An A* algorithm for very large vocabulary continuous speech recognition[1]

P. Kenny, R. Hollan[2], G. Boulianne, H. Garudadri, M. Lennig[2] and D. O'Shaughnessy

INRS–Télécommunications
3 Place du Commerce
Montreal, Quebec, Canada H3E 1H6

ABSTRACT

We present a new search algorithm for very large vocabulary continuous speech recognition. Continuous speech recognition with this algorithm is only about 10 times more computationally expensive than isolated word recognition. We report preliminary recognition results obtained by testing our recognizer on "books on tape" using a 60,000 word dictionary.

1. Introduction

In this paper we will give a preliminary report on our efforts to extend our earlier work on very large vocabulary isolated word recognition [16, 17, 19] to continuous speech tasks. We are aiming to perform speaker-dependent continuous speech recognition using a trigram language model and a vocabulary of 50,000–100,000 words.

Although the problem of very large vocabulary isolated word recognition has largely been solved [1, 19], no experiments have yet been conducted in continuous speech recognition with comparably large vocabularies because the search problem is so formidable. The best known approach to the search problem uses the word as the fundamental search unit and a stack decoding algorithm [11, 20] (also known as an A* search [14]). The effectiveness of this approach depends on having a good fast match strategy to identify candidate words whenever a word boundary is hypothesized. Many different fast match algorithms have been proposed [3, 4, 5, 15] but they have yet to be shown to perform satisfactorily on continuous speech tasks having a vocabulary larger than 5,000 words [2].

An alternative approach developped by Phillips [6] (on a 10,000 word vocabulary in German) uses the phoneme as the fundamental search unit and consists of a Viterbi search of the hidden Markov model obtained by combining phoneme HMMs with a Markovian language model (such as a trigram model). Aggressive pruning is necessary since the search space is very large. (For instance, if a trigram language model is used then one copy of the lexical tree is needed for every possible bigram.) The phoneme inventory is sufficently small that exact matches can be calculated whenever they are needed. However, it remains to be seen whether the phoneme unit is capable of attaining respectable accuracies on very large-scale recognition tasks.

A new type of bi-directional search strategy has emerged recently. The basic idea is to guide the search by means of a heuristic obtained by first carrying out an inexpensive search in the reverse time direction (subject to relatively weak linguistic and/or lexical constraints). This type of approach appears to have been discovered independently by several groups and has been shown to work effectively on a variety of applications [8, 9, 16]. In [16] we presented a very efficient algorithm for very large vocabulary isolated word recognition using this paradigm and the phoneme as the fundamental search unit. Our current efforts are devoted to extending this algorithm to continuous speech.

This isolated word recognition algorithm is an A* algorithm which uses a heuristic obtained by searching a phonetic graph [16] which imposes triphone phonotactic constraints on phoneme strings. This search is conducted using the standard Viterbi algorithm in the reverse time direction (starting from the end of the utterance). In addition to providing a very efficient heuristic, a major advantage in using triphone phonotactic constraints is that it enables us to identify the endpoint of the third-to-last phoneme in each partial recognition hypothesis with a high degree of accuracy, thereby substantially reducing the size of the search space. Another innovative feature of this algorithm is that it computes the acoustic matches of every segment of data with each of the phoneme models ('the point scores') before carrying out the A* search. (The principal reason for doing so is that this approach enables segment-level features such as phoneme durations to be modelled in an optimal way [16, 18, 7].)

The effectiveness of the triphone heuristic depends more on the quality of the phoneme models than on the size of the search space. Thus we have found that, even without any pruning, the isolated word recognition algorithm runs more quickly on a 60,000-word recognition task with clean speech and speaker-dependent models than on a 1,600-word task with telephone speech and speaker-independent models [16]. In the speaker-dependent case most of the computaion is taken up by the pre-processing (the calculation of the point scores

[1]This work was supported by the Natural Sciences and Engineering Research Council of Canada

[2]Also with Bell-Northern Research, Montreal, Canada

and the Viterbi search) and the A* search itself accounts for only about 1% of the total. In extending the algorithm to continuous speech (also with a 60,000 word vocabulary and speaker dependent models), we have found that the amount of pre-processing per unit time remains essentially the same, but the amount of computation needed for the A* search increases by three orders of magnitude. Hence, *the total computational demands of the algorithm only increase by a factor of about 10.*

The experiments reported here have been conducted using phoneme models, but the search algorithm can be extended to accommodate allophone models (including cross-word allophones) fairly easily.

