NOTICE WARNING CONCERNING COPYRIGHT RESTRICTIONS:
The copyright law of the United States (title 17, U.S. Code) governs the making of photocopies or other reproductions of copyrighted material. Any copying of this document without permission of its author may be prohibited by law.

CMU-CS-81-124

Minimizing Computational Cost for Dynamic Programming Algorithms

A.Waibel, N.Krishnan, R.Reddy

17 June 1981

Carnegie-Mellon University
Computer Science Department

This research was sponsored in part by the National Science Foundation, Grant MCS-7825824 and in part by the Defense Advanced Research Projects Agency (DOD), ARPA Order No. 3597, monitored by the Air Force Avionics Laboratory Under Contract F33615-78-C-1551.

The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Defense Advanced Research Projects Agency or the US Government.

UNIVERSITY LIBRARIES
CARNEGIE-MELLON UNIVERSITY
PITTSBURGH, PENNSYLVANIA 15213

Abstract

In this study we introduce and test several methods to reduce the computational cost in dynamic programming algorithms for isolated word recognition systems. Three methods will be discussed in detail: 1.) Pruning by preset thresholds 2.) Search based on the Branch and Bound technique 3.) Branch and Bound based search with additional pruning. Compared to conventional algorithms, Method 3.) could be seen to yield a speed up of approximately a factor of 5, at no loss of recognition accuracy. The branch and bound method with pruning is also ideally suited for research oriented systems, since pruning is independent of the parametrization used (eliminates the necessity for retuning thresholds). Additional features of this method, which are of importance to maintaining the flexibility and diagnosticity needed for such a system, will be discussed.

Table of Contents

List of Figures

1. Introduction

For the development of practical speech recognition systems, computation speed is one of the predominant design factors. Several commercially available systems still employ--in terms of recognition accuracy--inferior linear time normalization techniques to account for speaking rate variations, since the dynamic programming (DP) -technique is computationally very costly. Even in a research environment, the turn-around time for larger experimental runs over large speech data-bases can easily be in the order of days or weeks. Consequently, several methods have been employed to reduce the redundancies in isolated word recognition systems. Referring to the commonly used DP-matching techniques, as used by Sakoe and Chiba, Itakura, Rabiner and others[123], it can be seen that the bottleneck of nonlinear time normalization is given by the number of points within a matrix--defined by the frames of an unknown utterance x and a known reference utterance y--that are needed to find an optimum matching path. The computation needed for each of these points includes the computation of a distance between the particular test-frame and reference-frame under consideration and the derivation of a cumulative score defined by the constraints of the DP-matching algorithm in use. In a computer program that performs DP-matching, these operations will typically constitute the innermost loop and therefore be the most repetitious and most expensive in time. Finding less expensive warping constraints or distance functions, however, will in most cases yield a loss in recognition accuracy. Two other methods have been used by Sakoe & Chiba[1] and by Rabiner[3]. The first is the definition of a window[1] around the diagonal of the warping matrix that defines the boundaries of any allowable warping path. This definition is not only useful but also, for some warping functions, needed to prohibit possible non-linguistic paths through the matrix. Reduction of the width of this window thus increases computational speed significantly. It has been shown[4] that a window that restricts the warp search path to lead or lag behind a linearly time-normalized match by not more than 50 msecs is the optimal choice for an isolated word recognition system using an alpha-digit vocabulary. Such a window constraint was seen to not only provide a computational saving of up to 70% but also in some cases to increase recognition accuracy.

[1] It should be noted that, in that paper, the window was not mainly introduced for efficiency reasons.

[illegible]
CARNEGIE-MELLON UNIVERSITY
PITTSBURGH, PENNSYLVANIA 15213

Further methods have been suggested to increase computational efficiency. In the following chapter we will briefly describe a method suggested by Rabiner et al[3] and then introduce two alternate methods. In the third chapter we will report the results of extensive testing on all methods reported here.

2. Efficient Algorithms for Non-linear Time Warping

In this chapter we will describe three methods currently in use in our isolated word recognition system to perform dynamic programming in an efficient manner. Most methods are based on the idea that--analogous to the presumed strategy of human perception --selection of the correct candidate out of a reference vocabulary, can be performed in an anticipatory way, by process of elimination. In other words, particularly inappropriate candidates can be discarded comparably early in the matching process, i.e., the match can be aborted.

