuple `<X, Y>` is in this relation if and only if there is a sequence of one or more edges in relation `edge` leading from X to Y.

```prolog
path(X, Y) :- edge(X, Y).
```

The above derivation rules state that if there is an edge between X and Y then there is a path between X and Y; and that if there is some node Z such that there is an edge between X and Z, and a path between Z and Y, then there is a path between X and Y. Relation path is thus the transitive
closure of edge; any changes to the edge relation will be instantly reflected in the path relation.

A user of this database can make queries on the relation path as if it were a normal relational database relation. For example, to find out which nodes are reachable from a using the edges in edge, and which nodes one can reach node c from, one would write:

```prolog
?- path(a, Y).
Y = {b, c, d}.
?- path(X, c).
X = {a, b}.
```

These queries cannot be expressed directly in standard SQL because they involve recursion. To allow users to ask the first type of query, the database designer must write a program in a procedural programming language such as C using embedded SQL:

```sql
path_from()
{
  exec sql begin declare section;
  char source[20];
  int curr_count, prev_count;
  exec sql end declare section;
  exec sql connect;
  exec sql execute immediate
    create table prev (node char(20) not null);
  exec sql execute immediate
    create table curr (node char(20) not null);

  printf("from: ");
  scanf("%s\n", source);

  exec sql execute immediate
    insert into curr
      select distinct sink from edge where source = :source;
  exec sql execute immediate
    select count(distinct *) into :curr_count from curr;

  prev_count = 0;
  while (curr_count != prev_count)
  {
    exec sql execute immediate
      delete from prev;
    exec sql execute immediate
      insert into prev
        select * from curr;
    exec sql execute immediate
      insert into curr
        select edge.sink from prev, edge
      where prev.node = edge.source;
  }
}
```

The difference in complexity between this SQL/C program and the deductive database definition of relation path is very pronounced. When one considers that in a deductive database one can use the same definition to make four types of queries on relation path, each of which would require a separate SQL/C procedure in a relational database, the difference in expressive power looks even bigger.

Most deductive databases answer a query such as `?- path(a, Y)` by starting with the facts and continually deriving new facts from existing ones until they cannot derive any more (which is what the SQL/C code above does). In this case this means applying the first rule of path once and the second rule a variable (usually large) number of times. Because this method goes from the facts towards the query, it is called bottom-up computation, in contrast to the top-down computation strategies used by the implementations of most logic programming languages.

Raw bottom-up computation can be very inefficient: it can compute very large numbers of irrelevant facts, print out the ones the user asked for, and compute the same set of irrelevant facts for the next query. However, this is of absolutely no concern to the users of serious deductive database systems, since they all incorporate optimisations that focus query evaluation on the facts needed to answer
exec sql execute immediate
  select count(distinct *) into :curr_count from curr;
exec sql execute immediate
  select count(distinct *) into :prev_count from prev;
}

exec sql execute immediate
  drop table prev;
/* table curr contains the answers */
}

the query. Some deductive databases now internally
generate code that resembles to a greater or lesser degree
the SQL/C code that a programmer would write to execute
a given query (see section 4.2 for an example).

Besides aggressively incorporating the best relational
database technology into their systems, deductive data-
base researchers have found several optimisation algo-
rithms for queries that cannot be expressed in relational
databases at all (e.g. ?- path(a, Y)). At present, relational
database systems are still faster than deductive database
systems because much more effort has been spent on
tuning them. However, we predict that deductive data-
bases will catch up in almost all respects in five to ten years
time, and that the greater convenience and expressive
power of deductive databases will persuade many people
to migrate their applications from relational to deductive
databases as they once migrated from hierarchical or net-
work databases to relational databases.

2.2 Basic requirements for a deductive database system
Before we go on to describe Aditi itself, we would like to
define in general terms just what we consider a deductive
database system to be. We see a deductive database system
as a database system whose user interfaces are based on
predicate logic, or, equivalently, as a logic programming
system oriented towards manipulating the large amounts
of data found in typical database systems. We believe all
deductive databases should meet the following
requirements:

- Logic should be the only user-visible language used in a
deductive database system. That is, applications and
database queries should both be written in a language
based on logic and one should be able to embed queries
in application programs without any change in syntax
(unlike present systems where the dividing line between
(say) C and SQL is very clear and frequently annoying).
User application code and queries to the database
should be seamlessly integrated. A language such as
pure Prolog with extensions for type and mode declara-
tions, quantification, and aggregate operations would
meet these requirements.

- A deductive database system should allow information
to be shared by several different users and to persist
beyond the lifetime of any given application program
execution. This requires support for transactions, rec-
covery, integrity constraints and the ability to store rules
as well as facts in the database.

- To be accepted in a world full of relational database
systems, a deductive database system must have speed
that is competitive with the best commercial relational
database systems. Deductive database systems should
therefore make use of the best available database tech-
nology; this includes techniques such as query optimisa-
tion, parallel query evaluation, flexible indexing and
data clustering.

The first requirement separates deductive databases
from relational database systems, whereas the second and
third separate deductive databases from logic program-
ning systems.

There are four ways to create a deductive database
system.
1 Start with an existing database system and try to add a
veneer of logic on top.
2 Start with a logic programming system and try to mod-
ify it to handle large amounts of data.
3 Connect together an existing database system and an
existing logic programming system.
4 Start writing a deductive database system from scratch.

The second approach suffers from a problem called
impedance mismatch. One can modify a Prolog system
relatively easily to support predicates stored as permanent
relations on disk, but operations (e.g. joins) on these exter-
nal relations are still performed a tuple-at-a-time by the
Prolog system rather than being evaluated within a data-
base system that knows the structure of the indexing. This
effectively rules out the use of many of the most effective
relational database techniques, and thus places unacceptable limits on the performance of the system.

The third approach works only if the existing systems were originally designed with integration in mind, which is virtually never the case. Unfortunately, connecting relational database techniques, and thus places unacceptable limitations. The only problem with it is that it requires a lot of work.

Here at the University of Melbourne we have developed systems using the second and third approaches. We have modified NU-Prolog (Thom and Zobel, 1988) to allow the storage of facts in external relations using various indexing techniques (Ramamohanarao et al., 1988) and to use an existing SQL database as a backend processor (Zobel and Ramamohanarao, 1988). We have learnt a lot from these projects. Nevertheless we are convinced that deductive databases will not match the speed of relational databases until their designers start from a clean sheet of paper; the requirement to fit in with existing software simply constrains performance too much. Accordingly, we adopted the fourth approach for Aditi.

2.3 Definitions and terminology

In this paper we assume that the reader is familiar with standard database terminology. We also require some standard logic programming terminology, some of which the reader has met by now; we now define these terms formally.

The language we use is pure Prolog, which is essentially the first order predicate calculus with some restrictions and extensions and with some changes in notation. Our programs consist of variables, function and predicate symbols, and the logical connectives for conjunction, negation, and implication. Variables are represented by identifiers that start with upper case letters. Function and predicate symbols are represented by identifiers that start with lower case letters. Function symbols and predicate symbols have an associated arity that specifies the number of their arguments; function symbols of zero arity are also called constant symbols. Numbers (integer and floating point) are also constant symbols. Conjunction is represented by a comma \texttt{,}, negation is represented by the word \texttt{not}, and implication is represented by the symbol \texttt{:-}.

**Definition 2.1** A term is defined recursively as follows:

- a variable is a term
- a constant is a term
- if \( f \) is a function symbol of \( n \) arguments, and \( t_1, ..., t_n \) are terms, then \( f(t_1, ..., t_n) \) is a term.
- A term that is not a variable or a constant is also called a structured or complex term. A term that contains no variables is called a ground term.

**Example 2.1** The following are terms:

42, \( X \), \text{john}, \text{parent(father, Parent, child(fred, date(2,2,1990)))}.

**Definition 2.2** If \( p \) is a predicate symbol of arity \( n \) and \( t_1, ..., t_n \) are terms, then \( p(t_1, ..., t_n) \) is an atom. An atom or the negation of an atom is a literal.

**Definition 2.3** A derivation rule for a predicate \( p \) of arity \( n \) is of the form

\[ p(t_1, ..., t_n) :- L_1, ..., L_m, \]

where each \( L_i \) is a literal. \( p(t_1, ..., t_n) \) is the head and \( L_1, ..., L_m \) is the body. If the rule contains no variables and \( m \) equals zero, then the rule is written as \( p(t_1, ..., t_n) \) and is called a fact. A predicate is defined by one or more derivation rules. In this framework, relational database relations are represented by predicates all of whose derivation rules are facts. We call these extensional database predicates or EDB predicates, and we store their data the same way as a relational database would, except that our tuples may contain structured terms. All other predicates (i.e., predicates with non-fact rules) correspond to (possibly recursive) relational views. Aditi stores these intensional database or IDB predicates implicitly in the form of instructions that “materialise” the desired part of the IDB predicate whenever the user or another part of the system makes a query on that relation.

3. THE STRUCTURE OF ADITI

Aditi is based on the client/server model found in many commercial relational database systems. Users interact with a front-end process (FE) that is regarded as a client of the system. The client communicates with a back-end process (server) that performs the usual set of database operations, such as joining, merging, and subtracting relations, on behalf of the clients. Some systems have one server per client, while others have one server supporting multiple clients. Aditi is a hybrid of these two schemes: some of its server processes are dedicated to clients while others are shared by all clients.

The dedicated server process, called a Database Access Process (DAP), performs the initial authorisation clearance of the client as well as all tasks connected with query evaluation except the execution of relational algebra operations. Those operations are performed by a pool of server processes called Relational Algebra Processes (RAPs). These provide the relational operations required for query evaluation. The pool of RAPs is managed by a master process called the Query Server (QS). Figure 1 illustrates how the pieces fit together.

As a DAP evaluates a query, the relational algebra operations are sent to the QS for execution. If there is a free RAP, then the QS passes the task on to that RAP, otherwise the task is queued until a RAP becomes available. The RAP then performs the task and notifies the requesting process called the Query Server (QS). Figure 1 illustrates how the pieces fit together.
ADITI DEDUCTIVE DB

Figure 1. The structure of Aditi.

DAP of the result. The RAP also informs the QS that it is available for another task. To reduce communication overhead, we have implemented our own IPC mechanisms using shared memory; nevertheless we intend to implement a scheme in which a RAP can be assigned exclusively to a DAP for several tasks, the RAP being returned to the process pool only when the DAP is finished with it.

Here is a short summary of the properties of the various processes.

FE The clients of Aditi are called Front End processes. When making interactive queries on the database, one would use the query shell (see below) as a front end. When one wants to write applications using Aditi embedded in an interpreted language such as NU-Prolog, the front end would be the (modified) NU-Prolog interpreter. When one wants to write applications using Aditi embedded in a compiled language such as C, the front end would be the application program itself.

DAP Aditi requires each Front End process to access Aditi through a Database Access Process or DAP. DAPs are responsible for database security and they oversee the execution of queries. There is one DAP per live Front End process.

QS The Query Server or QS is responsible for managing the load on the machine. In operational environments, there will be one QS per machine (in our development environment, one can set up other query servers for testing).

RAP Relational Algebra Processes or RAPs carry out relational algebra operations on behalf of the DAPs. RAPs are allocated to DAPs for the duration of one such operation. The number of RAPs that can be active at a given moment is controlled by the QS within configurable limits, so there is no necessary connection between the number of DAPs and RAPs in the system.

3.1 The query shell
The current front end to Aditi is an interactive query shell. This shell, which is written in NU-Prolog, waits for the user to make queries to the database and evaluates them. The queries are expressed in Aditi-Prolog, which is simply pure Prolog with some extensions such as type and mode declarations and aggregate operators. The query shell transforms each query into a rule whose body is the query itself and whose head is an atom whose predicate is a special query predicate and whose arguments are the variables in the query. The query shell then uses the Aditi compiler (see section 4.3) to translate the query into the language of the DAP, and passes the translated query to the DAP to be executed. When the operation is complete, the DAP leaves its result in a temporary file; the query shell looks up this file and displays its contents to the user. In most cases, the temporary file is then deleted, although the user can save it and use it in later queries.

3.2 Database Access Process
The Database Access Processes are the portals through which users must pass to gain access to Aditi: the DAPs have the responsibility of enforcing the security restrictions (if any) on access to shared databases. DAPs are trusted processes; they always check whether the user on behalf of whom they are executing has the authority to carry out the operation he or she requests the DAP to perform. A front-end process (such as the query shell) can get access to the system in one of two ways. It can create a child DAP process for itself and set up a dedicated message channel to it, or it can have a copy of the DAP built into it. Though the latter method is faster, for obvious reasons it is confined to application programs that are themselves trusted.

After startup and the completion of the obligatory privilege check, DAPs go into a loop waiting for commands and executing them. These commands are not the original logic queries, but calls to procedures coded in a bytecode version of RL, the Aditi relational language whose operations are based on those of relational algebra. The DAP contains an interpreter to execute this code; this interpreter executes some operations itself and sends others to the QS for assignment to some RAP. Operations that require large amounts of computation, such as joins, unions etc are performed by RAPs; others, such as determining a relation's cardinality and creating and modifying relations are performed directly by the DAP.

Each DAP has a table containing the names and the code of the procedures it knows about. To execute an open query on a single predicate, a front-end process tells the DAP to execute the corresponding procedure in its procedure table. To execute any queries that are more complex, a front-end process compiles the query into RL, assembles it, and loads it into the DAP for immediate execution. The DAP leaves the results of queries in temporary relations;
the front-end can ask the DAP for the contents of these relations and may use them in later queries. The DAP also supports commands for updating permanent relations and for shutting down the system (the latter for privileged users only).

3.3 Query Server
The Query Server (QS) is the central management process of Aditi; it must be running for Aditi to be available to users. At startup, it is responsible for initialising the enter database system according to a specified global configuration file and starting the appropriate number of RAPs. During normal operation, its main task is to keep track of the state of each RAP and to allocate tasks to free RAPs. The QS listens on its message queue waiting for various requests and either performs them immediately or queues them until they can be performed. Requests can come from DAPs and RAPs. A DAP can make requests to log in to Aditi, to have a task executed, to get the status of the system, and to log out from Aditi. It can also send the QS a message to shut down the system if its user is authorised to do so. A RAP can tell the QS that it has finished an assigned task and is ready for another task, or it can inform the QS that it is aborting due to some fatal error.

By controlling the number of RAPs and DAPs running and keeping usage statistics for each user, the QS can perform load management for Aditi. Controlling the number of RAPs and DAPs allows the QS to prevent Aditi from crippling the host system. Recording usage statistics means that the QS can prevent individual users from overloading Aditi itself.

The QS can control the number of DAPs by refusing to log in new DAPs when the load on the system is too high. Since DAPs place relatively small loads on the system, such action should be quite rare. Control over the number of RAPs is more critical: relational algebra operations are the most expensive operations performed in a database system and can easily overload the host machine. The QS prevents this by not passing DAP requests onto the RAPs when the load on the machine is dangerously high, and by not creating more RAPs than the machine can support. This way many DAPs can be running, providing service to many users, but the overall number of jobs being performed is strictly controlled.

3.4 Relational Algebra Process
Relational Algebra Processes (RAPs) are the workhorses of Aditi. They execute the relational algebra operations required to evaluate queries. The set of operations supported include the standard ones (e.g. join, union, difference, select, project) as well as special combinations of these that we use to optimise the bottom-up evaluation of queries.

A RAP, when initially started, allocates a message queue for itself to receive jobs from the QS. It then notifies the QS that it is alive and what its message queue address is. During normal execution, the RAP cycles through: waiting for a job, executing the job, and sending the result to the requesting DAP. A RAP exits only when the QS commands it to shut down or it encounters some fatal error.

Messages sent to the RAP specify the name of the operation to be performed, the input relations, select conditions for these input relations, any data specific to the operation (e.g. join attributes), and a list of output relations along with projection information that specifies how each tuple in those relations is to be constructed from the result of the operation. All relational algebra operations can therefore be preceded by select operations and followed by project operations. The select conditions on the input relations often allow the RAP to confine its attention to the relevant parts of the input relations without incurring the overhead of creating temporary relations to hold those parts. In the same spirit, the projection information allows the RAP to allocate storage only for the part of the output that is actually required; the fact that more than one output relation can be specified (presumably each with a different projection) allows the RAP to avoid the overhead of scanning the results of the operation several times. A secondary benefit of pre-select and post-project is that they reduce the communication overhead between the DAP and the RAP.

The implementations of the various RAP operations (notably join) are currently based on an external sort-merge procedure (Knuth, 1973). The sort-merge procedure sorts a relation on a given list of attributes using prepended sort keys (Linderman, 1984). The sorted tuples can be written to a temporary relation or passed back to the calling routine during the final merge phase (thus avoiding an extra read and write of the tuples). More than one sort-merge can be in progress per RAP: a RAP can have one sort-merge running for each input relation. The sort-merge can leave the sort keys in place if the calling procedure wishes to use them. Thus relations can be sorted just once instead of once per operation. Such presorting on carefully chosen attributes can substantially reduce query time. The sort-merge procedure can also remove duplicates. This has proven essential for recursive queries; without it, the sizes of temporary files can grow exponentially.

Currently, each relational algebra instruction is carried out on a single RAP, but there is no fundamental reason why multiple RAPs cannot cooperate on a single task. We are working on schemes that would enable multiple RAPs to co-operate in sorting a single relation in parallel, and on parallel hash based join algorithms (Nakayama, Kitsuregawa and Takagi, 1988).
3.5 The database interface

An Aditi database is a set of relations implemented as files and subdirectories stored in one directory in the Unix file system. Every database must have a data dictionary relation, which contains schema information about all relations in the database.

The database interface (DBI) is a set of routines within Aditi that are used by both DAPs and RAPs to access these databases. The interface provides routines for creating and deleting temporary and permanent relations, inserting, deleting, and retrieving tuples from relations, and tuple-level and relation-level locking. The resemblance of this layer to standard relational database interfaces stems from Aditi’s internal use of relational database technology. Figure 2 gives the structure of the DBI.

The most important function of the DBI layer is to act as a switch between different indexing methods and to hide the differences between these methods. The indexing methods currently supported are B-trees, superimposed coding with and without multi-key hashing for clustering, and sequential files without indexing. We plan to add support for further indexing methods (e.g. bang files (Freeston, 1988)).

The DBI provides a set of routines for selecting the set of tuples from a relation that satisfy some condition. The compiler is responsible for splitting each selection condition into two parts: a first part that can take advantage of indexing and a second part that can’t. For example, consider a two-attribute relation p(X, Y) that is indexed only on its first attribute. A select condition that specified values for both attributes would have the condition X = a in its first part; this will be used to retrieve a set of pages that contain a superset of the answers for the whole selection. The second part of the condition Y = b will then be applied to this superset to select the final answers to the query.

Aditi provides three kinds of tuples. Relational database type relations can be stored in a relational database type tuple format: fixed length records with each attribute having a predefined type and maximum size. Deductive database type relations may require the flexibility of the other main tuple format, variable length records in which each field may be of arbitrary size, and may include function symbols nested arbitrarily deep. This latter feature is necessary to support the structured terms one may find in Aditi-Prolog programs. The third tuple format is intended for the storage of RL programs in Aditi relations. All tuples in a relation must have the same format. The database interface hides the differences between tuple types from the rest of Aditi.

The lowest layer of Aditi is the system interface library, which provides an OS-independent set of services to the rest of Aditi. The current version is based on Unix. It implements file locking, message passing and a disk buffer cache using the shared memory facilities available with Unix System V IPC. Synchronisation of accesses to this memory is handled via a combination of spinlocks and semaphores.

4. THE FRONT END OF ADITI

The administrators of a database system need to be able to create and destroy EDB and IDB relations; the users of a database system need to be able to update EDB relations and to make queries. In the final version of Aditi, the query shell and the Aditi-Prolog compiler will be responsible for providing this functionality. Due to the prototype nature of the current system, some of these facilities are at present implemented as Unix commands (e.g. mkrel and rmrel); later we plan to incorporate these commands into the query shell.

When supplying tuples for EDB predicates, defining IDB predicates or making queries, the users interact with Aditi using only Aditi-Prolog. The DAP, however, understands only its own machine language, which is a bytecode version of RL, the Aditi relational language. The Aditi compiler, which is written in NU-Prolog, converts facts, predicate definitions and queries from Aditi-Prolog to RL; an assembler written in C converts RL to the bytecode expected by the DAP. The query shell executes simple queries directly, but passes anything complicated (e.g. conjunctions) to the compiler for conversion into RL.

In sections 4.1 and 4.2 we describe Aditi-Prolog and RL programs and give examples of each. Section 4.3 describes the compiler that converts programs from Aditi-Prolog to RL.

4.1 Aditi-Prolog

Aditi-Prolog is essentially just pure Prolog augmented with declarations. Some declarations tell the compiler something about the properties of the predicate: e.g. which arguments will be known when the predicate is called. Others request specific rule transformations or evaluation strategies. Among those available are naive evaluation, differential or semi-naive evaluation (Balbin and Ramamohanaroa, 1987), evaluation by magic set interpreter (Port, Balbin, Meenakshi and Ramamohanaroa, submitted for publication), magic set transformation (Bancilhon,
Maier, Sagiv and Ullman, 1986; Beeri and Ramakrishnan, 1987), constraint propagation (Kemp, Ramamohanarao, Balbin and Meenakshi, 1989; Mumick, Finklestein, Pirahesh and Ramakrishnan, 1990), and transformations for linear rules (Han and Liu, 1989; Kemp, Ramamohanarao and Somogy, 1990; Naughton, Ramakrishnan, Sagiv and Ullman, 1989). Here is a full version of the example in the introduction.

?- mode(edge(f,f)).

?- mode(path(b,f)).
?- mode(path(f,b)).
?- flag(path, 2, diff).

path(X, Y) edge(X, Y).
path(X, Y) edge(X, Z), path(Z, Y).

The first line declares that the predicate edge has two arguments and that it expects to be called with both arguments free. Since this code has no definition for edge, it must be an EDB relation or a separately compiled IDB predicate.

The second and third lines declare that the predicate path has two arguments, and that it has two modes: it can be called either with the first argument bound to a ground term and the second argument free (e.g. ?- path(a, Y)) or with the second argument bound and the first argument free (e.g. ?- path(X, b)). The fourth line requests differential evaluation (Balbin and Ramamohanarao, 1987) for the path predicate with arity 2. The last two lines are the rules defining this predicate.

4.2 Aditi relational language

RL is a simple procedural language augmented with relational algebra operations. The primitive operations of RL fall into the following classes:

- the standard relational algebra operations such as join, union, difference, select and project
- extended relational algebra operations such as union-diff, which performs a union at the same time as a difference, thus saving the overhead of scanning the input relations twice
- data movement operations such as append (union without checking for duplicates), copy (which copies a relation) and assign (which copies a pointer to a relation)
- operations concerned with data structure optimisation such as presort-relation
- arithmetic and relational operations on integers and floating-point numbers and the usual operations on boolean values.

The DAP sends most of the operations in the first four classes to the QS for execution by some RAP; those in the fifth class and some others (e.g. copy-pointer-to-relation) are carried out by the DAP itself.

The control structures of RL are a similar mix of simplicity and sophistication: RL supports only gotos, conditional branches, and procedure calls (including recursion).

The procedures are the key to RL. An RL program has a procedure for every mode of every predicate in the Aditi-Prolog program it was derived from. A procedure corresponding to an EDB predicate with all arguments free merely returns a pointer to the permanent relation; a procedure corresponding to an EDB predicate with some arguments bound will simply perform a select on that permanent relation.

Procedures that implement IDB predicates are more complex, partly because they naturally require several steps to implement and partly because they offer more opportunities for optimisation. Here is an example:

procedure path2_1(init_path, final_path)
relation init_path, final_path;
{
relation new_path, diff_path, edge, nullary;
bool bool1;
int size1;

setrel(new_path, 2);
setrel(diff_path, 2);
setrel(edge, 2);
setrel(nullary, 0);
call("edge2_1", nullary, edge);
copy(edge, final_path);
copy(final_path, diff_path);
(Continued on next page)
This is the RL code corresponding to the first mode of `path`. Note the (intentional) similarity to the “look and feel” of a C program. The name of the RL procedure is derived from the name of the Aditi-Prolog predicate it implements (`path`), its arity (2), and the number of the mode it implements (1).

By convention, all RL procedures generated by the compiler have two arguments. The first is always a relation whose tuples represent the values of the input or bound arguments of the predicate it implements; the second is always a relation whose tuples represent the values of all the arguments of the predicate it implements. In this case, `init_path` has one attribute while `final_path` has two, because `path` has two arguments, only one of which is input in mode number 1. The idea is that when `path2_1` is called, the `init_path` relation must already be known, but that `path2_1` is responsible for determining the contents of `final_path` (in fact any tuples in `final_path` at the time that `path2_1` is called will be overwritten and thrown away).

The body of the procedure `path2_1` begins with the declaration of some local variables, and continues with the creation of (empty) relations of various arities to serve as the initial values of the relation-valued variables (setrel stands for “set temporary relation”).

The next two lines implement the first rule of the `path` predicate. The first line calls the RL procedure for the predicate `edge` with a zero-arity input relation and puts the result in `edge`. Since `edge` is an EDB predicate, its data is stored in a permanent relation, and `edge` will now contain a pointer to this relation. The second line copies the contents of `edge` into the relation `final_path` (like C, RL permits the use of parameters as temporaries, even though RL passes parameters by reference).

The rest of the procedure except for the final line implements the second rule of `path`. Since one cannot know in advance the number of times this rule must be applied, the code implementing it must be a loop; it must be a loop with the test at the bottom because the loop termination condition depends on information produced in the loop body. During the execution of the loop, `final_path` always holds the `path` facts currently known to be true; `new_path` always holds the `path` facts newly discovered to be true; and `diff_path` always holds the `new_path` facts that were not previously known to be true. The last of these is needed because one may rediscover facts that have been known to be true before.

`diff_path` is initialised in the line before the loop (`final_path` was already initialised by the first rule). The loop body starts out by implementing the join implicit in the conjunction `edge(X, Z), path(Z, Y)` by joining `edge` and `diff_path` with the join condition that the second argument of the former be equal to the first argument of the latter. (As both input relations and arguments are numbered from zero, the notation #(0,1) refers to input relation number 0 (i.e. `edge`) and argument number 1 (i.e. `Z`).) The join condition is split into two parts; the first is used for indexing while the second one contains that part of the join condition that can not use any available indexing (in this case the second part is empty). It is the responsibility of the compiler to ensure that the first part is appropriate for whatever kind of indexing is available for the relations to which it refers.

