Parallel Structure in an Integrated Speech-Recognition Network

M. Fleury, A. C. Downton, and A. F. Clark

Department of Electronic Systems Engineering,
University of Essex, Wivenhoe Park,
Colchester, CO4 3SQ, U.K
tel: +44 - 1206 - 872795
fax: +44 - 1206 - 872900
e-mail fleum@essex.ac.uk

Abstract. Large-vocabulary continuous-speech recognition (LVCR) speaker-independent systems which integrate cross-word context dependent acoustic models and n-gram language models are difficult to parallelize because of their interwoven structure, large dynamic data structures, and complex object-oriented software design. This paper shows how retrospective decomposition can be achieved if a quantitative analysis is made of dynamic system behaviour. A design which accommodates unforeseen effects and future modifications is presented.

1 Introduction

Two varieties of LVCR system exist: a pipelined structure in which components of acoustic matching and language modelling are separated; and an approach which integrates cross-word context dependent acoustic models and n-gram language models into the search. The former has been thought to be more computationally tractable [1], while the latter has delivered a low mean error rate, 8.2% per word in ARPA evaluation, for a 65k vocabulary, tri-gram language model [2]. This paper examines whether an integrated system could also be parallelised as has been achieved [3] for the pipelined structure.

On a high-performance workstation, even after introducing efficient memory management of dynamic data structures, and optimising inner loops, timings on a 20k vocabulary application, perplexity[1] 145, indicate that a further fivefold increase in execution speed is needed to achieve real-time performance. Increasingly complex future applications are likely to maintain this requirement even as uniprocessor performance increases through Moore's law. This paper proposes a minimum cost redesign of such a sequential system aimed at achieving real-time performance for prototyping, rapid performance evaluation, and demonstrations in the development environment. The imminent ETSI standard for front-end processing enables such systems to act as servers to thin clients possibly on mobile stations. To this end a preliminary parallelisation of a 20k vocabulary application has been made.

[1] Perplexity is a measure of average recognition network branching.

2 Scale of the problem

A standard stochastic modelling approach to speech recognition has both improved recognition accuracy and the speed of computation [4]. Mel-frequency cepstrum acoustic feature vectors, hidden Markov models (HMMs) [5] to capture temporal and acoustic variance, tri-phone sub-word representation, and Gaussian probability distribution mixture sub-word models [6] are amongst the algorithmic components that have led to the emergence of LVCR. Any parallelisation should seek to preserve this stable structure, onto which further algorithmic innovations have been conveniently hung. Tied states and modes within Hidden Markov models (HMM) for sub-word acoustic matching improve training accuracy for 'unseen' crossword triphones but imply shared data. Such common data also reduce computation during a recognition run on a uniprocessor or a multiprocessor with a shared address space but pose a problem for a distributed-memory parallel implementation.

Speaker-independent integrated systems are being contemplated for database enquiry by telephone services. Development of an LVCR system requires the considerable resources available to large organisations. It may be unrealistic to think that a parallel algorithm, for example [7], will now be newly applied to existing systems. British Telecom (BT) have developed a toolkit for constructing LVCR applications which employs a one-pass time-synchronous speech decoder where tokens carry scores (the sum of sub-word probabilities accrued), maintaining pointers to the n-best recognition network paths. The toolkit paradigm allows a variety of applications to be constructed from a core class library. One such application is considered in this paper. However, even when an application has been developed care must be taken that a subsequent unconsidered parallelisation prematurely fixes the algorithmic content. It is not always possible to predict the side-effects a change might have, or the restriction a change might have on future algorithmic development.

Achieving speaker-independent recognition in real time is significantly harder than speaker-dependent systems. Compare the IBM Tangora PC system [8] which uses an iterative search to reach real-time performance after the recognition network has been trained. Speaker-independent systems must model differences in speech intonation such as accent, dialect, age, and gender. Telephonic applications must also cope with a 10 kHz bandwidth restriction and noise reduction is needed on mobile stations. However, reduction of data-storage and algorithmic complexity is less of an obstacle than when squeezing a recognition interface onto a PC. A conversational interface, additionally requiring speech understanding, is possible on a larger system. The complete BT system is distributed, connected under the CORBA distributed object standard [9], with LVCR as one component. Portability, usually through standardized software, is also an important consideration if adding a parallel extension to the LVCR system. In this respect, the standard message-passing libraries, PVM and MPI, seem suitable.

