South African
Computer
Journal
Number 21
August 1998

Suid-Afrikaanse
Rekenaar-
tydskrif
Nommer 21
Augustus 1998

Computer Science and Information Systems

Rekenaarwetenskap en Inligtingstelsels

The South African Computer Journal
An official publication of the Computer Society of South Africa and the South African Institute of Computer Scientists

Die Suid-Afrikaanse Rekenaartydskrif
'n Amptelike publikasie van die Rekenaarvereniging van Suid-Afrika en die Suid-Afrikaanse Instituut vir Rekenaarwetenskaplikes

World-Wide Web: `http://www.cs.up.ac.za/sacj/`

Subscriptions

	Annual	Single copy
Southern Africa	R50.00	R25.00
Elsewhere	US$30.00	US$15.00

An additional US$15 per year is charged for airmail outside Southern Africa

to be sent to:

Computer Society of South Africa
Box 1714, Halfway House, 1685
Phone: +27 (11) 315-1319 Fax: +27 (11) 315-2276

Cultivating the information systems discipline

Niek du Plooy, Sub-Editor: Information Systems

Whether by 'information system' we mean a simple bookkeeping system for a small business, or a monolithic integrated 'management information system' for a global corporation, all organisations currently need information systems in order to function effectively. The computer and business community at large have readily adopted and accepted the use of the term 'information systems', but perhaps without too much real thought being given to any more profound meaning of the term. Departments bearing that name (or something very similar) are commonly found in organisations. But can the same be said for the 'academic' use of the term, as in describing the information systems discipline? Has it been 'accepted' as a separate scientific discipline?

The term 'discipline' is often loosely applied to indicate the scientific 'field', that is, the organised 'body of knowledge' or 'domains of discourse' within which (mainly) academic activities concerning a specific topic or a number of related topics, are conducted. [3] point out that a scientific discipline has a certain paradigm associated with it, meaning that researchers in that discipline are familiar with the research topics, the research methods and the accepted ways to interpret the results in their chosen field. A discipline is further strengthened and consolidated by the educational process whereby a researcher becomes a practitioner in that discipline, initially through the pursuit of academic degrees and thereafter, through recognition amongst his/her peers. Formal study in a particular discipline results in the value sets and exemplars (the 'paradigm') of that discipline being adopted by the student, either consciously or unconsciously.

Is 'information systems' truly a recognised scientific discipline such such as this? In the past, prominent authors such as Peter Keen did not think so [15, 16]. He deplored the lack of a cumulative tradition and advocated that one be built up, asking for a clarification of the reference disciplines of this new science and a definition of its dependent variable and the building of a cumulative tradition, amongst other things. [1] however, disputed Keen's position and pointed to strong links between research and practice found in their analysis. [11] showed clearly that 'orthodoxy' exists in many aspects of information systems, i.e. in information systems methodologies as well as in other areas of information systems development. This claim was supported by [13] who, in a detailed study based on papers in scientific journals, scientific conferences and textbooks, identified seven different but complementary 'schools of thought' within the field of information systems. In a study of leading universities and leading researchers in decision support systems, [9] provide exemplars, at least for that particular sub-discipline. [5] conclude from a citation study of journal influence during the period 1981 through 1985, that the discipline of information systems has attained stability and that it is in no danger of dying. It seems therefore, that Keen's despair is unfounded and that information systems have indeed grown into a separate, identifiable discipline, even if the field is best described as a 'fragmented adhocracy' ([3]).

The existence of an established scientific community in information systems has been given formal recognition by the recent formation of the Association for Information Systems, a professional society in the tradition of scientific societies, with 1400 members in 35 countries. A recently compiled directory of information systems academics contains entries on some 4,500 researchers from more than 1,000 institutions. A number of basic University and other curricula for information systems education have been published over the years [2, 6, 18]. The most recent of these is Curriculum '95, a joint effort by the ACM, AIS, DPMA, IAIM and ICIS [10, 7]. The most popular discussion group on the Internet (ISWorldNet) devoted exclusively to information systems matters has a membership which in 1997 approached 1829 from 53 countries [14]. A well-defined scientific community therefore exists.

In addition, if the existence of sound academic scholarship is further testimony to the existence of a 'discipline', then information systems can proudly point towards a dramatic growth over the past three decades in the number of scientific journals reporting on research in this area [12]. An even more recent study on research outlets showed that, amongst twenty-seven established journals carrying articles in this field, at least three of the most highly rated top ten are devoted exclusively to the discipline.

Yet, can it be said that the information systems discipline has been conclusively defined and that the research problems and research methodologies prescribed for it have been accepted by all who consider themselves to be working in this field? A re-examination and extension of an earlier (1988) list of keywords for use in classifying information systems literature [4] includes a list of the reference disciplines of information systems, as well as lists of the external environment, the technology, the organisational environment, etc., of information systems. We could argue that this very comprehensive list of keywords (nearly 1300) and other classifications define and describe the discipline of information systems accurately and usefully. For instance, the reference disciplines were listed as:

behavioural science, computer science, decision theory, information theory, organisation theory, management theory, language theories, systems theory, research, social science, management science, artificial intelligence, economic theory, ergonomics, political science, psychology. This list reflects the interdisciplinary or pluralistic nature of information systems.

In the same vein, [19] did a study on the themes of submissions to the journal Information Systems Research and produced a list of keywords, concepts and associations that characterise the categories into which they grouped the research questions of articles submitted. This list demonstrates conclusively that the subareas of the discipline (organisational, behavioural and managerial issues) are well established and attract a large number of researchers on a long-term basis. Swanson & Ramiller conclude by observing that the discipline still exhibits the 'fragmented adhocracy' identified by Banville & Landry, and is still topically diverse and '...based on appeals to significantly different and partly incommensurate reference disciplines'.

Thus, fragmentation can have adverse effects – something that information systems researchers should be aware of. However, fragmentation of the discipline of information systems may be evident in the field for a very long time. As has been pointed out [17, 8], the discipline as a whole follows trends in information technology, and researchers tend to build their interests around new technology (e.g. the earlier interest in expert systems and decision support systems, and current interest in computer-supported co-operative work). As information technology evolves, so the research interests will follow these new directions. Although we may wish it were different, it remains a fact that information technology is still a major reference discipline of information systems, and will remain so as long as researchers struggle to separate the fundamental or common issues in different fields from the technological ones.

Clearly, then, information systems is internationally well-established as a flourishing discipline. In the Southern African context it is important that the discipline should not merely flourish but be seen to flourish. To this end, this editorial calls on academics and especially on practitioners to add your contributions, via a submission to SACJ.

References

[1] M. Alavi and P. Carlson. A review of mis research and disciplinary development. *Journal of Management Information Systems*, 8(4):45–62, 1992.

[2] R.L Ashenhurst. Curriculum recommendations for graduate professional programs in information systems. *Communications of the ACM*, 15(5):364–398, 1972.

[3] C. Banville and M. Landry. Can the field of mis be disciplined? *Communications of the ACM*, 32(1):48–60, 1989.

[4] H. Barki, S. Rivard, and J. Talbot. A keyword classification scheme for is research literature: An update. *MIS Quarterly*, pages 209–226, June 1993.

[5] R.B. Cooper and M. Blair, D.and Pao. Communicating mis research: A citation study of journal influence. *Information Processing & Management*, 29(1):113–127, 1993.