2.1 Preset Thresholds

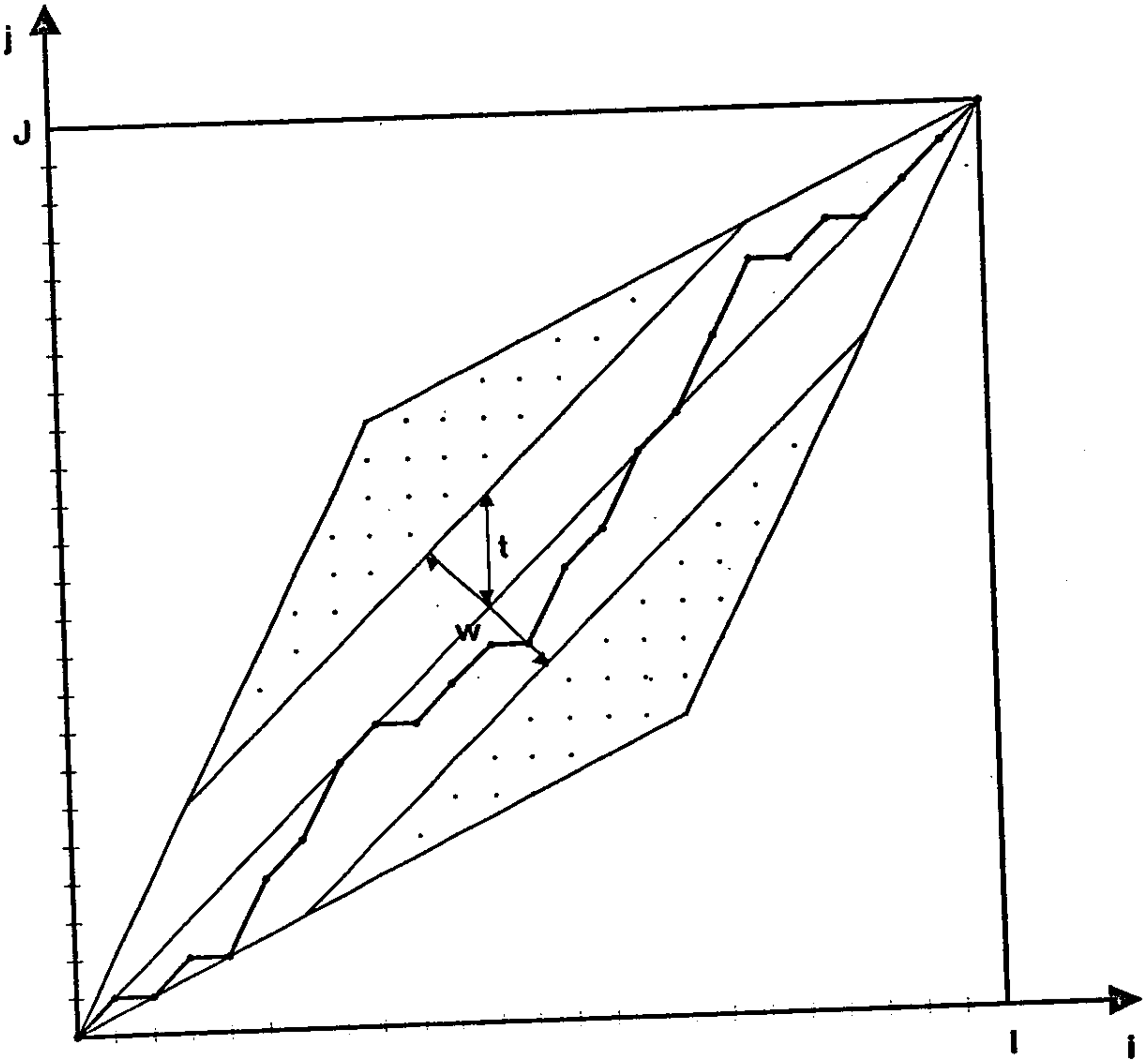
In this way, Rabiner et al. have obtained significant reductions in computation cost. Two thresholds are predefined, denoted Tmin and Tslope. The computation of the warp is performed by computing the distances and the Itakura warping function[2] between a given frame *i* in the test token and a column (specified by the search space) of reference frames (see Fig.2-1). For each of these grid points a cumulative dissimilarity score of the best path leading to this point is obtained in this fashion. The minimum score out of these cumulative scores--"localmin"--is determined and compared to the threshold Tj.[2]

If localmin>Tj the warp is aborted and recognition proceeds to the next candidate; Tj is given by

$$Tj = (Tmin + i \cdot Tslope) \cdot N$$

where N is the number of frames in the test utterances. Referring to Fig.2-2, it can be seen that Tslope can be viewed as N times the average distance that can be added to the cumulative score along the search path without causing the pruning mechanism to abort the match. The factor N provides a further adjustment depending on utterance length. Both Tmin and Tslope have to be set in such a fashion that they minimize computation (for efficiency) but are generous enough to not degrade recognition performance (e.g., by aborting "a good match").