The result of the join is a relation with three attributes, representing the variables X, Y and Z. Since the head of the second rule contains only X and Y, the result is projected onto X and Y (#(0,0) and #(1,1) respectively) before it is assigned to relation `new_path`, which thus contains tuples corresponding to the `path` facts we have discovered in this iteration. We then put any tuples in `new_path` that were not in `final_path` into the relation `diff_path`, and add all tuples in `new_path` to the relation `final_path`. This maintains the invariant that `diff_path` contains tuples for the `path` facts not previously known to be true.
while **final_path** contains tuples for all **path** facts now known to be true. The **uniondiff** instruction carries out both these operations at the same time (its third argument allows one to specify arguments on which the output relations should be sorted; this capability is not used in this example).

The application of the second rule of **path** is now complete; the only thing that remains to be done is to find out whether the loop should continue. To this end, we put the cardinality of the **diff_path** relation into the integer variable **size1**, and test whether this number is less than or equal to zero (the “less than” part is just defensive programming). If it is; we jump out of the loop to **label2**; otherwise, we go back to the start of the loop at **label1**.

When the loop exists, **final_path** will contain the entire **path** relation. **path2** proceeds to throw most of it away by joining **final_path** with **init_path**, thus ensuring that the **final_path** returned to the caller contains only those tuples whose first arguments are in **init_path** — in other words, the tuples the caller asked for. The wastefulness of this should be self-evident. Fortunately, there are optimisation methods that compute only the required subset of the **path** relation, thus avoiding this inefficiency (Bancilhon, Maier, Sagiv and Ullman, 1986; Veeri and Ramakrishnan, 1987; Kemp, Ramamohanarao and Somogyi, 1990; Naughton, Ramakrishnan, Sagiv and Ullman, 1989; Ullman, 1989). Aditi implements many of these optimisations; we chose to show the unoptimised version for the sake of exposition. Using the preferred optimisation for this query, the NRSU-transformation (Naughton, Ramakrishnan, Sagiv and Ullman, 1989), the compiler produces RL code that does essentially the same things as the SQL/C code in section 2.1.

### 4.3 The compiler

The compiler that turns programs written in Aditi-Prolog into RL is written in NU-Prolog (Thom and Zobel, 1988). Unlike most compilers, it represents programs in not one but two intermediate languages, which we call HDS and LDS (for “high-level data structure” and “low-level data structure” respectively). HDS provides an easy-to-manipulate representation of Prolog rules while LDS provides an easy-to-manipulate representation of RL programs. The compiler has three main stages: Aditi-Prolog to HDS, HDS to LDS, and LDS to RL, in addition to optional optimisation stages that transform HDS to LDS or LDS to LDS.

The first main stage, Aditi-Prolog to HDS, is concerned mainly with parsing the input and filling in the slots in the HDS representation of the program. Some of the tasks required for the latter are trivial, such as finding the scopes of all variables, while others can have great impact on the performance of the final code, such as selecting an appropriate sideways information-passing strategy or sip (Ullman, 1985).

The HDS to HDS level is where the compiler implements the optimisations that are defined in terms of source-to-source transformations. These optimisations include magic set transformation, counting set transformation (Bancilhon, Maier, Sagiv and Ullman, 1986; Beeri and Ramakrishnan, 1987), constraint propagation (Kemp, Ramamohanarao, Bilbin and Meenakshi, 1989; Mumick, Finklestein, Pirahesh and Ramakrishnan, 1990), and transformations for linear rules (Kemp, Ramamohanarao and Somogyi, 1990; Naughton, Ramakrishnan, Sagiv and Ullman, 1989).

The second main stage, HDS to LDS, is responsible for converting a predicate calculus oriented representation of the program into a representation geared to relational algebra operations. Among other things, this requires the transformation of recursive Prolog rules into RL procedures containing iteration. If several predicates are mutually recursive, then the compiler generates a single procedure containing one big iteration that computes values for all these predicates. It also generates an interface procedure for each predicate involved; these call the procedure containing the iteration and select the data they need from it.

The translation from HDS to LDS can take any one of several different paths. The paths currently implemented are naive and differential implementations of the standard bottom-up interpreter, naive and differential implementation of the magic set interpreter (Port, Balbin, Meenakshi and Ramamohanarao, submitted for publication), and an implementation of the predicate semi-naive technique (Ramakrishnan, Srivastava and Sudershank, 1990). Each of these implementation schemes is good in some circumstances and bad in others. It is an open research problem to find the domains of optimality of each; it is a problem we intend to pursue with the help of Aditi.

LDS to RL optimisations come from the programming language and relational database literature (in contrast to HDS to LDS optimisations, which come mainly from the logic programming/deductive database literature). They include techniques such as common subexpression elimination, loop invariant removal, moving selections before joins and the intelligent exploitation of any available indexing.

The third main stage, LDS to RL, is necessary for two main reasons. First, LDS has if-then-else and while loops, whereas RL has labels and gotos, and second, LDS has no notion of the bundling of several operations together as occurs with pre-select and post-project as well as with operations such as union-diff. (These differences are caused by the fact that the aim of LDS is easy optimisation while the aim of RL is easy assembly and fast execution.) So the LDS to RL translator includes a peephole optimiser that can convert a sequence of LDS operations into a single RL instruction.

The output of the compiler is human readable RL code.
(like the example in the previous section). The translation of RL into the bytecode needed by the DAP is the task of a small, fast assembler written in C. This design makes it convenient to inspect the output of the compiler, making debugging the compiler easier. It also allows us to write RL code ourselves. This allows us to exercise much of Aditi even without a working compiler, which is very important when the compiler itself is under development and hence sometimes not working correctly. It also allows us to try out hand optimised queries on the system; we can thus experiment with optimisations before deciding to incorporate them in the compiler.

5. PERFORMANCE

The current version of Aditi, though functional, is essentially a prototype. We have given functionality and completeness higher priority than performance. Hence there are no absolute performance measures available at this time.

However, we have taken some performance measurements to determine the effectiveness of various algorithms within Aditi. For example, the union diff operation (which performs both a union and a difference of its input files in one pass) in the RAP can reduce query execution time by 15 per cent compared to separate union and diff operations. Similarly, presorting of relations can reduce execution time by a factor of two for some queries.

We have also used Aditi to evaluate the relative effectiveness of some optimisations that use source-to-source transformations. We have included the results from these tests as a very preliminary indicator of performance in Aditi. See Kemp, Ramamohanarao and Somogyi (1990) for background and information on the predicate definitions and transformations being used. The results are given as elapsed time to measure how a user would see the actual benefit of these transformations. This takes into account the effect of such things as context switches, waiting for disk I/O, etc. that cannot be measured easily otherwise. It also gives an idea of the overall performance of Aditi.

In these tests we compared our own context transformation against the magic set transformation and the null transformation. We tried out each of these algorithms on three programs. We ran each of the resulting nine programs on three sets of data, asking two queries with three programs. We ran each of the resulting nine programs on three sets of data, asking two queries with three programs. We ran each of the resulting nine programs on three sets of data, asking two queries with three programs.

Each entry in Table 1 gives the number of seconds of real time required to execute the open query to q on the given combination of program and data sets, and (in parentheses) the speedup compared to the untransformed program for the same data sets. All computations used the differential evaluation strategy (Balbin and Ramamohanarao, 1987).

Every figure we report is based on the average of several runs. When t contained just one tuple, the repetition served only to reduce timing errors. When t contained several tuples, the repetition served to eliminate the influence of the large variation in performance between different sets of input values: each measurement is an average of several runs with different t relations. For the large times, two or three runs were enough, but in some cases we needed four or five runs to establish a reasonably firm value for the mean. The majority of the raw results lie within about five per cent of the mean.

All the testing was performed on an Encore Multimax 320 with 64 megabytes of memory under UMAX 4.2 (revision 3.3.1). The test database was stored on an NEC D2362 disk drive connected via an asynchronous SCSI interface with a 1.5 Mb/s maximum transfer rate.
Table 1. Timing results using the Aditi deductive database.

<table>
<thead>
<tr>
<th>Table 1. Timing results for right-linear query evaluation</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>depth = 9</strong></td>
</tr>
<tr>
<td># t = 1</td>
</tr>
<tr>
<td>.orig</td>
</tr>
<tr>
<td>magic</td>
</tr>
<tr>
<td>context</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Table 1. Timing results for left-linear query evaluation</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>depth = 9</strong></td>
</tr>
<tr>
<td># t = 1</td>
</tr>
<tr>
<td>.orig</td>
</tr>
<tr>
<td>magic</td>
</tr>
<tr>
<td>context</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Table 1. Timing results for multi-linear query evaluation</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>depth = 9</strong></td>
</tr>
<tr>
<td># t = 1</td>
</tr>
<tr>
<td>.orig</td>
</tr>
<tr>
<td>magic</td>
</tr>
<tr>
<td>context</td>
</tr>
</tbody>
</table>

It must be remembered that no profiling or performance tuning of Aditi has been done yet, and in some cases we have implemented the simplest algorithms possible to provide required functionality as quickly as possible. It is certain that the execution times given above will improve once we turn our attention to increasing the performance of Aditi.

Table 1 shows the importance of source level optimisations. The magic set transformation gives substantial speedups, and it is applicable to all predicates. The context transformation performs even better, but it is applicable only to predicates whose rules all belong to the class of rules known as linear recursive rules.

In our experience, the effectiveness of many optimisations is strongly influenced by the number of iterations required to complete the bottom-up computation. Sometimes, a technique will introduce a one-off overhead that cannot be compensated for by the reduced cost per iteration because there are not enough iterations in the bottom-up computation. Presorting of relations is just one such technique. At other times, an optimisation technique may speed up each iteration but require more iterations to answer a query. Unfortunately, one cannot always decide in advance whether a given optimisation is worthwhile; the answer is often query and data dependent. More experimentation is required to determine whether any useful heuristics exist. We envisage the compiler exploiting such heuristics by generating code that switches at run-time between two or more ways of evaluating a query depending on the characteristics of the data.

6. OTHER DEDUCTIVE DATABASE SYSTEMS

There are many other research teams developing deductive database systems. In this section, we describe a representative sample of substantial implementation projects.

The ECRC research group in Munich, West Germany, is focusing on fast implementations. They have developed several systems, including DedGin* (Vieille, 1988a) and EKS-V1 (Vieille, Bayer, Kuchenhoff and Lefebvre, 1990) which used the QoSaQ top-down query evaluation algorithm (Vieille, 1988b). EKS-V1 is layered on top of the MegaLog system (Boccia, Dahman, Freeston, Macartney and Pearson, 1989), which provides efficient storage methods (including bang files) updates, transactions, concurrency and recovery.

The LDL project (Naqvi and Tsur, 1989) at MCC in Austin, Texas, is oriented towards bottom-up computation methods, together with optimisation techniques such as magic sets. However, LDL puts a lot of emphasis in the programmer interface. The LDL language is a distant relative of Prolog: it extends the logic language we presented in section 2.3 both declarative and procedurally. On the declarative side, it provides support for set-valued variables and aggregation; on the procedural side, it...
includes constructs for updating base relations and allows programmers to embed procedural code in otherwise declarative rules.

The Nail project (Morris, Ullman and Gelder, 1986) at Stanford University, California, has resulted in a system that can use different strategies, both bottom-up and top-down, for evaluating queries. The Nail-Prolog rules are compiled into an intermediate language (ICODE) which is executed by an interpreter implemented on top of an existing SQL database. This has been a cumbersome arrangement, as relational database systems find it difficult and expensive (though not impossible) to represent and manipulate structured Prolog terms such as lists.

The Nail group believe that declarative languages like Prolog are not sufficient to express database updates, nor do they allow the expression of certain efficient special-purpose database operations. They have therefore developed Glue, a procedural language with some declarative flavour (Phipps, 1990); it lies somewhere between Aditi-Prolog and RL and has features from both. Nail and Glue procedures can each call procedures written in either language. The intention is that users will write the declarative parts of programs in Nail and the procedural parts in Glue.

The project is focusing on the programming language features of Glue, some of which seem hostile to optimisation (e.g. overly constrained execution order, and the use of arbitrary terms as predicate names).

MegaLog is a disk-based system: it stores relations on disk and uses main memory as a cache. At present, LDL and Glue are memory-based; they keep all their data in main memory while running and use secondary storage only to preserve the database from one invocation to the next. This simplifies their implementation but restricts their applicability to domains whose data (permanent as well as temporary) is small enough to fit into main memory. More seriously from a research point of view, it also prevents them from studying the effect of recursive query optimisation techniques on disk traffic patterns, buffering strategies and other such aspects of real (i.e. large) databases.

7. FURTHER WORK
A prototype of Aditi is functional and we are able to evaluate most queries, but much research and development remains to be done. We outline briefly some of the more important areas we are currently examining.

Transactions: At present, Aditi has no transaction mechanism, although some of the hooks required are present. A single transaction may require the co-operation of several processes (the client DAP and any RAPs used); we need to research into concurrency control methods that allow this. We are also looking at how best to resolve simultaneous updates. As recursive computations can take relatively long periods of time, overlapping updates are more likely in deductive databases than in relational databases. We do not want to lock relations for long periods because this can drastically reduce concurrency. However, optimistic methods that force restarts of computations could cause starvation and waste too many computation resources. We would prefer a hybrid solution that minimises these problems.

Parallelism: The available parallelism still needs to be fully utilised. We are looking at both fine and coarse grained parallelism. Fine grained parallelism is applicable to relational algebra operations. Currently, we are implementing a parallel hash-join algorithm and we also plan to implement parallelised versions of the other relational algebra operations in the future. Coarse grained parallelism can be exploited at the relational language level. All the base rules of a predicate can be evaluated in parallel, and all the recursive rules can be evaluated in parallel as well. We have therefore added constructs to RL that allow several RAP operations to execute in parallel. The interpreter for this extended language (currently being implemented) maintains a context for each thread of parallel execution and communicates with several RAP processes.

Negation and aggregates: The compiler currently generate code for programs with negation as long as the program is stratified in the standard sense, i.e. has no cycles in the predicate graph involving negation. Soon this will be extended to include programs with aggregation operations such as \texttt{count}, \texttt{sum}, \texttt{max} and \texttt{min} as long as the program is stratified with respect to aggregates, i.e. has no cycles involving aggregation operations. We are investigating what modifications to the compiler would be required for dealing with non-stratified programs for both negation and aggregation. We conjecture that a compiler can generate code for sufficiently stratified programs (Dung and Kanchanasut, 1989) with little or no alterations required either to the interpreter or to the relational back-end.

Mixed tuple-at-a-time and set-at-a-time: An advantage of a deductive database system over a logic programming system, is its ability to use relational database techniques for performing computations involving joins with large relations. However, there are often situations where it is far more efficient to perform some computations in a tuple at a time manner. For example, predicates for list manipulation such as list reverse and append should be compiled into code which takes each tuple in the input relation, performs the required list manipulation on that tuple, and then places it in the output relation. This avoids the I/O involved in maintaining the intermediate relations containing the sublists generated by the bottom-up computation of such predicates. Predicates to be computed tuple at a time can be hidden from set at a time computations by encapsulating them in RL procedures that call NU-Prolog to perform their evaluation.

Rule Transformations: We are continuing our investigations into general rule transformation techniques, build-
ing on our experience with techniques such as magic set
interpreters (Port, Balbin, Meenakshi and Ramamohan­
arao, submitted for publication), constraint propagation
(Kemp, Ramamohanarao, Balbin and Meenakshi, 1990) and
special optimisations for linear recursions (Kemp, Ramamohanarao and Somogyi, 1990). In particular, we
intend to undertake an extensive evaluation of the perfor­
mance of each optimisation technique. We intend to use
the derived cost models in the implementation of a strategy
module in the compiler that will select the rule transforma­
tions to be applied to each predicate.

8. SUMMARY
We have presented an overview of the structure of Aditi, a
disk-based deductive database system under continuous
development at the University of Melbourne.

Users interact with Aditi using a variant of Prolog, a
logic programming language that makes it easy to write
applications involving recursion and function symbols.
Aditi's internal operations are based on relational technol­
y, but the system also employs several optimisations
specific to deductive databases. Examples include differen­
tial evaluation, magic set transformation, magic set
interpreter, constraint propagation, and context transfor­
mation for linear rules. Several of these were developed at
the University of Melbourne (Balbin, Port and Ramamo­
hanarao, 1990; Balbin and Ramamohanarao, 1987; Kemp, Ramamohanarao, Balbin and Meenakshi, 1989;
Kemp, Ramamohanarao and Somogyi, 1990; Port, Balbin,
Meenakshi and Ramamohanarao, submitted for publication).

We are currently using Aditi as a tool for research into
deductive databases; it has been the vehicle for the de­
velopment and evaluation of several of the above optimisa­
tion techniques. We aim to eventually use Aditi to prove
that deductive database systems can achieve performance
comparable to that of commercial relational database
systems.

ACKNOWLEDGEMENTS
Many people have contributed significantly to Aditi both
by performing research and by developing software. We
would like to acknowledge the contributions of Kim Mar­
riott, John Shepherd, David Keegel, Warwick Harvey, Tim
Leask, Simon Lienert and Jeff Schultz in this respect. We
would like to thank Hayley Veen for her comments on a
draft of this paper.

This research was supported by a grant from the Aus­
tralian Research Council through the Machine Intelli­
gence Project.

REFERENCES
Set Computations for Stratified Databases, Journal of Logic Pro­
gramming, to appear.

of the differential approach to recursive query evaluation. Journal of

Magic sets and other strange ways to implement logic programs. In
Proceedings of the Fifth Symposium on Principles of Database Systems,
pages 1-15, Washington DC.

Proceedings of the Sixth ACM Symposium on Principles of Database
Systems, pages 269-283, San Diego, California, March.

BOCCA, J., DAHMEN, M., FREESTON, M., MACARTNEY, G. and
PEARSON, P. (1989). KB-Prolog, a Prolog for very large knowledge
bases. In Proceedings of the Seventh British National Conference on
Databases, pages 163-184, Edinburgh, Scotland, July.

to declarative semantics of logic programs. In Proceedings of the First
North American Conference on Logic Programming, pages 604-625,
Cleveland Ohio, October.

Gray and R. Lucas, editors, Prolog and databases, pages 188-211,
Ellis Horwood, Chicester, England.

GALLAIRE, H. and MINKER, J. (1978). Editors. Logic and databases,
Plenum Press.

Proceedings of the First North American Conference on Logic Pro­
gramming, pages 816-830, Cleveland, Ohio, October, MIT Press.

KEMP, D.B., RAMAMOHANARAO, K., BALBIN, I. and MEENAK­
Programming, pages 981-998, Cleveland, Ohio, October.

Right-, left-, and multi-linear rule transformations that maintain
context information. In Proceedings of the Sixteenth Interna­
tional Conference on Very Large Data Bases, pages 380-391, Brisbane, Aus­
tralia, August.

KNUTH, D.E. (1973). Sorting and searching. volume 2 of The art of
computer programming, chapter 5.4.1. Addison-Wesley,
Massachusetts.

working sort routine. Bell Laboratories Technical Journal, 63(8(Part
2)): 1827-1843, October.

MORRIS, K., ULLMAN, J.D. and GELDER, A.V. (1986). Design overview
of the NAIL system. In Proceedings of the Third International Conference on

MUMICK, I.S., FINKELSTEIN, S.J., PIRAHESH, H. and RAMAK­
ville, Tennessee, April.

Hash-partitioned join method using dynamic destaging strategy. In
Proceedings of the Fourteenth Conference on Very Large Data Bases,
pages 468-477, Los Angeles, August.


NAUGHTON, J.F., RAMAKRISHNAN, R., SAGIV, Y. and ULLMAN,

NAUGHTON, J.F., RAMAKRISHNAN, R., SAGIV, Y. and ULLMAN,
Fifteenth Conference on Very Large Data Bases, pages 173-182, Am­
sterdam, The Netherlands, August.

(extended abstract). Technical report, Stanford University, Califor­
nia, August. To be published.

PORT, G., BALBIN, I., MEENAKSHI, K. and RAMAMOHANARAO,
K. Magic sets made simple. Technical Report 88/20, Department of Computer Science, University of Melbourne, Melbourne, Australia. Submitted for publication.


BIODGRAPHICAL NOTES

Jayen Vaghani received his honours degree in Computer Science from the University of Melbourne in 1987. He joined the Machine Intelligence Project as a full-time researcher and programmer in 1988. He is the main designer and implementor of Aditi and hopes to submit a PhD about it.

Kotagiri Ramamohanarao received his PhD from Monash University. He joined the Computer Science Department of the University of Melbourne in 1980 and was appointed professor in 1989. Professor Kotagiri leads the Machine Intelligence Project, which performs research in the areas of logic programming and deductive databases. His other research interests include extended relational database systems, parallel processing, and computer architecture. Professor Kotagiri has been a program committee member and an invited speaker at several international conferences.

David Kemp is a PhD student in the department of computer science, University of Melbourne. His research work has involved the development of program transformation and query evaluation techniques in deductive databases, and he has co-authored several papers that have been presented at international conferences. David Kemp is an enthusiastic member of a canoe-polo team.

Zoltan Somogyi is a lecturer in the Department of Computer Science at the University of Melbourne and a member of the Machine Intelligence Project. His PhD was in the field of parallel logic programming; his current interests include deductive databases, computer architecture, operating systems and system software.

Peter Stuckey received his PhD from the Monash University in 1987. He is currently a lecturer in Computer Science at the University of Melbourne, working with the Machine Intelligence Project. His other research interests include constraint logic programming and linear arithmetic.
1. INTRODUCTION
Relational database systems have become widely used in both business and scientific applications. However applications of database systems are becoming more varied and more complex — particularly in multi-media applications such as image processing, computer-aided design and document databases.

The idea of the nested relational model was developed around the need to extend the relational model to support complex objects in addition to atomic valued attributes. This arose partly from semantic data modelling research (Hull and King, 1987) which indicated that such structures occurred frequently in the real world and partly from the need to efficiently implement relational databases at the physical level by supporting complex objects such as repeating groups.

The original not-necessarily-normalised relations (Makinouchi, 1977) simply allowed relation-valued attributes as well as atomic-valued attributes. Databases allowing relation-valued attributes and other complex objects have also been called nested relational databases or non-first normal form databases.

Nested relational databases are an important area as they combine the efficiency of pre-relational systems with many of the advantages of the relational approach. Furthermore, nested relations can be used to represent complex objects in object-oriented database systems. Object-oriented extensions to some nested relational query languages have already been proposed, for example X-SQL/NF (Korth and Roth, 1989).

A number of major research projects have been developed which have been based on the nested relational model. These include the AIM Project at IBM Heidelberg Scientific Centre in Germany (Pistor and Dadam, 1989), the VERSO project at INRIA in France (Scholl et al., 1989), the DASDB project at Darmstadt in Germany (Schek and Scholl, 1989), the work SQL/NF (Roth et al., 1987) and formal query languages (Roth et al., 1988) at the University of Texas at Austin and the Titan+ project at Melbourne in Australia (Thom et al., 1990).

Over the past decade a number of nested relational database languages have been developed as part of these research projects. In this paper we survey some of the languages developed for nested relational databases and examine in detail the TQL language which has been designed to support advanced text-based applications at RMIT and The University of Melbourne.

Section 2 introduces the nested relational model and compares it with the standard relational model. A nested relational algebra and nested relational calculus are described in this section; as are two query languages based on SQL.

Section 3 describes the details of TQL, the Titan+ query language (Thom et al., 1990). The Titan+ system is a research prototype information system supporting...
advanced features such as the ability to support hierarchical data structures (including nested relations), structured data types and reference pointers for linking database objects. None of these features are supported in standard relational systems. The Titan+ system has also been designed to efficiently support text processing applications (Sacks-Davis et al., 1990) and so TQL contains text data type and a number of text operators.

2. NESTED RELATIONAL MODEL

To illustrate the nested relational model consider a fictitious database about houses and the people who live in them. Consider the following nested relational schema

\[
\text{Houses} \{ \text{street, town, pcode}\}
\]

\[
\text{Occupants} \{ \text{firstname, lastname}\}
\]

\[
\text{History} \{ \text{occupy, vacate}\}
\]

which defines one table, Houses which contains a nested table Occupants which contains a further nested table History. An instance of this schema is shown in Table 1. (Examples within this paper use the following conventions: keywords are shown in all uppercase; table and nested table names begin with an uppercase letter followed by lowercase letters; and tuple names and atomic attribute names are shown in all lowercase.)

In the equivalent normalised relational schema the information about a single house and the people who live in it is fragmented over three relations.

\[
\text{Houses1} \{ \text{street, town, pcode}\}
\]

\[
\text{KEY} = (\text{street, town})
\]

\[
\text{Occupants1} \{ \text{street, town, firstname, lastname}\}
\]

\[
\text{KEY} = (\text{street, town, firstname})
\]

\[
\text{History1} \{ \text{street, town, firstname, occupy, vacate}\}
\]

\[
\text{KEY} = (\text{street, town, firstname, occupy})
\]

If the physical design reflects the logical schema definition (as is almost always the case in relational databases) then the data will be stored in three physical tables. This will make many common queries expensive to evaluate, for example to retrieve the information about a single house, three relations rather than a single table must be accessed. In addition, the query on the relational database will require that two join operations be evaluated. The SQL query to retrieve information about each house can be expressed as

\[
\text{SELECT street, town, pcode, firstname, lastname, occupy, vacate}
\]

\[
\text{FROM Houses1, Occupants1, History1}
\]

\[
\text{WHERE Houses1.street = Occupants1.street}
\]
Using the Houses example introduced above we shall introduce some formal query languages for the nested relational model; an algebra in subsection 2.1, a calculus in subsection 2.2. SQL has become a de facto standard for relational database systems and is described in most standard database textbooks (see for example Date, 1985; Elmasri and Navathe, 1989; Korth and Silberschatz, 1986). Because of its prevalence in the database community, many of the languages proposed for post-relational databases supporting nested tables have been designed to have an SQL flavour. Two of these languages SQL/NF (Roth et al., 1987) and NF2 (Dadam et al., 1986; Pistor and Andersen, 1986) are described in subsection 2.3. Section 3 examines one other query language, TQL, in more detail.

2.1 Nested Algebra

In a nested relational algebra the basic relational algebra operators — union (U), set difference (−), Cartesian product (×), projection (π) and selection (σ) — can behave as in the standard relational algebra except that the domains of these operators may be set-valued as well as atomic. In query 1 we project Houses over the atomic attribute town as well as the nested table which is the result of a query on the nested table Occupants returning the first names of people whose surname is Smith.