[6] J.L. Couger. Curriculum recommendations for undergraduate programs in information systems. *Communications of the ACM*, 16(12):727–749, 1973.

[7] J.L. et al. Couger. Guidelines for undergraduate is curriculum. *MIS Quarterly*, 19(3):341–360, 1995.

[8] P. Ein-Dor and E. Segev. A classification of information systems: Analysis and interpretation. *Information Systems Research*, 4(2):166–204, 1993.

[9] S.B. Eom and S.M. Lee. Leading us universities and most influential contributors in decision support systems research (1971-1989). *Decision Support Systems*, 9:237–244, 1993.

[10] J.T. et al. Gorgone. Information systems '95 curriculum model — a collaborative effort. *Data Base*, 25(4):5–8, 1994.

[11] R.A. Hirschheim, J. Iivari, and H. Klein. An assumption analysis of the five emergent approaches to information systems development (isd): A contribution to the paradigmatic foundations of alternative approaches to isd. Working paper, Binghamton, 1995.

[12] C.W. Holsapple, L.E. Johnson, H. Manakyan, and J. Tanner. Business computing research journals: A normalized citation analysis. *Journal of Management Information Systems*, 11(1):131–140, 1994.

[13] J. Iivari. A paradigmatic analysis of contemporary schools of is development. *European Journal of Information Systems*, 1(4):249–272, 1991.

[14] B. Ives. Posting to `listserv.hea.ieon` 21 July 1997.

[15] P.G.W. Keen. Mis research: Reference disciplines and a cumulative tradition. In *Proceedings of the First International Conference on Information Systems, Philadelphia*, pages 9–18, 1980.

[16] P.G.W. Keen. Relevance and rigor in information systems research: Improving quality, confidence, cohesion and impact. In H-E. Nissen, H.K. Klein, and R.A. Hirschheim, editors, *Information Systems Research: Contemporary Approaches & Emergent Traditions*. North-Holland, Amsterdam, 1991.

[17] K. Lyytinen. Information systems and critical theory. In M. Alvesson and H. Willmott, editors, *Critical Management Studies*. Sage Publications, London, 1992.

[18] J. Nunamaker, J.L. Couger, and G.B. Davis. Information systems curriculum recommendations for the 80s: Undergraduate and graduate programs. *Communications of the ACM*, 25(11):781–805, 1982.

[19] E.B. Swanson and N.C. Ramiller. Information systems research thematics: Submissions to a new journal, 1987-1992. *Information Systems Research*, 4(4):299–330, 1993.

A Performance Analyser for the Numerical Solution of General Markov Chains

William Knottenbelt[a] Pieter Kritzinger[b]

Data Network Architectures Laboratory, Department of Computer Science, University of Cape Town, Rondebosch 7700, South Africa
[a]wjk@doc.ic.ac.uk, [b]psk@cs.uct.ac.za

Abstract

Despite many advances in queueing theory and other modelling paradigms, one persistenty discovers real life stochastic systems which do not yield neatly to existing methods for their performance analysis. In most such cases, the only alternative, other than simulation, is to resort to modelling the process as a Markov chain and to solve that. The immediate problem which presents itself, however, is the very familiar one of state-space explosion. In this paper we present a new probabilistic dynamic storage management technique based on hash-compaction which allows large state spaces to be explored with a low state omission probability. The other important consideration in the computation of the steady-state distribution of large Markov chains is the solution of large sparse sets of linear equations. Recent Krylov subspace techniques and new decompositional techniques address this problem in innovative ways. We provide an overview of these methods and implement them, together with our new hash-compaction technique, in a performance analysis tool called DNAmaca. We conclude by modelling a typical real-life example of a teletraffic switch and analysing it with DNAmaca.
Keywords: *Performance analysis, Markov chains, state space generation, probabilistic dynamic storage, steady state solution, Krylov subspace techniques, DNAmaca*
Computing Review Categories: *E.2, F.2.1, G.1.3, G.3*

Introduction

The performance analysis of complex stochastic systems has been the focus of research for many years. Since work began in earnest we have seen major advances such as multi-class queueing networks [2], stochastic Petri nets [3] and solution techniques for complex queues [17]. However, real life stochastic systems do not always yield neatly to existing methods. The example described in Sec. is one illustration of such a system. In most such cases, the only alternative besides simulation is to generate and solve a large Markov chain derived from the model. It thus makes sense to have a tool which can be used to solve general, large-scale Markov chains. The first such tool that the authors are aware of is USENUM [20], developed around 1987 at the University of Dortmund and to which the authors have gratefully had access for several years. Another tool is MARCA [24, 25] which was first developed at Queen's University of Belfast and which was later improved upon at the University of North Carolina.

The contributions of DNAmaca, the Markov chain specification, generation and solution tool described here are twofold. Firstly, DNAmaca's state generator uses a new dynamic storage management technique based on hash-compaction. In contrast to conventional exhaustive storage methods, the memory usage of this technique is very low and is independent of the length of the state descriptor. This enables DNAmaca to generate and store up to 2000000 states in only 64 Mb RAM. Secondly, DNAmaca's steady state solution engine incorporates a rich variety of sparse linear equation solvers. DNAmaca offers 12 different solution methods to solve general, unrestricted Markov chains with up to 2 400 000 million entries in their state-transition matrix using only 64 Mb RAM.

Note that if the underlying model has certain pre-existing structural properties, or if the only objective is to decide the correctness of the system being modelled, then techniques exist to handle very large state spaces [18, 5]. However, the objective here is to generate and solve the state space of *unrestricted* systems for the purpose of *performance analysis.*

As with other tools, DNAmaca goes through four main phases to solve any Markov chain. In the first phase the user specifies a translation from a modelling formalism to a Markov chain by using the high-level interface language described in Sec. . Next, the state space and state transition graph are generated from the model description. Sec. describes a new probabilistic dynamic hash-compaction method which is used to store states during this process. The third phase of DNAmaca's solution process involves solving a large set of steady-state equations to determine the long-run proportion of time that the system spends in each of its states. In Sec. we describe the recent Krylov subspace solution techniques and the new decomposition-based steady-state solvers which we have implemented. The final phase of the solution process involves synthesising the low-level state probabilities into more meaningful high-level performance statistics such as throughput or mean buffer length. In Sec. we illustrate the application

of DNAmaca by using it to assess the performance of a teletraffic switch.

DNAmaca user interface

The underlying Markov chain of a real-life stochastic system is likely to involve many thousands, if not millions of states and transitions. DNAmaca includes a high-level modelling interface to avoid explicit enumeration of these states and transitions. The interface also provides the user with facilities to control the state generation and solution processes, and to specify target performance measures.

Model description

The interface specifies the descriptor of a general *state* of the system, the rules for *transitions* between states and an *initial state* of the system. This model description is powerful enough to support a variety of formalisms such as Generalised Stochastic Petri nets, queueing networks etc.

Transitions are specified by using general C++ expressions and assignment statements to describe:

- one or more *enabling conditions* involving elements of the state vector corresponding to the *current state*.
- an *action* to be taken if the transition is executed; this will involve an assignment to the state vector elements of the *next state*.
- an indication of whether the transition from the current to the next state is *timed* or *instantaneous*.
- a *rate* (for timed transitions) or *relative weight* (for instantaneous transitions) is specified. These rates and weights may be denoted by (possibly state-dependent) arbitrary expressions.