2. Block Processing

In developping our algorithms, we have decided to work with commercially distributed books on tape (analog recordings of well known novels). Half of each recording is used as training data and the other half for testing and we use an optical character recognizer to read the accompanying texts. Since this data is not segmented into sentences we have designed our training and recogniton algorithms to work with chunks of data of arbitrary size. This means that the data has to be processed in blocks which can fit comfortably into memory.

Our approach is to use an A* search in each block which is similar to the isolated word recognition algorithm except insofar as word boundaries are not known in advance and a trigram language model is used in the scoring procedure. As in the isolated word case, an admissible heuristic is obtained by means of an initial Viterbi search through a graph which imposes triphone phonotactic constraints on phone strings. The A* search generates a list of theories (partial phonemic transcriptions together with word histories) for the speech data up to the end of the block[3] As soon as the list of theories for the current block has been obtained, the block is swapped out of memory and the search of the next block begins using this list to initialize the stack.

This list of theories plays the same role as the beam used in a time synchronous Viterbi search. The Markov property of the trigram language model allows us to merge theories that have identical recent pasts but different remote pasts so the number of theories that have to be generated at the end of each block (the 'beam width') can held fixed without running the risk of losing the optimal theory. In order to pursue the search in subsequent blocks, the only information needed concerns the recent pasts of these theories. By logging the information concerning the remote pasts to disk we are able to ensure that the memory required to recognize a file is independent of its length (instead of increasing exponentially with the length of the file as would be necessary without merging and block processing).

For the last block in a file it is only necessary to generate a single recognition hypothesis and, once the last block has been processed, the transcription of the entire utterance can be obtained by back-tracking. The recognition algorithm can therefore be viewed globally as a beam search and locally as an A* search.

3. The Heuristic

Broadly speaking, an A* search of the data in a block proceeds as follows. At each iteration of the algorithm, there is a sorted list (or 'stack') of theories each with a heuristic score. This heuristic score is calculated by combining the exact likelihood score of the speech data accounted for by the theory (using phoneme HMMs and the language model) with an overestimate of the score of the remaining data on the optimal extension of the theory permitted by the lexicon and the language model. The theory with the highest heuristic score is expanded, meaning that for each of the one-phoneme extensions permitted by the lexicon the heuristic score of the extended theory is calculated and the extended theory is inserted into the stack at the appropriate position. This process is iterated until sufficiently many theories satisfying a suitable termination criterion have been generated.

For the time being, we have decided to ignore the issue of overestimating language model scores altogether in constructing the heuristic (that is, we use an estimate of 1 for the language model probability of any extension of a given theory). Our strategy for overestimating acoustic scores is essentially the same as in the isolated word case, that is, we conduct an exhaustive search backwards in time through a phonetic graph which imposes triphone phonotactic constraints on phoneme strings rather than full lexical constraints and enables the third-to-last phoneme in a given partial phonemic transcription to be accurately endpointed. Naturally, the triphone phonotactic constraints must take account of triphones which occur at word boundaries. The simplest graph with these properties is specified as follows:

1. *Nodes:* there is one node for every possible diphone fg
2. *Branches:* for every legitimate triphone fgh (that is, a triphone that can be obtained by concatenating the phonemic transcriptions of words in the dictionary) there is a branch from the node corresponding to the diphone fg to the node corresponding to the diphone gh
3. *Branch Labels:* if fgh is a legitimate triphone then the branch from the node fg to the node gh carries the label f.

[3]More precisely, each of the theories generated has the property that all of the hypothesized end points for the third-to-last phoneme in the partial phonemic transcription are beyond the end of the block. The partial phonemic transcription need not end at a word boundary.

Denote this graph by G^*. It is easy to see that this graph imposes triphone constraints on phoneme strings, that is, if $g_1, g_2, g_3 \ldots$ is the sequence of phoneme labels encountered on a given path through G^* then every triple $g_k g_{k+1} g_{k+2}$ $(k = 1, 2, \ldots)$ is a legitimate triphone. The labelling scheme (3) is chosen so that the endpointing condition is satisfied (see the next section).