Given the logistic difficulties of developing the BT system, there are also short-term benefits from parallelising a LVCR system. Though algorithmic development is ongoing, there is an interim need to demonstrate real-time behaviour to potential clients. An additional benefit of speed up is a reduction in performance-tuning run times. Test code in the BT system is included for all major modules to validate correct behaviour. However, in comparative performance runs, the complete system must be reset even if a minor change is made. For this purpose, speed-up need not be real time. In short, any improvement in speed is welcome whether it results from code optimisations, algorithmic innovation within the existing system, parallelisation or a combination of effects.

3 Effect of object-oriented design

The BT LVCR system has been written in C++. In a run for a 20k word vocabulary, 1188 different functions were called. Object-orientation is a necessary way of coping with this complexity. Even so, a class-browser, such as SNiFF [10], is essential for retrospective analysis of the system. Object-orientation involves partitioning of data. Within an object, data are normally held privately or in protected state (available only to derived classes). Partitioning of data deters performance optimisations arising from merging functions.[2] Equally, careful consideration has to be given to how a system is decomposed as a prelude to parallelisation. Experimental systems to combine parallelism and the object-oriented paradigm are documented in [11]. A class of synchronisation and communication primitives which do not extend the language is provided in [12], based on Parmacs. However, [13] notes that simply adding these primitives does not encapsulate parallelism suggesting adding path expressions to remove the inflexibility. In the present design, we were constrained by pragmatic considerations in introducing a parallel structure.

4 Processing difficulties

Processing on workstations is an order of magnitude away from real time, assuming a 10 ms frame acquisition window, if an $n-$best single-pass search is made. Formation of the initial feature vector is a task that is well understood and can be delegated to Digital Signal Processors (DSP's). The Viterbi search algorithm [14], based on a simple maximal optimality condition, has made the subsequent network search at least feasible on uni-processors. The Viterbi search is breadth first and synchronous, not asynchronous and depth first which might be more suited to parallel computation. A beam search [15] is a further pruning option, whereby available routes through the network are thresholded. Beam-pruning with two-tiered score thresholding, signatures [16], and path merging [17] have been added to the BT system to further reduce search complexity. However, it is at the network decoding phase that more processing power still needs to be deployed if no further radical pruning heuristics are forthcoming.

[2] In this context, the term 'function' seems more appropriate than the object-oriented term 'method'.

BBN's HARC system took about twenty times real time for an $n-$best search on the $992-$word vocabulary DARPA test with word pair perplexity of 60. Dual TI C30 DSP's were used to find the feature vectors, while a Sun4 workstation performed network parsing. To improve the speed [18] to double real-time, a two-pass iteration was made, though language parsing must be added on to this time. AT&T implemented a system for the same task on a symmetric multiprocessor (SMP) with four MIPS R3000 processors, but for a full search again recorded twenty times real time. A multicomputer was designed [3] with 128 DSP's in a proprietary interconnection topology, though the system employed to take a standard DARPA test was one node consisting of sixteen processors. Using localised memory and store-and-forward communication, on the AT&T system, results in a non-linear growth in communication overhead and hence in the number of processors needed for larger vocabularies.

The AT&T system has a pipeline architecture; possible because the component parts of the decoder system are separable as feature extraction, mixture scoring, phone scoring, word scoring, and phrase scoring. Other systems such as the Cambridge University HTK system, also with high accuracy scoring, are organised as a token-passing network [19], which is not easily broken into a pipeline. As mentioned in Section 1, the BT LVCR system is also of the token-passing network type.