State space generation control

A Markov chain state generator maps the high-level model description onto a low-level system consisting of the state space and a labelled state graph where the labels describe the rate of transitions between states. This mapping is performed using a breadth first search.

Depending on the application domain, there may be functional properties which should be checked during the state generation process, such as system invariants and the existence of deadlocks. These invariant conditions can be expressed as C++ expressions in DNAmaca. The generator will issue a warning if it encounters any state which violates an invariant.

The generator also allows the control of certain aspects of the state space generation process such as the *maximum number of states* to be generated or the *maximum CPU time* that should be spent on the generation.

During this phase the infinitesimal generator matrix Q is computed using the information about the rate of transitions between states. Q is not stored in main memory during this phase but is written to secondary storage row-by-row for later use by a steady state solver.

Solution control

Once the state space has been generated, the resulting infinitesimal generator matrix Q must be solved for its steady state distribution. The steady state probability solution process can be guided through a choice from any of the solution algorithms described in Sec. .

The choice of algorithm will depend on the characteristics of Q. For very small state spaces direct methods are generally more efficient than iterative methods, while decompositional methods are useful when the Markov chain is nearly completely decomposable.

The required accuracy and the maximum number of iterations can be specified for iterative methods. These methods will terminate after i iterations with an "accuracy" of ε if:

$$\frac{||x^{(i)} - x^{(i-k)}||_\infty}{||x^{(i)}||_\infty} < \varepsilon$$

where $x^{(i)}$ is the steady-state vector after iteration i and k is a small positive integer (the exact value depending on the method and the number of iterations required). ε can vary between 10^{-2} and 2^{-52}.

For the successive overrelaxation method (SOR), the relaxation parameter can be specified, or a simple dynamic parameter estimation technique can used. Finally, it is possible to specify the initial probability distribution vector; this is useful when performing a batch of experiments with similar parameter values.

Performance measures

Performance measures provide a backward mapping from low-level results like probabilities of states and rates of transitions to higher-level quantities such as throughput or mean buffer occupancy. Performance measures can generally be classified as *state* or *count* measures.

A *state measure* is used to determine the mean and variance of a real expression which is defined at every state in the system such as, for example, the average number of tokens on a particular place of a Petri net or some transition's enabling probability. The mean, variance, standard deviation and distribution of state measures can be computed in DNAmaca.

A *count measure* is used to determine the mean rate at which a particular event occurs. The occurrence of an event is controlled by three conditions:

- a *precondition* which holds on the current state.
- a *postcondition* which holds on the next state.
- *transitions* which must be fired during the transition from the current to the next state.

The conditions are specified as C++ expressions while the transitions are given in a list. Note that only the mean

of count measures is available, since computation of higher moments requires transient analysis.

Storage techniques

The process of mapping a high level model onto its underlying state space is a similar to a conventional breadth first search traversal of a directed graph. Most state space generation techniques therefore use two data structures: a breadth first search (BFS) queue and a table of explored states.

The BFS queue is a FIFO queue which is accessed sequentially at either end and is limited by the breadth of the state graph; thus it is not critical to memory requirements. However, the table of explored states must hold enough information to determine whether states popped off the BFS queue are new, or have in fact been encountered before. That is, one has to be able to rapidly store and retrieve information about *every* reachable state. Consequently, the layout and management of the explored state table is crucial to both the time and space efficiency of a state space generation technique.

Background

There are two ways of allocating memory for the table of explored states: *Static* techniques pre-allocate large blocks of memory, while *dynamic* techniques allocate memory as the state space is being generated.

Another characteristic which distinguishes the various state space generation methods is the *reliability* with which one can conclude that every state is uniquely represented in memory. *Exhaustive* generation techniques store the full state descriptor of every state and thereby ensure that every possible state in the state space is stored and uniquely identified. Unfortunately, this is impractical in most cases, because the state descriptor is usually too large (typically tens or even hundreds of bytes) to allow for the storage of a large number of states. One therefore resorts to some hashing technique which maps state descriptors onto a compressed representation of the state; the compressed representation is then stored in place of the full state descriptor. However, since no hashing function can guarantee a 1–1 mapping, these *probabilistic* techniques cannot guarantee that every unique state descriptor will be stored in memory. Any subsequent performance analysis will therefore be only partially correct and such techniques need to quantify the probability of inadvertently omitting a state.

Exhaustive dynamic techniques insert the complete state descriptor of every explored state into a dynamic array or linked list. Establishing whether or not a state has been explored requires a search of the entire list. This method is straightforward to implement, but there is a large search time overhead and memory requirements are high. This method is used by the MARCA analyser [25].

A variation of the above is to use a hash table to break the list of state descriptors up into several shorter lists where each list forms a row of the hash table. States are assigned to particular hash table rows by using a hash function. This method is used by the USENUM analyser [20]. The result is a faster search time (since only one row has to be searched when looking for a state), but memory requirements are now higher because of the space requirements of the hash table.

In order to reduce the size of the explored state table, Holzmann proposed his well-known bit-state hashing technique [15, 16] where a a hash function is used to map each state onto a single bit position in a large bit vector. This method is clearly a *probabilistic static* technique since more than one distinct state descriptor can map to the same hash value. To quantify the likelihood of the latter, Wolper and Leroy [30] approximate the probability p of no collisions as:

$$p \approx e^{-\frac{n^2}{t}}$$

where n is the number of states and t is the size of the bit vector. The size of the bit vector required to keep p very low is, however, still impractically large, even with the double hashing technique proposed in [15].

The problem with Holzmann's bit-state hashing method is that the ratio of the number of states to table entries must be kept very low to ensure a good probability of complete state coverage. Consequently, a large amount of the memory allocated for the bit vector is wasted. Wolper and Leroy [30] observed that it is more space efficient to store only those hash table entries which have been marked as explored. This can be done by using a hash compaction technique which hashes state descriptors onto compressed values of k bit keys. These keys can then be stored in a smaller hash table which uses a collision resolution scheme. Given a hash table with $t > n$ slots, this approach simulates a bit-state hashing scheme with a table size of 2^k; the probability p of no collision is given by:

$$p \approx e^{-\frac{n^2}{2^k}}$$

In standard hash compaction, it is implicitly assumed that the hash values (used to determine where in the hash table to store the k-bit compressed values) are calculated using the compressed values. Stern and Dill [23], however, noted that the omission probability can be dramatically reduced in two ways. Firstly, two separate hash functions can be used to calculate the hash values and compressed values independently. Secondly, a collision resolution scheme which keeps the number of probes per insertion low can be employed. This improved technique is so effective that it requires only 5 bytes per state in situations where Wolper and Leroy's standard hash compaction requires 8 bytes per state. Given that all slots in the hash table are occupied by states, Stern and Dill derive a straightforward formula for the approximate maximum omission probability q:

$$q \approx \frac{1}{2^k} n(\ln n - 1)$$

which shows that the omission probability is proportional to $n \ln n$. Increasing the number of bits per state k by one halves the omission probability.

A new probabilistic dynamic storage technique

None of the methods mentioned above has the advantage of being both probabilistic and dynamic. In this section we propose a new technique which uses dynamic storage allocation while yielding a good collision avoidance probability. We use a hash table of linked lists (as used in an exhaustive dynamic technique) but instead of storing full state descriptors in the lists, we store compressed state descriptors (as in hash compaction).