4. Searching a block

In order to search a block extending from times T_1 to T_2, we first construct the hidden Markov model corresponding to the graph G^* [16] and, for a suitably chosen positive integer Δ, we perform a Viterbi search backwards in time through this HMM from time $T_2 + \Delta$ to time T_1. (The condition used to determine the parameter Δ is given below.) The boundary condition used to initialize the search is that the backward probability at every state in the model at time $T_2 + \Delta$ is 1. For each node n in G^* and each $t = T_1 - 1, \ldots, T_2 + \Delta - 1$ we thus obtain the Viterbi score of the data in the interval $[t + 1, T_2 + \Delta]$ on the best path in G^* which leaves n at time t and is subject to no constraints on the state in the model occupied at time $T + \Delta$; denote this quantity by $\beta_t^*(n)$.

Suppose we are given a partial phonemic transcription $f_1 \ldots f_k$. Let n be the node corresponding to the diphone $f_{k-1} f_k$ and for each time t, let $\alpha_t(f_1 \ldots f_{k-2})$ denote the Viterbi score of all of the data up to time t (starting from the beginning of the utterance) for the truncated transcription $f_1 \ldots f_{k-2}$. Since $\beta_t^*(n)$ is the Viterbi score of the data in the interval $[t + 1, T + \Delta]$ on the best path in G^* which leaves n at time t and the construction of G^* constrains this path to pass first through a branch labelled f_{k-1} and then through a branch labelled f_k, it is reasonable to estimate the endpoint of the phoneme f_{k-2} as

$$\operatorname*{argmax}_t \alpha_t(f_1 \ldots f_{k-2}) \beta_t^*(n).$$

In the case of clean speech and speaker-dependent models, this estimate turns out to be exact almost all of the time [16] but it is safer to hypothesize several end points (for instance by taking the five values of t for which

$$\alpha_t(f_1 \ldots f_{k-2}) \beta_t^*(n)$$

is largest).

A stack entry (theory) θ is a septuple $(\mathbf{w}, \mathbf{f}, m, n, \sigma, \{\alpha_t\}, S)$ where

1. $\mathbf{w} = w_1 \ldots w_n$ is a word history.
2. $\mathbf{f} = f_1 \ldots f_k$ is a partial phonemic transcription which may extend into a word following w_n (but there are no complete words after w_n in the partial transcription **f**)
3. m is a node in the lexical tree [16] corresponding to the part **f** which extends beyond w_n, if any; m is the root node of the lexical tree otherwise
4. n is the node in the graph G^* which corresponds to the diphone $f_{k-1} f_k$
5. σ is the current state of the trigram language model; there are three possibilities depending on whether the word following w_n is predicted using a trigram distribution $P(\cdot|w_{n-1} w_n)$, a bigram distribution $P(\cdot|w_n)$ or a unigram distribution $P(\cdot)$
6. for each endpoint hypothesis t, α_t is the Viterbi score of the data up to time t against the model for the truncated transcription $f_1 \ldots f_{k-2}$
7. S is the heuristic score which is given by

$$S = P(\mathbf{w}) \max_t \alpha_t(f_1 \ldots f_{k-2}) \beta_t^*(n)$$

where $P(\mathbf{w})$ is the probability of the word string **w** calculated using the trigram language model.

The reason why both **w** and **f** have to be specified is that different words may have the same transcription and different transcriptions may correspond to the same word. Obviously it is redundant to specify m, n and σ in addition to **w** and **f** but it is convenient to do so.

A stack entry is said to be *complete* if all of its hypothesized endpoints are to the right of T_2. The parameter Δ is determined empirically by the condition that the exact endpoint of a complete stack entry should always be included among the hypothesized endpoints. (Since it is not actually possible because of memory limitations to carry around sufficient information with each theory to be able to generate its segmentation, we test this condition by verifying that the acoustic score of the global transcription found by the recognizer of the data in each file is the same as the score found by the training program when it is run with this transcription.)

At the start of the search, the stack is initialized using the list of theories generated by searching the previous block (ending at time T_1). Each of these has the property that all of its hypothesised endpoints are to the right of T_1, so the speech data prior to the beginning of the current block is no longer needed. The search terminates when sufficiently many complete theories have been generated at which point the next block is swapped into memory and a new search begins.