[2] Note that the techniques described here would have to be altered if different warping algorithms were used. The Itakura algorithm appears particularly practical for these methods.



Restriction of the Search Space via an Adjustment Window

The dotted area indicates computational saving through the use of the window constraint. Tolerance t is used as a measure of the width as well as the saving achieved.

Figure 2-1: Warping Plane Indicating the Search Space of the Itakura Algorithm

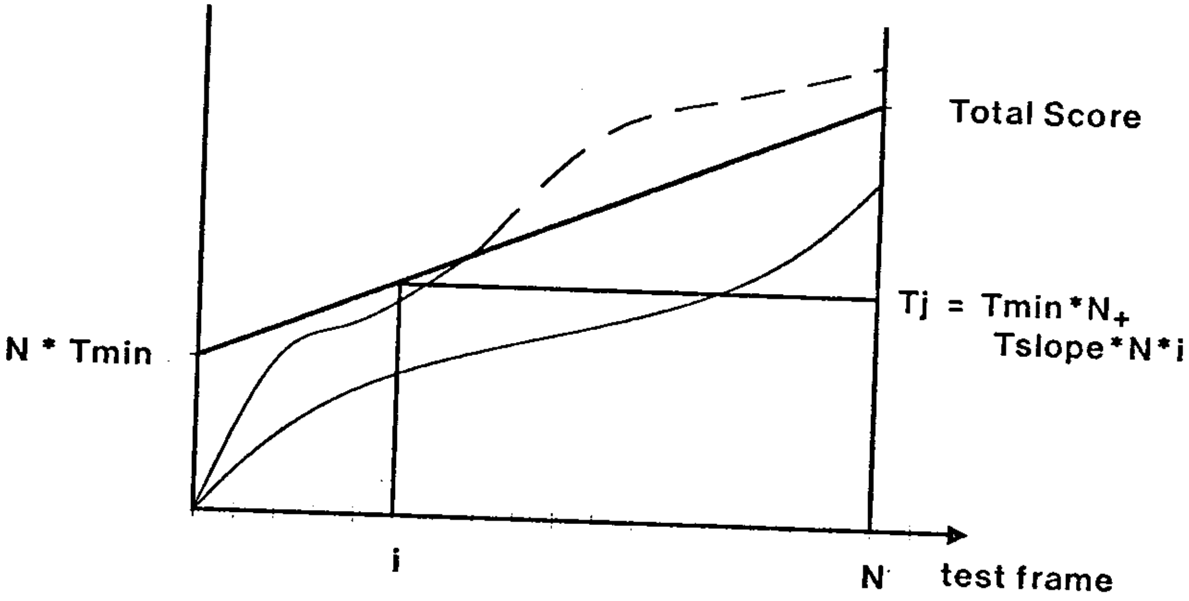


Figure 2-2: Pruning Using the Preset Thresholds Tmin and Tslope

2.2 Branch and Bound

In a research oriented speech recognition system it is for experimentation sometimes desirable to ensure that recognition results are not affected by pruning mechanisms, i.e., that they are guaranteed to reflect the differences in the overall dissimilarity scores derived from all matches, only. Nevertheless, one would want to avoid unnecessary computation. This is provided by a method that is based on the "branch and bound" search technique. This technique requires that the various matches of a recognition be performed in parallel.

What we mean by this "parallel warping" technique is illustrated in Fig.2-3. Instead of performing all matches sequentially, each frame *i* in the test-token is matched with the