Two independent hash functions are used. The *primary* hash function h_1 is used to determine which hash table row should be used to store a compressed state and the *secondary* hash function h_2 is used to compute the compressed state descriptor values. Both h_1 and h_2 are assumed to distribute states randomly and independently of one another; the H_3 class of hash functions defined by Carter and Wegman [6] satisfies this property. If a state's secondary key is already present in the hash table row given by its primary key, then the state is deemed to have been explored and no further action is taken. Otherwise, the secondary key is added to the hash table row and its successors are pushed onto the state exploration queue. This scheme is illustrated in Fig. 1. Note that two states s_1 and s_2 are

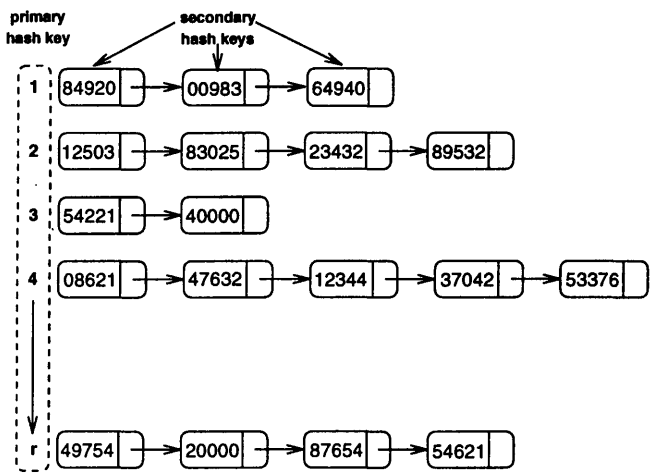


Figure 1: Hash table with compressed state information

classified as being equal if and only if $h_1(s_1) = h_1(s_2)$ and $h_2(s_1) = h_2(s_2)$; this may happen even when the two state descriptors are different, so that collisions may occur (as in all other probabilistic methods).

Reliability analysis

Let X_ℓ^n be a random variable denoting the number of states allocated to row $\ell, 1 \leq \ell \leq r$, given that there are n unique state identifiers to be inserted into the table. Then, since we assumed that h_1 distributes states randomly, X_ℓ^n will have a binomial distribution with parameters n and $p = 1/r$, i.e.,

$$P\{X_\ell^n = j\} = \binom{n}{j} (\tfrac{1}{r})^j (1-\tfrac{1}{r})^{n-j} = \binom{n}{j} \frac{(r-1)^{n-j}}{r^n}$$

Denoting the number of clashes in row ℓ by C_ℓ and considering the case when there are j states in row ℓ, we have:

$$P\{C_\ell = 0 | X_\ell^n = j\} = \frac{t(t-1)(t-2)\ldots(t-j+1)}{t^j} = \frac{t!}{(t-j)!\,t^j}$$

where $t = 2^b$ is the number of unique secondary key values and b is a positive integer denoting the number of bits used to store the secondary key. Then,

$$P\{C_\ell = 0\} = \sum_{j=0}^{n} P\{C_\ell = 0 | X_\ell^n = j\} P\{X_\ell^n = j\} = \frac{1}{r^n} \sum_{j=0}^{n} \binom{n}{j} \frac{(r-1)^{n-j} t!}{(t-j)!\,t^j}$$

If C_r denotes the total number of clashes across all rows r of the hash table, the probability p of no clash in *any* row of the hash table is simply given by:

$$p = P\{C_r = 0\} = (P\{C_\ell\})^r = \left(\frac{1}{r^n} \sum_{j=0}^{n} \binom{n}{j} \frac{(r-1)^{n-j} t!}{(t-j)!\,t^j} \right)^r \quad (1)$$

since it is assumed that the primary hash function distributes states randomly. The probability q of omitting at least one state is of course simply $q = 1 - p$.

An experiment was conducted to compare the values of p computed from Eq. (1) against values obtained from a simulation. Using a small hash table of $r = 128$ rows and $b = 10$ bit keys, experiments were performed with $n = 50, 100, 150, \ldots, 500$ states. Each experiment was repeated 10000 times and the proportion of clash-free runs was noted. The simulated results with 95% confidence intervals are presented in Fig. 2.

Approximation

Evaluating the right hand side of Eq. (1) requires performing $O(n^2)$ operations. However, an $O(1)$ approximation can be found through an approach similar to that used by Stern and Dill [23] in their analysis of improved hash compaction.

In the appendix we show that if $n(n-1) << 2rt$ (as will be the case in practical schemes where $q << 1$) we can use the fact that $e^x \approx (1+x)$ for $|x| << 1$ to approximate for the probability p of no omission when inserting all n states, giving:

$$p \approx 1 - \frac{n(n-1)}{2rt}$$

so that probability q of an omission is:

$$q \approx \frac{n(n-1)}{2rt} = \frac{n(n-1)}{r2^{b+1}} \quad (2)$$

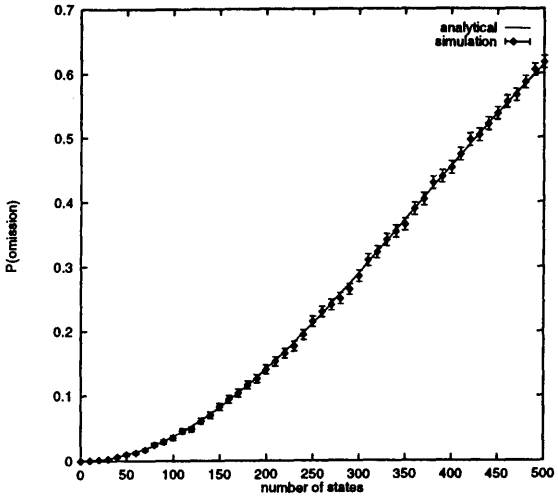


Figure 2: Analytical vs. simulated results for the probability of state omission q

Thus the probability q of omitting a state is proportional to n^2 and is inversely proportional to the hash table size r. Increasing the size of the compressed bit vectors b by one bit halves the omission probability.

Space complexity

If we assume that the hash table rows are implemented as dynamic arrays, the number of bytes of memory required by the new scheme is:

$$M = hr + nb/8. \tag{3}$$

Here h is the number of bytes of overhead per hash table row. For a given number of states and a desired omission probability, there are a number of choices for r and b which all lead to schemes having different memory requirements. How can we choose r and b to minimise the amount of memory required?
Rewriting Eq. (2):

$$r \approx \frac{n(n-1)}{2^{b+1}q}$$

and substituting this into Eq. (3) yields

$$M \approx \frac{hn(n-1)}{2^{b+1}q} + \frac{nb}{8}$$

Minimizing M with respect to b gives:

$$\frac{\partial M}{\partial b} \approx -\frac{n(n-1)(\ln 2)h}{2^{b+1}q} + n/8 = 0$$

Solving for b yields:

$$b \approx \log_2\left(\frac{h(n-1)\ln 2}{q}\right) + 2 \tag{4}$$

Table 1 shows the the optimal memory requirements in megabytes (Mb) for corresponding values of b and r for state space sizes ranging from 10^5 to 10^8. We assume a hash table row overhead of $h = 8$ bytes/row.