Query 1. For each town give the first names of all residents (current and former) whose surname was Smith.

\[ \pi_{\text{town}}(\pi_{\text{firstname}}(\sigma_{\text{lastname} = 'Smith'} \cdot \text{Occupants})) \cdot \text{Houses} \]

<table>
<thead>
<tr>
<th>town</th>
<th>Occupants firstname</th>
</tr>
</thead>
<tbody>
<tr>
<td>Port Melbourne</td>
<td>Alexander</td>
</tr>
<tr>
<td></td>
<td>Catherine</td>
</tr>
<tr>
<td>South Melbourne</td>
<td>Alexander</td>
</tr>
<tr>
<td></td>
<td>Catherine</td>
</tr>
<tr>
<td></td>
<td>Daniel</td>
</tr>
<tr>
<td>Ballarat</td>
<td></td>
</tr>
<tr>
<td>Carlton</td>
<td></td>
</tr>
</tbody>
</table>

Query 1

To handle nested tables the traditional relational algebra needs to be extended with nest (v) and unnest (u) operators (Roth et al., 1988). The nest operator takes a table, groups the tuples on the unspecified attributes and creates a nested table using specified attributes. The unnest takes a nested table and flattens it.

Query 2. Restructure the Houses table into a table giving for each person and a list of current and former addresses.

\[ \text{Addresses} = (\pi_{\text{street}, \text{town}, \text{pcode}}(\mu_{\text{Occupants}} \cdot \text{Houses})) \]

<table>
<thead>
<tr>
<th>first name</th>
<th>last name</th>
<th>street</th>
<th>town</th>
<th>pcode</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alexander</td>
<td>Smith</td>
<td>1/11 Bay Road</td>
<td>Port Melbourne</td>
<td>3207</td>
</tr>
<tr>
<td>Belinda</td>
<td>Jones</td>
<td>2 High Street</td>
<td>South Melbourne</td>
<td>3205</td>
</tr>
<tr>
<td>Catherine</td>
<td>Smith</td>
<td>1/11 Bay Road</td>
<td>Port Melbourne</td>
<td>3207</td>
</tr>
<tr>
<td>Daniel</td>
<td>Smith</td>
<td>2 High Street</td>
<td>South Melbourne</td>
<td>3205</td>
</tr>
<tr>
<td>Edward</td>
<td>Jones</td>
<td>3 Main Street</td>
<td>Ballarat</td>
<td>3350</td>
</tr>
<tr>
<td>Fiona</td>
<td>Anderson</td>
<td>3 Main Street</td>
<td>Ballarat</td>
<td>3350</td>
</tr>
<tr>
<td>Harold</td>
<td>Jones</td>
<td>3 Main Street</td>
<td>Ballarat</td>
<td>3350</td>
</tr>
<tr>
<td>Greta</td>
<td>Brown</td>
<td>3 Main Street</td>
<td>Ballarat</td>
<td>3350</td>
</tr>
<tr>
<td>Isabella</td>
<td>Jones</td>
<td>3 Main Street</td>
<td>Ballarat</td>
<td>3350</td>
</tr>
</tbody>
</table>

Query 2

Other algebraic operators have been proposed, such as extended union (Uε), extended intersection (γ), extended difference (−ε), extended natural join (×ε), and extended projection (πε). Use of these extended operators maintains a desirable property for nested relational databases called partitioned normal form (Roth et al., 1988).

2.2 Nested Calculus

The traditional tuple relational calculus needs to be extended for nested relational databases. The tuple variables in the tuple calculus for nested relations need to range over attributes which can be set valued as well as atomic valued. To allow tuple variables to range over set valued attributes it is necessary that equality comparison of relations (including set constants) is allowed and that attributes can be equated with tuple calculus queries.

In the tuple calculus for nested relations the queries in last section become:

- query 1

\[ \{ t_{\text{Houses}} (\exists \cdot) \cdot (\exists \cdot) \cdot (\exists \cdot) \cdot h \in \text{Houses} \]

\[ \land t[1] = h[1] \]

\[ \land t[2] = \{ s[1] \mid s[1] = p[1] \land p[2] = \mathcal{m}' \mathcal{S} \mathcal{M} \mathcal{I} \mathcal{H} \} \]

- query 2

\[ \{ r_{\text{Houses}} (\exists \cdot) \cdot (\exists \cdot) \cdot (\exists \cdot) \cdot h \in \text{Houses} \land p \in h[4] \]

\[ \land r[1] = p[1] \]


The notion of safety can be extended to nested relational calculus; a safe expression is one in which the answer can be computed in finite time by examining only the relations and constants mentioned in the expression.
The nested relational algebra containing the operators union ($\cup$), set difference ($-$), Cartesian product ($\times$), projection ($\pi$), selection ($\sigma$), nest ($\nu$) and unnest ($\mu$) is a minimal relational algebra of equivalent power to the nested relational calculus restricted to safe expression (Roth et al., 1988).

2.3 SQL-based Languages

2.3.1 SQL/NF

SQL/NF (Roth et al., 1987) was developed as an extension to SQL to handle nested relations. In the process of extending SQL a number of improvements were made to make the language more orthogonal. TQL, which will be described in detail in section 3, closely follows the syntax of SQL/NF.

In SQL/NF the schema for the Houses table would be defined:

```
SCHEMA TABLE Houses
  ITEM street CHARACTER 20
  ITEM town CHARACTER 20
  ITEM pcode INTEGER
  ITEM (TABLE Occupants
    ITEM firstname CHARACTER 15
    ITEM lastname CHARACTER 15
  )
  ITEM (TABLE History
    ITEM occupy INTEGER
    ITEM vacate INTEGER
  )

In SQL/NF, a select-from-where (SFW) expression is allowed anywhere in a query that a relation could appear, for example in query 1:

```
SELECT town,
  (SELECT lastname
   FROM Occupants
   WHERE lastname = 'Smith')
FROM Houses.
```

Nested algebra operations of nest and unnest allow arbitrary restructuring and can be used with an aliasing feature (AS) to rename attributes; for example in query 2:

```
NEST (SELECT firstname, lastname, street, town, pcode
  FROM (UNNEST Houses ON Occupants))
ON street, town, postcode AS Addresses.
```

The aliasing feature can also simplify some queries with repeated subqueries.

Other advantages of SQL/NF are that functions are applied to relations not attributes, and the elimination of GROUP BY and HAVING clauses of SQL.

2.3.2 NF2

The extended NF2 databases described in (Pistor and Traunmueller, 1986) provides a model and language that includes tables both ordered (lists of tuples) and unordered (relations, that is sets of tuples). These lists (or sets) are not restricted to having elements of type tuple but may be lists (or sets) of scalars (atomic data types such as numeric, character or boolean) or may be lists (or sets) of tables (either ordered or unordered). These are included in the language in an orthogonal way so that the basic SFW expression is a generic constructor for both sets and lists.

Sets are represented by pairs {...}, lists by pairs <...> and tuples by pairs <|...|>. To create the Houses example table the following NF2 declaration would be required:

```
CREATE OBJECT Houses {
  <| street: CHAR,
     town: CHAR,
     pcode: INTEGER,
     Occupants: {
       <| firstname: CHAR
          lastname: CHAR
       |
     };
     History: <
       <| occupy: INTEGER,
           vacate: INTEGER
       >|
     >
    |
}
```

In NF2 the queries become:

- query 1

```
SELECT <| town: x.town,
         Occupants: { y.firstname }
```
TQL – QUERY LANGUAGE

FROM x IN Houses,
y IN x.Occupants
WHERE y.lastname = 'Smith'

• query 2

SELECT <| person: y.person,
   address: { <| x.street, x.town,
x.pcode |> }
|>
FROM x IN Houses,
y IN x.Occupants

Amongst possible further extensions to NF2 described by Pistor and Traunmueller (1986) were the provision of a reference data type. These were included in the query language SQL/W (Larson, 1988). SQL/W was designed by Per-Å Larson for LauRel, a prototype database system being developed at the University of Waterloo. TQL includes the concept of references and implicit joins from SQL/W (Larson, 1988) and also the concept of a structured tuple type.

3. TQL

Like many other post-relational query languages TQL has an SQL flavour but is not a strict extension of SQL. Although TQL closely follows the syntax of SQL/NF a major difference between TQL and the other nested relational query languages is the support for applications using free text retrieval. In TQL text is treated as an atomic type and a number of operations on that type are supported.

To illustrate some of the design features for TQL, we will extend our fictitious database about households, the people who live in them, and their family relationships. Consider the following TQL schema which defines three tables, Households, People and Parents with the obvious interpretations. An instance of this schema is shown in the appendix. This schema has been designed to illustrate TQL features and is not proposed as the best design for the information represented.

Consider the Households table which contains entries consisting of the household address and details of the occupants. Information about addresses is stored in a structured attribute, Address, which is a tuple consisting of three components, namely a street, town and pcode. The ability to support tuples in this way means that addresses can be treated as atomic objects. The other structured attribute type supported by TQL is the nested table and in the example the information about the household occupants is stored in a nested table, Occupants. Thus a household consists of zero or more occupants and all information about the occupants can be viewed directly as being associated with the corresponding household. The Occupants table contains two attributes, person and History where History is a nested table containing the dates that each person occupied and subsequently vacated the household — the atomic attributes occupy and vacate. Note that a person may re-occupy the household on a number of occasions. The attribute person is a reference attribute and provides a reference to the full details of each person in the table People. A reference attribute has a domain consisting of the (possibly composite) key values of the referenced table — thus person has a domain of the pid values in the People table. For efficient processing of user queries, the actual address of the referenced tuple may be stored with the key value for a reference attribute but this is an implementation issue and would be done transparently to the TQL user.

The table People contains information about the first, middle and last names, sex, year and city of birth of each person. The table Parents contains three references to the People table connecting a person with their mother and father.

3.1 Select-from-where

Like SQL, the basic construct of a query in TQL is a SELECT-FROM-WHERE (SFW) expression. An SFW (select-from-where) expression specifies a selection constraint on a subexpression (which restricts which tuples to return) and a projection list of attributes, or expressions based on attributes, to project from each tuple retrieved. An SFW expression can also be used to define joins. Note that the list of expressions for projection can return atomic values or tuples or nested tables.

Query 3. List the names of each person both in Castlemaine.

SELECT name
FROM People
WHERE birthplace = 'Castlemaine';
TQL also supports a functional notation for the representation of SFW queries where the table on the from line is specified first followed by an optional select line in square brackets. The functional notation has been advocated in a number of query languages based on object oriented data models (Hull and King, 1987). Using the functional notation the query 3 could be expressed as:

```
People[name] WHERE birthplace = 'Castlemaine';
```

Unpacking attributes which are tuples is straightforward, the following query unpacks the tuple structure name. Unnesting/unfolding attributes which are tables is more complex and is dealt with in section 3.3.4.

**Query 5. List all the names of people whose last name is Smith.**

```
SELECT name.*
FROM People
WHERE lastname = 'Smith';
```

or simply

```
SELECT firstname, Middle, lastname
FROM People
WHERE lastname = 'Smith';
```

<table>
<thead>
<tr>
<th>People</th>
</tr>
</thead>
<tbody>
<tr>
<td>(Fiona, Frances, Anderson)</td>
</tr>
<tr>
<td>(Llewelyn, Lloyd, Jones)</td>
</tr>
</tbody>
</table>

**Query 3**

It is sometimes necessary to define an alias for a table, tuple or attribute identifier. Examples of where this is necessary include the requirement to disambiguate references to variables (when, for example a table is joined to itself) and the requirement to name attributes or tables in those cases where no identifier exists. An alias only lasts for the current query. New tuples attributes can be constructed in the SELECT clause by using ( and ).

**Query 4. List the first and last name in one tuple structure and the place and year of birth of another tuple structure.**

```
SELECT name(firstname AS first, lastname AS last),
(birthplace AS place, birthyear AS year) AS birth
FROM People;
```

<table>
<thead>
<tr>
<th>People</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alexander, Alfred, Smith</td>
</tr>
<tr>
<td>Belinda, Andrew</td>
</tr>
<tr>
<td>Catherine, Christine, Smith</td>
</tr>
<tr>
<td>Daniel, David, Smith</td>
</tr>
<tr>
<td>Edward, Jones</td>
</tr>
<tr>
<td>Fiona, Anderson</td>
</tr>
<tr>
<td>Greta, Brown</td>
</tr>
<tr>
<td>Harold, Jones</td>
</tr>
<tr>
<td>Isabella, Jones</td>
</tr>
<tr>
<td>James, MacDonald</td>
</tr>
<tr>
<td>Kerry, MacDonald</td>
</tr>
<tr>
<td>Llewelyn, Jones</td>
</tr>
</tbody>
</table>

```
name(first, last) birth(place, year)
Alexander Smith
Belinda Jones
Catherine Smith
Daniel Smith
Edward Jones
Fiona Anderson
Greta Brown
Harold Jones
Isabella Jones
James MacDonald
Kerry MacDonald
Llewelyn Jones
```

**Query 4**

In order to support nested tables, TQL has a recursively defined syntax so that SFW expressions can contain nested SFW expressions. As such, it allows an expression that returns a table to be used within a query at any place that a table name can be used. This property of TQL, referred to as orthogonality, allows for the support of an incremental approach to the formulation of complex queries and also simplifies query formulation in many cases. In TQL it is possible to use a query expression in a SELECT or FROM clause as well as in a WHERE clause. Other examples are provided in Roth et al. (1987) to illustrate the importance of supporting an orthogonal syntax and the advantages of a more recursively defined SQL.

3.1.1 Explicit and implicit joins

The recursive syntax allows a table to be joined at any level within a hierarchical structure. The following query joins the nested table Occupants with the outer-level table People.

**Query 6. For each household in Port Melbourne list the...**
address with the names of anyone who has occupied that household.

```
SELECT address, (SELECT People.firstname, People.lastname
FROM Occupants, People
WHERE Occupants.pid = People.pid)
FROM Households
WHERE town = 'Port Melbourne';
```

<table>
<thead>
<tr>
<th>Households</th>
<th>Occupants</th>
</tr>
</thead>
<tbody>
<tr>
<td>(1/11 Bay Road, Port Melbourne, 3207)</td>
<td>Alexander Smith</td>
</tr>
<tr>
<td></td>
<td>Belinda Jones</td>
</tr>
<tr>
<td></td>
<td>Catherine Smith</td>
</tr>
</tbody>
</table>

Query 6

Reference attributes allow implicit joins between tuples in a table containing the reference and tuples in the referenced table. Each of the attributes in the referenced tuple is directly available (through the implicit join) when querying on the tuple containing the reference. Using an implicit join query 6 becomes

```
SELECT address, (SELECT person.firstname, person.lastname FROM Occupants)
FROM Households
WHERE town = 'Port Melbourne';
```

If there is no name clash it is not necessary to specify the reference attribute name. Also it is possible to combine the functional and SFW notations by using `Occupants[firstname, lastname]` as the functional notation `SELECT firstname, lastname FROM Occupants`. Thus the query could be written as:

```
SELECT address, Occupants[firstname, lastname]
FROM Households
WHERE town = 'Port Melbourne';
```

3.1.2 Inner and outer joins

Consider the following query which contains one explicit and two implicit joins between People and Parents.

**Query 7. List all people and their parents.**

```
SELECT People.firstname, mother.firstname AS mum, father.firstname AS dad
FROM People, Parents
WHERE People.pid = Parents.person.pid
PRESERVE People;
```

Names of people with no corresponding parents tuples will be included in the join and padded with null values.

<table>
<thead>
<tr>
<th>firstname</th>
<th>mum</th>
<th>dad</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alexander</td>
<td>NULL</td>
<td>NULL</td>
</tr>
<tr>
<td>Belinda</td>
<td>Fiona</td>
<td>Edward</td>
</tr>
<tr>
<td>Catherine</td>
<td>Belinda</td>
<td>Alexander</td>
</tr>
<tr>
<td>Daniel</td>
<td>Belinda</td>
<td>Alexander</td>
</tr>
<tr>
<td>Edward</td>
<td>NULL</td>
<td>Llewelyn</td>
</tr>
<tr>
<td>Fiona</td>
<td>NULL</td>
<td>NULL</td>
</tr>
<tr>
<td>Greta</td>
<td>NULL</td>
<td>NULL</td>
</tr>
<tr>
<td>Harold</td>
<td>Fiona</td>
<td>Edward</td>
</tr>
<tr>
<td>Isabella</td>
<td>Greta</td>
<td>Edward</td>
</tr>
<tr>
<td>James</td>
<td>NULL</td>
<td>NULL</td>
</tr>
<tr>
<td>Kerry</td>
<td>Isabella</td>
<td>James</td>
</tr>
<tr>
<td>Llewelyn</td>
<td>NULL</td>
<td>NULL</td>
</tr>
</tbody>
</table>

Query 8

3.2 Support for text attributes

Since Titan+ is designed for applications supporting free text, a number of text operators are supported by TQL. These operators support the searching of free text on a word by word basis and the transformation of text to soundex, stemmed and case converted forms. In TQL text is treated as an atomic type.

There are a number of ways a word in a query can be compared to a word of text. For example, it is often useful to compare the stems of words so that a query containing...
'computer' would match a document containing 'computing' or 'computers'. Another form of comparison is based on the use of the soundex algorithm and results in matches if a query term sounds the same as a document term. Exact matching and case insensitive matching are also supported. For free text queries, the default type of comparison used is that which is defined in the database schema — described in section 3.5. Typically, stemming is used for free text, while soundex is often required for names of people or places. Although the default form of comparison is defined in the scheme, it is possible to specify a query in which the default is overridden. For example stemming algorithms sometimes result in unrelated words reducing to the same stem. It may therefore be necessary to specify exact matching in some free text or string expressions. To achieve this, TQL provides single prefix characters (transformation marks) to select the method of comparison. The possible transformations are:

- exact match (!),
- ignore case (~),
- stemming ('), and
- soundex (@).

The absence of such a prefix transformation indicates that the default comparison is to be used. If there is no default comparison "exact match" comparisons are done. If the two text operands have different types of comparison defined in the schema and there is no prefix defined on the comparison operator then an error is reported.

It is also possible to specify indexing using these transformations, this is described in section 3.5.2.

Any of the transformations can be applied to the equality and inequality operators. Thus ! = means the two text operands match exactly (that is they are identical), = means the two text operands are equal if case is ignored, ~ = means the stems of the two text operands are the same, and = means the two text operands sound the same. Similarly for inequality operators ! <>, <= >, ~<> and <>.

**Query 9.** Find all people whose surname sounds like mcdonald.

```
SELECT name,
FROM People
WHERE lastname @ = 'mcdonald'
```

```
People
name(firstname,Middle[name],lastname)
(James, Jonathan, MacDonald)
(Kerry, [ ], MacDonald)
```

**Query 10.** Find all people whose firstname ends with the letter 'a'.

```
SELECT name
FROM People
WHERE name.firstname LIKE '%a';
```

```
People
name(firstname,Middle[name],lastname)
(Belinda, [ ], Jones)
(Fiona, Frances, Anderson)
(Greta, [ ], Brown)
(Isabella, [ ], Jones)
```

**Query 11.** Find all addresses in Bay Road or High Street.

```
SELECT address
FROM Households
WHERE street CONTAINS ('Bay' 'Road') | ('High' 'Street')
```

```
Households
address(street, town, pcode)
(1/11 Bay Road, Port Melbourne, 3207)
(2 High Street, South Melbourne, 3205)
```
Given two subexpressions of type text, + concatenates the two expressions. Thus 'computer' + ' ' + 'science' is equal to 'computer science'.

3.3 Table and tuple operators
The equality and inequality operators (= and <>) can be used on tables and tuples as well as atomic values. However relative operators (<?, <=, >, >=) only apply to atomic attributes. Two tuples are equal if they are type compatible and the pairs of values for each attribute in the tuples are equal. Two tables are equal if they are type compatible and each table has the same tuples in the same order.

3.3.1 Quantifiers and set comparison
EXISTS returns true if the table subexpression contains at least one tuple, false otherwise. There are two other forms which are automatically mapped to EXISTS expressions. ANY expr WHERE cond returns true if any tuple in the table subexpression matches the WHERE condition and ALL expr WHERE cond returns true if all the tuples in the table subexpression matches the condition.

SUBSET and SUPERSET take two tables of identical structure and return true if all the tuples in one table exist in the other.

The IN operator returns true if the left atomic or tuple subexpression is in the set of values returned by the right table subexpression, for example:

Query 12. Find all people who are parents.
SELECT name
FROM People
WHERE pid IN (Parents[mother.pid]) OR pid IN (Parents[father.pid])

The HAS operator is identical to IN except that the operands are reversed and different types of comparison (including text transformations) can be used.

3.3.2 Aggregate functions
In TQL, the argument to an aggregate function such as SUM, AVG, MAX, or MIN is a SFW-expression that returns a vector (a table with one attribute) of numeric values. The function COUNT takes as an argument any table. This differs from SQL/NF which also allows table arguments to SUM. Like SQL/NF, TQL was designed to incorporate an orthogonal implementation of functions. SQL, on the other hand, is weak in this respect (Date, 1984b). As Date has observed (Date, 1984a), in SQL, the argument to a function is specified in a most unorthodox manner and that, as a consequence, function references can only appear in a small set of special-case situations.

3.3.3 Finding a row in a table
ROWNUM returns the tuple number of the current tuple. The current tuple is the tuple retrieved on the FROM line of a SFW expression. Rownum's for nested tables are sequenced 1, 2, 3 and so on. For outer level tables however, rownum is the actual record number on the file. Thus if a record is deleted, the rownums will no longer be contiguous.

Query 13. Select the first listed occupant for each household.
SELECT address, (Occupants[1][firstname, lastname])
FROM Households;

ROWNUM can not be applied to all table expressions but only outer level tables (GLOBAL, LOCAL or TEMP) and nested tables but not tables generated as intermediate results in a query. If the two does not exist ROWNUM returns NULL. As an abbreviation for WHERE ROWNUM = number a particular row of a table can be selected by adding {number} after the table name. Thus query 13 could be expressed

SELECT address, (Occupants{1}[firstname, lastname])
FROM Households;

3.3.4 Operations on tables

A number of the relational algebra operators (UNION, INTERSECT, MINUS, TIMES and JOIN) are provided as abbreviations for SFW expressions. In addition TQL provides three special operators (UNNEST, UNFOLD and GROUP) to restructure tables.

The UNNEST operator is used to flatten the result of a table subexpression which contains a nested table. For each tuple in the nested table being unnested on, a copy of all the other attributes is made. If the nested table does not contain any tuples, then a single tuple is returned with the nested table attributes being given null values. UNFOLD is similar to UNNEST but when an empty subtable is unfolded the corresponding tuple(s) are removed. The following two queries illustrate the difference between the UNNEST and UNFOLD operators.

Query 14. Produce a 1NF table of addresses and residents.

```
SELECT address, person
FROM (UNNEST Households ON Occupants);
```

<table>
<thead>
<tr>
<th>Households</th>
<th>person(pid)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(1/11 Bay Road, Port Melbourne, 3207)</td>
<td>(1)</td>
</tr>
<tr>
<td>(1/11 Bay Road, Port Melbourne, 3207)</td>
<td>(2)</td>
</tr>
<tr>
<td>(1/11 Bay Road, Port Melbourne, 3207)</td>
<td>(3)</td>
</tr>
<tr>
<td>(2 High Street, South Melbourne, 3205)</td>
<td>(1)</td>
</tr>
<tr>
<td>(2 High Street, South Melbourne, 3205)</td>
<td>(2)</td>
</tr>
<tr>
<td>(2 High Street, South Melbourne, 3205)</td>
<td>(3)</td>
</tr>
<tr>
<td>(2 High Street, South Melbourne, 3205)</td>
<td>(4)</td>
</tr>
<tr>
<td>(3 Main Street, Ballarat, 3350)</td>
<td>(5)</td>
</tr>
<tr>
<td>(3 Main Street, Ballarat, 3350)</td>
<td>(6)</td>
</tr>
<tr>
<td>(3 Main Street, Ballarat, 3350)</td>
<td>(7)</td>
</tr>
<tr>
<td>(3 Main Street, Ballarat, 3350)</td>
<td>(8)</td>
</tr>
<tr>
<td>(3 Main Street, Ballarat, 3350)</td>
<td>(9)</td>
</tr>
</tbody>
</table>

Query 15. Produce a 1NF table of addresses and residents, without retaining any addresses which have no residents.

```
SELECT address, person
FROM (UNFOLD Households ON Occupants);
```

Wherever a attribute within a nested table is referred to on the SELECT using the dot notation the nested table is automatically unfolded. Thus query 15 could also be expressed using automatic unfolding as:

```
SELECT address, Occupants.person
FROM Households;
```

The GROUP operator builds a nested table by grouping tuples with common values in unnested attributes together to form a single new tuple. For example, grouping the table Parents[person(pid), mother(pid), father(pid)] on the attributes mother and father would result in a table of the form Parents[Child[person(pid)], mother(pid), father(pid)]. This new table would have all the children for each pair of parents in a nested table Child.

Query 16. Convert the 1NF table Parents to a non first normal form relation grouping on both parents.