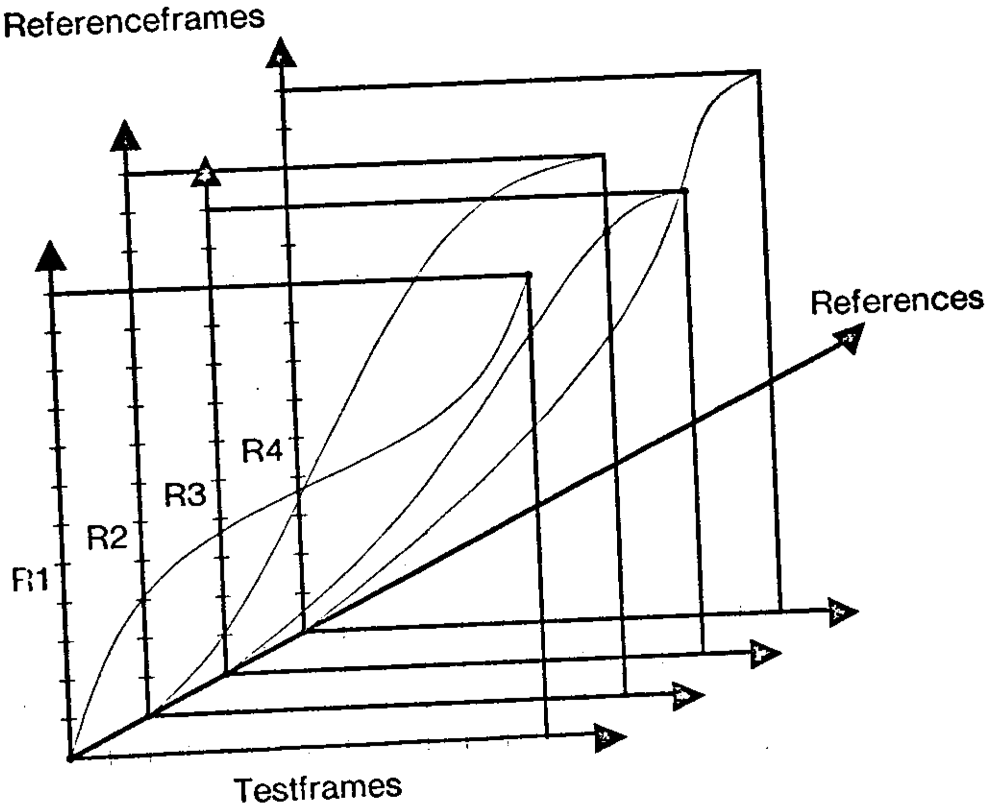


Figure 2-3: Parallel Warping Planes

corresponding frames of the K reference tokens of a K-word vocabulary. Fig.2-3 illustrates this technique by adding a dimension (k) to the warping process (usually depicted as a warping plane). In this fashion K warping planes are considered at a time. Information about the goodness of the matches with all the tokens in the reference vocabulary is available at all intermediate stages i_p during the warp. Several methods to prune comparatively bad matches suggest themselves. For the "branch and bound"-based technique, however, we do not prune away a bad match. Rather, only the so far least expensive match (the one with the so far lowest "localmin"-value) is expanded. This means that, instead of warping a particular test-frame i_p against the various frames of the K reference tokens, the so far best match out of the K matches is warped (thus proceeding in *i*) regardless of the momentary position in *i* of its search path. This method is illustrated by Fig.2-4, which depicts the projection of the search paths onto the ik-plane for the parallel warp and the "branch and bound"--based parallel warp. Clearly, in the branch and bound method bad matches--i.e. matches between strongly differing speech signals--will accumulate high distances and therefore be left behind. As soon as the best match reaches the end of the test utterance, the recognition process is completed. Thus, implicit pruning is performed on all other matches. This method has the advantage of guaranteeing that the lowest dissimilarity score will be found and thus it provides identical recognition results as if no pruning were performed. As an additional advantage for research oriented system, it should be noted that users can specify a value n to obtain the n best matches in the recognition, while the least amount of computation is being performed necessary to obtain the n best matches. However, if n>>1, of course, the computational saving will be minimal.

2.3 Branch and Bound with Pruning

In many cases, such as practical recognition systems as well as during large production runs of research oriented recognition systems, it often does not matter to preserve the exact individual recognition outcomes, as long as the overall number of errors is not increased when pruning is performed. If this is the case, the branch and bound method, described above, can be further extended to further reduce computation time. Thus, every time a path is expanded by means of continuing its warp, the number of frames that its search path is then leading before or lagging behind any other path is determined. If this number exceeds the threshold Leadt, this other path is pruned off. Leadt is given by

$$\text{Leadt} = P/100 \cdot N + 1$$

where P is a user-defined percentage and N the number of frames in the test utterance.