The Markov property of the trigram language model enables us to merge theories that have identical recent pasts but different remote pasts. Specifically, suppose we have two theories $\theta = (\mathbf{w}, \mathbf{f}, m, n, \sigma, \{\alpha_t\}, S)$ and $\theta' = (\mathbf{w}', \mathbf{f}', m', n', \sigma', \{\alpha_t'\}, S')$ such that $m = m'$, $n = n'$ and

$\sigma = \sigma'$. (In this case we will say that θ and θ' are equivalent.) The future extensions of both theories which best account for the data starting at any given time (subject to lexical and language model constraints) will be identical. Thus if it happens that t is on the list of hypothesized endpoints for both theories and

$$P(\mathbf{w}')\alpha'_t < P(\mathbf{w})\alpha_t$$

then we can remove t from the hypothesis list for the second theory without running the risk of losing the optimal path. In practice, the condition $n = n'$ means that the list of hypothesized endpoints for both theories will be the same (except in very rare cases). Furthermore, if this inequality holds for one such t then it is typically because the first theory gives a better fit to the remote past than the second theory; hence it will usually be the case that if the inequality holds for one t then it will hold for all t and the second theory can be pruned away completely.

We can take advantage of this fact to speed up the A* search by maintaining a list of 'merge buckets' consisting of all the equivalence classes of theories encountered in the course of the search. Associated with each equivalence class we have an array of forward scores $\{A_t\}$ which is updated throughout the search. For each t, A_t is defined to be

$$\max_{\theta} P(\mathbf{w})\alpha_t$$

where θ extends over all theories $(\mathbf{w}, \mathbf{f}, m, n, \sigma, \{\alpha_t\}, S)$ in the given equivalence class that have been encountered so far (in the course of searching the current block). When a new theory $\theta' = (\mathbf{w}', \mathbf{f}', m', n', \sigma', \{\alpha'_t\}, S')$ in this equivalence class comes to be inserted into the stack we can test to see if the inequality

$$P(\mathbf{w}')\alpha'_t < A_t$$

holds for each hypothesized endpoint t. If it does, then we can prune this endpoint hypothesis before entering the theory into the stack; if not, then A_t is updated and the endpoint hypothesis has to be retained.

We have not been able to implement this scheme fully because of memory limitations. In practice, we only invoke merging when a word boundary is hypothesized so the only merge buckets generated in the course of the search are those which correspond to theories for which m is the root node of the lexical tree. (However, before starting the search we prune the list of hypotheses generated by searching the previous block by merging at arbitrary phoneme boundaries and we use this pruned list to initialize the stack.)

5. Experimental Results

Our first experimental results obtained from two books on tape (analog recordings) appear in Table 1. The first book *"White Fang"* by Jack London was recorded by a male speaker; the second book *"Washington Square"* by Henry James was recorded by a female.

Book	Training Set Size	Test Set Size	Accuracy %
WF	18,012	730	51.4
WS	16,257	786	73.2

Table 1: Preliminary recognition results.

The second and third columns in this table give the training and test set sizes in words; the third column gives the accuracy which is calculated as

$$\frac{N - (\text{Substitutions} + \frac{1}{2}[\text{Deletions} + \text{Insertions}])}{N}$$

where N is the size of the test set.

These experiments were run using 41 phonemic mixture HMMs for each speaker, a 60,000 word dictionary which was edited to include all of the words in both books (1.5% of the words in each of the books had to be added) and a trigram language model which was trained on 60,000,000 words of newspaper texts. No attempt was made to tailor the language model to the task domains. The test set perplexity was 1,743 in the case of *"White Fang"* and 749 in the case of *"Washington Sqaure"*. These perplexities can be reduced to 576 and 347 respectively by smoothing the language model statistics using word frequencies collected from the training set but we did not take advantage of this in running our experiments.

The CPU time required to run the *"Washington Square"* experiment was 120 times real time on a HP 720 workstation. We had to use a block advance of only 10 frames (1 frame = 10 ms) in order to keep the stack size within reasonable bounds. The parameter Δ was set at 140 frames. The stack was implemented as a heap and the maximum number of stack entries was set to 60,000. (When this figure is reached, the size of the stack is cut back to 30,000). The number of theories passed from one block to the next was 3,000. In the case of *"White Fang"* a larger stack was needed to prevent search errors and the execution time was longer. We will have to run the recognizer on several more speakers before attempting to optimize these parameters.