```
GROUP Parents ON mother, father FORMING Child
```

<table>
<thead>
<tr>
<th>Parents</th>
<th>Child</th>
</tr>
</thead>
<tbody>
<tr>
<td>mother(pid)</td>
<td>father(pid)</td>
</tr>
<tr>
<td>(NULL)</td>
<td>(12)</td>
</tr>
<tr>
<td>(2)</td>
<td>(1)</td>
</tr>
<tr>
<td>(3)</td>
<td>(6)</td>
</tr>
<tr>
<td>(6)</td>
<td>(5)</td>
</tr>
<tr>
<td>(7)</td>
<td>(5)</td>
</tr>
<tr>
<td>(9)</td>
<td>(10)</td>
</tr>
</tbody>
</table>

Like SQL, duplicates are not removed in an ordinary TQL query. The function DISTINCT takes a table as an argument and removes duplicates. Two tuples are considered to be duplicates if all atomic values are equal and any nested tables contain the same set of tuples where the ordering of tuples is significant. The table returned has the same structure as the subexpression but may contain less tuples. The ordering of tuples in the returned table is not guaranteed. Another way to express query 16 which does not require the GROUP operator is as follows:

```
DISTINCT (SELECT (SELECT person
FROM Parents AS Child
WHERE Parents.mother = Child.mother)
FROM Households)
```
AND Parents.father  = Child.father),
mother,
father
FROM Parents)

3.4 TQL data manipulation language
The DML command INSERT is used to add new tuples to a table or when used with UPDATE can add new tuples to a nested table. There are two forms of the INSERT command — INSERT VALUES and INSERT :=. The first form inserts tuples. The second form wipes the table first before inserting the new tuples. That is, it replaces the contents of the table with a new set of tuples.

Query 17. To add data to the People table, the INSERT command is used.

INSERT People VALUES
[ (1, m, ('Alexander', [('Alfred'), ('Andrew')], 'Smith'), 1955, 'Melbourne') ];

The result of any query can be used as data for insertion, not just constant tuples. Also a list of attributes names can be specified allowing the attributes names to be ordered as desired within the INSERT command. Any attributes that are not included are assigned the value NULL.

DELETE removes matching tuples from a table (or nested table). A WHERE clause is used to restrict which tuples to delete. A DELETE command can be used on nested tables by nesting the command in an UPDATE.

The UPDATE command is used to update tuples in a table. Update commands can be used to assign a new value to an atomic field, or modify nested tables using a nested DML command. Thus an update command can contain nested INSERT, DELETE or UPDATE commands. The set clauses are evaluated in the order of appearance.

Query 18. Change the year of birth for Llewelyn Jones to 1895.

UPDATE People
SET birthyear = 1895
WHERE name.firstname = 'Llewelyn' AND name.lastname = 'Jones';

3.5 TQL data definition language
TQL supports three types of outer level tables — global, local and temporary. Global tables reside in a shared database and are accessible by all users who connect to that database. All shared tables must be global and only global tables can be indexed.

Local tables are local to each user. One user cannot access a local table belonging to another user. Local tables do not support indexing and are intended for local storage space. Local tables are independent of the database that the user connects to and so can be used to transfer data between databases. Local tables are also useful for applications for which it is required to keep user dependent parameters in a database table.

Temporary tables are local to a session and are deleted at the end of a session. They are intended for temporary scratch space.

In order to create an outer level table using TQL, it is necessary to modify the current schema definition. As in SQL, TQL contains commands to both create and drop tables. The table creation command can provide formatting information for the new table. A separate command is used to create an index for a table.

TQL also supports query-only views which provide a powerful mechanism for the support of high level user abstractions of the data. Views are also a very useful mechanism for complex query formulation. Once created a view can then be treated as a normal table in queries.

3.5.1 Creating and deleting tables
The table creation command to create the People table including defaults for formatting and text transformations:

CREATE GLOBAL TABLE People [
  pid INTEGER,
  sex 'TEXT FORMAT 'width=1',
  name ( firstname @TEXT FORMAT 'width=10',
        Middle[name @TEXT FORMAT 'width=10'],
        lastname @TEXT FORMAT 'width=10' ),
  birthyear INTEGER,
  birthplace @TEXT FORMAT 'width=10'
] KEY = (pid);

Prefix characters are used to define the default type of equality matching for text fields. In the absence of prefix characters, exact matching is used as the default.

For all fields, it is also possible to specify a format for the output of queries in TQL. This is an important feature for displaying text fields on a terminal without the aid of a forms package. Formats are used by the TQL command interface to format the output of a query. TQL supports variable length fields of infinite length. When displaying the result of a query however, a column width is needed to output the result of the query in a readable form. Columns can be assigned default formats when the table is defined by adding a FORMAT clause to one or more of attribute defined in the CREATE command. Alternatively the FORMAT statement can be used in a query to override the default format for the result of an expression.

Tables can be deleted using the DROP command, for
example to remove the global table People from the current database:

DROP GLOBAL TABLE People;

3.5.2 Index creation
To create an index on a global table, the CREATE INDEX command must be used. Titan+ uses superimposed coding schemes (Sacks-Davis et al., 1987) to index the contents of tables and these schemes require a number of parameters to be specified. To make life easier for users, the ANALYSE INDEX command analyses a table of data and the index currently defined and automatically selects appropriate parameters.

Consider the following command

CREATE INDEX People [  
  !pid,  
  @name.firstname,  
  'name.firstname,  
  @name.lastname,  
  'name.lastname,  
  !birthyear,  
  @birthplace  
 ];

This command will cause the fields pid, firstname, lastname, birthyear and birthplace to be indexed; the fields sex, Middle.name are not indexed; and two indexes are constructed for both the firstname and lastname fields. Each attribute name is prefixed by a transformation character. This effects the actual value used for indexing. For example, !pid causes the exact value of the people number to be indexed on. By specifying both firstname and !birthyear, both the firstname mapped to upper case and the soundex version of the firstname will be indexed.

Once the index has been created, data can be loaded into the table. The ANALYSE INDEX command can then be used to design the index parameters. The table must contain some data before the ANALYSE INDEX command can be used as it samples the data in the table to determine information such as the average number of index terms per record.

ANALYSE INDEX People;

By supporting nested relations and efficient indexing methods based on superimposed coding TQL and Titan+ are able to provide support for document databases. Documents can be treated as single objects by the system and efficient access of these objects is supported by the powerful access methods.

4. CONCLUSION
We have described a language TQL for nested relations. The language provides support for non-first normal form relations including text attributes, tuples, nested tables and implicit joins using references. Compared with other languages proposed for the nested relational model, TQL has powerful support for text. We are currently investigating how TQL can support document applications such as hypertext databases. A node can be stored as a complex object using TQL with hypertext links supported by reference attributes. We are also investigating problems of query optimisation for the nested relational model. Particular issues are the complexity of join operators for nested tables and the efficient support of operators such as nest and unnest.

REFERENCES


APPENDIX

<table>
<thead>
<tr>
<th>Households</th>
<th>Occupants</th>
<th>History</th>
<th>Person(pid)</th>
<th>Address(street, town, pcode)</th>
<th>History</th>
<th>Person(pid)</th>
<th>Address(street, town, pcode)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>occupy</td>
<td></td>
<td>(1) 1978 1962</td>
<td></td>
<td></td>
<td>(1) 1978 1962</td>
</tr>
<tr>
<td></td>
<td></td>
<td>vacate</td>
<td></td>
<td>(2) 1978 1962</td>
<td></td>
<td></td>
<td>(2) 1978 1962</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>(4) 1983 NULL</td>
<td></td>
<td></td>
<td>(4) 1983 NULL</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>(5) 1948 NULL</td>
<td></td>
<td></td>
<td>(5) 1948 NULL</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>(6) 1948 1962</td>
<td></td>
<td></td>
<td>(6) 1948 1962</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>(7) 1966 1985</td>
<td></td>
<td></td>
<td>(7) 1966 1985</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>(8) 1988 NULL</td>
<td></td>
<td></td>
<td>(8) 1988 NULL</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>(9) 1988 1966</td>
<td></td>
<td></td>
<td>(9) 1988 1966</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Parents</th>
<th>Person(pid)</th>
<th>Mother(pid)</th>
<th>Father(pid)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(1)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(2)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(3)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(4)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(5)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(6)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(7)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(8)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(9)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(10)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(11)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(12)</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

BIографICAL NOTES

James Thom has been with the Department of Computer Science at RMIT since 1987 after working for 18 months with the Machine Intelligence Project at the University of Melbourne. His current position is a senior lecturer.

Alan Kent was the RMIT Computer Science department’s first PhD graduate, his thesis was completed in 1988. Since then he has been working as a post-doctoral fellow in the Key Centre for Teaching and Research in Knowledge Based Systems.

Ron Sacks-Davis is currently the research director of the Collaborative Information Technology Research Institute. He is also the director of the Intelligent Databases program, Commonwealth Research Centre for Intelligent Decision Systems and co-director of the Key Centre for Teaching and Research in Knowledge Based Systems.

THE AUSTRALIAN COMPUTER JOURNAL, VOL. 23, NO. 2, MAY 1991 65
Action programmes for teaching and researching in information systems

DE Avison
Department of Accounting and Management Science
University of Southampton, Southampton, SO9 5NH, UK

Information systems is a practical discipline and teaching and research should be relevant to the world of the information systems practitioner. Action programmes aim to reflect this pragmatic view of the discipline of information systems: action learning in our teaching and action research in our research. This paper discusses the advantages and problems of these programmes and in each case describes one project in detail.

Keywords and Phrases: information systems, action learning, action research, case study.

CR Categories: H.O. H. 1

1. THE NEED FOR ACTION PROGRAMMES
There is a growing realisation that information systems is a practical subject and teaching and research ought to be relevant to the industrial and commercial worlds of the public and private sector. Students with a degree in information systems ought to have experienced in their course 'real world' problems and, before graduating, worked on projects of the kind carried out at their potential future employers. Similarly, some research in information systems ought to be carried out along with practitioners in the public and private sectors with the researchers and practitioners acting together to move the frontiers of information systems knowledge. It is not the argument of this paper that conventional learning and researching methods are not valid, but that information systems departments at universities ought also to be carrying out action programmes. Nor is it claimed that action programmes are new — they are a feature of teaching in many disciplines at some universities (for example in education), but they are not yet known to all in the information systems community, nor are they accepted by all who are aware of the approach. It is the purpose of this paper to bring action learning and action research to a wider audience and discuss some of the strengths and weaknesses of the approach. This paper describes two action programmes undertaken by the information systems group at Aston University in the UK and compares the programmes with those at universities elsewhere. It also describes typical action learning and action research projects.

2. ACTION LEARNING
2.1 Teaching Methods in Information Systems
The conventional way of teaching information systems in universities is to give a series of lectures backed by a textbook, along with selected readings taken from journals. This is supplemented by tutorial sessions which give students the opportunity to tackle problems. Sometimes this is backed up by case studies which are more substantial pieces of work, very often stemming from real-life examples, ideally gained from the tutor's own experience, but very often taken from a text. Assessment is largely made by a formal examination.

2.2 Problems with Conventional Methods
As Buckingham, Hirschheim, Land and Tully (1987) and Nunamaker, Cougar and Davis (1982) make clear, the conventional methods of teaching described in section 2.1 do not provide a good enough grounding for information systems people ready to enter the commercial world, because of a lack of real-world experience in such a programme. Examinations test what can be memorised and reproduced from lectures and texts, and it is difficult to devise questions which show true understanding of the subject matter. The problems given in examinations have to be simple because of the time factor. Problems suitable
for tutorial work need to be fairly mechanical, such as the drawing of entity-relationship diagrams and data flow diagrams from narrative descriptions, or normalising a set of relations.

The traditional way of providing ‘real’ experience, the case study, is by no means ideal, indeed Hammersley (1981) provocatively titles his paper on the subject “The Case Study is Dead”. Although they usually stem from real-life, case studies are simplified, and very often simplified in a way that greatly reduces their usefulness, because what is omitted are often those difficulties that real-world situations have. In particular, the complexities of the real-world, often related to the links between various aspects of the problem situation or complexities due to ‘people problems’, are frequently omitted to keep the case study ‘within bounds’.

Such simplifications often lead to students having a view that there is one ‘solution’ to a case. Indeed, some books of case studies provide lecturers’ aids which suggest ‘model solutions’. Of course there can be many good design alternatives (as well as thousands of bad ones). Much will depend on people factors, not technological ones, but these people factors are rarely explored in case studies.

The case study still has a place, the traditional ways of teaching information systems will be appropriate to the large classes of students who are interested in the subject in the early years of a degree course. But information systems specialists need real experience before they graduate. Action learning enables students who have already completed courses in information systems to practice using some of the methodologies, techniques and tools learnt in a formal course to a real-world situation.

2.3 Action Learning Principles
Action learning usually takes place in outside organisations. The programmes of the University of East Anglia in the UK, and the University of New South Wales and Griffith University in Australia, described in Wood-Harper and Flinn (1983), Lawrence and Brooks (1980), and Little and Margetson (1989) respectively, are carried out in industry and commerce. Indeed, the first attempts at Aston under this programme concerned a group project carried out in industry. An example can be found in the appendix of Avison (1985). Such an approach is also recommended in Buckingham et al. (1987). This would seem to be the ideal, and was successful for some Aston projects, but not all.

There are a number of problems associated with action learning in industry, for example:

1. It is difficult to contain projects in the time available as university timetables tend to allocate courses in three month, six month or one year slots starting at particular times of the year. This requirement may conflict with those of the outside organisation. Timing difficulties may also make it difficult to provide continuity between years.

2. It is difficult to have control over the projects in the outside organisation, partly because of distance between it and the university, and partly because of changes in personnel or policy that may occur there.

3. It is also difficult to provide all the desirable learning experiences in one commercial project. Indeed there may be potential clashes between the requirements of organisation with those of the university.

At Aston University, we have modified the approach so that the work takes place at the university itself. In an in-house application, the work can be as ‘real-world’ as that in industry and the ‘learning experience’ is at least as important as the ‘product’. With students working in groups, it is possible for the groups to adopt different methodologies, techniques and tools, and to discuss their experiences, which enhances the total learning experience, as each group presents its work to the whole class.

2.4 An Action Learning Project
Work carried out as an action learning project by students at Aston University is described in this section. The students are either final year undergraduates or masters students, and in both cases they have had courses in systems analysis and information systems methodologies. Further details of the scheme and the application can be found in Avison (1989 and 1990a respectively).

The in-house application concerns the development of an integrated information system for an academic department. The full information system aims to fulfil the information requirements of staff and students of the department and will take some years to develop. Groups have tackled a number of subsystems such as those for admissions, courses, timetabling, industrial placement, examinations, publications, equipment, accounting, careers, consultancy and conferences. Already there are prototype versions of these subsystems which produce standard regular reports, provide statistical analyses and handle ad-hoc queries.

An action learning programme also exists at the Department of Information Systems, University of New South Wales. This programme has one major difference with the Aston approach in that the groups of students carry out the same project (it is similar to that described in Wood-Harper and Flinn (1983) in this respect). This is easier to set up and control, but does have weaknesses for the overall learning experience. First, the groups do not gain the benefits of intra-group work. In particular the various subsystems in the Aston programme need to integrate — one subsystems’ output is another’s input. Problems in subsystems’ integration is often a weakness of commercial systems and it is important to make students aware of this requirement. Second, at the University of New South Wales the groups tend to work in competition.
Cooperation and communication between project teams in industry is vital to help in developing successful information systems.

Action learning programmes at Aston University require the cooperation of staff (both academic and non-academic) and the general body of students. They have been interviewed, completed questionnaires, attended formal presentations, used the prototypes and commented on their design and use. Students on the programme did not know most of these members of staff well, and the relationship between staff and the students on the programme was, on the whole, similar to that of clients and analysts. One of my colleagues even 'played' an unhelpful role, and it was useful for students to experience and react to this situation. Hopefully the reward to academic staff, for example, will be an information system that will help them cope with increasing administrative loads so that they can spend more time in their own teaching and research, and provide improved information to students.

2.5 Problems with Action Learning

There are a number of problems with action learning projects which include the difficulty in ensuring that the subsystems developed by students can be integrated and also that there are overall standards agreed, including those for validation, security and control. For this reason, the project is time-consuming for the lecturers concerned. Nevertheless, we see our role as one of guidance and being available if any difficulties are experienced. Recently, the department in question appointed a computer officer who is responsible for standards (assuming the role of the database administrator) and the implementation of operational systems.

From the students' point of view, the course takes a greater than average amount of time, but this is partly of their own doing. They enjoy the work, see the value of it and therefore are willing to put considerable effort into the success of their prototype. They gain experience in interviewing, group and intra-group work, project control, presenting seminars, training users, report writing, learning about the roles of the people in the application area, and using and evaluating contrasting methodologies, techniques and tools. The work is not a programming project, though most groups develop a prototype using a fourth generation system.

A major problem concerns assessment, there being no formal examination for the course. A group mark is given by two assessors on the basis of the report and the formal presentation to staff and other students. The handling of questions is also considered in this assessment. Tutors meet groups regularly and have a fair idea of the contributions of the individual group members. But each group member also provides an assessment of his or her own contribution and that of the other group members. I do worry that these marks could be distorted by elements such as racism or sexism, and though I am not aware of this having happened, this does not mean that it has not happened.

2.6 Further Observations with Action Learning

Although some groups tackle new applications in the area, others take over previous work. This can also be a valuable experience as students learn from enhancing the work of others. For example, they learn why the various choices were made, whether, in retrospect, they were good choices, and how to assess the standard of work. Such document examination provided. I have been made aware of the flaws in previous work that were not detected when assessing the reports —the course also provides useful action learning for information systems teachers!

The action learning project does seem to be of interest to prospective employers who seem aware of its significance. Many students have reported that this course was the main topic of conversation in their initial job interviews.

3. ACTION RESEARCH

3.1 Alternative Research Methods

Many of the arguments for action learning also apply to action research. There are many types of research in information systems. Ten methods of research which are relevant in information systems include mathematical modelling, conceptual study and laboratory experiment, at the quantitative and scientific end of the continuum, through to case study research and action research at the qualitative and non-scientific end (Avison, 1990b). In the spirit of the view that 'many flowers should bloom' (Klein and Lyytinen, 1985), it is not the purpose of this paper to attack any one approach. Each research method has its place, and Jenkins (1985) discusses criteria that might be used when assessing which one to use for a particular situation.

However, action research is less well-known and respected in universities when compared to scientific and quantitative research, though it can be of great value to the information systems community.

Research in information systems is frequently carried out at the university by researchers who may be developing or tuning tools (such as CASE tools) to support information systems, proposing variations on techniques (such as the entity-relationship model), or establishing a theory of information systems. Others may be statistically analysing data obtained from field work or survey material (such as the impact of fourth generation tools on systems development). This is valid and useful research to the information systems community, but action research can also be as useful.

3.2 Action Research Method

Action research, described fully in Checkland and Scholes (1990), is somewhat different. In action research,
researchers test and refine principles, tools, techniques and methodologies that may have been part developed at the university to address real-world problems. Further, it is characteristic of action research that researchers may come with a framework to the problem situation which will be adapted in response to the needs of the application. Another characteristic of action research is that the practitioners as well as the researchers participate in the analysis, design and implementation processes and contribute at least as much as the researchers in any decision-making. Thus there is a synergy between the researchers and practitioners, the researchers building up theories and modifying them on the basis of practical experience and the practitioners using and modifying research ideas for solving real-world problems.

This can have the effect of overcoming the real problem of persuading users to adopt new techniques, and in general, help overcome the divide or culture clash that frequently exists between two groups of information systems people: academics and practitioners, and should therefore ensure that managers get real benefits from the programme that they are, after all, paying for.

Action research is often confused with case study research, but, as Benbasat, Goldstein and Mead (1987) have shown, whereas case study research examines phenomena in its natural setting with the researcher an independent outsider (and, it is argued, 'objective'), in action research the researcher is a participant in the implementation of a system and simultaneously evaluates a particular approach. An action researcher would argue that the researcher ought to be 'useful'.

### 3.3 Advantages of Action Research

Action research gives researchers an opportunity to test in a natural setting the ideas, principles, tools and techniques, which have been developed in the conventional research arena. It is at this time that such research makes a contribution to society.

### 3.4 Problems with Action Research

Action research does have critics, however. The results are qualitative rather than quantitative. There is no such thing as a repeatable experiment in this context, all organisations and the people in them being different. Further, the laboratory is open to outside influences; indeed it is a tenet of this approach that human subjects in that real world contribute to the research, the researcher admitting a lack of objectivity.

A particular difficulty that universities have is persuading research funding bodies that this type of research is as valid and as useful as conventional methods of scientific research. The open-endedness of such research and the flexibility necessary in writing a research proposal also provide additional difficulties. Further, although the researcher's intent is to conduct research while effecting change, it has also been accused of being nearer consultancy than research. The lack of scientific discipline in such research makes it difficult for the work to be assessed for the award of research degrees and for publication in academic journals. It is hoped, however, that this paper exposes the many possible gains of action research.

### 3.5 An Action Research Project

This section discusses an action research project carried out at Darlington Health Authority, a district within the UK National Health Service. Some aspects of the work are described further in Avison and Catchpole (1988).

#### The Problem

In an average health district in the UK (there are 220 in total), the normal health care activity of the community (non-hospital) health services creates tens of thousands of health care interactions annually. These relate to a whole range of activities, from the domiciliary midwifery service to the domiciliary care of the terminally ill elderly patient. Current information about these interactions is stored in several discrete manual record systems. There is surprisingly little comprehensively organised information available about them and it is difficult to evaluate how effective the services are. The need for the systematic collection, collation and analysis of information would be a valuable information base for planning and organising health care activities. Such a system would provide feedback to the professionals working in these areas:

- A UK government steering group on health service information (Korner, 1982, 1984a and 1984b) has also recommended a minimum amount of data which should be recorded to provide the statistics required by the Department of Health and Social Security.

An increasing emphasis is being placed on caring for clients in the community situation and the potential of community information systems is large: at Darlington Health Authority the number of beds in the district general hospital is 750; one community service, school nursing, has 20,000 registered clients. Data relating to this work could provide valuable epidemiological and demographic information. The opportunity is presently being lost. The action research project concerns the analysis, design and implementation of a prototype information system for the community health services of Darlington Health District, whilst fulfilling the related statutory requirements of the NHS Korner steering group.

#### Purpose

The purposes of this particular action research project were many. For example, from the researchers' point of view, we wanted to take a further look at Multiview, which is an approach to information systems development, in 'action'; so that we could respond to problems that occurred when using the approach. Indeed, the differences in the descrip-
tion of the Multiview framework found in Avison and Wood-Harper (1990) when compared to Wood-Harper, Antill and Avison (1985) can largely be attributed to experience gained in action research projects such as this. The 1990 text describes six action research projects.

From the users' point of view, the purpose was to develop a prototype information system for community health workers for a local area health authority. From senior management's point of view, it was to fulfil the requirements of government to provide statistics related to community health.

Research Content
We wished to test and refine the principles, tools and techniques developed by the information systems research group at Aston University in a real-world problem situation. As argued in Avison and Wood-Harper (1991), it is unreasonable to rely on a single methodology for all situations. Most importantly, this would tend to preclude the real (rather than lip-service) participation of the users, the methodology being a strait-jacket preventing them from contributing effectively. People ought to have control over their work environment wherever possible, including the computer systems that they use. For this reason the methodology framework chosen should be appropriate for the application area and flexible enough for user groups to have a real influence on decisions made.

Thus the framework to be adopted should lend itself to participation of user groups at all stages of the development of the information system, not only the design of the human-computer interface. A second and related requirement was to enable prototyping (see also Avison and Wilson, 1991). This facilitates the demonstration of aspects of a potential system to be made without any irreversible work being done on the real system.