Thus using the illustration in Fig.2-4, if we were expanding path 1 to i_1 and if i_1-i_2>Leadt, match 2 would be aborted. In addition to drastically decreasing the computational effort, this pruning method is entirely independent of the numerical values of the distances, scores, and

UNIVERSITY LIBRARIES
CARNEGIE-MELLON UNIVERSITY
PITTSBURGH, PENNSYLVANIA 15213

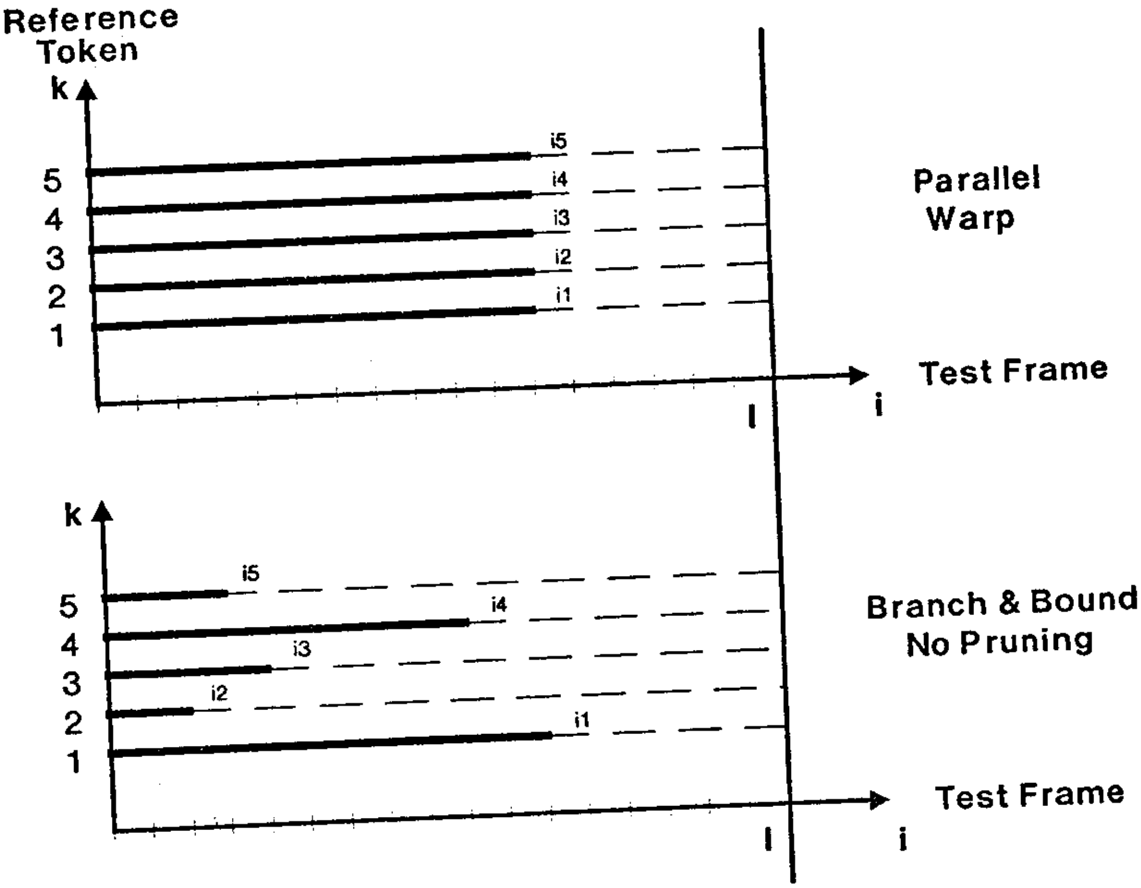


Figure 2-4: Expanding Search Paths in Parallel Warping Algorithms

spectral coefficients. It is therefore ideally suited for systems in a developmental stage. Using other pruning methods, frequent changes in the representation of the speech signal would cause the necessity for repeated retuning of thresholds to optimally trade off recognition accuracy and computational saving.

3. Testing

As a measure of the computation needed using the algorithms described above, we use the total number of grid points (of the warp search space) that were computed for each speaker and the run time. As testing conditions, the algorithms were run on 5 data sets, 36 utterances each (the alpha-digit vocabulary) for 8 speakers (4 male, 4 female). As reference data-set for each speaker, a 36-utterance reference set was generated from 5 additional readings of the vocabulary [5]. A detailed description of the recognition system can be found elsewhere[4]. It should be pointed out, however, that entirely automatic endpoint detection was used; no manual tuning was performed. Some of the recognition errors reported in these results are due to errors in the endpoint detection.