5 The BT System

5.1 Problems to be overcome

The existing BT design, Fig. 1 [20], resists decomposition due to the close coupling of the network update procedures. 49-way acoustic feature vectors (frames) arriving every 10 ms, are applied to each active node of the recognition network. Real, noise, and null nodes embody models for respectively speech, noise, and word connections. The nodes are kept in global lists, necessary because a variety of update procedures are applied. In particular, dormant nodes are reused from application-maintained memory pools without variable delay due to system memory allocation. Large networks, for unconstrained speech or language models beyond bi-gram, are dynamically extended when a token reaches a network boundary. Network extension makes parallel decomposition by statically forming sub-networks problematic because of the need to load balance and hence repeatedly re-divide the network.

5.2 Execution analysis of the LVCR system

The top-20 functions call graph, Fig. 2, for 97 utterances on the 20k Wall Street Journal test with bi-grams, showed 67% of total computation time including 3% load-time, was taken up by the 'feedforward' update. The branch of function calls, Fig. 3, resulting in the calculation of state output probabilities, bprobs, was uncharacteristically free of sub-function calls which otherwise can give rise

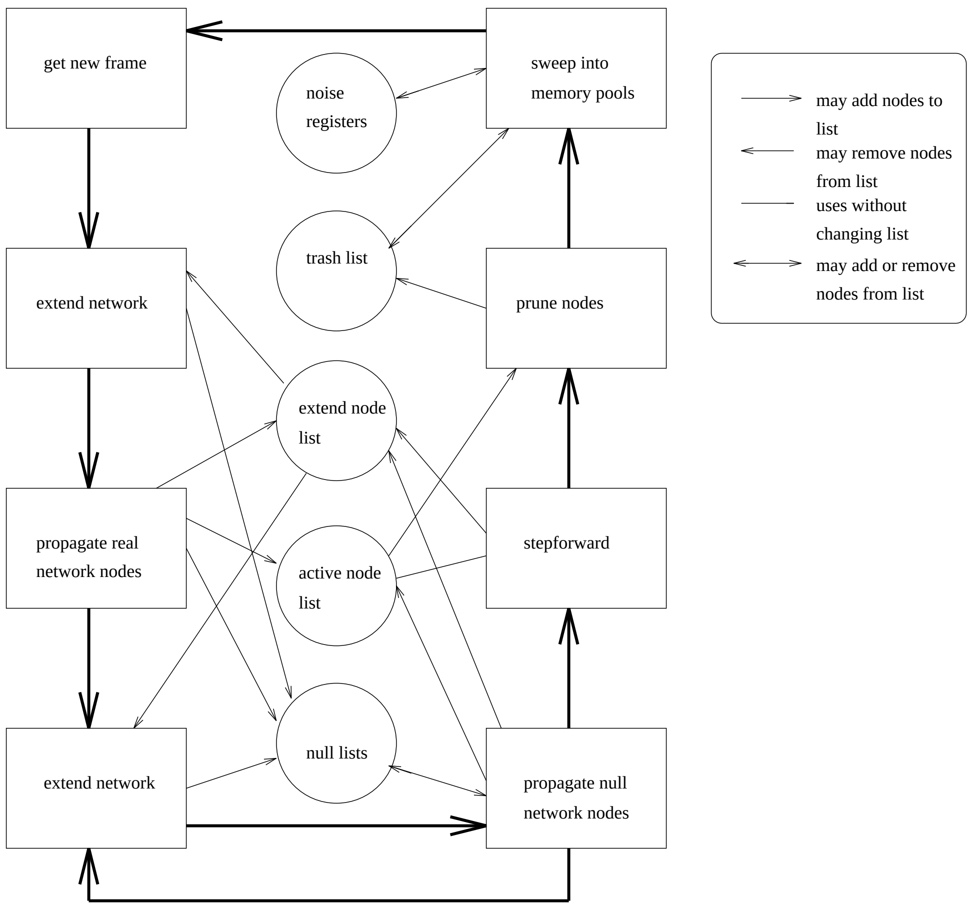

Fig. 1. Network processing cycle

to unforeseen data dependencies. Other parameters such as state transition probabilities, aprobs, remain fixed. The seemingly redundant level of indirection for mode-level checks enables future sharing of modes, which model variety in speech intonation. 44% of time is spent calculating a quadratic part of the sum forming the mixture of unimodal Gaussian densities which comprise the core of any state. 6,641 nodes were present in the mean for 395 frames representing 4s of speech.