The framework for the methodology to develop the information system was an adaptation of Multiview (Avison and Wood-Harper, 1990). This framework has been developed and used on a number of action research projects, and adapted by the researchers and practitioners as appropriate for each project.

Method
The Multiview framework was used to set up a prototype information system. Two staff groups, chosen by the design group, carried out the pilot study. One nursing group (school nurses who carry out 'group session' type activities) and one para-medical group (chiropodists as they carry out 'face-to-face contract' type activities) were chosen because they provided a contrast to test different parts of the system. The pilot system was evaluated from the point of view of effectiveness, that is the proportion by which the system meets organisational objectives and goals, and efficiency, that is a measure of the 'mechanical' aspects of the system such as the accuracy, timeliness and speed of access to information. We also wished to review the training and design process, essentially reviewing the participative and prototyping approach to information systems development, and opinions were solicited about the system from the senior managers as well as user groups and individual users. The methods used to assess the system were questionnaires with follow-up interviews and group discussions.

Results
Respondents to the questionnaire considered that the system has helped people to perform their work tasks more effectively, drawing attention to what is actually being done, and has resulted in staff deciding to change their working practices. The system has improved the handling of case loads, including timetabling and utilisation of clinical time. More up-to-date, accurate and easily accessible information will help in both short and long term service planning. Respondents also suggested a number of enhancements which could be added to the system. On the negative side, respondents did point out that, at least in the short term, data collection was slower. On working practices, comments such as "it was satisfying to see what work had been done at the end of the month" and "I have realised what a large amount of time is spent travelling and walking to homes and schools" are revealing.

<table>
<thead>
<tr>
<th>Views on Using the New System</th>
<th>-2</th>
<th>-1</th>
<th>0</th>
<th>+1</th>
<th>+2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Difficult to learn</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>X</td>
<td>Easy to learn</td>
</tr>
<tr>
<td>Difficult to use</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>X</td>
<td>Easy to use</td>
</tr>
<tr>
<td>Troublesome</td>
<td>*</td>
<td>X</td>
<td>*</td>
<td>*</td>
<td>Few problems</td>
</tr>
<tr>
<td>Time consuming</td>
<td>*</td>
<td>X</td>
<td>*</td>
<td>*</td>
<td>Time saving</td>
</tr>
<tr>
<td>Imposing</td>
<td>*</td>
<td>X</td>
<td>*</td>
<td>*</td>
<td>Unimposing</td>
</tr>
</tbody>
</table>

Figure 1. Views of the new system.

Figure 1 represents one of the fourteen tables derived from responses which lend themselves to some sort of statistical analysis, though objectivity is not claimed in the assessment procedures. It looks at the various views of the users concerning the use of the new system. This is the most revealing table because it is critical of the new system. It highlights in particular the time necessary to use the new system. We pursued this aspect further in follow-up interviews, and this was seen by most people as a 'once-and-for-all' investment of time, and, on the whole, it was felt that the adoption of the prototype had many advantages which outweighed the costs.

As for the method of implementation, comments such as: "... as it is the community health professionals who have to use the system, we are in the best place to help design it; we know what information is important and relevant for recording"; "It is good to be involved with
Further Comments

We were fortunate to receive the full cooperation from user groups. These included management grades, such as the director of nursing services for the community, the six assistant directors of nursing services, the senior clinical medical officers, and district heads and non-management grades such as health visitors, school nurses, community midwives, district nurses, and dieticians. These formed design groups which met regularly to decide an overall strategy as well as detailed designs.

Users were also happy to try new tools and techniques such as portable hand-held data recorders as an alternative to the redesigned forms. The portable data recorders, no bigger than hand calculators, enable direct data entry into the computer system (see Catchpole, Avison and Peart, 1987, for a discussion of this trial). This experiment was not imposed on staff—we as facilitators made the staff groups aware of the various ways of collecting data and the staff groups chose to try these in our prototypes.

4. CONCLUSION

Although not underestimating the importance of conventional methods of teaching and research, there is also a need for action learning and action research in information systems. They help both the student and researcher gain an understanding of the problems when developing information systems in the real-world. Although there are problems associated with both action programmes, they provide valuable feedback on the usefulness of methodologies, techniques, and tools taught and developed in universities.

ACKNOWLEDGEMENTS

I am grateful to Marcus O'Connor who leads the action learning programme at the University of New South Wales for the opportunity of participating in that scheme and to Michael Lawrence, Dan Moody and Sandy Grant who commented on a draft of the paper whilst I was Visiting Fellow at UNSW. The paper has also gained from the comments of those attending a research seminar at the University in October 1989. I wish to acknowledge the contribution of Paul Catchpole and Julie Horton, research students at Aston University, who contributed to the action research programme described in the paper and to Darlington Health Authority and the Northern Region Health Authority in the UK who have part financed the project. Many thanks also to the students who have worked on the action learning project at Aston University during my time as Senior Lecturer there.

REFERENCES


24 April 1991

The Editor
Australian Computer Journal

Dear Sir


While the problems addressed by this article, namely shortages of IS personnel and high employee turnover, are themselves real enough, I believe that this publication does little to shed light on either of them for the Australian context. This is simply because this study used an invalid sampling frame, an invalid instrument, and failed to abandon the work when an inadequate response was received.

The survey was distributed to the membership of the ACS, which is simply not appropriate for the nature of the study. While the paper title refers to “information systems professionals”, a general term covering many levels from programmers to chief information officers, the survey form is aimed at programmers. The ACS is not a stronghold of mainstream programmers, the majority of programmers in Australia are not members. The research needed to attempt to define the population under investigation and then make some attempt at sampling it effectively.

The prior work validating this instrument demonstrated its usefulness and effectiveness for US-based programmers. As soon as one moves away from that specific culture and this particular category of employees, the instrument needs to be validated and adjusted or redesigned. Hofstede’s seminal works on cultural effects in the workplace1 demonstrate that we cannot simply view Australia, or any other country for that matter, as extensions of the United States of America. The message is simple, if you want to use an American instrument in Australia, it must be validated.

The third major shortcoming is the failure to stop work when insufficient response was received. The original instrument is designed for a focus on programmers and analyst programmers. As an ACS member receiving this questionnaire I found it meaningless to my present situation, as no doubt it was to many other members. This is a likely cause of the huge non-response rate. Ninety per cent plus of those surveyed did not respond, the usable responses being only 8.2 per cent of the total. This is noticeably below the (scraping the barrel) threshold of ten per cent normally used in surveys. Investigators who continue with their analysis when their response rate is less than 10 per cent can be fairly said to be clutching at straws.

Returning to my original point, the paper addressed important issues, but, by neglecting to adopt appropriate methodological standards, failed to add significantly to the debate.

Yours faithfully
Ernest Jordan, MACS
Department of Information Systems
City Polytechnic of Hong Kong
83 Tat Chee Avenue
Kowloon, Hong Kong

1See for example, Culture’s Consequences, G. Hofstede, Sage, 1980.
A direct search algorithm for global optimisation of multivariate functions

K.K. Benke and D.R. Skinner
Materials Research Laboratory (MRL)-DSTO, PO Box 50, Melbourne, Victoria, 3032

We describe a direct search method for locating the global optimum of a multimodal function. This is an adaptive probabilistic algorithm suitable for any function of many variables subject to arbitrary constraints. The algorithm is based on a simple model for noise reduction and uses an iterative method for averaging random perturbations in the parameter estimates. No prior assumptions are required about the continuity of the search domain, or about the continuity, differentiability and modality of the function. The algorithm is very simple, requiring little preparation and is suitable for application to functions that are non-linear, noisy and of high dimensionality. The performance of the algorithm is demonstrated by a number of numerical examples, including a highly dimensioned problem in pattern recognition.

Keywords and Phrases: global optimisation, direct search procedures, adaptive random search, numerical methods, learning algorithms, pattern recognition, artificial intelligence, neural networks.

CR Categories: F.2.1, G.1.6, I.2.8, I.5.

1. INTRODUCTION

An important goal in the design of adaptive systems, for use in pattern recognition and cybernetics, is the global search for a set of optimum operating conditions. The estimation of the optimal parameters of a mathematical model can be accomplished using special sequential methods, known as search procedures. In the general case, these search procedures require the existence of an explicit functional relationship predicting a response for each combination of control variables (Emshoff and Sisson, 1970).

The problem of finding an optimum on a multi-dimensional response surface has received considerable attention and various methods, both deterministic and probabilistic, are described in the literature (see, for example, Emshoff and Sisson, 1970; Gottfried and Weisman, 1973; Hillier and Lieberman, 1974; Taha, 1976; Glorioso and Osorio, 1980). Traditionally, numerical search procedures have often been based on gradient approaches, whose origins can be found in the theory and practice of differential calculus. At each step, the gradient is evaluated and a new search direction is chosen. Each new value differs from the previous value, depending on the specified step size, which determines search position and speed of convergence. Computation time increases rapidly with higher dimensionality, and there is a risk of entrapment at a local extremum since the final value achieved is effectively determined by the starting point. Stochastic methods, where the points at which a function evaluation occurs are chosen in a random manner, have been developed in an effort to improve the probability of locating the global extremum.

Many functions met in practice are non-differentiable and noisy, and may defy attempts to achieve an analytic solution (moreover, a noisy response surface may also introduce uncertainty in the calculation of the gradient). For these analytically intractable problems, statistical approaches are often the only recourse available. Swann (1974), and Jarvis (1975), in reviewing advantages associated with random search techniques, note that they are untroubled by plateaus, holes or discontinuities—which can cause serious problems with deterministic techniques.

Probabilistic search methods are particularly suitable for finding the global optimum on a multimodal surface, and are characterised by their extreme simplicity, especially when applied to non-linear functions of high dimensionality (Swann, 1974). Constrained simple random sampling can suffer, however, from the problem of slow convergence, requiring many trials for statistical accuracy. As shown by Gottfried and Weisman (1973), a deterministic gradient-based method is generally more efficient if the system is well behaved and can be characterised by a simple function that is smooth and continuous, unimodal and of low dimensionality.
A purely random search strategy is unmodified by any accumulated data, requiring little input and making very little use of any new information. The proximity of the final solution to the optimum is dependent on the number of trials, but independent of surface topography. For example; the probability of locating the maximum of a one-dimensional function to within a fraction $\beta$ of the permissible range of the independent variable, after $k$ trials, is given by $P(\beta) = 1 - (1 - \beta)^k$.

A random process selects the control parameters in a non-sequential, or parallel, manner and if the statistical properties of the search procedure itself can be controlled, then greatly improved convergence is possible. An adaptive probabilistic search is an attempt to provide faster convergence and greater statistical accuracy, whilst retaining the fundamental advantages of the simple random search.

In this context, a learning or adaptive system may be defined (Narendra and Thathachar, 1974) as "one characterised by its ability to improve its behaviour with time, in some sense tending towards an ultimate goal". In this paper, such a process is taken to include one for which:

(i) an operation is applied iteratively to the data,
(ii) the operation is a function of a number of parameters,
(iii) at each iteration, there is an opportunity to adjust some or all of these parameters, and
(iv) parameter adjustment is directed towards improving a performance index.

This description of learning is consistent with the usage of Narendra and Thathachar (1974), Shimura (1978) and Kohonen (1984), as applied to stochastic automata and artificial neural networks.

In the past, probabilistic learning algorithms, known as stochastic automata, have been applied to optimisation problems (McMurtry and Fu, 1966; Shapiro and Narendra, 1969; Jarvis, 1969; Wiswanathan and Narendra, 1973; Devroye, 1976). The stochastic automaton uses a reinforcement scheme designed to modify the search pattern. This involves a penalty or reward, based on the relative success of previous trials. Successes lead to transitions to higher states analogous to those occurring in a discrete Markov process.

Surveys of these learning automata reveal that they have potential in many areas of adaptive control, pattern recognition and systems analysis (Narendra and Thathachar, 1974; Narendra and Lakshmivarahan, 1977; Tsyplkin and Poznyak, 1977; Oomen and Hansen, 1984). However, they can be mathematically complex in terms of their structure and computational implementation. In addition, the application of learning automata to parameter optimisation is still in a developmental stage. For systems with a large number of dimensions, convergence can be very slow. This method of optimisation cannot be applied directly if the parameter space is defined for a continuous range of values.

Another recent approach to multivariate optimisation, referred to as simulated annealing, uses time-dependent probabilistic rules as an aid to avoiding entrapment at a local optimum (Kirkpatrick, Gelatt and Vecchi, 1983; Hinton and Sejnowski, 1983). These rules are formally equivalent to the conditions for thermally induced escape from a potential well, and, as the optimisation proceeds, are gradually made more stringent in a manner analogous to the annealing of solids. This approach has been developed into a learning algorithm for an artificial neural network known as the Boltzmann machine (Ackley, Hinton and Sejnowski, 1985). Unfortunately, convergence is inherently slow, and attempts to speed up the process by a too-rapid tightening of escape criteria can lead to local entrapment.

In the next section, we describe an optimisation algorithm that is simple, requires little preparation and is applicable to a wide range of problems. It has the fundamental advantages of the probabilistic search process, viz. improved performance in non-linear or multimodal environments of high dimensionality, and alleviates its major weakness, i.e. slow convergence and poor statistical accuracy. We show that, as a proposed global algorithm, it compares very favourably with a number of common global algorithms used in the past as benchmarks. The numerical examples are chosen mainly for ease of comparison between global algorithms, and in some cases could have been solved by other methods.

### 2. THE OPTIMISATION ALGORITHM

We consider specifically the maximisation of a function of many variables that is everywhere positive and finite, but application to most practical forms of optimisation represents a trivial extension. The function to be maximised is of arbitrary complexity and is treated as a non-linear mathematical programming problem with prescribed constraints. The approach used for optimisation has been applied to a number of diverse problems and consistently finds the global optimum, while converging much faster and more accurately than many competing global search algorithms.

Suppose we have a positive figure of merit, $f$, which is a function of a large number of variables expressed as a vector $\mathbf{X} = (x_1, x_2, ..., x_d)$ defined in a $d$-dimensional Euclidean space. The problem is to find the set of components $x_i$ which will maximise $f$ subject to a given set of constraints. The function may be non-differentiable and non-linear, so that the problem is not amenable to established techniques based on differential calculus or linear programming. The components may appear in the constraints as non-linear combinations, and the parameters may or may not be restricted to integral or positive values.

The problem may be expressed more formally as follows. Let $f$ be an objective function defining a hypersurface in a $d$-dimensional real Euclidean space, $\mathbb{R}^d$. Then the...
constrained optimisation problem is:

Maximise \( f = f(X) \); \( f > 0 \)
subject to \( X \in \mathbb{F}^d \); \( \mathbb{F}^d \subseteq \mathbb{R}^d \)

where \( \mathbb{F}^d \) is the subspace defining the feasible region. We employ a probabilistic search for \( f_{\text{max}} \) that is, we attempt to generate a sequence of vectors \( \{X_k\} \), sampled from within \( \mathbb{F}^d \), which converge to the optimal vector \( X_{\text{opt}} \) by a stochastic process, such that

\[
\lim_{k \to \infty} X_k = X_{\text{opt}}
\]

or, more specifically,

\[
\text{Prob} \{ \|X_k - X_{\text{opt}}\| > \delta \} \to 0 \quad \text{as} \quad k \to \infty \quad ; \quad \delta > 0
\]

so that \( X_k \) tends to \( X_{\text{opt}} \) in probability.

Suppose now we have two independent estimates for \( X_{\text{opt}} \), denoted by \( X_1 \) and \( X_2 \). These may be expressed as

\[
X_1 = X_{\text{opt}} + \Delta_1 \quad \text{(1)}
\]
\[
X_2 = X_{\text{opt}} + \Delta_2 \quad \text{(2)}
\]

where \( \Delta_1 \) and \( \Delta_2 \) represent error vectors. By analogy with electronic noise reduction through signal averaging, we may attempt to reduce the error vector by taking a weighted-mean vector of the form

\[
X_3 = \alpha X_1 + (1 - \alpha) X_2 \quad \text{(3)}
\]

where \( 0 \leq \alpha \leq 1 \). If we define

\[
\Delta_3 = \alpha \Delta_1 + (1 - \alpha) \Delta_2 \quad \text{(4)}
\]

then

\[
X_3 = X_{\text{opt}} + \Delta_3 \quad \text{(5)}
\]

(If the feasible region is disconnected or not convex, the value of \( X_3 \) obtained by weighted averaging may not obey the problem constraints. We have not specifically considered such problems, but for connected regions we suggest dividing in the ratio \( \alpha : (1 - \alpha) \) the shortest line that connects \( X_1 \) and \( X_2 \) and that lies wholly within the feasible region, provided such a line can be found in practice.)

We wish to know the value of \( \alpha \) that would be expected to minimise the magnitude of the vector \( \Delta_3 \), which is given by

\[
\| \Delta_3 \| = \left\{ (\alpha \Delta_1 + (1 - \alpha) \Delta_2) \cdot (\alpha \Delta_1 + (1 - \alpha) \Delta_2) \right\}^{1/2}
\]

\[
= \{ \alpha^2 \| \Delta_1 \|^2 + (1 - \alpha)^2 \| \Delta_2 \|^2 + 2 \alpha (1 - \alpha) \langle \Delta_1, \Delta_2 \rangle \}^{1/2} \quad \text{(6)}
\]

In practice, we will attempt to minimise the square of the magnitude, and for this purpose we will define

\[
Z \equiv E [ \| \Delta_3 \|^2 | \Delta_1, \Delta_2 ]
\]

that is, \( Z \) is the conditional expectation value for \( \| \Delta_3 \|^2 \) given \( \Delta_1 \) and \( \Delta_2 \).

In the absence of any information about the components \( \delta_{i,j} \) and \( \delta_{2,j} \) of the error vectors of \( \Delta_1 \) and \( \Delta_2 \), we assume that each component comes from a finite set of identical uncorrelated ensembles of random values with zero mean. This implies that

\[
E \left[ \sum_{j=1}^{d} \delta_{i,j} \delta_{2,j} \right] = 0 \quad \text{(7)}
\]

Hence

\[
E [ \Delta_i \cdot \Delta_2 ] = 0 \quad \text{(8)}
\]

and therefore

\[
Z = \alpha^2 \| \Delta_1 \|^2 + (1 - \alpha)^2 \| \Delta_2 \|^2 \quad \text{(9)}
\]

By differentiating this last equation with respect to \( \alpha \), it is easily shown that \( Z \) will be minimum for the condition

\[
\alpha = \| \Delta_2 \|^2 / (\| \Delta_1 \|^2 + \| \Delta_2 \|^2) \quad \text{(10)}
\]

that is, the vectors should be weighted inversely as the square of their error vectors. The expectation value for the square of the magnitude of the weighted-mean error vector will then be

\[
Z = \| \Delta_1 \|^2 \| \Delta_2 \|^2 / (\| \Delta_1 \|^2 + \| \Delta_2 \|^2) \quad \text{(11)}
\]

If the distributions of \( \Delta_1 \) and \( \Delta_2 \) are concentrated near their mean values, we can replace \( \| \Delta_1 \|^2 \) and \( \| \Delta_2 \|^2 \) by their expectation values. It follows that \( Z \) must be less than both \( E [ \| \Delta_1 \|^2 ] \) and \( E [ \| \Delta_2 \|^2 ] \). Therefore, the expected error in the weighted mean vector \( X_3 \) should be smaller than that in either \( X_1 \) or \( X_2 \).

Consider now the case where the two vectors \( X_1 \) and \( X_2 \) are independent random selections of vectors within a search domain, and the expectation value for the square of the magnitude of error vector for such a random selection is \( E [ \| \Delta \|^2 ] = Z_1 \). If we could combine \( X_1 \) and \( X_2 \) inversely as the squares of their error vectors, we would expect the magnitude of the new error vector to be about \( Z_2 = Z_1 / 2 \). In general, if \( Z_k \) is the expectation for \( \| \Delta \|^2 \) after the optimal combination of \( k \) vectors, optimal combination of the parameter vector \( X \) with a new random selection will give

\[
Z_{k+1} = Z_1 Z_k / (Z_1 + Z_k) \quad ; \quad k > 1 \quad \text{(12)}
\]

and hence

\[
Z_k = Z_1 / k \quad \text{(13)}
\]

The important point to note about this variance reduction is that the ratio of \( Z_k \) to \( Z_1 \) is independent of the number of dimensions, \( d \), in which the vectors are defined. (However, we might surmise that \( Z_1 \) would be roughly proportional to \( d \) for similar problems and that the requirements for the precision of \( X_{\text{opt}} \) would become more stringent as \( d \) increased.) We would therefore expect (and indeed demonstrate later) that, in the case of multi-dimensional search, the estimate for \( X_{\text{opt}} \) would converge to the true value much faster than would the estimate from constrained random search for which it may be shown that

\[
Z_k = Z_1 / k^{2d} \quad \text{(14)}
\]

where \( k \) here refers to the number of function evaluations.
but returns to near the expectation value when the performance of the algorithm for the cases where lower probability of finding a vector near the optimum, for 

occupies most of the domain, whilst for it is trivial to show that \( \| A \|_2 \) is equal to 

% probable because weighted means are rejected when they represent no improvement. Finally, the figure shows the results of a simple constrained random search, in which, as would be expected for a problem of such high dimensionality, there is no significant reduction in \( \| \Delta \|_2 \) even after 10 000 iterations. These results confirm that the estimate of \( Z_k / Z_1 \), which is independent of the number of dimensions \( d \), can be appropriate over a significant number of iterations, depending on the function \( f \). The results also confirm the counter-intuitive notion that the search can converge more quickly to a narrow peak than to a broad one. What they do not show is how well the search converges to an optimum on a multimodal surface, and, because this will be very dependent on the nature of the function, it appears that this can be established only by experiment.

It may be that there are certain symmetry operators \( S(X) \) that can be applied to the vector without altering the figure of merit, i.e.

\[
\text{f}(X) = \exp(-b\|X\|^2 / d)
\]

where

\[ \forall j, -1 \leq x_j \leq 1; 1 \leq j \leq d \]

and \( d \) again refers to the number of dimensions in \( X \). For this function, it is trivial to show that \( Z_k \) is equal to \( d/3 \), independent of \( b \), so the value of \( Z_k \) as defined above would also be independent of \( b \). Figure 1 shows the expectation \( Z_k \) as a function for \( k \) for the case of \( d = 1000 \) (a very highly dimensional function), together with a typical actual performance of the algorithm for the cases \( b = 2, 10 \) and 25. For \( b = 2 \), the central peak (to the point of inflexion) occupies most of the domain, whilst for \( b = 10 \) and 25 it occupies progressively less. We would therefore expect that the estimate for \( \| \Delta \|_2 \) after a large number of iterations would be poor for the first case, and much better for the others. This is borne out by the graph, which shows the actual \( \| \Delta \|_2 \) diverging from \( Z_k \) after about five iterations for \( b = 2 \), but after about 100 iterations for \( b = 10 \). For \( b = 25 \), the value of \( \| \Delta \|_2 \) diverges early from \( Z_k \) due to the lowered probability of finding a vector near the optimum, but returns to near the expectation value when the weighted averaging becomes effective.

\[ f(S(X)) \equiv f(X) \]

A typical case would be sign inversion, where \( S(X) = -X \).

Where \( n \) such operators exist, two estimates for the optimum vector may, in fact, approximate to different but equivalent vectors \( X_{opt} \). In this situation, the feasible region \( F \) actually comprises \( n + 1 \) equivalent subspaces. If we form the weighted mean not only with each new random vector, but also with the new vector after application of each symmetry operator, we effectively need to investigate only one of the subspaces. The volume of the feasible region has thus been reduced by a factor of around \( n + 1 \), and we would therefore expect to reduce the number of
iterations to reach a given precision by a similar factor. However, the number of function evaluations for each iteration of the algorithm will now be \( n_s + 2 \) instead of 2, so the required number of function evaluations will be multiplied by a factor of about \( (n_s + 2) / (2n_s + 2) \), e.g. \( 3/4 \) for \( n_s = 1 \).

In order to make a systematic and global search for an optimal solution, whilst taking maximum advantage of the variance-reduction strategy described above, we propose the following algorithm for iterative optimisation, for the case where there is a single symmetry operator.

Given that there exists an operator \( S(X) \) such that

\[
\begin{align*}
    f(S(X)) &= f(X) \\
\end{align*}
\]

and defining

\[
\begin{align*}
    X_b(k) &= \text{best estimate for } X_{\text{opt}} \text{ after } k \text{ iterations} \\
    X_r(k) &= \text{random vector, } X_r(k) \in F^d \\
    X_{w1}(k) &= \frac{f(X_b(k))X_b(k) + f(X_r(k))X_r(k)}{f(X_b(k)) + f(X_r(k))} \\
    X_{w2}(k) &= \frac{f(X_b(k))X_b(k) + f(X_r(k))S(X_b(k))}{f(X_b(k)) + f(X_r(k))} \\
\end{align*}
\]

and

\[
T(k) = \{ X_b(k), X_r(k), X_{w1}(k), X_{w2}(k) \}
\]

then

\[
\forall X_b(k), X_r(k), X_{w1}(k), X_{w2}(k) \in F^d
\]

\[
X_b(1) = X_r(1) \text{ and for } k \geq 1
\]

\[
X_b(k + 1) =
\begin{cases}
    X_b(k) \text{ if } f(X_b(k)) = \max_{X \in T(k)} f(X) \\
    X_r(k) \text{ if } f(X_r(k)) = \max_{X \in T(k)} f(X) \\
    X_{w1}(k) \text{ if } f(X_{w1}(k)) = \max_{X \in T(k)} f(X) \\
    X_{w2}(k)
\end{cases}
\]

Where there is no symmetry operator \( S(X) \), the deletions relevant to \( X_{w2}(k) \) are obvious, as are the extensions for more than one \( S(X) \).

The third possible choice for \( X_b(k + 1) \) is, of course, a weighted sum, or centroid, between the current best estimate and a noise injection, whilst, where \( S(X) = -X \), the fourth possibility represents a weighted difference. It has been found, in practice, that allowance for a weighted difference can speed convergence even when \( f(-X) \neq f(X) \). The conditions under which this represents a computational saving are currently under investigation.

Pseudo-code for this algorithm, which we call the “centro-roid algorithm”, is given in Table 1. After several iterations have been completed, \( X_1 \) represents the evolving trend, \( X_2 \) a random input and \( X_3 \) and \( X_4 \) perturbations to the trend. In each iteration, step 3.7 makes a decision on substitution for \( X_1 \) after comparing several inputs. Usually, there are only a few early substitutions due to random inputs. After a limited number of iterations, usually about two or three, the substitutions are predominantly learned rather than random, that is, are from \( X_3 \) or \( X_4 \).

### Table 1. Pseudo-Code for Centroid Algorithm.

1. Generate random parameter vector \( X_1 \) within specified constraints.
2. \( f_1 = f(X_1) \)
3. FOR required number of trials DO 3.1 to 3.7
   
   BEGIN find improved parameter vector
   
   3.1 \( f_2 = f(X_2) \)
   
   3.2 Compute the weighted mean \( X_3 \) of \( X_1 \) and \( X_2 \) according to
   
   \[
   X_3 = (f_1 X_1 + f_2 X_2) / (f_1 + f_2)
   \]
   
   3.3 Compute a further weighted mean \( X_4 \) according to
   
   \[
   X_4 = (f_1 X_1 + f_2 S(X_2)) / (f_1 + f_2)
   \]
   
   3.4 \( f_4 = f(X_4) \)
   
   3.5 IF \( f_4 = \max(f_1, f_2, f_3, f_4) \) THEN
   
   no action
   
   ELSE IF \( f_3 = \max(f_1, f_2, f_3, f_4) \) THEN
   
   BEGIN replacement due to random vector
   
   \( X_1 = X_2 \)
   
   \( f_1 = f_2 \)
   
   END
   
   ELSE IF \( f_2 = \max(f_1, f_2, f_3, f_4) \) THEN
   
   BEGIN replacement due to first weighted mean
   
   \( X_1 = X_3 \)
   
   \( f_1 = f_2 \)
   
   END
   
   ELSE \( f_1 \) must be the maximum, and
   
   BEGIN replacement due to second weighted mean
   
   \( X_1 = X_4 \)
   
   \( f_1 = f_4 \)
   
   END
   
   END
   
4. Optimum parameter vector is current value of \( X_1 \)

Optimum function value is current value of \( f_1 \)
5. END of algorithm

### 3. NUMERICAL EXAMPLES

In this section, we apply the algorithm to several problems with known solutions. In the first case, a multimodal problem is solved using the centroid algorithm and also a purely random search for comparison. In the second and third problems, the procedure is applied to multimodal functions and compared with several algorithms used for global optimisation. The final problem considered is relevant to pattern recognition and digital spatial filtering, and illustrates the performance under conditions of high dimensionality, non-linearity and multiple constraints.

#### Problem 1. A Multimodal Function with Closely Competing Optima

For the first problem, we devised a function specifically to test the global performance of the centroid algorithm and
GLOBAL OPTIMISATION OF FUNCTIONS

compare it with that of a simple random search (sometimes referred to as a Monte Carlo optimisation). This function is the sum of five two-dimensional Gaussian distributions and represents a multimodal surface with a number of peaks closely contesting for optimum status. The task is to find the global maximum of the following function, independent of starting point, using an iterative algorithm:

\[ f(x, y) = \sum_{i=1}^{5} a_i \exp\left(-\frac{(x - x_i)^2 + (y - y_i)^2}{s_i^2}\right) \quad (16) \]

\[-2 \leq x \leq 2, \quad -2 \leq y \leq 2\]