6. Future Work

There is obviously a substantial amount of work to be done to improve both the accuracy and speed of our recognition algorithm. These problems are not independent of each other: we expect that the search algorithm will run faster with a better language model and better acoustic models and conversely

improvements in the search algorithm will lead to fewer search errors and hence higher recognition rates.

At present, the only types of pruning that have been implemented are the merging of theories having identical recent pasts and the limitations on the size of the stack used in searching a block as well as the length of the list of theories passed from one block to the next. Several other possibilities remain to be explored. We may be able to get away with a beam search in the calculation of the β^*'s. It may be possible to prune hypotheses based on poor local acoustic matches (evaluated using the point scores or the β^*'s or a combination of the two). Since the branching factor at the root node of the lexical tree is 41, we would expect a big payoff if this type of pruning can be made to work successfully whenever a word boundary is hypothesized. Also the limitations on the stack size and the length of the hypothesis lists passed from one block to the next should probably be made threshold-dependent rather than preset.

In our current implementation, we have not taken full advantage of the sparseness of the language model statistics (the the number of bigrams w_1w_2 for which we have trained trigram distributions $P(\cdot|w_1w_2)$ is relatively small and these distributions are typically concentrated on very small subsets of the dictionary). Our present implementation gets some mileage out of this fact by using the notion of a language model state (σ) to determine when theories can be merged, but more work remains to be done. Adding a language model component to the heuristic would probably help as well.

As for acoustic modelling, we can expect a major improvement by using allophone models. From the way we have presented the algorithm, it may appear that we have locked ourselves into the choice of the phoneme as the modelling unit so it may come as a surprise to learn that our algorithm can accommodate allophone models in a natural way (without unduly increasing the amount of computation needed). The only restriction is that the allophones of a given phoneme should be defined by looking at contexts which extend no more than two phonemes to the right (there is no restriction on left contexts).

Since this is an important issue, we take the time to explain what is involved here. Certainly, we would encounter problems if we were to proceed in a straightforward manner and recompile the lexicon in terms of allophonic transcriptions rather than phonemic transcriptions. Firstly, the structure of the lexical tree would have to be radically altered to accomodate allophones defined by contexts which extend across word boundaries. Secondly, with a reasonably large allophone inventory (say a few thousand), the size of the graph G^* would become so large as to make the computation of the β^*'s practically infeasible. So the approach is to retain the structure of the lexical tree and the graph G^* determined by the phonemic transcriptions and perform the translation to allophonic transcriptions on-line. (The same method could be used to incorporate phonological rules whose domain spans word boundaries.)

Suppose we have a theory θ whose partial phonemic transcription is $f_1 \dots f_k$. We have to explain how $\alpha_t(f_1 \dots f_{k-2})$ and $\beta_t^*(n)$ are computed when allophone models are used.

In calculating $\alpha_t(f_1 \dots f_{k-2})$, we simply use the appropriate allophonic models for each of the phonemes $f_1, \dots, f_{k-2}$. (Note that sufficent information concerning the right contexts is available to determine which allophones to use).

It is natural to organize the calculation of the β^*'s in terms of the point scores. To see how this goes, consider first the case of phoneme models. The β^*'s can be calculated using the block Viterbi algorithm [16]. Recall that, for a given node n, the first two phoneme labels on any path in G^* which starts at node n are uniquely determined. Denote the first phoneme by f and the second by g. The recursion formula is

$$\beta_t^*(n) = \max_{t'>t} V([t+1,t']|f) \max_{n'} \beta_{t'}^*(n')$$

where n' ranges over all nodes such that (n, f, n') is a branch in G^* and $V([t+1,t']|f)$ denotes the Viterbi score of the data in the interval $[t+1,t']$ calculated using the f model.

In the case of allophone models, we can calculate the backward probabilities using the recursion formula

$$\beta_t^*(n) = \max_{t'>t} \max_{\phi} V([t+1,t']|\phi) \max_{n'} \beta_{t'}^*(n')$$

where, as before, n' ranges over all nodes such that (n, f, n') is a branch in G^* and ϕ ranges over all the allophones of f determined by the condition that the phoneme immediately following f is g. It is obvious that the backward probabilities calculated in this way provide an overestimate of the acoustic score of the data which has not yet been accounted for on the optimal extension of the theory θ so the admissibility condition is satisfied. Of course, it is not possible to endpoint the phoneme f_{k-2} exactly in this case since the allophone models needed to score f_{k-1} and f_k cannot be determined until the theory has been extended. This does not present a problem since we already have a mechanism in place for handling multiple endpoint hypotheses.