The results of these experimental runs are shown in figures 3-1 through 3-6.

The computational cost of the various algorithms tested is presented in figures 3-1 and 3-2. The criterion for these graphs was to minimize cost under the constraint of maintaining the same or reducing error rate as compared to a conventional algorithm. The results are presented in Fig.3-1 in terms of the number of grid points needed to compute the 180 recognition of the test data base and in 3-2 in terms of the average run time per recognition in msec. The first measure was chosen to provide a machine independent estimate of the savings obtained. As can be seen from Fig.3-2 in comparison to Fig.3-1, this does not directly translate into run time improvements, as we reduce the number of grid points. This is so, since in those cases, the number of grid points ceases to be the predominant factor contributing to computational cost and the overhead outside the innermost warping loop has to be considered. In both graphs, algorithm 1 -labeled no pruning, no window- performs an exhaustive search of the Itakura warp[2], algorithm 2 (no pruning, t=5 window) is algorithm 1 with the additional adjustment window constraint, that was previously reported[4] to yield better performance in accuracy and efficiency. Finally, the results for the algorithms 3, 4, 5, are shown, i.e., for the branch and bound with no pruning (i.e., P=100), the method of preset thresholds and the branch and bound method with pruning (P=15), as described earlier. Using the fastest algorithm, our particular implementation of the recognition system (running on a VAX-780) operates in less than 2.5 times real time.