The values for \(a_i, x_i, y_i\), and \(s_i\) are provided in Table 2. A contour map of the function is plotted in Figure 2 (a) and an isometric projection depicted in Figure 2 (b). The global maximum of the function is:

\[ f(-0.01356, -0.01356) = 1.29695 \]

The global maximum is located on a peak which rises above all local maxima over an area of about 0.05% of the search domain. This peak appears in Figure 2 (b) as a small spur on the side of a ridge, which itself consists of a barely perceptible saddle point between two local maxima,

\[ f(-0.289, -0.206) = 1.217 \]
\[ f(-0.206, -0.289) = 1.217 \]

There are also two more distant well-resolved maxima,

\[ f(-0.003, 0.994) = 1.207 \]
\[ f(0.994, -0.003) = 1.207 \]

The difference in height (above the base plane) between the global maximum and any of the four local maxima is no more than 7%. The five peaks are therefore competing very closely for optimum status. The function also contains one minimum and several saddle points. The combined effect of all these stationary points is to surround the global peak and thus thwart any gradient-based deterministic method. On the outer perimeter of the active area of the function, far from the stationary points, there are flat regions providing very little information for gradient-based methods (see Figure 2 (b)).

We will compare the performance of the centroid algorithm with that of a pure random search, ignoring the symmetry operator \(S((x, y)) = (y, x)\). For assessment purposes, we will use two indices of performance. The first index is a measure of global performance and is defined as the number of function evaluations required to find a position on the global peak, with a function value higher than all local maxima (similarly, in the case of minimisation).

Table 2. Parameters for Multi-Gaussian Test Function.

<table>
<thead>
<tr>
<th>i</th>
<th>(a_i)</th>
<th>(x_i)</th>
<th>(y_i)</th>
<th>(s_i)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.5</td>
<td>0.0</td>
<td>0.0</td>
<td>0.1</td>
</tr>
<tr>
<td>2</td>
<td>1.2</td>
<td>1.0</td>
<td>0.0</td>
<td>0.5</td>
</tr>
<tr>
<td>3</td>
<td>1.0</td>
<td>0.0</td>
<td>-0.5</td>
<td>0.5</td>
</tr>
<tr>
<td>4</td>
<td>1.0</td>
<td>-0.5</td>
<td>0.0</td>
<td>0.5</td>
</tr>
<tr>
<td>5</td>
<td>1.2</td>
<td>0.0</td>
<td>1.0</td>
<td>0.5</td>
</tr>
</tbody>
</table>

Figure 2 (a). Contour plot of the multi-Gaussian function specified by Table 2 and used in the first numerical example.

Figure 2 (b). Isometric plot of the function of Figure 2 (a), as seen looking towards the origin from the (+x, +y) quadrant.

Figure 2 (c). Probabilities of finding the global peak, and achieving a function value of 99% of the maximum, for both the centroid algorithm and constrained random search when maximising the function of Figures 2 (a) and 2 (b). Results, in terms of function evaluations, are averaged over 100 experiments.
tion we require a position within the global valley having a value lower than all local minima). The second index is a measure of speed of convergence and is defined as the number of function evaluations required to solve the problem to a specified precision (for example, 99% of the maximum function value). The latter index is commonly used in unimodal search (see Box, 1965).

To achieve statistical confidence, we averaged the results of 100 independent experiments for each algorithm. Each experiment was started at a random point within the feasible region and halted after 5000 function evaluations. The results are plotted in Figure 2 (c). After 5000 function evaluations, the random search found a position on the global peak in 85% of experiments. The centroid algorithm found the global peak with a probability of 85% after only 1200 function evaluations, representing an improvement of 4:1 in global performance. This is an intriguing result, since a purely random search is generally regarded as the best method for finding the global peak (Gottfried and Weisman, 1973; Bekey and Ung, 1974).

As expected, the random search method was unable to match the centroid algorithm with respect to speed of convergence to the maximum function value. The random search converged to a function value within 99% of the global maximum in 20% of experiments after 5000 function evaluations. The centroid algorithm achieved the same result in fewer than 200 function evaluations, representing an improvement of 25:1 in speed of convergence. It is interesting to note that the plotted results for the global performance and the speed of convergence of the centroid algorithm are almost identical. In other words, the algorithm performs simultaneously as a global search technique and as a hill-climbing technique.

The advantage of the centroid algorithm over constrained random search is not as clear-cut when the global maximum is totally isolated and far from the domain of other significant function values. If a sixth peak is added to the function of Figure 2, using the specifications

\[ a_6 = 1.35, x_6 = -1.5, y_6 = -1.5, s_6 = 0.1, \]

the new global maximum becomes a thin, isolated peak rising from the floor to a value only 4% higher than the previous maximum, as shown in Figure 3 (a). The results of an investigation into convergence in this case, again ignoring the symmetry operator, are shown in Figure 3 (b). In terms of function evaluations, even for this difficult case, the centroid algorithm still converged to a function value of 99% of the maximum much faster, by a factor of around 3, than did random search. However, it was now not much better than half as fast as the constrained random search in finding a position on the global peak itself, since the adaptive component of the algorithm did not contribute significantly to the early stages of the search (the authors are indebted to one of the reviewers, who suggested the possibility of this behaviour).

**Problem 2. Comparison with Other Adaptive Random Search Techniques**

The second problem studied is to find the location of the global minimum of the following non-linear function, which was studied originally by Uosaki, Imamura, Tasaka and Sugiyama (1970), and later in greater detail by Bekey and Ung (1974):

\[ f(x, y) = \left(1 - 8x + 7x^2 - \frac{3}{2}x^3 + \frac{1}{4}x^4\right) y^2 \exp(-y) \]  

(17)

\[ 0 \leq x \leq 5, 0 \leq y \leq 6 \]

This surface has two minima at

\[ f(1,2) = -1.12779 \text{ (local minimum)} \]

\[ f(4,2) = -2.34581 \text{ (global minimum)} \]

Uosaki et al investigated the properties of an algorithm based on a modified Kiefer-Wolfowitz procedure (itself an extension of the stochastic approximation procedure of Robbins and Munro (Shimura, 1978)). Their algorithm attempts to find the global optimum by the simple expedient of adding to each observation noise with zero mean...
and non-zero variance which decreases with each observation. Their experiments showed that from a starting point of (1.0, 4.5), which is near the local minimum, the global minimum could be located. Bekey and Ung found, however, that this could not be achieved reliably and that the global performance depends on the random number sequence generated during the course of the optimisation.

Nevertheless, the algorithm of Uosaki et al produced superior global performance relative to the “complex” method of Box (1965), which is a probabilistic extension of the simplex methods described by Spendley, Hext and Himsworth (1962), and Nelder and Mead (1965). The test results of Uosaki et al indicate that the method of Box can find the global extremum in only about 50% of experiments. Recent research by Rangaiah and Krishnaswamy (1987) shows also that the method of Box is lacking somewhat in global performance; in their tests it converged to the global optimum in no more than 50% of experiments.

Bekey and Ung solved the above problem much more efficiently using an algorithm based on a modified random-creep procedure. This algorithm first locates a local minimum and then searches the parameter space with vector steps whose mean length gradually increases. Their results are reproduced in Table 3. Starting at the point (1.0, 4.5), the algorithm required 349 function evaluations to reach the global valley and a minimum function value was obtained after 451 function evaluations. The algorithm produced greatly improved global performance relative to the previous methods, finding the global minimum within 500 function evaluations in 26 out of 27 experiments.

When the centroid algorithm was applied to this problem (noting that minimising $f(x, y)$ is equivalent to maximising $-f(x, y)$, and ignoring symmetry conditions for simplicity), it required only 20 function evaluations to reach the global valley and the location of the minimum function value was obtained with greater precision after only 106 function evaluations (see Table 3). At this point, the error from the true minimum was an order of magnitude lower than that of the Bekey-Ung algorithm. The global valley was found within 20 function evaluations in 27 out of 27 experiments. The median number of function evaluations required to find the global valley was only 6 (the variation being due to the particular sequence of random numbers generated after starting from the initial point (1.0, 4.5)). The centroid algorithm is therefore faster than the Bekey-Ung algorithm by a factor of at least 16:1 in the number of function evaluations required to reach the global valley, and by a factor of 4:1 in the number of function evaluations required to converge to the minimum function value.

It is interesting to note that, since the starting point was near the local minimum, both algorithms converged to this stationary point first before proceeding to the global minimum. The centroid algorithm converged almost to the local minimum value within four function evaluations before switching to the global minimum, whereas the algorithm of Bekey and Ung required 85 function evaluations to match this precision and was still lingering on the local minimum after 108 function evaluations.

Many algorithms claimed to have a global capability are in fact simply probabilistic versions of existing deter-

### Table 3. Comparative Performance of Direct Search Algorithms.

(a) Bekey-Ung Algorithm.

<table>
<thead>
<tr>
<th>Function Evaluations</th>
<th>$x$</th>
<th>$y$</th>
<th>$f$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1.0000</td>
<td>4.5000</td>
<td>-0.46866</td>
</tr>
<tr>
<td>3</td>
<td>1.07864</td>
<td>4.14765</td>
<td>-0.56397</td>
</tr>
<tr>
<td>5</td>
<td>0.88992</td>
<td>3.85862</td>
<td>-0.64812</td>
</tr>
<tr>
<td>6</td>
<td>0.73725</td>
<td>3.63153</td>
<td>-0.68240</td>
</tr>
<tr>
<td>7</td>
<td>0.78464</td>
<td>3.58483</td>
<td>-0.71928</td>
</tr>
<tr>
<td>17</td>
<td>0.79484</td>
<td>3.27404</td>
<td>-0.81487</td>
</tr>
<tr>
<td>19</td>
<td>0.89123</td>
<td>3.23241</td>
<td>-0.85388</td>
</tr>
<tr>
<td>23</td>
<td>1.12829</td>
<td>3.05294</td>
<td>-0.90724</td>
</tr>
<tr>
<td>25</td>
<td>0.93272</td>
<td>2.91856</td>
<td>-0.95516</td>
</tr>
<tr>
<td>27</td>
<td>0.85677</td>
<td>2.75890</td>
<td>-0.98259</td>
</tr>
<tr>
<td>28</td>
<td>0.76319</td>
<td>2.58384</td>
<td>-0.99818</td>
</tr>
<tr>
<td>30</td>
<td>0.70626</td>
<td>2.42343</td>
<td>-0.99820</td>
</tr>
<tr>
<td>35</td>
<td>0.68263</td>
<td>2.06214</td>
<td>-1.02059</td>
</tr>
<tr>
<td>42</td>
<td>0.75268</td>
<td>2.10778</td>
<td>-1.06371</td>
</tr>
<tr>
<td>44</td>
<td>0.80029</td>
<td>2.03451</td>
<td>-1.08912</td>
</tr>
<tr>
<td>46</td>
<td>0.83687</td>
<td>1.84182</td>
<td>-1.09569</td>
</tr>
<tr>
<td>47</td>
<td>1.10311</td>
<td>1.84612</td>
<td>-1.11297</td>
</tr>
<tr>
<td>56</td>
<td>0.97345</td>
<td>1.83422</td>
<td>-1.11904</td>
</tr>
<tr>
<td>71</td>
<td>0.90508</td>
<td>1.96726</td>
<td>-1.11954</td>
</tr>
<tr>
<td>85</td>
<td>0.92422</td>
<td>1.90217</td>
<td>-1.12004</td>
</tr>
<tr>
<td>87</td>
<td>0.97399</td>
<td>2.04229</td>
<td>-1.12669</td>
</tr>
<tr>
<td>103</td>
<td>1.02506</td>
<td>1.98295</td>
<td>-1.12721</td>
</tr>
<tr>
<td>108</td>
<td>1.00528</td>
<td>1.98600</td>
<td>-1.12771</td>
</tr>
<tr>
<td>349</td>
<td>4.08324</td>
<td>3.53003</td>
<td>-1.57444</td>
</tr>
<tr>
<td>350</td>
<td>3.99468</td>
<td>3.28736</td>
<td>-1.74914</td>
</tr>
<tr>
<td>352</td>
<td>3.99070</td>
<td>3.07293</td>
<td>-1.89385</td>
</tr>
<tr>
<td>354</td>
<td>3.93506</td>
<td>2.99573</td>
<td>-1.93899</td>
</tr>
<tr>
<td>355</td>
<td>4.16349</td>
<td>2.93650</td>
<td>-1.94227</td>
</tr>
<tr>
<td>358</td>
<td>4.04506</td>
<td>2.84550</td>
<td>-2.03576</td>
</tr>
<tr>
<td>361</td>
<td>4.06679</td>
<td>2.54815</td>
<td>-2.19399</td>
</tr>
<tr>
<td>364</td>
<td>4.13033</td>
<td>2.15666</td>
<td>-2.30275</td>
</tr>
<tr>
<td>367</td>
<td>3.88618</td>
<td>2.03345</td>
<td>-2.32544</td>
</tr>
<tr>
<td>370</td>
<td>3.90530</td>
<td>2.06060</td>
<td>-2.32991</td>
</tr>
<tr>
<td>373</td>
<td>4.05143</td>
<td>1.93819</td>
<td>-2.3391</td>
</tr>
<tr>
<td>385</td>
<td>4.00596</td>
<td>1.95753</td>
<td>-2.34470</td>
</tr>
<tr>
<td>396</td>
<td>4.00477</td>
<td>1.97394</td>
<td>-2.34540</td>
</tr>
<tr>
<td>451</td>
<td>3.99093</td>
<td>2.01405</td>
<td>-2.48666</td>
</tr>
</tbody>
</table>

(b) Centroid Algorithm.

<table>
<thead>
<tr>
<th>Function Evaluations</th>
<th>$x$</th>
<th>$y$</th>
<th>$f$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1.0000</td>
<td>4.5000</td>
<td>-0.46866</td>
</tr>
<tr>
<td>2</td>
<td>1.16650</td>
<td>2.37807</td>
<td>-1.07383</td>
</tr>
<tr>
<td>4</td>
<td>0.95133</td>
<td>2.01478</td>
<td>-1.12573</td>
</tr>
<tr>
<td>20</td>
<td>3.30404</td>
<td>2.38261</td>
<td>-1.77300</td>
</tr>
<tr>
<td>24</td>
<td>3.86722</td>
<td>2.03921</td>
<td>-2.31837</td>
</tr>
<tr>
<td>32</td>
<td>3.89228</td>
<td>2.12935</td>
<td>-2.31876</td>
</tr>
<tr>
<td>38</td>
<td>3.94973</td>
<td>2.04625</td>
<td>-2.34059</td>
</tr>
<tr>
<td>68</td>
<td>3.95468</td>
<td>2.02009</td>
<td>-2.34233</td>
</tr>
<tr>
<td>86</td>
<td>3.95867</td>
<td>1.98391</td>
<td>-2.34295</td>
</tr>
<tr>
<td>106</td>
<td>4.00289</td>
<td>2.00373</td>
<td>-2.34557</td>
</tr>
</tbody>
</table>
GLOBAL OPTIMISATION OF FUNCTIONS

ministic algorithms. For example, Beltrami and Indusi (1972) developed an adaptive random search algorithm by refining the algorithm of Lawrence and Steiglitz (1972), who in turn had randomised the pattern search method of Hooke and Jeeves (1959). The algorithm has a limited global search capability, for example, detecting the global minimum of the following function on 50% of occasions:

\[ f(x, y) = 1.41x^4 - 12.76x^3 + 39.91x^2 - 51.93x + 24.37 + (y - 3.9)^2 \]  

(18)

The global minimum occurs at \( f = -3.98 \) at \((3.48, 3.9)\) and the centroid algorithm will find this consistently in under 100 function evaluations, despite starting at random points located within the active area of the local minimum itself at \((1.4, 3.9)\).

The Rastrigin function (see Polovinkin, 1981) is a more complicated example of a multimodal function. We have applied the centroid algorithm to the location of the minimum of this function, which is given by

\[ f(x, y) = x^2 + y^2 - \cos(18x) - \cos(18y) \]  

with \(-1 \leq x \leq 1, -1 \leq y \leq 1\)  

(19)

where the accuracy required for locating the global minimum point, \( f(0,0) = -2 \), is \( x^2 + y^2 \leq 4/(5917^n) \). This function (used as a test in the past by Soviet mathematicians) is smooth and continuous, with 50 local minima in a lattice arrangement, and is quite amenable to a random method incorporating some form of gradient-descent procedure for local search. Even with this type of function, the centroid algorithm with no prior knowledge or arbitrary parameters, and using one symmetry condition, i.e. \( X_{w1} \) and \( X_{w2} \) representing weighted mean and difference of \( X_b \) and \( X_r \), typically required only about 380 function evaluations.

In contrast, Polovinkin reported that Monte Carlo search required 5917 evaluations, and multi-start random and multi-start gradient methods 1176 and 556, respectively. He did find, however, that an involved algorithm of “competing points” could accomplish this in as few as 111 function evaluations, but only on the basis of considerable \textit{a priori} information (e.g. volume of global minimum is larger than that of all local minima), assumptions that were clearly invalid for general application. Experimental details were lacking, but preparation and running times appear to be significant. In the general literature, the conditions of experimental comparisons are not always completely described, making evaluation of relative efficiencies of algorithms (which may vary from problem to problem) a somewhat hazardous occupation.

Problem 3. Comparison with Stochastic Automata

The third demonstration of this algorithm is an application to a very noisy bimodal test function (Figure 4 (a)). The function \( E(x) \), defined only for integral abscissae, was used by Shapiro and Narendra (1969), and others (Wiswanathan and Narendra, 1973; Devroye, 1976) to test the performance of various stochastic automata for the purpose of parameter optimisation. The noisy measurements, \( f(x, r) \), are related to the expected values, \( E(x) \), by the following function:

\[ f(x, r) = E(x) + r \]  

(20)

The random noise variable, \( r \), is uniformly distributed over the range \((-2.0 \leq r \leq 2.0)\). A particular value of the measurement \( f(x, r) \), at a given iteration during optimisation, has a uniform probability of occurring within a band about the function \( E(x) \) to be maximised. The function was designed to preclude the use of many conventional optimi-
sation methods, such as gradient-based approaches. The function is multimodal and the variations in $E(x)$ are much smaller than the variance of the added noise (that is, the signal is effectively submerged in the noise).

We carried out 1000 experiments, using different sets of random numbers, and at each iteration in each experiment recorded the value of $x$ which, with the added random noise, gave the highest function value. This information was used to compute the probability of obtaining the optimum value of $x(6)$ as a function of the number of iterations (see Figure 4 (b)). This probability is 0.95 after 260 trials, calculated using the centroid algorithm with no symmetry operator. For comparison, we also plotted the performance of several stochastic automata. The best is the algorithm of Shapiro and Narendra (1969), which reaches 0.95 in 7400 trials. If the computing loads for performance of several stochastic automata. The best is the algorithm of Shapiro and Narendra (1969), which reaches 0.95 after 7400 trials. If the computing loads for iterations of the two algorithms are comparable, then the centroid algorithm is an order of magnitude faster in convergence.

**Problem 4. A 13-Dimensional Problem Relevant to Computer Vision**

The final problem is an application in the field of machine vision, and requires the generation of a discrete convolution mask, or spatial filter (Benke and Skinner, 1987). The mask is a square array of integer weights $w(m, n)$, and operates on a digitized image which takes the form of a rectangular array of pixels with grey levels expressed as integers $g(i, j)$. The discrete convolution operation (Pratt, 1978) generates a new image $h(i, j)$, the filtered image, according to the equation:

$$h(i, j) = \sum_{m=-N}^{N} \sum_{n=-N}^{N} w(m, n) g(i - m, j - n)$$

(21)

where $1 \leq i \leq I, 1 \leq j \leq J; \ \forall i, j, 0 \leq g(i, j) \leq G$

and $G, I, J$ and $N$ are parameters specific to each problem.

The filtered image is, of course, defined only over an array of size $(I - 2N) \times (J - 2N)$. The problem is to find the mask parameters $w(m, n)$ that maximise the normalised variance, or energy, $\nu(w)$ of the filtered image $h(i, j)$ when convolved with a given image $g(i, j)$ according to Equation (20), where

$$\nu(w) = \frac{\sum_{i=1}^{I-N} \sum_{j=1}^{J-N} h^2 (i, j)}{(I - 2N)(J - 2N) G^2 \sum_{m=-N}^{N} \sum_{n=-N}^{N} w^2 (m, n)}$$

(22)

This energy is independent of multiplicative scaling changes in $g$ and $w$, and for a uniformly patterned image is largely independent of the size of the image sample $(I \times J)$. The optimisation is the machine-vision analogue of finding a feature detector in the human visual system that is optimally matched to an observed pattern, and represents a learning algorithm for visual pattern recognition.

### Table 4. Generation of Convolution Mask using Centroid Algorithm.

<table>
<thead>
<tr>
<th>Iteration</th>
<th>Substitutions</th>
<th>Energy</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>66</td>
<td>0.33</td>
</tr>
<tr>
<td>10</td>
<td>53</td>
<td>0.44</td>
</tr>
<tr>
<td>100</td>
<td>4</td>
<td>0.65</td>
</tr>
</tbody>
</table>

In the particular problem under consideration, the mask is a $5 \times 5$ matrix, that is $N = 2$, subject to the constraints

$$\sum_{m=-2}^{2} \sum_{n=-2}^{2} w(m, n) = 0$$

$$w(m, n) = w(-m, n)$$

$$-2 \leq m \leq 2, -2 \leq n \leq 2$$

$$\max |w(m, n)| \leq 100$$

(These constraints are for illustration only — we have also optimised $21 \times 21$ masks with real coefficients and without any symmetry constraints, i.e. investigated a function in 439 dimensions.) The problem is therefore a search for a maximum of a function defined at approximately $10^{30}$ discrete points in a 13-dimensional space. An exhaustive search of all possible solutions is clearly impracticable, and, for most images, it is not feasible to find the optimum mask by theoretical analysis. However, it is simple to prove that, when all the edges contained in the binary image are vertical, the optimum mask should have the element values shown in the last array in Table 4. There is one symmetry operator to take into account here, since the figure of merit, the filtered image energy, is invariant to sign inversion of the mask.

In order to generate random convolution masks efficiently within the problem constraints, sets of thirteen uniformly distributed random numbers were multiplied by a $13 \times 25$ constraint matrix to give the values for the mask elements (this technique is applicable in all cases where the constraints are linear). The results of a search for the optimum mask for a vertical pattern are shown in Table 4. The computing load can be assessed from the fact that the number of function evaluations is $3 \times (\text{Iteration number}) - 2$, since $n_i$ is equal to unity and the initial random mask generation is considered to be iteration number 1. This table also shows, as a percentage of its maximum possible value, the energy obtained using each mask.

It can be seen that the first estimate appears truly random within the constraints. After ten iterations, each of which caused a new matrix to be substituted for the current
Table 5. Generation of Convolution Mask using Constrained Random Search.

<table>
<thead>
<tr>
<th>Iteration 1</th>
<th>Substitutions 1</th>
<th>Energy 3.31</th>
</tr>
</thead>
<tbody>
<tr>
<td>90</td>
<td>1 -100</td>
<td>1 90</td>
</tr>
<tr>
<td>75</td>
<td>-62 -100 -62</td>
<td>75</td>
</tr>
<tr>
<td>31</td>
<td>37 -95 37</td>
<td>31</td>
</tr>
<tr>
<td>57</td>
<td>30 -94 30</td>
<td>57</td>
</tr>
<tr>
<td>38</td>
<td>-42 -61 -42</td>
<td>38</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Iteration 10</th>
<th>Substitutions 4</th>
<th>Energy 33.42</th>
</tr>
</thead>
<tbody>
<tr>
<td>-66</td>
<td>54 39 54 -60</td>
<td></td>
</tr>
<tr>
<td>-75</td>
<td>-14 100 -14 -75</td>
<td></td>
</tr>
<tr>
<td>-57</td>
<td>-31 100 -31 -57</td>
<td></td>
</tr>
<tr>
<td>-37</td>
<td>63 72 63 -37</td>
<td></td>
</tr>
<tr>
<td>-58</td>
<td>23 95 23 -58</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Iteration 100</th>
<th>Substitutions 5</th>
<th>Energy 34.72</th>
</tr>
</thead>
<tbody>
<tr>
<td>-74</td>
<td>56 89 56 -89</td>
<td></td>
</tr>
<tr>
<td>-52</td>
<td>-29 100 -29 -52</td>
<td></td>
</tr>
<tr>
<td>-19</td>
<td>74 81 74 -81</td>
<td></td>
</tr>
<tr>
<td>-51</td>
<td>-28 100 -28 -51</td>
<td></td>
</tr>
<tr>
<td>-57</td>
<td>-26 100 -26 -57</td>
<td></td>
</tr>
</tbody>
</table>

best estimate, there is a significant grouping of negative values towards the centre. After 100 iterations, with 55 substitutions, the matrix is recognisably close to the theoretical form, with a deficit of less than 1% in the figure of merit. The experiment was continued to 1000 and 10 000 iterations, with the known ideal matrix finally appearing at number 4492. It was found that different starting points for the pseudo-random number generator gave the illustrated matrix or the sign-inverted version with about equal probability, and that around 200 iterations generally produced a figure of merit within 0.5% of the optimum value.

In order to demonstrate that the adaptive mechanism constitutes a significant part of the mask-generation process, the above trial was repeated with the mask averaging removed, i.e. with a purely random search for matrix elements within the constraints. The results of this exercise are shown in Table 5, where it can be seen that, after 10 000 iterations, the deficiency in the figure of merit is greater than 20% and individual parameters are subject to significant error. Comparison with Table 4 amply demonstrates the efficacy of the learning process, even taking into account that, in this case, the computing load per iteration is about three times greater for the centroid algorithm than for constrained random search. (For example, the performance of the centroid algorithm at 100 iterations, or 300 function evaluations, is far better than that of constrained random search after 10 000 function evaluations.)

4. DISCUSSION

The algorithm is most useful for problems which may be theoretically intractable, and where data may be discontinuous, partial or approximate, and perhaps very noisy. The probabilistic direct search approach appears to be particularly effective for functions which are non-linear, highly dimensional and which may contain plateaus or discontinuities. Swann (1974) notes that a direct search algorithm may often prove to be more reliable and stable than a gradient-based method, as no assumptions are made about the function, and much less preparation is required.

Unlike many traditional techniques, such as methods using Lagrange multipliers, the algorithm obviates the need for solving large sets of simultaneous equations. Only \((n_r + 3)(n_r + 1)\) feasible values are stored, where \(n_r\) is the number of symmetry operators and \(n_p\) is the number of parameters. In fact, the software for the optimisation of masks of size up to 21 x 21 is currently in use for vision research on a desk-top computer and incorporates an efficient new algorithm for fast iterative convolution.

For each iteration, the parameter set is selected from within the feasible region using a random process (this can be achieved very efficiently when the constraints are linear). Each parameter in the set is selected independently for each iteration, which is generally not true for gradient methods. This suggests that for problems of extremely high dimensionality, i.e. with thousands of variables, the algorithm can be implemented even more rapidly using parallel processing techniques.

It is clear, particularly from Table 4, that, as with the constrained random search strategy, the optimum can be found within an arbitrarily small error margin which is dependent on the number of iterations in the simulation, so that if the optimum itself is not found by either a random or learned input during one of the iterations, then the algorithm converges to it in a probabilistic sense.

Although the algorithm is suitable for general application, it may not converge as rapidly as an efficient gradient method in the case of, say, a unimodal function which is simple and smooth, continuous and differentiable, and of low dimensionality. Even for this limited class of problem, there are other considerations influencing overall efficiency. Himmelblau (1972) notes that the total cost associated with the solution of an optimisation problem must combine both the preparation time and the computer time used. He concludes that a solution obtained using a less efficient algorithm with easily prepared code may in fact cost much less than using a highly efficient code that requires many hours of preparation. He stresses that finding analytic derivatives is a major source of human error apart from being very time-consuming. It is clear that the preparation time required for the centroid algorithm is very small, being only slightly more than for a pure random search.

Swann (1974) observes that, in order to improve preparation time for a gradient-based method, a numerical approximation to differentiation can lead to truncation or cancellation errors which may nullify the underlying theory of the algorithm, resulting in slow or non-existent convergence. In many problems, a measure of efficiency based on the number of function evaluations may unfairly favour a gradient-based algorithm. For example, in the case of complex functions of high dimensionality, the time required to determine the point for the next function eva-
evaluation can often be many times greater than that required for the evaluation of the function itself (see Himmelblau, 1972). One reason for this is the requirement for the evaluation of the first and, if necessary, the second derivatives. As an example of preparation time for the centroid algorithm, using a shell program written in Pascal for an IBM AT compatible desk-top computer, only 15 minutes additional programming time was required to solve the Rastrigin problem (see Equation 19).

Most gradient methods are supported by optimality proofs, whereas most direct search methods are not, and it is generally not possible to derive convergence criteria for direct search (see Swann, 1974, and the review by Shimura, 1978, who cites a number of heuristic algorithms published without convergence proofs). Whilst we have not provided a rigorous analysis of the convergence behaviour of the centroid algorithm, considerations based on expectations for noise reduction suggest that the size of the error vector, \( \Delta = X - X_{opt} \), between the unknown optimal solution and the best estimate after \( k \) iterations of the algorithm, is largely independent of the number of dimensions in the domain of \( X \). This is supported by an investigation of the variation of \( \Delta \) during optimisation of a simple unimodal function (Equation 15).

5. SUMMARY

We describe a global direct search algorithm suitable for the optimisation of an arbitrary multivariate function. The centroid algorithm is noteworthy for its extreme simplicity, requiring very little coding and preparation time, and is applicable to functions which may be non-linear, noisy or highly dimensional, and may have discrete or continuous domains. A fundamental feature of the algorithm is its performance on multimodal functions, especially the difficult cases where approximate solutions are normally obtained by recourse to random search techniques.

The performance of the algorithm is compared with well known global algorithms including random search, various adaptive and guided random search methods, and stochastic automata. Faster and more accurate convergence to a global solution was demonstrated on benchmark problems (selected for the convenient comparison of global algorithms rather than for their complexity or the absence of other methods of solution), and also in some practical applications. The numerical examples include a number of non-linear multimodal functions, illustrating performance under conditions of noise, non-differentiability and high dimensionality, and with discrete search domains.

In addition to producing significantly improved global performance, the centroid algorithm has no arbitrarily selected parameter settings, such as step size or search direction, and appears to be particularly effective in applications where very little information is available on the function topography. If required, the convergence can be accelerated by systematically reducing the domain during the course of the optimisation, but at the expense of a diminishing global capability. Likewise, the precision can be improved by performing a second optimisation over a reduced neighbourhood. The procedure has potential applications in pattern recognition, adaptive filtering, computer vision and in the training of artificial neural networks.

6. ACKNOWLEDGEMENTS

We would like to express our appreciation to Dr Peter Beckwith and Dr John Ternan for many technical discussions concerning the algorithm and this paper. We would also like to thank Mr David Walker for discussions and for suggesting an efficient software implementation for the convolution experiment. Finally, we are grateful for the constructive criticism which was offered by the anonymous reviewers of this paper and which resulted in improved presentation and significant clarification.

7. REFERENCES


GLOBAL OPTIMISATION OF FUNCTIONS


BIOPGRAPHICAL NOTES
Kurt Benke received BSc and MSc degrees in Physics from the University of Melbourne, and a PhD in Mathematics and Computing (for studies in computer vision) from Deakin University. He also holds a postgraduate diploma in Applied Statistics from the Royal Melbourne Institute of Technology.