6 Parallel architecture

The parallel architecture that was arrived at can be considered to be a pipeline, Fig. 4, though no overlapped processing takes place across the pipeline stages because of the synchronous nature of the processing. The first of the two pipeline

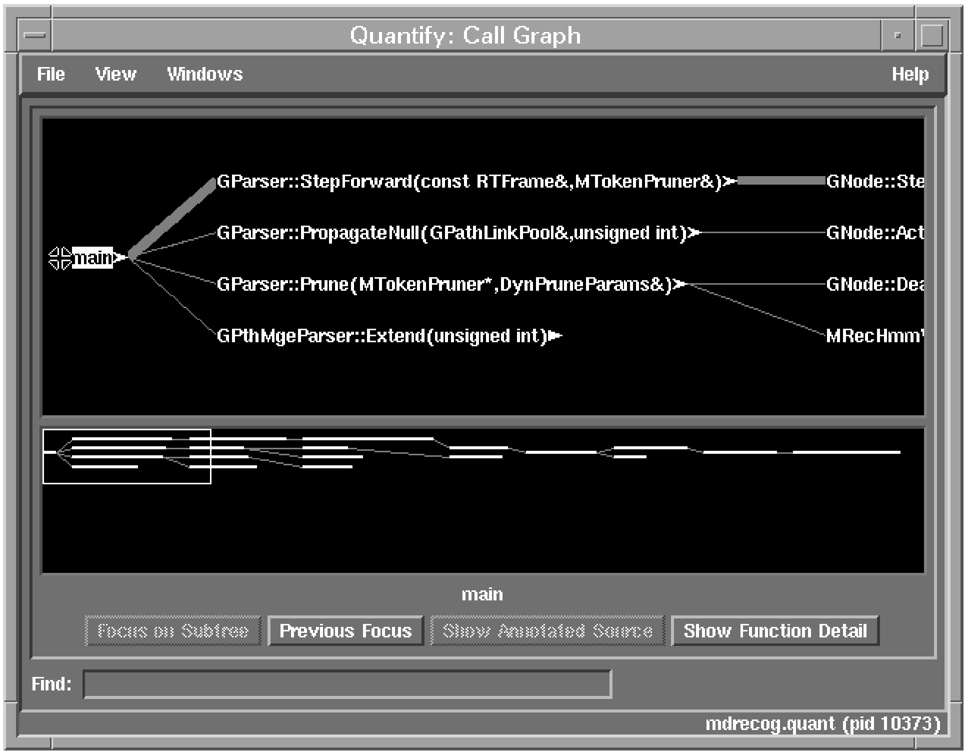
Quantify: Call Graph
File
View
Windows
Help
GParser::StepForward(const RTFrame&,MTokenPruner&)
GParser::PropagateNull(GPathLinkPool&,unsigned int)
GParser::Prune(MTokenPruner*,DynPruneParams&)
GPthMgeParser::Extend(unsigned int)
main
GNode::Ste
GNode::Act
GNode::Dea
MRecHmm
main
Focus on Subtree
Previous Focus
Show Annotated Source
Show Function Detail
Find:
mdrecog.quant (pid 10373)

Fig. 2. Extract from high-level call graph, showing call intensity by link width

stages employs a data-farm. A data-manager farms out the computationally-intensive low-level probability calculations to a set of worker processes, with some work taking place local to the data-farmer while communication occurs.

The standard PVM library of message-passing communication primitives was used in the prototype, run over a network of HP Apollo 700 series workstations. Worker processes each hold copies of a pool of 1,954 tri-state models (and 50 one-state noise models). State-level parallelism requires broadcast of the current model identity (4 bytes), the prototype system needing no global knowledge if started at the 55% point in Fig. 3. By a small breach of object demarcation, whereby at the node object level (with embedded HMM) the state object update history was inspected, the number of messages was sharply reduced, as only one in twelve state bprobs for any frame were newly calculated. A small overhead from manipulation of the node active list enables the parallel ratio to reach the 67% point, collection of thresholding levels then being centralised. Mode level checking when introduced would check for replications at the local level thus limiting the loss in efficiency.