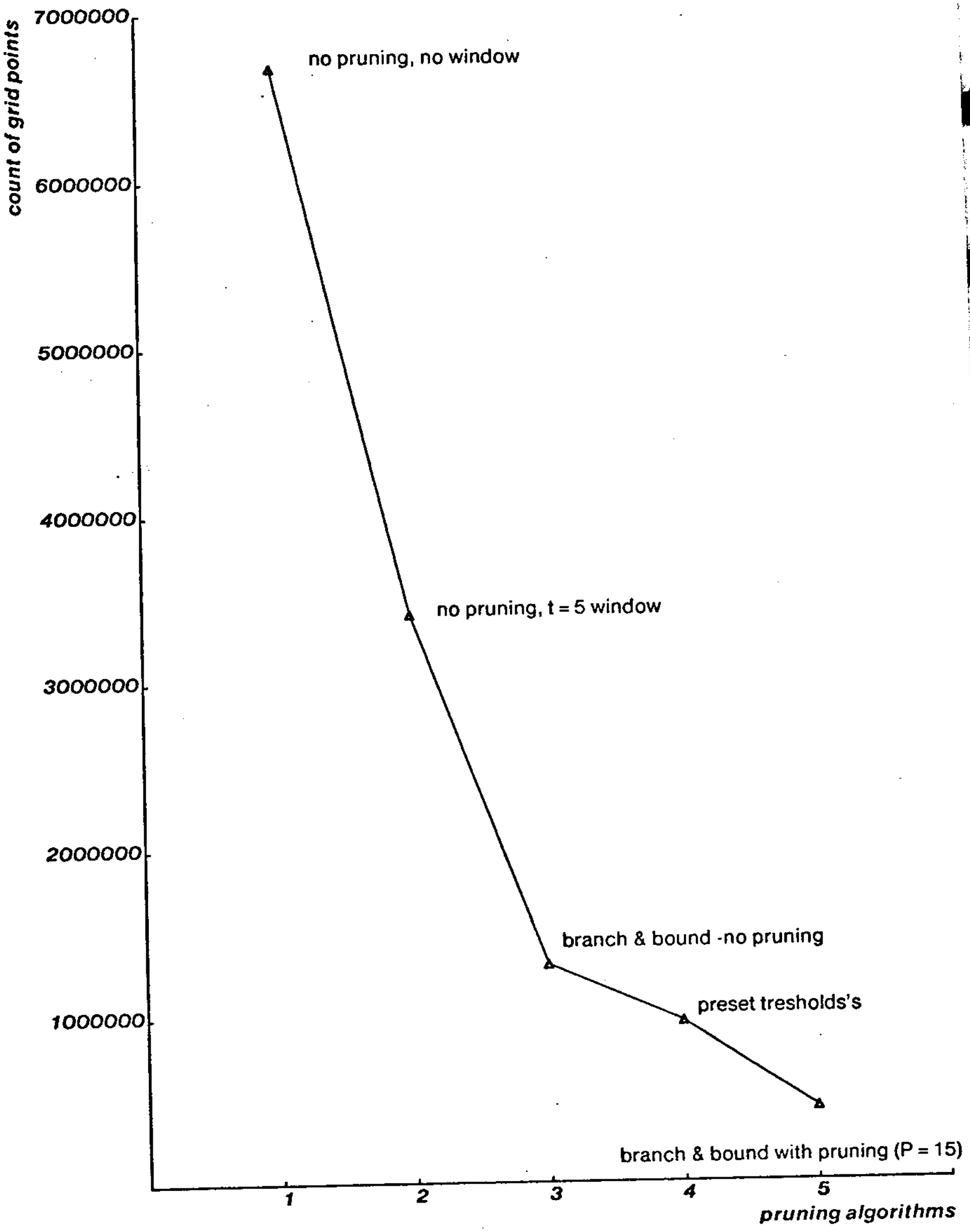


Figure 3-1: Number of Grid Points over 180 Recognitions for the Various Algorithms Tested