The DNAmaca analyser implements this scheme using a hash table with 2^{14} rows and 32 bit secondary keys. Fig. 3 compares the memory utilisation of our technique with that of an exhaustive dynamic storage technique for the *benchprod* Stochastic Petri Net model [7]. *Benchprod* is a scalable model of the Oki Electric Company (Japan) production line. It is not important to understand all the details of the model; it suffices to note that there is a scalable parameter k, and that as k increases, so does the number of states in the model. The results were obtained on a Sun SPARCclassic with 64 Mb memory. The space saving advantages of using a probabilistic technique are clear: while DNAmaca is able to explore state spaces of up to 2 100 000 states in under 32 Mb memory, the exhaustive dynamic scheme implemented by USENUM is limited to under 200 000 states.

The table in Fig. 3 presents the corresponding state space generation times (CPU and system time, as given by the *clock()* system call) for systems of up to 2 100 000 states. It is interesting to note that, even on a SPARCclassic (a machine only approximately 1.5 times as powerful as a 33MHz 486), our state exploration method outperforms a parallel exhaustive exploration technique running on a CM-5 with 32 nodes, each of which corresponds to a SPARC2 workstation with 32 Mb RAM [7]. For a 511 588 state *benchprod* model, the CM-5 generates the state space at a rate of 1.507 milliseconds per state, while we measured 0.864 milliseconds per state.

Because of the limitations of exhaustive methods, state space exploration used to be regarded as the most resource-intensive phase of the performance analysis pipeline; we believe that our new technique dramatically shifts the bottleneck away from state space exploration onto later stages in the pipeline.

Steady-state solution methods

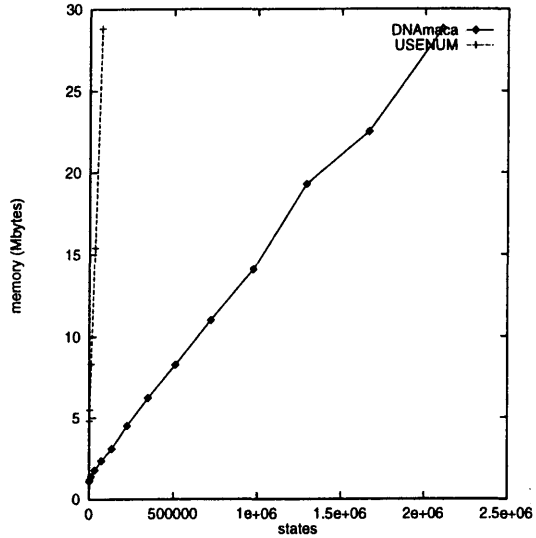
Once the state space has been generated, the next stage is to solve the set of steady-state equations

$$\pi Q = 0 \quad \text{subject to} \quad \sum_{k=0}^{n} \pi_k = 1$$

where $\pi = (\pi_1, \pi_2, \ldots, \pi_n)$ is the n-vector of steady-state probabilities and Q is the $n \times n$ infinitesimal generator matrix of transition rates between states. In the following discussion of methods to solve linear systems of the form $Ax = 0$, where $A = Q^T$ and $x = \pi^T$, we distinguish between those solution methods which have been known for many years and more recent ones. All the methods described have been implemented in DNAmaca and can be specified as the preferred solution method by the user (see Sec.).

q	number of states											
	10^5			10^6			10^7			10^8		
	Mb	b	r	Mb	b	r	Mb	b	r	Mb	b	r
0.001	0.4061	31	2328	4.483	34	29104	48.96	38	1818997	530.7	41	2273737
0.01	0.3649	28	1863	4.061	31	23283	44.83	34	291038	489.6	38	1818989
0.1	0.3238	24	2980	3.649	28	18626	40.61	32	116415	448.3	34	2910383

Table 1: Optimal values for memory usage and the values for b and r used to obtain them for various system state sizes and omission probabilities q



k	number of states	generation time (seconds)	time/marking (millisecs)
1	172	0.12	0.709
2	2 359	1.51	0.641
3	11 386	8.05	0.707
4	32 653	24.17	0.740
5	71 560	55.01	0.769
6	133 507	108.26	0.811
7	223 894	186.01	0.831
8	348 121	292.60	0.841
9	511 588	442.14	0.864
10	719 695	645.47	0.897
11	977 842	887.32	0.907
12	1 291 429	1 215.89	0.942
13	1 665 856	1 625.18	0.976
14	2 106 523	2 147.10	1.019

Figure 3: Comparative memory use between exhaustive (USENUM) and probabilistic (DNAmaca) state space exploration techniques (left) and state space generation times for DNAmaca (right) for the *benchprod* model

Historical methods

Direct methods compute an exact solution in a fixed number of arithmetic operations determined by the size of the problem and have general complexity $O(n^3)$. Historical direct methods implemented in DNAmaca are sparse Gaussian elimination [26, §2.2.1] and Grassmann's method [12]. Grassmann's method is a very accurate subtraction-free variant of Gaussian elimination specifically designed to solve for the steady-state distribution of Markov Chains.

Iterative methods, on the other hand, form a sequence of vectors $x^{(0)}$, $x^{(1)}$, $x^{(2)}$... which converges to the solution of $Ax = 0$. Techniques which have been known for decades are Gauss-Seidel and SOR [29, §3.1]. These methods are characterized by low memory requirements (storage for matrix A plus 2 vectors) and smooth convergence. However, convergence is slow, and the methods cannot be easily parallelised. SOR also requires estimation of the over-relaxation parameter. Iterative methods are preferred over direct methods because they are more time efficient for large values of n, do not modify the matrix A and mostly only involve matrix-vector operations of the form Ax or $A^T x$.

Recent methods

We next consider two classes of relatively recent iterative methods, namely Krylov subspace techniques and decompositional techniques.

Krylov subspace techniques

Krylov subspace techniques implemented in DNAmaca include BiCG [9], CGNR [31, §2.5], BiCGSTAB [28], BiCGSTAB(2) [21], CGS [22] and TFQMR [10]. Krylov subspace techniques are particularly attractive for the following three main reasons:

- The methods are parameter free, yet still provide good rates of convergence. The original conjugate gradient algorithm, for example, provides the same order of convergence rate as optimal SOR but without the need for dynamic parameter estimation.
- Krylov subspace techniques have become increasingly competitive with classical iterative methods in terms of memory utilization. This is because the most recently developed conjugate gradient-type algorithms for non-symmetric matrices (eg. CGS, BiCGSTAB, TFQMR) do not require storage of large sequences of vectors (as does GMRES [19]), nor do they require multiplication with the transpose of the coefficient matrix (as do BiCG, QMR [11] and CGNR/CGNE).

- The methods are well suited to implementation on parallel and vector computers. Most Krylov subspace methods compute one or two matrix-vector products and several vector inner products at every iteration; the methods are thus easily parallelised by distributing the matrix among processing nodes and by using the inner products as synchronisation points. In practice, superlinear speedups (corresponding to efficiencies of over 100%) have been achieved in both symmetric multiprocessing environments and high-speed distributed environments [4].

The development of Krylov subspace techniques began in the early 1950s with the conjugate gradient (CG) algorithm of Hestenes and Stiefel [13]. This algorithm is used to solve $n \times n$ linear systems of the form $Ax = b$ where A is a symmetric positive definite coefficient matrix. The CG method is regarded as an attractive algorithm for two main reasons. Firstly, the algorithm has very modest memory requirements because it uses simple three-term recurrences. Secondly, the algorithm has good convergence properties since the residual is minimized with respect to some norm at each step. The generated residuals are also mutually orthogonal, which guarantees finite termination.