Finally, we have recently embarked on a project to parallelize the search algorithm with a view to obtaining a real-time response on a platform supplied by ALEX Informatique containing 48 i860's and 48 T800 transputers.

References

1. Averbuch A., "Experiments with the Tangora 20,000 word speech recognizer," *Proc. ICASSP 87*, pp. 701-704, 1987.
2. Bahl, L.R. et al. "Large Vocabulary Natural Language Continuous Speech Recognition", *Proc. ICASSP 89*, pp. 465–467, 1989.

3. Bahl, L.R., De Gennaro S.V., Gopalakrishnan, P.S., Mercer, R.L., "A fast approximate acoustic match for large vocabulary speech recognition", personal communication.

4. Bahl, L.R., Bakis, R., de Souza, P.V., Mercer, R.L., "Obtaining candidate words by polling in a large vocabulary speech recognition system", *Proc. ICASSP 88*, pp. 489–492, 1988.

5. Fissore, L., Laface, P., Micca, G., Pieraccini, R., "Lexical access to large vocabularies for speech recognition", *Proc. ICASSP 89*, pp. 1197–1213, 1989.

6. Steinbiss, V., "A 10 000-word continuous speech recognition system", *Proc. ICASSP 90*, pp. 57–60, 1990.

7. Sagayama, S., "A matrix representation of HMM-based speech recognition algorithms", *Proc. Eurospeech 91*, pp. 1225–1228, 1991.

8. Soong, F.K., Huang, E.-F., "A tree-trellis based fast search for finding the N best sentence hypotheses in continuous speech recognition", *Proc. ICASSP 91*, pp. 705–708, 1991.

9. Zue, V. et al., "Integration of speech recognition and natural language processing in the MIT VOYAGER system", *Proc. ICASSP 91*, pp. 713–716, 1991.

10. Austin, S., Schwartz, R., Placeway, P., "The forward-backward search algorithm", *Proc. ICASSP 91*, pp. 697–700, 1991.

11. Jelinek, F., "A fast sequential decoding algorithm using a stack", *IBM Journal of Research and Development*, **13**, pp. 675–685, 1969.

12. Jelinek, F., "Continuous Speech Recognition by Statistical Methods", *Proc. IEEE*, **64**, 1976.

13. Seitz, F., Gupta, V., Lennig, M., Kenny, P., Deng, L., and Mermelstein, P. "Dictionary for a very large vocabulary word recognition system," *Computer, Speech and Language*, **4**, pp. 193–202, 1990.

14. Nilsson, N., "Principles of artificial intelligence," Tioga Publishing Company, 1982.

15. Gupta, V., Lennig, M., and Mermelstein, P., "Fast search strategy in a large vocabulary word recognizer," J. Acoust. Soc. Am. **84**(6), 2007-2017, 1988

16. Kenny, P., Hollan, R., Gupta, V., Lennig, M., Mermelstein, P., and O'Shaughnessy, D., "A* - admissible heuristics for rapid lexical access", to appear in *IEEE Transactions on Signal Processing*, November 1992.

17. Deng, L., Kenny, P., Lennig, M., Gupta, V., Seitz, F., and Mermelstein, P., "Phonemic hidden Markov models with continuous mixture output densities for large vocabulary word recognition," *IEEE Transactions on Signal Processing*, **39**, pp. 1677–1681, 1991.

18. Kenny, P., Parthasarathy, S., Gupta, V., Lennig, M., Mermelstein, P., and O'Shaughnessy, D., "Energy, Duration and Markov models", *Proc. Eurospeech 91*, pp. 655–658, 1991.

19. Lennig, M., Gupta, V., Kenny, P., Mermelstein, P., O'Shaughnessy, D., "An 86,000-Word Recognizer Based on Phonemic Models", *Proc. DARPA Speech and Natural Language Workshop*, pp. 391–396, 1990.

20. Paul, D., "Algorithms for an optimal A* search and linearizing the search in the stack decoder", *Proc. ICASSP 91*, pp. 693–696, 1991.