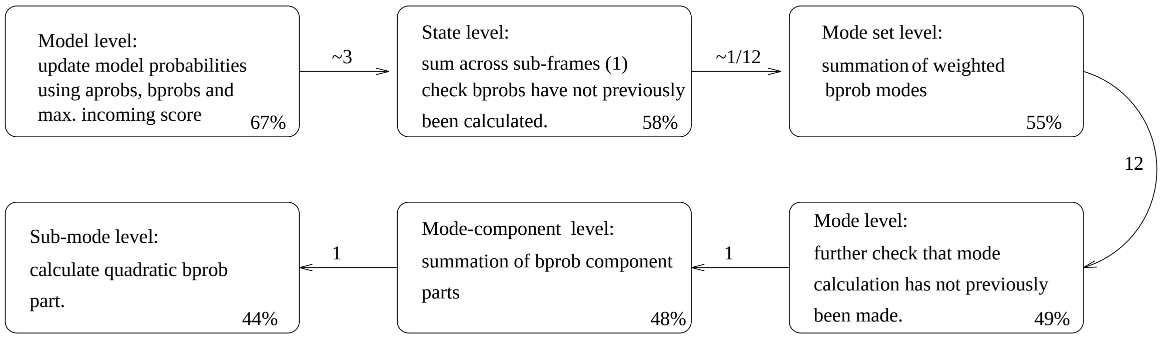
Model level:
update model probabilities using aprobs, bprobs and max. incoming score
67%
~3
State level:
sum across sub-frames (1)
check bprobs have not previously been calculated.
58%
~1/12
Mode set level:
summation of weighted bprob modes
55%
12
Mode level:
further check that mode calculation has not previously been made.
49%
1
Mode-component level:
summation of bprob component parts
48%
1
Sub-mode level:
calculate quadratic bprob part.
44%

Fig. 3. Probability calculation function hierarchy with call ratios

POSIX-standard pthreads are proposed in the second pipeline stage whereby the residual system is parallelised. Propagation of null nodes and real nodes, sweeping-up nodes (thus avoiding over-use of free store), and recognition network pruning functions all have a similar structure. For example, the prune function first establishes pruning levels, which are then globally available for all spawned threads. Once spawned, the host thread of control, i.e. the prune function, is descheduled until it is reawoken by the completion of its worker threads. Worker threads proceed by taking a node(s) from the active list, deciding whether pruning should take place, and updating the trash list and active list if pruning takes place. The large number of active nodes allows granularity to be adapted to circumstances and the few points of potential serialization, requiring locks, increases the potential scale-up.

7 Future implementation on an SMP

We considered whether a widely-available type of parallel machine would be sufficient to parallelize the complete system. On a symmetric multiprocessor (SMP), the thread manager would share one processor with the data manager. Efficient message-passing is available for SMPs [21] in addition to threads. Triphones, usual for continuous speech, restrict potential parallelism but with node level decomposition, Table 1, an eight processor machine would approach the required fivefold speed-up while a four processor machine would reduce turnaround during testing. The estimate assumes conservatively that half of the residual system is parallel, while scaling of the system to this level is irrespective of the frame processing workload distribution over time. Inlining of some functions is available as a further sequential optimisation.