Several algorithms have since been devised to generalise the CG algorithm to allow for arbitrary coefficient matrices. Unfortunately, algorithms for non-symmetric coefficient matrices cannot maintain both the short recurrence formulation and the minimization property [8]. Thus, by trading off certain optimality conditions against the amount of memory required, three main classes of CG variants have been developed:

- Algorithms which attempt to preserve both properties by transforming a linear system based on a non-symmetric coefficient matrix A to an equivalent system based on the symmetric positive definite matrix A^TA (CGNR) or AA^T (CGNE). This approach is known as conjugate gradient applied to the normal equations.

- "Pure" algorithms for non-symmetric matrices A which are based on maintaining *either* the short recurrence formulation (eg. BiCG [9]) *or* the minimization property (eg. GMRES [19]).

- "Hybrid" methods for non-symmetric matrices A which seek to combine elements of the short recurrence formulation with minimization properties that are either heuristic (eg. CGS), localized (eg. BiCGSTAB) or quasi-optimal (eg. QMR). This class includes most of the more recently developed CG-type methods such as CGS, QMR, TFQMR, BiCGSTAB and BiCGSTAB(l).

Decomposition techniques

Touzene's 1995 Aggregation-Isolation (AI) algorithm [27] and its improved variant Aggregation-Isolation Relaxed (AIR) are new algorithms specifically designed for solving large scale Markov Chains. Memory requirements are low (matrix plus 3 n-vectors) and convergence is rapid and smooth. However, the method is not suited to parallelisation. DNAmaca features the first full-scale implementations of both algorithms, including optimizations such as storing and solving the 3×3 aggregation matrix in registers and using an effective table-driven relaxation parameter estimation scheme for AIR.

Fig. 3 shows the convergence behaviour of some steady state algorithms for a queueing Petri net model of the INRES protocol [14] with 73 735 tangible states. Notice the fairly smooth but slow convergence of SOR, the erratic but superlinear convergence of the Krylov subspace methods and the rapid convergence of the AIR method.

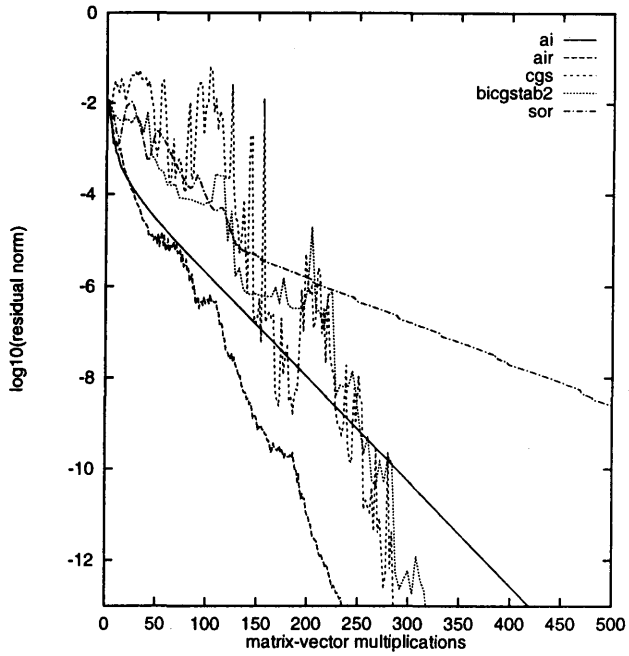


Figure 4: Convergence behaviour of some steady-state algorithms

Example

The schematic diagram in Fig. 4 is of a multimedia teletraffic switch designed to handle delay sensitive (eg. voice) and delay insensitive (eg. data) traffic [1]. The switch has a capacity for s calls and is designed to give priority to voice calls. If the switch is full and the number of data calls in the system exceeds a certain threshold n, an arriving voice call may preempt a data call. If there are fewer than n data calls and no free circuits in the switch, arriving voice calls will be blocked. Waiting or preempted data calls are stored in a buffer with capacity b.

There are v potential sources of voice calls. Each of these sources is governed by a two-state Markov process which alternates between a silence phase and a talkspurt phase. The mean duration of the silence phase is $1/\lambda_1$ and the mean duration of a talkspurt phase is $1/\mu_1$. The data arrival process is simpler, being Poisson with parameter λ_2. Data calls are served at a rate of μ_2 per server.

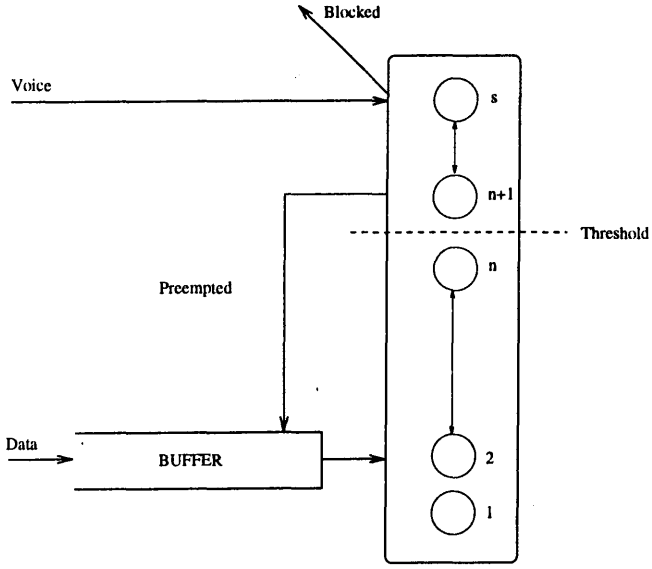


Figure 5: A multimedia switch for handling voice and data traffic

The performance analysis of this switch, with both blocking and preemptive priority service as well as a two-phase arrival process, is not easily accomplished using conventional queueing theory. This makes the model a good candidate for DNAmaca's unrestricted analysis capabilities.

We used DNAmaca to model a switch with capacity $s = 72$, buffer size $b = 200$ and a threshold value $n = 32$. There were $v = 1000$ voice sources, with $\lambda_1 = 0.04$ and $\mu_1 = 1.0$. The data arrival rate was $\lambda_2 = 43.0$, and the data service rate was $\mu_2 = 1.2$ per server.

The model generates a Markov chain with 17 301 tangible states. Fig. 6 shows the distributions for the number of voice and data calls in the system and the corresponding distribution for the number of data calls in the buffer. Since voice calls have priority over data calls, it makes sense that the mean number of voice calls in the system should be higher than the mean number of data calls. The distribution of calls in the buffer shows that a buffer size of 150 should be more than adequate to deal with almost all calls. Note that the number of voice calls drops off sharply once the preemption limit is reached.

Conclusions

In this paper we introduced and analysed a new probabilistic dynamic hash-compaction storage technique. Our technique has a low state omission probability and a low memory requirement that is independent of the length of the state descriptor. This makes it possible to explore state spaces that are an order of magnitude larger than those obtainable using conventional exhaustive techniques. We also described the many linear equation solution methods for the computation of the steady state probability distribution and put them in context. These various solution methods and the new hash-compaction technique are implemented in DNAmaca, a software tool for the solution of continuous time Markov chain models of state-transition systems. Experiments with DNAmaca show that, even with a modest workstation such as a Sun SPARCclassic with 64 Mb of RAM, one can generate Markov chains of as many as 2 000 000 states; furthermore one can solve for the steady state distributions of Markov chains with up to 2 400 000 entries in their state transition matrices. In many real situations, however, even this capability may not be sufficient and future work will concern the parallelisation of the state space exploration technique as well as the linear equation solver in order to address models with significantly larger state spaces.