He was originally employed with the Kodak Research Laboratory, where he gained research experience in theoretical and experimental optical physics, X-ray physics, and electromagnetic scattering. He is currently a Senior Research Scientist with the DSTO Materials Research Laboratory, and has been actively involved in research in human and machine vision, robotic inspection, pattern recognition and sonar surveillance.

David Skinner received a BSc in Physics from the Queen’s University of Belfast, Northern Ireland, in 1957, and until 1960 worked with Short Brothers and Harland on the design of aircraft transducers. Since then he has been with the DSTO Materials Research Laboratory, Melbourne. He is a Principal Research Scientist and Task Manager for research in sidescan sonar image analysis. His research experience includes work in theoretical and experimental xerography, laser physics and its applications, remote colorimetry, and in visual camouflage and other countersurveillance. Research interests include digital image processing, visual perception, optical physics and adaptive systems.

CALL FOR PAPERS
SPECIAL ISSUE ON ALGORITHMS
The May 1992 issue of the Journal will focus on the theme of Algorithms. Papers are sought on the broad spectrum of topics in Algorithms, including but not limited to:

* complexity of algorithms and data structures
* parallel algorithms
* computational geometry
* VLSI algorithms
* randomised algorithms

Papers should be addressed to the broad Australian Computer Journal audience, and should not be too narrow, except when giving examples to illustrate broad concepts.

Papers must be presented in conformance with this Journal’s style, including Harvard-style citations and reference list, and excluding footnotes and endnotes.

Potential authors are requested to register their interest with either of the Guest Editors by 12 August 1991 indicating a provisional title and a (non-binding) 50-200 word outline of the intended contribution. A package of Instructions for Authors will be provided by return mail.

Dr Peter Eades
Department of Computer Science
The University of Queensland
St Lucia Queensland 4072 Australia
Telephone +61 07 365 2904
Facsimile +61 7 365 1999

Dr Alistair Moffatt
Department of Computer Science
University of Melbourne
Parkville Victoria 3052 Australia
Telephone +61 03 344 5229
Facsimile +61 3 348 1184

Each time I open this book, I am surprised by something practical I had not noticed before. When one considers what should go into a book on parallel programming, one thinks of the gross factors such as the notation of the language, creating and destroying processes, synchronizing them, communicating between them and sharing memory. These things are in the book and a newcomer to the world of parallel programming will benefit from a good read. However, an old hand will be surprised by the names given to some functions. For example, the busy-wait method of synchronizing using a flag variable is called a spin-lock and the method of ensuring that server synchronizing them, communicating between them and sharing memory. It performs independent parallel tasks terminate before proceeding is to use a variable is called a spin-lock and the method of ensuring that server synchronizing them, communicating between them and sharing memory. It is easy to read and I found it easy to move around because it is well organised, has a detailed contents listing and a good index. The content of the book covers time sharing of processors, processes, shared memory, parallel programming techniques, scheduling of parallel tasks, converting serial to parallel programs including a treatment of data dependencies. It then discusses performance issues such as the positioning of the process fork, the effect of the number of processes and the use of cache. Finally discrete event, discrete time simulation is covered and a number of applications discussed. The last chapter presents semaphores and events. This material should have been presented earlier.

One of the surprises was the use of a subset of Fortran. The author justifies this by claiming that the subset is really a subset of almost any language one cares to name. Perhaps this allows the programmer new to parallel programming to see how he can do things in his favourite language. However, it denies the user the syntactical advantage obtained from the elegance of Inmos OCCAM or even from Encore Parallel Fortran described in an appendix. Parallelism is achieved in the same way as in Unix using fork and process identifiers to determine what branch the child should be following, whereas in OCCAM this is done by using the PAR construct.

As well as the unhelpful syntax, the constructs offered for parallel programming provide the user no protection from misuse or oversight. For example, process—join must be called with the number of parallel processes which have been forked. This should be unnecessary and will cause many obscure faults. There should be a discussion about these issues.

Problems are interspersed throughout the text. Some are answered or have partial answers but the reader will find them valuable. The appendices discuss the relationship between C and Fortran, Encore Parallel Fortran and parallel programming with Unix. The software presented in the text is offered for PCs on a floppy disc.

I think the author’s choice of Fortran limits the audience by excluding the undergraduate who will expect a more formal course. There is no discussion of languages for parallel programming, and yet the text is introductory. I think the market for this text consists of those people with access to parallel machines with a background in Fortran. It may prove to be useful to someone teaching in the area, but the expert will find the book of limited benefit. The price quoted is very reasonable.

Peter Horan
Deakin University, Geelong


This is an excellent book for all those involved in the process of negotiating data processing contacts. It covers all aspects of the IT industry — hardware, software and services. The book’s audience includes manufacturers, vendors, software developers, contractors, the associated legal profession and of course users. In fact, the book is primarily aimed at this last group and offers copious advice in obtaining the best possible contract with a supplier.

It starts with the base position of most vendors, viz. ‘If you want to use our product, you have to sign our form’. The authors go on to state that the vendor is usually in a better bargaining position than the user which has resulted in grossly inadequate contractual relationships. One reason given for this (which this reviewer does not necessarily agree with) is that the computer industry started as a hardware rental industry. It continues ‘This practice lies at the root of the contractual problem in the industry; a rental contract is not a complex document, and since it is ostensibly terminable on short notice, it does not require that the obligations of the parties be spelled out in great detail.’

This is the third edition (spanning 15 years) and this latest edition reflects changes and possible maturing of law and the computer industry. I found it well written and without a legal background easy to read and understand. It begins with a number of narrative chapters covering the contracting environment, negotiating methodology and strategy and the various types of data processing contracts. The book concludes with detailed appendices on contract clauses and checklists. There are plenty of example letters and contracts to be found throughout the text.

For most of the contract clauses the authors give examples of ‘ideal phrasing’ (from the purchaser’s point of view) and fall back alternatives if the ideal situation is not achieved. The checklists are copious, for example there is one for hardware contracts which notes 142 separate points and another for facilities management contracts covering 173 points.

One minor criticism is that, although published last year it still quotes Burroughs and Sperry as separate companies. Also the authors make no mention of workstations and the influence of UNIX and OSF in their otherwise comprehensive summary of the current computer industry. And of course, it is written primarily for the US market. There is a whole chapter on US contract law. However since the majority of IT vendors are still US based, it gives some insight into the ‘standard contracts’ presented by them.

Finally, as the authors note, it is a guide to contracting in the computer industry and should be used in conjunction with appropriate legal counsel.

In conclusion, I recommend this book despite my comments above, for all those involved in negotiating contracts in the computer industry. The high price is more than offset by the potential gains to be made through following the rich advice outlined in the text.

Lawrie Hanson
Digital Equipment Corporation

SKIENA, Steven.(1990): Implementing Discrete Mathematics - Combinatorics and Graph Theory with Mathematica, Adisson-Wesley, 334pp., $US 43.25 (Hardcover).

I sat down to read this book at home and very soon realised that I needed to read it at work! For in many ways it is more of a reference manual for a computer software package than a textbook. It is essential to have the program Mathematica in order to use this book. One can then obtain the combinatorics package Combinatoriaca (which is essentially what this book is all about) to run within Mathematica, via FTP from the author in the U.S. (or via disk from Wolfram Research Inc.)

Being rather a novice user of Mathematica, I was surprised to find how easy it was to use the various combinatorial functions given in the book. Each one is illustrated with an example, and the expected input and output for each is given, together with a little explanation. There are six chapters, the first two covering ‘counting’ aspects of combinatorics, such as permutations and combinations (including a little on Polya theory), partitions, and Young Tableaux; even Gray codes are included. The remaining four chapters are graph-theoretic: Representing Graphs; Generating Graphs; Properties of Graphs; Algorithmic Graph Theory. It is in the area of graph theory, of course, that the graphics capabilities of Mathematica are fully utilised.

The book also contains both Exercises and Research Problems at the end of each chapter, and it includes Appendices on Mathematical preliminaries, Mathematica preliminaries, a glossary of functions used in the package (and given explicitly in the book),
and 17 pages of references. The references are particularly useful to
direct the reader to the original source of the algorithms implement-
ed. It is pointed out that both the time and space are lacking to
fully develop the theory behind the algorithms used in the book.
I can envisage this being an extremely useful package to incorpo-
rate in an undergraduate course on graph theory. The graphics one
can produce are great fun; I particularly liked the function Shake-
Graph, which moves the vertices of a graph around to change the
picture, thus clarifying existence of certain edges if three or more
vertices were collinear. However, in comparison with other software
I have used for combinatorics, such as Brendan McKay's nauty or
John Cannon's GAPLEY, I believe that this combinatorics package
is less suited to research use.

In any future updates I should like to see the areas covered
extended to include topics such as, for instance, Latin squares (input
a partial one; complete in all possible non-isomorphic ways?); Room
squares; block designs (is a given design resolvable?). However,
evertheless, in these areas of combinatorics more than other ar-
eas, the computational time might become too great too quickly.

To summarise, if the reader has access to Mathematica, is inter-
ested in combinatorics and wants to try out some combinatorial
algorithms without the agony of writing them him/herself, then I
thoroughly recommend this book, together with the accompanying
Combinatorica package.

Elizabeth J. Billington
The University of Queensland

460 pp., $537.50 (hardcover).

This book is rather unusual, it is in an area that I normally
wouldn't consider but is rather interesting and relevant. It consists
of a number of essays about technical writing and online text, some
of which were presented at the Fifth Annual Conference: Writing
for the Computer Industry directed by the editor. The author is a
Lecturer in the Writing Program at MIT, which, among other
things, is looking at the role of writers in the computer industry.
He seems to have assembled an impressive list of authors for the
various essays. Judging by the citations in a number of the essays
many of the leading authorities in online text have contributed to
this work.

The book is divided into four sections, each of which explores a
different area in the role of online text and the authors of such
text. These sections are titled:

- Hypertext and Hypermedia: Designing Systems for the
  Online User,
- Multimedia and Nonlinear Information Architectures,
- The Social Perspective: Writers, Management, and the
  Online Environment, and
- Sensemaking, Learning, and the Online Environment.

Although hypertext and hypermedia are prominently mentioned
in many of the titles this is only a small part of the topics covered.
The areas covered range from highly theoretical (Such as Effects in
Online Text) to very practical (How to Manage Educational Com-
puting Initiatives - Lessons from the First Five Years of Project
Athena at MIT), from experimental (Learning by Doing With Simu-
lated Intelligent Help) to implemented (From Database to Hyper-
text via Electronic Publishing: An Information Odyssey), from
hardware (Investigations in Multimedia Design Documentation)
to people (Consulting Skills for Technical Writers).

The first section was probably the most useful to me, in that it
opened my eyes to an area I hadn't thought about. As the title of
the section indicates, it is about the design of systems for online
users. The essays cover some of the history of online text retrieval,
but the majority of the section covers the current developments in
online text. Although this includes hypertext, it really discusses all
online text, including a comparison of the printed manuals and on-
line help, factors affecting the acceptance of online text, methods
of using hypertext for online text and interface design for text re-
trival systems.

The next section examines the current state of the art in hypertext
and hypermedia, giving numerous examples of the design and use
of these systems. It presents a number of competing views, from
those who suggest that there are big developments to come, those
suggesting that current techniques have much to learn, to those that
suggest that a simple text processing is sufficient for many applica-
tions.

The following section deals with the role of technical writers. It is
mainly of interest to technical writers, but includes an essay on how
Project Athena at MIT is managed which will be of interest to a
much wider audience.

The final section describes the results of a number of different
experiments and studies into online text. For example, it includes a
comparison of the effectiveness of a number of hypertext systems
based on the same original text and a discussion of a natural
language processor being developed at MIT.

After reading this book I am still at a loss describing who it is
aimed at. It certainly has information relevant to technical writers,
but as an application developer I found that it also has much of
relevance to me. It is also essential to researchers into online text
and related areas, and finally of interest to managers of any project
that involve technical writing (from manuals to online help to hy-
perlink systems). Ultimately many different groups will benefit from
reading appropriate essays from this book because as Lawrence
Levine states in Consulting Skill for Writers:

"Technical writer's primary expertise is need clarification and
problem solving in the learning and communication of
technical information."

and isn't that what we all need to do at some stage.

Frank Crawford
QANTEK

FUKUNAGA, Keinosuke (1990): Introduction to Statistical Pattern
591 pp., $59.95 (hardcover).

The main subjects of this book are statistical estimation and deci-
sion-making as applied to pattern recognition. It is intended to be
used as a text for introductory courses in pattern recognition and as
a reference for people working in this field. The material in this
book has been taught in a graduate course at Purdue University,
where the author is professor of Electrical Engineering.

As with most text books the material is presented in chapters
with sets of computer projects, exercises and references at the end
of each chapter.

The author starts off by an introduction to pattern recognition,
that is, given a set of observable characteristics the problem is to clas-
sify the object into one of a set of different classes. In general there
are n characteristics are being considered we have an n dimensional
vector. Further this vector is a random vector (X), as a number of
samples (or observations) are taken before a decision can be made.
In order to design a classifier in a general n dimensional space, we
must study the characteristics of the distribution of X for each cat-
egory, or class, and find a proper discriminant function (or a
boundary function which gives a separation between the categories).
The author states:

"Pattern recognition, or decision-making in a broader sense,
may be considered as a problem of estimating density func-
tions in a high-dimensional space and dividing the space into
the regions of categories or classes.

As probability density functions and parameters of distributions
of random vectors are of fundamental importance in the book, a
chapter is included which reviews mathematical statistics and linear
algebra. Topics such as moments, transformations, eigenvalues, ei-
genectors and matrices are covered.

Following the introduction, various classifiers together with error
estimations are presented. Initially the theoretically best classifier is
considered assuming random vectors whose distributions are given.
The Bayes classifier is the best classifier which maximises the proba-
bility of classification error. However its implementation is difficult
in practice. Parametric classifiers are considered in the next section.
In these methods the density function or discriminants are assumed
to be of a certain form and are expressed in terms of parameters such as expected vectors or covariance matrices. Linear, quadratic and piecewise classifiers are considered and methods for estimation of the unknown parameters are covered.

When no parametric structure can be assumed for the density function, nonparametric techniques such as the Parzen and k-nearest neighbour approaches for estimating the density functions must be used. In the following sections of the book the statistical properties of these estimates are given and applied to classification problems together with error estimation.

In the final three sections, successive estimation, feature extraction and clustering are considered:

- Rather than estimating parameters from all of the data, successive estimation is discussed, where we approximate the parameters by an initial guess and update by observing a subset of the data.
- Using a large number of features (or high-dimensionality) for classification results in pattern recognition problems being very difficult. To reduce the complexity of the problem the more important features have to be extracted and considered for the problem. In feature extraction a mapping from the n dimensional space to a lower-dimensional space is considered.
- Finally, clustering is discussed, where a given distribution is decomposed into a number of clusters.

A set of appendices is also included which cover mathematical formulas and other tables as needed by the body of the book. I will certainly be referring to this book in any further pattern recognition work I do. I’ve recently worked in the area of pattern recognition, and would very much have appreciated having this book at the time. The material is well presented and easy to follow, and I feel that the author certainly meets his aim in producing a good teaching and reference text. I enjoyed reading the book, but obviously it is not a book that one sits down and reads cover to cover without some practical work. Jagoda Crawford ANSTO


This book describes and discusses an interesting collection of parallel programming primitives.

A short Chapter 1 presents the need for parallel computation. Chapters 2 to 5 introduce the Paralation model informally, essentially by giving many examples. Chapter 6 gives a formal description of the model in Common Lisp. Chapter 7 deals with possible implementations on different parallel architectures. Chapter 8 discusses how the Paralation Model can deal with the problems of locality and communication which are important in distributed computing. Chapters 9 and 10 compare the Paralation Model with other models of parallelism. Three appendices follow giving a simplified code for the Model, and a glossary. The full code for the Paralation Model is available at extra cost, and it is not being reviewed here.

The book does not state who are the intended readers. It starts very slowly, explaining even the basic features of Lisp, but very quickly starts using a fairly sophisticated Lisp. Chapter 6 again starts by sketching various types of semantics before giving the operational one for the Paralation Model. The subtitle of the book describes the Model, fairly, as Architecture Independent. Although Common Lisp is used as a base language, language independence is stressed, and alternative base language C is briefly sketched.

All the new concepts in the book are introduced informally through examples, and I found it difficult to go through full half of the book without a more secure base than what can be deduced from examples. Still, this is my only real complaint as examples are well chosen, and most questions arising are eventually answered. Perhaps the comparisons with other parallel models could have been more extensive, and I would also like more detailed discussion of complexity. The book devotes only one paragraph to complexity issues, and tries to explain them by a single example with very little discussion.

Many books on parallelism stress architecture too much. This is not the case with this book, it talks to the programmer and the architecture is discussed only where necessary.

Although the book tends to give the impression that no background knowledge is needed to understand it, I think it is more suitable for senior undergraduates and postgraduate students of computer science. The book puts forward quite interesting ideas and I do recommend it for everybody interested in parallel computing.

I. Fris
University of New England


There is no branch of computer science with more popular appeal than computer graphics. Its wizardry has livened the entertainment and information industries, able on the one hand to portray the supernatural and surreal, on the other to portray increasingly complex realistic images of our world complete with subtleties like mirages and the gentle defocussing of a scene viewed through glass.

So how is it done? Short of actually attending a SIGGRAPH Conference, I could recommend no better way to find out than to read this book. It is based on a one day course designed for a mixed audience - hackers to graphic artists - and presented at SIGGRAPH '88. It consists of 9 chapters by various contributors, all solid figures in the field, with a lucid preface by the editor which summarises the book and signposts routes through the contents for differing reader's expertise. There is a 136-item bibliography and a glossary which manages to pick up most of the jargon introduced in the book.

In his clear prose, Andrew Glassner uses the introductory chapter to motivate the succeeding contributions. He gives a stylized trace of a light ray backward from the eye to the objects where it has been reflected/refracted, back to its source. He touches on why the physics of the photon-surface interactions will need to be addressed, and why the reader must think about which rays to trace, given that every ray can't be traced.

The second chapter by Eric Haines will bring the reality of graphics work home to the beginner: ray tracing involves a lot of pretty unexciting geometry, repeated over and over. Haines deals with the all important reflection of rays at object boundaries, in the process being careful of non-mathematicians to the extent of defining the discriminant of a quadratic. Chapter 3, by Pat Hanrahan, presents the more experienced through more complicated representations of objects and surface ray interactions, presuming a great deal more mathematical sophistication. The next chapter by Glassner expands on his introduction, describing colour and optics from an elementary beginning and going on to the 24 parameter Hall shading model of directional energy intensities. Robert Cook's chapter advances the reader through stochastic sampling which aids anti-aliasing and allows effects like motion blur and penumbrae to be simulated. It is a classic exposition. The last two chapters return to the essential nitty-gritty: Arvo and Kirk describe various methods for speeding up computation, with most emphasis on various hierarchical subdivisions of space to eliminate as early as possible computations which involve rays which do not intersect a particular surface. The last chapter by Paul Heckbert guides the reader through the actual writing of a ray tracer.

The book is very well presented, from its quality paper to its layout, with 16 colour plates offering a selection of 1988's finest graphics-refections, transparencies and (limited) natural objects. On the down side, there are the occasional definition and diagram missing, the occasional false statement, the occasional incomprehensible diagram, and there are the inevitable editing problems: repetitions between chapters which could have been weeded out, terminology not fully standardised eg. 'transparency rays' or 'transmitted rays'. Content wise, I would have liked to see a more leisurely treatment of object modelling than that provided by Hanrahan. After all, advances in graphics are largely because increased compute power allows more
rare to have been for many years attempts to use optics to perform
products: optical techniques have the theoretical edge on CMOS, ECL,
is the stuff of science fiction. For many years engineers have been
read for computer graphic students. And a fair portion of it is a
good read for anyone who has merely wondered ‘how do they do it?’
Janet Aisbett
Defence Science and Technology Organisation

FORD, J. (1990): Pascal by Example: An introductory course, NCC
Blackwell Ltd, 266pp., $32.95 (paperback).

Whether you teach or study Pascal (and/or TurboPascal), this is a
very useful text. If you have some experience with any other pro-
gramming language, and wish to teach yourself Pascal, this book
would be an excellent choice. It covers all the features of standard
(ISO) Pascal and TurboPascal, as well as the important and oft-
neglected art of structured program design.

For each concept, the author introduces a well-chosen illustrative
Pascal program, and the discussion of the example is the primary
means by which she conveys the concept. This has the advantage of
providing, right from the start, examples of good programming
style and the process of program design.

Ms Ford does not neglect syntax diagrams, which are used
throughout to reinforce (and perhaps correct) the syntax the student
might glean from the examples.

The discussion of examples is clear and to the point. There are
plenty of exercises, and answers to selected ones are provided.
Differences between standard (ISO) Pascal and TurboPascal are
pointed out at each relevant place in the text.

My only quarrel with the author is her choice of indentation
scheme. Statements bounded by begin and end are not indented re-
late the begin. Indentation schemes are, of course, a matter of
personal taste. I also feel that the end of a function or loop body
should be marked by a comment. I remain uncertain about whether
a semi-colon should be used before an end;. It certainly is easier for
the novice to treat the semi-colon as a terminator, but I find it clut-
ters the visual presentation.

In my opinion Example 2 below gives better visual clues to the
program structure than Example 1, which is part of the program on
p282 of the book.

Example 1.

Example 2.

function Cubic (x:real):real;
begin
  cubic := x*x*x-4*x;
end;
begin {main body}
  ScaleFactor := 0.15;
end {main body}

Despite my typographical preferences, I recommend this as a use-
ful and somewhat different contribution to the plethora of Pascal
textbooks.

Marshall Harris
University of Queensland


Optical Computing - holographic memory, 3-D laser circuitry, etc. -
is the stuff of science fiction. For many years engineers have been
searching for technologies with ever smaller speed-bandwidth prod-
ucts: optical techniques have the theoretical edge on CMOS, ECL,
and Josephson-Junctions, at least at room temperature. Therefore
there have been for many years attempts to use optics to perform
computation, both as analog and digital computers.

At first, optical computation was confined to the things light can
compute naturally - a form of analog computing. A lens performs a
two-dimensional Fourier transform in the time it takes for light to
traverse it; convolution and deconvolution are natural operations.
One of the best known commercial successes was the analog com-
puter for Synthetic Aperture Radar.

Now, people in many nations are attempting to design digital op-
tical computers. As yet, they haven't got very far. The Americans
seem to be the most obvious exponents of the art, but the Japanese
are doing good work.

This book is a collection of survey articles by various authors
outlining the state-of-the-art in Japan. It concentrates mostly on
digital techniques, but does cover a few analog systems, including
an analog speech processing system. It is a book that may be useful
to researchers in the field; it is not an introduction to the art.

To understand many of the articles, a working knowledge of opt-
tics is required. Moreover, most of the articles are too short to cov-
er their subject adequately: they follow the normal form for a sci-
etific paper, but in fewer pages. The shortness of the articles
means that their introductory paragraphs sometimes squeeze the re-
al content matter into a few paragraphs at the end. The book's
real value lies in the copious references to published literature, ra-
er than in the research described directly.

Amongst other things, articles cover:

- The fundamental limitations to optical computing - the
  photon limit, quantum effects, etc.
- Descriptions of various logic systems - shadow logic, in-
terference logic, etc. - and how to build logic gates
  using them.
- Elementary Neural Networks implemented using some of
  these logic systems.
- Chip-to-chip optical interconnection, and optical busses
  for multiprocessor work.
- Optical devices, including electro-optic bistable devices,
lasers, modulators, etc.

Because the literature on Optical Computing is so widely distrib-
uted through the journals, and is difficult to find, Optical Compu-
ting in Japan will find a place on the shelves of researchers in the
field. For a gentler introduction, however, Feitelson's survey (1988)
is probably more appropriate.

References.

puter Scientists, MIT Press

Peter Chubb
Softway Pty. Ltd.


This book contains 12 papers and 2 panel discussions from a spe-
cialist Workshop held at Rutgers University in April 1989. Many
papers describe new ideas for solving problems in traditional appli-
cations areas, such as industrial inspection and defect detection, 3-
D distance and motion estimation, and 3-D object recognition from
features. Some of these are quite relevant to work being done at
present in Australia, for instance the stripe rangefinding techniques
of Alexander and Ng at Monash University. Several papers embody
the theme of extending 2-D techniques to suit 3-D situations, e.g.
generalisation of the Sobel operator, and interpretation of 3-D
medical data.

As with many books based on selected conference papers, this
book contains only 12 papers and 2 panel discussions from a spec-
cialist Workshop held at Rutgers University in April 1989. Many
papers describe new ideas for solving problems in traditional appli-
cations areas, such as industrial inspection and defect detection, 3-
D distance and motion estimation, and 3-D object recognition from
features. Some of these are quite relevant to work being done at
present in Australia, for instance the stripe rangefinding techniques
of Alexander and Ng at Monash University. Several papers embody
the theme of extending 2-D techniques to suit 3-D situations, e.g.
generalisation of the Sobel operator, and interpretation of 3-D