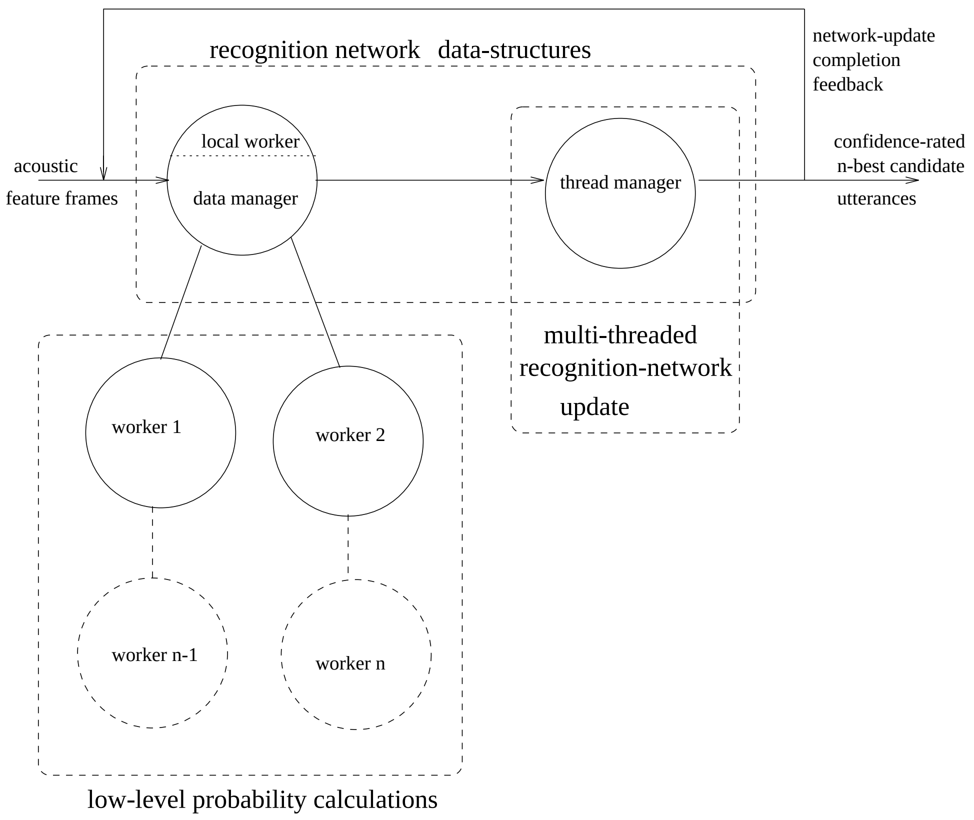

Fig. 4. Synchronous LVCR process pipeline

parallelization	stage 1		stage 1 & 2	
level/processors:	4	8	4	8
state	1.58	1.92	1.88	1.94
node	2.01	2.42	2.68	3.71

Table 1. Speed-up estimate (Amdahl's law)

8 Conclusion

Speech recognition exhibits many features of present day systems that must be taken into account if parallelisation is used as a performance enhancing technique:

– The parallelisation should be conservative, i.e. preserve as much of the established structure as possible, given that the structure is well-proven and has been arrived at after considerable effort, under a sequential programming model.

– The system is complex resulting in considerable logistical difficulties at the testing phase, which a parallel structure may ameliorate.
– Object-oriented design is a way of managing complexity but can impose restrictions which should be considered. Given the advantages of object-oriented design in terms of managing large projects, a parallel design cannot ignore this issue.
– Software standardisation is important for this type of system, which means standards for parallel systems such as PVM and MPI and POSIX threads should be used and developed.
– Parallelisation is not the only way to improve performance and indeed algorithmic innovation in speech recognition, for example tied states, can result in greater improvements.
– The system exhibits algorithmic discontinuity which requires the combination of two different parallel programming paradigms.

Presented with a large-scale system, the prospect may seem daunting to the system analyst. It is important to examine the dynamic behaviour as well as the static structure of the system. Indeed doing this allowed a credible parallel architecture to be formed from the BT LVCR, combining two parallel programming paradigms: the data-farm and the multi-threaded shared-memory model. A high-level design with low-bandwidth message-passing has been developed and prototyped to cope with the major part of the processing. The complete parallel system, which is portable, hardly disturbs the existing integrated structure, but with modest hardware outlay is confidently estimated to bring significant performance gain.

Acknowledgements

This work was carried out under EPSRC research contract GR/K40277 'Parallel software tools for embedded signal processing applications' as part of the EPSRC Portable Software Tools for Parallel Architectures directed programme.