References

[1] H Akimaru and K Kawashima. *Teletraffic: theory and applications*. Springer-Verlag, 1993.

[2] F Basket, K Chandy, R Muntz, and F Palacios. 'Open, closed and mixed networks of queues with different classes of customers'. *Journal of the ACM*, **22**:248 – 260, (1975).

[3] F Bause and P Kritzinger. *Stochastic Petri net theory*. Verlag Vieweg, Wiesbaden Germany, 1995.

[4] B Boulter. 'Performance evaluation of HPF for scientific computing'. In *Lecture Notes in Computer Science 919*. Springer-Verlag, (1995).

[5] J Burch *et al.* 'Symbolic model checking for sequential circuit verification'. *IEEE Transactions on Computer-Aided Design of Integrated Circuits*, **13**(4):250 – 263, (1994).

[6] J L Carter and M N Wegman. 'Universal classes of hash functions'. *Journal of Computer and System Sciences*, **18**:143–154, (1979).

[7] S Caselli, G Conte, and P Marenzoni. 'Parallel state exploration for GSPN models'. In *Lecture Notes in Computer Science 935: Proceedings of the 16th International Conference on the Application and Theory and Petri Nets*. Springer Verlag, Turin, Italy, (June 1995).

[8] V Faber and T Manteuffel. 'Necessary and sufficient conditions for the existence of a conjugate gradient method'. *SIAM Journal on Numerical Analysis*, **21**(2):352–362, (April 1984).

[9] R Fletcher. 'Conjugate gradient methods for indefinite systems'. In *Lecture Notes in Mathematics*, volume 506, pp. 73–89. Springer-Verlag, (1976).

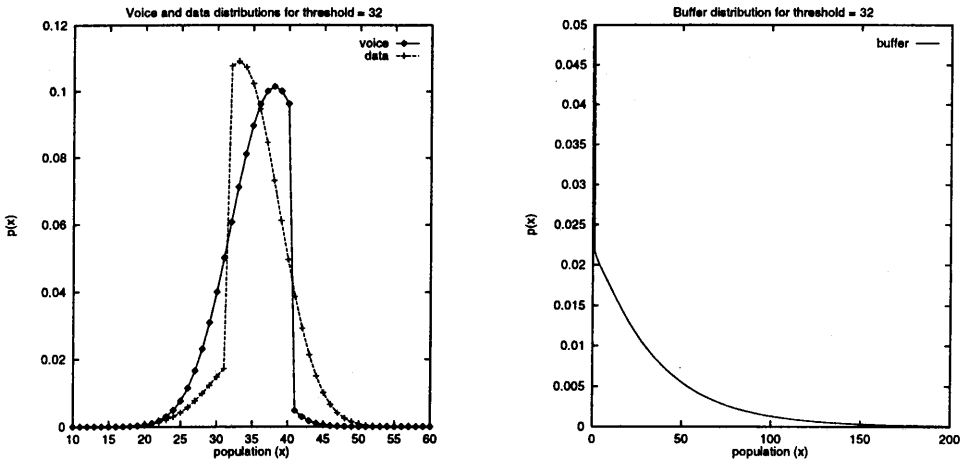


Figure 6: Distributions for the number of voice and data calls in the switch (left) and the number of data calls in the buffer (right).

[10] R W Freund. 'A transpose-free quasi-minimal residual algorithm for non-Hermitian linear systems'. *SIAM Journal on Scientific Computing*, **14**(2):470–482, (March 1993).

[11] R W Freund and N M Nachtigal. 'QMR: a quasi-minimal residual method for non-Hermitian linear systems'. *Numerische Mathematik*, **60**:315–339, (1991).

[12] W K Grassmann, M I Taskar, and D P Heyman. 'Regenerative analysis and steady state distributions for Markov chains'. *Operations Research*, **33**(5):1107–1116, (1985).

[13] M Hestenes and E Stiefel. 'Methods of conjugate gradients for solving linear systems'. *Journal of Research of the National Bureau of Standards*, **49**:409–435, (1952).

[14] D Hogrefe. *Estelle, LOTOS and SDL.* Springer-Verlag, 1989.

[15] G J Holzmann. *Design and Validation of Computer Protocols.* Prentice-Hall, 1991.

[16] G J Holzmann. 'An analysis of bitstate hashing'. In *Proceedings of IFIP/PSTV95: Conference on Protocol Specification, Testing and Verification.* Chapman and Hall, Warsaw, Poland, (June 1995).

[17] F Kelley. *Reversibility and Stochastic Networks.* Wiley and Sons, 1979.

[18] P Kemper. 'Numerical analysis of superposed GSPNs'. In *Proc. of the Sixth International Workshop on Petri Nets and Perfromance Models*, pp. 52–62. IEEE Computer Societ Press, (1994).

[19] Y Saad and M H Schultz. 'GMRES: A generalized minimal residual algorithm for solving non-symmetric linear systems'. *SIAM Journal on Scientific and Statistical Computing*, **7**(3):856–869, (July 1986).

[20] M Sczittnick. Technicken zur funktionalen und quantitativen Analyse von Markoffschen Rechensystem-modellen. Diplomarbeit, Universität Dortmund, October 1987.

[21] G L Sleijpen and D R Fokkema. 'BiCGSTAB(*L*) for linear equations involving unsymmetric matrices with complex spectrum'. *Electronic Transactions on Numerical Analysis*, **1**:11–32, (September 1993).

[22] P Sonneveld. 'CGS, a fast Lanczos-type solver for nonsymmetric linear systems'. *SIAM Journal on Scientific and Statistical Computing*, **10**(1):36–52, (January 1989).

[23] U Stern and D L Dill. 'Improved probabilistic verification by hash compaction'. In *IFIP WG 10.5 Advanced Research Working Conference on Correct Hardware Design and Verification Methods*, (1995).

[24] W J Stewart. *Markov Analysis of Operating System Techniques.* PhD thesis, Queen's University of Belfast, N. Ireland, 1974.

[25] W J Stewart. 'MARCA: Markov Chain Analyser'. In *Numerical Solutions of Markov Chains*, pp. 37–62. Marcel Dekker, New York, (1991).

[26] W J Stewart. *Introduction to the Numerical Solution of Markov Chains.* Princeton University Press, 1994.

[27] A Touzene. 'A new iterative method for solving large-scale Markov chains'. In *Lecture Notes in Computer Science 977: Proceedings of the 8th International Conference on Modelling Techniques and Tools for Computer Performance Evaluation.* Springer Verlag, Heidelberg, (September 1995).

[28] H van der Vorst. 'Bi-CGSTAB: A fast and smoothly converging variant of BiCG for the solution of non-symmetric linear systems'. *SIAM Journal on Scientific and Statistical Computing*, **13**(2):631–644, (March 1992).

[29] R S Varga. *Matrix Iterative Analysis*. Prentice-Hall, 1962.

[30] P Wolper and D Leroy. 'Reliable hashing without collision detection'. In *Lecture Notes in Computer Science 697*, pp. 59–70. Springer-Verlag, (1993).