medical data.

As with many books based on selected conference papers, this
book contains only 12 papers and 2 panel discussions from a spec-
cialist Workshop held at Rutgers University in April 1989. Many
papers describe new ideas for solving problems in traditional appli-
cations areas, such as industrial inspection and defect detection, 3-
D distance and motion estimation, and 3-D object recognition from
features. Some of these are quite relevant to work being done at
present in Australia, for instance the stripe rangefinding techniques
of Alexander and Ng at Monash University. Several papers embody
the theme of extending 2-D techniques to suit 3-D situations, e.g.
generalisation of the Sobel operator, and interpretation of 3-D
medical data.

As with many books based on selected conference papers, this
book contains only 12 papers and 2 panel discussions from a spec-
cialist Workshop held at Rutgers University in April 1989. Many
papers describe new ideas for solving problems in traditional appli-
cations areas, such as industrial inspection and defect detection, 3-
D distance and motion estimation, and 3-D object recognition from
features. Some of these are quite relevant to work being done at
present in Australia, for instance the stripe rangefinding techniques
of Alexander and Ng at Monash University. Several papers embody
the theme of extending 2-D techniques to suit 3-D situations, e.g.
generalisation of the Sobel operator, and interpretation of 3-D
medical data.
The three papers of greatest interest to me were in fact by the least widely known authors. Their appeal was due both to the innovative nature of the ideas and to the clarity of expressing the motivation and technical details for those ideas. A Framework for 3D Recognition (Bolle & Cali)ano emphasised the importance of feature abstraction and multiresolutional data for understanding scenes in a general and modular way. The Free-Form Surface Matching Problem (Beres) provided a lengthy discussion of the difficulties of this process and suggested a novel maximum-likelihood approach for establishing surface curvature and correspondence properly. Occlusion-Free Sensor Placement Planning (Tsal & Tarrabian) offered a solution to the problem of sensor placement for multiple object visibility, using convex decomposition of object space.

The Workshop origins of the papers have inspired an explanatory style and an enthusiastic tone from all the authors and the result is a book more readable than many journal papers on similar topics. The context of appropriate problems has no doubt enhanced the understanding obtained by all but the most specialised of readers.

Anthony Maeder
Monash University


Immediately striking about this book is the list of contributing authors, which in itself suggests that this is top quality stuff. Many of the world's foremost authorities in a reasonably wide range of fields have been gathered together in this volume in which theoretical and empirical knowledge relevant to the design of pilot performance models are outlined and reviewed. The book is the product of a study conducted and directed by the Panel on Pilot Performance Models for Computer-Aided Engineering (CAE) formed by the U.S. National Research Council (NRC). The objective of the study was to review current models of human performance so as to identify the most useful of these for CAE facilities. The book focusses upon the visual and associated cognitive functions required of pilots in the operation of advanced helicopters. The work aims of the study explain the division of the material into vision- and cognitive models, a mixture which may otherwise seem slightly curious. In addition to reviewing particular classes of models, the book also provides a framework for designing, testing and selecting suitable models from a number of different possible ones. Overall findings/recommendations of the study as such are presented last.

Each of the 23 Chapters portrays the state-of-the-art of a particular area of Human Factors research. Typically, the relevant features of the most important models are reviewed critically, drawing attention to the advantages, constraints and limitations of each, thereby giving an accurate account of what might be gained from applying a given model to a particular problem. This approach is, however, not adopted in all Chapters. Stu Card, for example, in his review of models of working memory divides some 32 phenomena into seven categories addressing different aspects of memory functions. He then uses these categories as a basis on which to review and summarise the literature in a manner that enables empirical findings to be presented as a list of useful guidelines. Along similar lines, Card's Chapter on modeling scenarios for action provides many useful hints both for design and research as well as reviewing the literature. The Chapter I enjoyed the most was modeling and predicting human error by David Woods, addressing an area that has so far proven too difficult to model. Woods defines what errors are, identifies sources of errors and gives many ideas for ways in which to tackle research into the nature of human error. The vision model Chapters explore various aspects of human vision, and note the implications for, and limitation of, each for cockpit design. My own bias towards the cognitive Chapters in favour of the vision model Chapters reflects my research interests rather than the quality of the papers presented.

The entire volume constitutes a wealth of reference materials, giving key references and source materials and pointing to directions for Human Factors research throughout. Yet, although the book concentrates on models in those areas directly relevant to human performance in pilots, and hence in cockpit design, the content is equally relevant to researchers and designers of other complex interactive systems. For the researcher, it offers direction and focus of research; for the designer, it offers questions to be asked, issues to be raised, in the design process. Through critical review and analysis of models, the volume also provides a healthy dose of scepticism towards the simplistic belief that optimal conditions for perfect human performance can be readily attained by adhering to certain principles and/or guidelines alone.

If I were a project leader in large design/development teams of complex interactive systems that are demanding in the resource allocation of human operators, I would regard this book as an absolute must on the team reading list as well as on my own library shelves. As a HCI researcher, I would certainly want access to it, although perhaps more as a general library resource instead, due to tight purse strings which do not allow me to go much beyond the needs dictated by my immediate research. This does not currently involve models of human vision. Expectations raised by the first glance at the list of world-class contributing authors were certainly borne out whilst reading the book, and at the price, it most definitely is a volume well worth having for Human Factors researchers as well as designers/developers of large interactive systems.

Güte Lindgaard
Telecom Australia Research Laboratories


This book is a collection of scientific papers presented at a conference held in Antibes in September 1989. There are 9 invited papers and 41 contributed papers, each around 10 pages long, mostly written by French scientists. As the book title suggests, a broad range of topics is covered, some with a predominantly 'statistical' flavour (e.g. factor analysis, clustering algorithms, data smoothing techniques, time series analysis), others with an 'artificial intelligence' flavour (e.g. knowledge acquisition, concept learning).

The focus of the conference was to bring together workers from different schools of thought with interests in similar problems, in the hope that the different approaches may enrich one another. Whether or not this aim was achieved cannot be judged from the book. Each of the articles is a brief account of the authors' current work, seen from their current perspective and new approaches to future work will only be evident later. The techniques discussed are 'modern' and presented without marketing hype but, as the papers are primarily statements of technical results, they are not self-contained accounts of the technology and certainly not suitable reading for the faint-hearted.

For readers of this journal the book is not without interest. Through its wide coverage of topics it exposes techniques that the reader may be unfamiliar with. This may help to decide if such techniques are potentially applicable to their data analysis problems.

In summary, if this book is in your library then definitely browse through it. If it is not in your library and you are actively involved in theoretical aspects of data analysis, then consider recommending it for purchase.

Liz Sonenberg
The University of Melbourne


Considering the number of books already published on ALGORITHMS, it is difficult to justify writing yet another. However, I believe that Introduction to Algorithms turns out to be a truly valuable addition to the textbooks on this subject.

As well as an introductory section, this book is divided into seven further sections. Part one is a mathematical foundation from which
most students will considerably benefit. This section covers in reasonable depth the notions of the growth of functions which are used for describing the asymptotic running time of algorithms; summations; recurrences; concepts related to sets, relations, functions, graphs, and trees; and elementary probability theory. Part two deals with two problems, namely, sorting and selection. Part three covers some common data structures which are often required to design algorithms. Part four is concerned with some techniques for designing efficient algorithms and amortized analysis. Part five deals with advanced data structures including B-trees, and heaps. Graph algorithms are considered in part six, where algorithms for graph searching; finding minimum spanning trees, and shortest-paths; and the maximum flow problem are presented. Part seven concentrates on some advanced selected topics including parallel algorithms, matrix algorithms, algorithms for FFT and string matching, computational geometry, NP-Completeness, and approximation algorithms.

The book is an excellent treatise on each of the topics covered in these seven sections. Although the authors cover a wide range of problems and present algorithms for them, they take sufficient care to discuss every algorithm in considerable depth including a proof of correctness in as many cases as possible. The reader should agree with the authors’ claim that the ‘book is designed to be both versatile and complete’. The book, in fact, covers much more material than can be taught in a one-semester course on algorithms. This book can be used as a text for an Algorithm course as well as being a useful reference for a course on Data Structures.

In summary, this well presented book may be recommended as a text on Algorithms for undergraduate classes. It contains over 900 exercises and over 120 problems. The price of the book (US$49.95) is quite reasonable, and at this price it is worth acquiring.

Pranay Chaudhuri
University of New South Wales


Adaptive User Interfaces is a record of the four-year Adaptive Intelligent Dialogues project in the UK Alvey program. Adaptive User Interfaces (AUIs) are defined as those that change automatically in response to experience with users. AUI technology is of great interest to designers of interfaces that will receive a large and varied audience.

The book goes through the motivation, theory, methodology, techniques and evaluation of adaptive systems. It is an excellent starting point for researchers who wish to create adaptive interfaces and will cross many of the same hurdles the authors did. Every designer of advanced interfaces should have a copy on the shelf.

The book is a monograph and the material assumes a basic understanding of user interface terminology. It is thus aimed at the expert rather than the novice. References to relevant work in the supporting disciplines are excellent.

Ken Yap
CSIRO Division of Information Technology


This is an excellent book written for experienced practitioners of SSADM. It is also useful in general to software engineers who are experienced in other structured methodologies. It assesses the various features of SSADM, compares it with other approaches, and proposes enhancements and alternative uses of the method in areas where it was not originally designed.

The book is divided into five chapters. A review of the evolution of structured methodologies is presented in Chapter 1. The strengths and weaknesses of SSADM are discussed in Chapter 2. Chapter 3 presents readers with a bag of ‘pearls of wisdom’, or useful advice on how to handle SSADM in various practical situations. In Chapters 4 and 5, extensions of SSADM are suggested in real-time systems, distributed systems, conversational systems, expert systems and object origraphy is provided in the book. Readers who would like to take the advice of the author and venture into expert systems, for example, will not know where to read further.

The title of the book may suggest that its main objective is to present the advanced features of SSADM, rather than to provide an assessment of the method. Perhaps a subtitle is suggested in its second edition.

The subsection headings (down to the fourth level) look uncomfortably similar in many places. It would be useful if the major section heading could be printed at the top of every right hand page. In this way, the reader can easily distinguish those pages that cover the strong points of SSADM from those that cover the weaknesses. He can find out at a glance whether we are discussing real-time, distributed or conversational systems, so that he does not have to flip-flip between the table of contents and the main text.

In spite of the shortcomings, this book is strongly recommended to all practitioners who have a fair amount of exposure to SSADM or other structured methods and who would like to increase the breadth or depth of their own experience.

T.H. Tse
University of Hong Kong


This book is basically a collection of papers derived from the Second European Natural Language Generation Workshop held in Edinburgh in April 1989, but revised for this edition. For students of, or active researchers in, this field it is an interesting and timely collection, but due to its highly technical treatment, it is certainly not a book for the casual reader or one interested in obtaining introductory overview of natural language generation (NLG).

The book is organised around four main themes, each of which would interest workers or students in the field. The first section contains 11 papers by many of the world’s leading researchers in NLG. The book begins by looking at two issues that represent a hierarchical approach to NLG: firstly, planning what to say and how the text should be organised, followed by deciding how to actually say it. While this is the traditional approach, each paper views the problems in new ways. Of particular interest are the two papers on multi-modal interaction.

The third main theme is about generating text that is organised ‘naturally’ with a natural use of expressions that refer to other parts of the text. An interesting paper by Reiter focusses on the problem of generating text that is appropriate to the needs of the reader for information.

The last theme of the book is in many ways the book’s greatest attraction. Here, two papers describe new approaches to NLG. The first describes the learning and recall of phonemic (sub-word) sequences by a connectionist network; the second describing the simultaneous understanding and generation of language in a simultaneous Japanese-English speech interpreting system. Both papers are ambitious and innovative.

In summary, the book is highly technical, but interesting selection of papers describing the state of the art in natural language generation. It would be of particular interest to students of the field as much as active researchers. For the casual reader, its lack of introduction to the field or historical perspective would probably prove to be daunting.

Chris Rowles
Telecom Research Laboratories


This book is an excellent introduction to numerical analysis for engineers and science students. To understand this book readers need to have covered a course in calculus and algebra up to 2nd year undergraduate level. They would not have had to done any programming to be able to read this book. It is a book for the practicing engineer who is interested in understanding the theory in
Numerical Analysis: not as a ‘cookbook’ for finding program source listings.

The book has many items in its favour compared to others in its genre. Many authors on this subject concentrate on the programming aspects and are too hardware or software orientated. This tends to leave the students confused and they miss the point by concentrating on the programming aspects rather than the mathematics involved. (Conte and De Boor (Elementary Numerical Analysis (2nd ed.)) is an example where the emphasis is on writing the code not the underlying mathematics). This book makes a brief reference to the hardware in the opening chapter by quoting IEEE standard for floating point arithmetic, and mentions its implementation in the INTEL 8087 and MOTOROLA 6881. This is the closest it comes to being hardware specific. By concentrating on the mathematical aspects of numerical analysis the book will be around for a while and not outdated.

The book covers enough material for a semester course in Numerical Analysis, beginning with errors and computer arithmetic through functions, nonlinear equations, interpolation, differentiation, integration, systems of linear equations, approximations and differential equations.

Knuth in his preface to Algorithms (The Art of Computer Programming) speaks about the importance of the reader doing the exercises to have a proper understanding of the material.

"It is difficult, if not impossible for anyone to learn a subject by reading about it, without applying the information to specific problems ..."

Good pedagogical textbooks have exercises with answers, as does this one at the end of each chapter. All the exercises require only pencil and paper, hence students can work independently. At the end of each chapter there are references which may be of interest to students who want a more extensive treatment, however, I found the text sufficiently self contained.

Don’t be put off by the names of the authors, or the fact that it was translated from an original Swedish text. The style of writing is concise, easy to read and clear.

At $39.95 hardcover with 340 pages, it is a book which is an extremely affordable to student, or as a useful reference to the practicing engineer who wants more than just a cookbook in Numerical Analysis.

Peter Radonyi
UNENR


Although the title of this book creates a guessing game about its contents and the author with-holds any explanation of who should read it and why, this book will nevertheless be very useful to anyone wanting to become, or currently practicing as, a manager in any computer-related field. The book is particularly relevant for information systems planning, design, implementation, operation and control. It is up-to-date, gives a mercifully brief description of hardware developments and hardware issues that matter (e.g. data communications, distributed processing) and contains a very good glossary and bibliography. The emphasis is on information systems within large private or public sector organisations.

One of the advantages of the book is that it places DP as a service function within an organisation and follows through on this to describe the role of the user in system development, the importance of planning, the importance of quality and the relevance of standards. Computer-based systems are related to users’ jobs in a brief chapter on motivation and satisfaction. Although human needs are mentioned, the book is stronger in describing various types of computer support that may be familiar terms to managers yet not understood by them. For example, there are useful chapters on Decision Support Systems and Expert Systems.

Sachdeva’s book is straightforward, concise and efficient in its presentation of information. It is well organised and each chapter can be read on its own. As such, it is a good reference book for bringing a manager up-to-date on a particular topic. It appears to espouse no particular philosophy and is quite rational (or dry) in style and, by implication, Sachdeva could be said to view the use of Information Technology as a completely rational process. Of course it is not, but the reader has to translate the information in this book into the environment of his or her own organisation.

I don’t think the book really pushes the state-of-the-art and as such would not use it for teaching computing to computing students. It does not provide a theoretical framework or any particular base for the development of new ideas. However, that is not its purpose. As a reference text for managers I would certainly buy it and think it is very good value at $A39.95.

P.A. Collings
University of Canberra


This book contains a collection of papers from a IB Summer School on Advanced Programming Methodologies, IE held in Italy in September 1987. The aim of the school, and of the book, was to inform participants about new methodologies, languages and tools for advanced programming, especially object-oriented programming. Apart from two American and two Polish authors, all the authors were Italian. The whole volume is in English, some of it in fairly poor English, but only occasionally bad enough to obscure understanding.

About half the papers are survey articles, covering topics such as database languages, data abstraction and modularity in LISP, an overview of object-oriented programming and implementation techniques for storage management and module support.

Two other papers introduce little known programming languages: .IB Loglan ’82 .IE and .IB Paragon .IE Loglan supports abstract data types, inheritance, coroutines, processes and exceptions and is based on a theory of algorithmic logic similar to Dijkstra’s weakest preconditions. Paragon is also object-based, but restricts inheritance so that the exact type of all objects is known at compile time. This reduces flexibility (no subtype polymorphism) but gives a safer and more efficient language than a full object-oriented language.

One of the most interesting papers is about a technique called .IB lambda abstraction .IE (or .IB higher order generalization) .IE for deriving functional programs. The author shows that it can be used to simulate pointers, avoid multiple traversals of input data and avoid local recursion. And all this within a call-by-value language. He claims that lambda abstraction often makes alternative transformation techniques such as lambda lifting, supercombinators and lambda hoisting unnecessary. Another paper proposes a new implementation technique for logic programs. It uses relational algebra operators and has a more declarative semantics than the usual execution model of Prolog.

I was disappointed with much of this book. The title is promising, but many of the papers fail to live up to the promise of .IB advanced .IE programming methodologies. Researchers working in these areas will probably find little to interest them, and the .IB Computing Surveys .IE journal is a source of better survey papers than those in this book.

Mark Utting
University of New South Wales
NEWS BRIEFS

"News Briefs" is a regular feature which covers local and overseas developments in the computer industry including new products and other topical events of interest.

MACQUARIE UNIVERSITY AND MICROSOFT INSTITUTE JOIN FORCES TO RAISE LEVEL OF AUSTRALIAN IT RESEARCH

Macquarie University and the Microsoft Institute of Advanced Software Technology have joined forces in a unique collaboration designed to significantly improve the level of computer-related research and training conducted in Australia.

The agreement between Macquarie University and the Microsoft Institute was formalised at a special ceremony on 7 May. It is a world first for Microsoft and integrates the work of an international IT company into a major university. This integration will also include the CSIRO as a research partner.

According to Vice Chancellor of Macquarie University, Professor Di Yerbury, the agreement with the Microsoft Institute creates a research association with all the necessary ingredients to achieve new levels of research and to ensure the results of the research are quickly adapted for commercial use.

"Our existing relationship with CSIRO has led to the creation of a Joint Centre for Information Technology Research. The prospect of this centre working with the commercial world's largest software company is extremely exciting for Australia and has massive potential," Professor Yerbury said.

"Apart from the national benefits, the agreement will provide students at the university with access to a far wider range of software tools, involvement in Microsoft's internal research projects in areas such as natural language and multimedia, high-level work experience for Bachelor of Technology students, a stronger understanding of commercial operations and the ability to transfer information through seminars and lectures."

Director of the Microsoft Institute, Professor Vance Gledhill, said the agreement would help the Institute address the three objectives it was established to achieve: improve Australian software development, create new software products and provide access to international markets when a product is developed.

"This agreement, and the involvement of the CSIRO, will improve the level of Australian research through the sharing of resources, expose students to more relevant and competitive research as well as provide a vehicle for Australian ideas to reach international markets," Professor Gledhill said.

"We have established a research fund to provide a number of Microsoft Fellowships while access to Macquarie University staff will enhance the Institute's reputation in the region. It is also anticipated that staff from Microsoft Research Laboratories in the US will spend time at the Institute which will further assist the transfer of technology."

Director of CSIRO's Institute of Information Science and Engineering, Dr Bob Fraser, described the agreement as "an important extension to CSIRO's collaboration with Macquarie University."

"In the IT industry, it is essential that researchers are aware of commercial requirements, while it's also in the interests of commercial organisations to be able to capture the benefits of successful research," Dr Fraser said.

"This relationship will facilitate the process of resource sharing and improve Australia's ability to pick up commercial opportunities in the highly competitive international computer market."

SYSTEMS 91: ONE DAY LONGER

In an effort to provide those who would normally find it difficult to visit the trade fair during the week with an opportunity to visit SYSTEMS, the Munich Trade Fair Corporation and the exhibitor's advisory board have agreed to extend this year's trade fair for computers and communication by one day to include Saturday. Thus SYSTEMS 91 will begin on Monday, 21 October and run until Saturday, 26 October 1991.

The sixth fair day is intended to address target groups whose members are tied to their place of work from Monday to Friday.
HOMER software in the future, to potential in expanding its modules of areas and contribute to the ongoing specific user needs in their respective and finance. These groups identify ing areas such as nursing, supplies, thea­

dures with the use of specialised company's Series 18 computer. Admis­

sions, finance, nursing and radiology are active in national user groups, represent­

ems of the hospital information systems. McDonnell Douglas Information Sys­

was found computerised technology a key investment.
The Royal Victorian Eye & Ear is a 139-bed hospital which provides medi­cal, paediatric and surgical services in the fields of eye, ear, nose and throat to its patients. It is the only hospital of its kind in Australia and has earned a worldwide reputation for its specialised areas of out-patient and in-patient care.

Through continued upgrades of both hardware and software, the Royal Vic­torian Eye & Ear has shown its commit­ment to computerised efficiency in such far reaching areas as appointment scheduling, investment management, financials, medical records and pharmacy.
The Royal Victorian Eye & Ear was the first hospital in Australia to actually beta test the Reality operating system developed by McDonnell Douglas. The hospital found the system was well suited to its needs and is the first Australian hospital to run the newly released Reality operating system version 7.0.

Because the Reality operating sys­tem is a fully integrated system, the benefits are realised by executives, department heads and hospital employees alike. Strengths of its new 7.0 version include users’ immediate access to database information, and specialised functions, such as quick and efficient report generation. With input of a brief description of the parameters of data required, a report is produced presenting all information available from the database.

Many departments throughout the hospital have been equipped with McDonnell Douglas Information Sys­

tems' HOMER software, running on the company’s Series 18 computer. Admis­sions, finance, nursing and radiology are only a few of the departments which have upgraded their operating proce­dures with the use of specialised modules of the hospital information management system, HOMER.

Many hospital staff members are active in national user groups, represent­ing areas such as nursing, supplies, theat­res and finance. These groups identify specific user needs in their respective areas and contribute to the ongoing developments of the HOMER products.
The Royal Victorian Eye & Ear sees potential in expanding its modules of HOMER software in the future, to include the areas of food services, theatre management and clinical costing.

---

**Data Security courses to meet your needs**

Data security is emerging as one of the most significant aspects of information technology in the 1990s as computer and communications systems become increasingly complex and vulnerable.

QUT’s Information Security Research Centre – a national leader in the field – offers a modular program of courses to cater for a range of training needs in data security, from one/two-day special topic seminars to formal qualifications. The one/two-day seminars are the building blocks for one-week short courses which are, in turn, building blocks for a formal award: Graduate Certificate in Information Technology (Data Security). The program is designed to meet requirements of the Training Guarantee Act and fees are tax deductible for organisations.

### One/two-day seminars

Each seminar is on a discrete topic. Participants can register for any one or more seminars. No prerequisites are necessary. The fees include lunch and course notes.

The second series of one/two-day seminars is scheduled as follows:

#### Communications Technologies and Network Management
- 6 July Understanding the Basics of Data and Computer Communications ($290)
- 9 July Network Architecture ($180)
- 10 July High Speed and Local Area Networks (LANs) ($180)
- 11 July Value Added Networks (VANS) and Communications Equipment ($180)
- 12 July Network Management and Control ($180)

#### Network Security, Performance and Risk Management
- 15-16 July Network Security Architectures and Technologies ($360)
- 17-18 July Network Performance ($360)
- 19 July Risk Management in Communications ($180)

### One/two-week short courses

Each one-week course comprises one-day and two-day seminars (at a discounted cost) which together form a major topic. The short courses are for staff with specific data security responsibilities, or for educators and trainers within the field.

#### Short courses for 1991/92:
- 8-12 July Communications Technologies and Network Management ($800)
- Nov-Dec Risk Management
- March 1992 Management Issues
  - Technical Issues

#### Graduate certificate ($5500)

The graduate certificate requires participation in all six one/two-week courses together with satisfactory performance in assessment and completion of a project. Candidates will normally be graduates with some experience in data security. Non-graduates with extensive experience may be considered.

The graduate certificate may be credited as half a graduate diploma or a quarter of a masters degree if students want to continue.

For more information about these programs, phone David Hall at QUT's Office of Educational Services on (07) 864 2196 or fax (07) 221 0313.

---

**QUT**

A university for the real world

Queensland University of Technology Gardens Point campus:

2 George Street GPO Box 2434 Brisbane Q 4001