The software described in this paper was developed at BT Laboratories, Martlesham Heath, UK. The authors are indebted to BT Speech Technology Unit for permission to publish the experimental work on parallelising speech recognition. Opinions expressed in this paper are the authors' personal views, and should not be assumed to represent the views of BT.

References

1. L. R. Rabiner, Juang B.-H., and C.-H. Lee. An overview of automatic speech recognition. In Lee C.-H., F. K. Soong, and K. K. Paliwal, editors, *Automatic Speech and Speaker Recognition Advanced Topics*. Kluwer, Boston, 1996.

2. P. C. Woodland, C. J. Leggetter, J. J. Odell, V. Valtchev, and S. J. Young. The 1994 HTK large vocabulary speech recognition system. In *ICASSP'95*, volume I, pages 73–76, 1995.
3. S. Glinski and D. Roe. Spoken language recognition on a DSP array processor. *IEEE Transactions on Parallel and Distributed Systems*, 5(7):697–703, July 1994.
4. R. Moore. Recogition – the stochastic modelling approach. In C. Rowden, editor, *Speech Processing*, pages 223–255. McGraw-Hill, London, 1993.
5. L. R. Rabiner. A tutorial on Hidden Markov Models and selected applications in speech recognition. *Proceedings of the IEEE*, 77:257–285, February 1989.
6. L. A. Liporace. Maximum likelihood estimation for multivariate observations of Markov sources. *IEEE Transactions on Information Theory*, 28(5):729–734, September 1982.
7. W. Turin. Unidirectional and parallel Baum-Welch algorithms. *IEEE Transactions on Speech and Audio Processing*, 6(6):516–523, November 1998.
8. S. K. Das and M. A. Picheny. Issues in practical large vocabulary isolated word recognition: The IBM Tangora system. In C-H. Lee, F. K. Soong, and K. K. Paliwal, editors, *Automatic Speech and Speaker Recognition Advanced Topics*, pages 457–479. Kluwer, Boston, 1996.
9. S. Baker. *CORBA Distributed Objects Using Orbix*. Addison-Wesley, Harlow, UK, 1997.
10. TakeFive Software GmbH, Stichting Mathematisch Centrum, Amsterdam, the Netherlands. *SNiFF+ Release 2.2 User's Guide and Reference*, 1996.
11. G. V. Wilson and P. Lu, editors. *Parallel Programming Using C++*. MIT, Cambridge, MA, 1996.
12. B. Beck. Shared-memory parallel programming in C++. *IEEE Software*, 7(4):38–48, July 1990.
13. Y. Wu and T. G. Lewis. Parallelism encapsulation in C++. In *International Conference on Parallel Processing*, volume II, pages 35–42. Pennsylvania State University, 1990.
14. G. D. Forney. The Viterbi algorithm. *Proceedings of the IEEE*, 61(3):268–278, March 1973.
15. R. Umbach and H. Ney. Improvements in beam search for $10,000-$word continuous-speech recognition. *IEEE Transactions on Speech and Audio Processing*, 2(2):353–356, April 1994.
16. S. P. A. Ringland. Application of grammar constraints to ASR using signature functions. In *Speech Recognition and Coding*, pages 260–263. Springer, Berlin, 1995. Volume 147 NATO ASI Series F.
17. S. Hovell. The incorporation of path merging in a dynamic network parser. In *ESCA, EuroSpeech97*, volume 1, pages 155–158, 1997.
18. S. Austin, R. Schwartz, and P. Placeway. The forward-backward search algorithm. In *International Conference on Acoustics, Speech, and Signal Processing*, volume 1, pages 697–700, 1991.
19. S. Young. A review of large-vocabulary continuous-speech recognition. *IEEE Signal Processing Magazine*, pages 45–57, September 1996.
20. D. Ollason, S. Hovell, and M. Wright. Requirements and design of the new continuous speech recognition parser – the Grid. Technical report, BT Laboratories, Martlesham Heath, Ipswich, IP5 3RE, UK, 1998.
21. S. S. Lumetta and D. E. Culler. Managing concurrent access for shared memory active messages. In *IPPS/SPDP'98*, 1998. 7 pages from http://now.CS.berkeley.EDU/Papers2.