[31] U M Yang. 'Preconditioned conjugate gradient-like methods for nonsymmetric linear systems'. Technical Report 1210, Center for Supercomputing Research and Development, University of Illinois at Urbana-Champaign, (July 1994).

Received: 10/97, Accepted: 3/98

A Reliability of the new hash-compaction scheme

We consider inserting n states into the hash table, one at a time. Let N_k be the event that no omission takes place when the $(k+1)$st state is inserted into the hash table, given that the previous k states have been inserted without any omissions.

The probability of event N_k will depend on the number of secondary key comparisons that have to be made when inserting state s_{k+1} into the target row given by it's primary hash key $h_1(s_{k+1})$. If there are j items in the target row,

$$P\{N_k|X^k_{h_1(s_{k+1})}=j\}=1-\frac{j}{t}\approx(1-\frac{1}{t})^j$$

since $(1+nx)\approx(1+x)^n$ for $|x|<<1$ (here $1/t$ is very small). We have already established that

$$P\{X^k_{h_1(s_{k+1})}=j\}=\binom{n}{j}(\frac{1}{r})^j(1-\frac{1}{r})^{n-j}$$

Thus by the law of total probability,

$$\begin{aligned}P\{N_k\} &= \sum_{j=0}^{k}P\{N_k|X_{h_1(s_{k+1})}=j\}P\{X_{h_1(s_{k+1})}=j\}\\ &\approx \sum_{j=0}^{k}(1-\frac{1}{t})^j\binom{k}{j}(\frac{1}{r})^j(1-\frac{1}{r})^{k-j}\\ &= \sum_{j=0}^{k}\binom{k}{j}\left(\frac{1}{r}(1-\frac{1}{t})\right)^j(1-\frac{1}{r})^{k-j}\end{aligned}$$

Applying the binomial theorem yields:

$$\begin{aligned}P\{N_k\} &= \left(\frac{1}{r}(1-\frac{1}{t})+1-\frac{1}{r}\right)^k\\ &= \left(1-\frac{1}{rt}\right)^k\\ &\approx e^{-\frac{k}{n}} \qquad (5)\end{aligned}$$

Now let N'_k be the unconditional event that no omission takes place when inserting the $(k+1)$st state into the hash table. Stern and Dill show that the probability p of no omission when inserting all n states is given by:

$$\begin{aligned}p &= P\{N'_{n-1}\wedge N'_{n-2}\wedge\ldots\wedge N'_0\}\\ &= P\{N'_{n-1}|N'_{n-2}\wedge\ldots\wedge N'_0\}P\{N'_{n-2}\wedge\ldots\wedge N'_0\}\\ &= P\{N_{n-1}\}P\{N'_{n-2}\wedge\ldots\wedge N'_0\}\end{aligned}$$

which, when applied recursively, yields:

$$p=\prod_{k=0}^{n-1}P\{N_k\}$$

Now we can substitute the expression for $P\{N_k\}$ from Eq. (5) to yield:

$$\begin{aligned}p &\approx \prod_{k=0}^{n-1}e^{-\frac{k}{n}}\\ &= e^{-\sum_{k=0}^{n-1}\frac{k}{n}}\\ &= e^{-\frac{n(n-1)}{2n}}\end{aligned}$$

If $n(n-1)<<2rt$ (as will be the case in practical schemes where $q<<1$), we can use the fact that $e^x\approx(1+x)$ for $|x|<<1$ to approximate p by:

$$p=1-\frac{n(n-1)}{2rt}$$

so that the probability of state omission q is:

$$q\approx\frac{n(n-1)}{2rt}=\frac{n(n-1)}{r2^{b+1}}$$

Notes for Contributors

The prime purpose of the journal is to publish original research papers in the fields of Computer Science and Information Systems, as well as shorter technical research notes. However, non-refereed review and exploratory articles of interest to the journal's readers will be considered for publication under sections marked as Communications of Viewpoints. While English is the preferred language of the journal, papers in Afrikaans will also be accepted. Typed manuscripts for review should be submitted in triplicate to the editor.

Form of Manuscript
Manuscripts for *review* should be prepared according to the following guidelines:

- Use wide margins and $1\frac{1}{2}$ or double spacing.
- The first page should include:
- the title (as brief as possible)
- the author's initials and surname
- the author's affiliation and address
- an abstract of less than 200 words
- an appropriate keyword list
- a list of relevant Computing Review Categories
- Tables and figures should be numbered and titled.
- References should be listed at the end of the text in alphabetic order of the (first) author's surname, and should be cited in the text according to the Harvard. References should also be according to the Harvard method.

Manuscripts accepted for publication should comply with guidelines as set out on the SACJ web page,

`http://www.cs.up.ac.za/sacj`

which gives a number of examples.

SACJ is produced using the LaTeX document preparation system, in particular LaTeX 2_ε. Previous versions were produced using a style file for a much older version of LaTeX, which is no longer supported. Please see the web site for further information on how to produce manuscripts which have been accepted for publication.

Authors of accepted publications will be required to sign a copyright transfer form.

Charges
Charges per final page will be levied on papers accepted for publication. They will be scaled to reflect typesetting, reproduction and other costs. Currently, the minimum rate is R30.00 per final page for contributions which require no further attention. The maximum is R120.00, prices inclusive of VAT.

These charges may be waived upon request of the author and the discretion of the editor.

Proofs
Proofs of accepted papers may be sent to the author to ensure that typesetting is correct, and not for addition of new material or major amendments to the text. Corrected proofs should be returned to the production editor within three days.

Letters and Communications
Letters to the editor are welcomed. They should be signed, and should be limited to about 500 words. Announcements and communications of interest to the readership will be considered for publication in a separate section of the journal. Communications may also reflect minor research contributions. However, such communications will not be refereed and will not be deemed as fully-fledged publications for state subsidy purposes.

Book Reviews
Contributions in this regard will be welcomed. Views and opinions expressed in such reviews should, however, be regarded as those of the reviewer alone.

Advertisement
Placement of advertisements at R1000.00 per full page per issue and R500.00 per half page per issue will be considered. These charges exclude specialised production costs, which will be borne by the advertiser. Enquiries should be directed to the editor.

South African Computer Journal

Number 21, August 1998
ISSN 1015-7999

Suid-Afrikaanse Rekenaar-tydskrif

Nommer 21, Augustus 1998
ISSN 1015-7999

Contents

Editorial
N. du Plooy 1

Research Articles

Text Categorization as an Information Retrieval Task
H. Paijmans 4

Preliminary Investigation of the RAMpage Memory Hierarchy
P. Machanick and P. Salverda 16

Changes in Entry-Level University Students' Attitudes to Computers from 1985 to 1997
M.C. Clarke and G.R. Finnie 26

A Performance Analyser for the Numerical Solution of General Markov Chains
W. Konttenbelt and P. Kritzinger 34

Specific Acquisition of Collective Belief Knowledge for Socially Motivated Multiagent Systems
V. Ram 44

Communications and Viewpoints

Fractal Image Compression
E. Cloete and L.M. Venter A49

Applied Lambda Calculus: Using a Type Theory Based Proof Assistant
L. Pretorius A55

Recursive Specifications and Formal Logic: What Benefits for Intelligent Tutoring Systems?
Lot Tcheeko A63