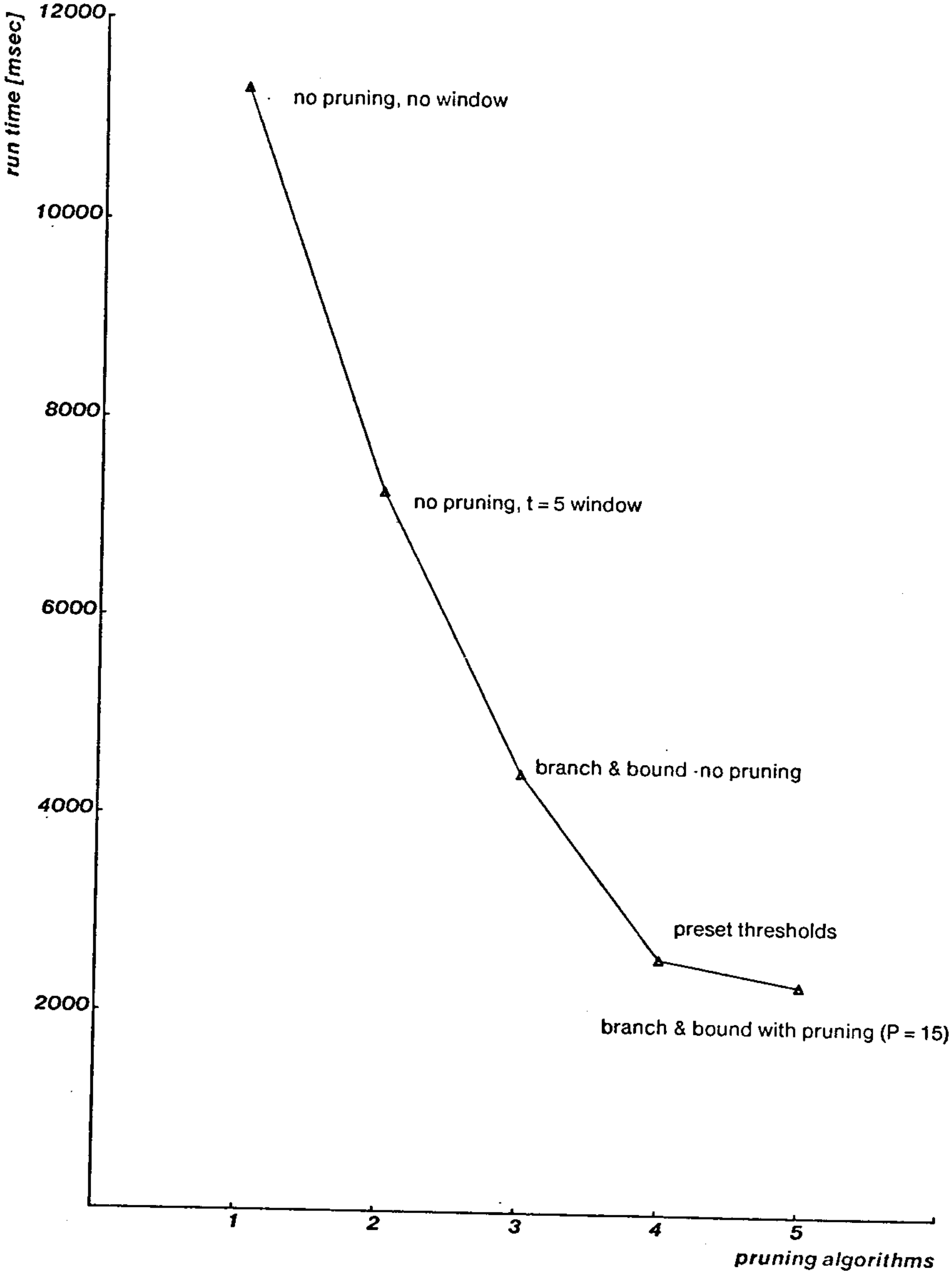


Figure 3-2: Average Run Times per Recognition for the Various Algorithms Tested

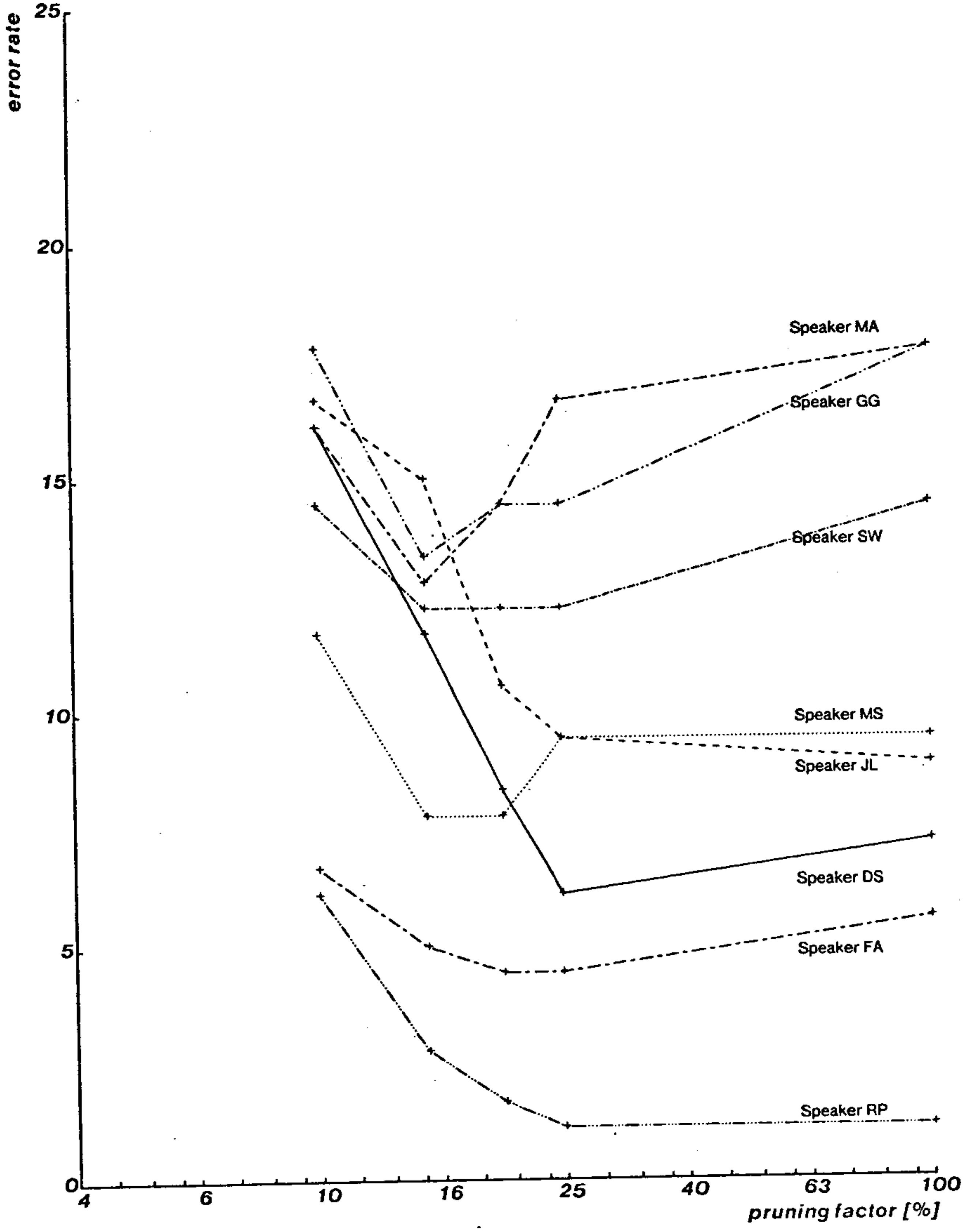


Figure 3-3: Error Rates for 8 Different Speakers vs Pruning Factor P

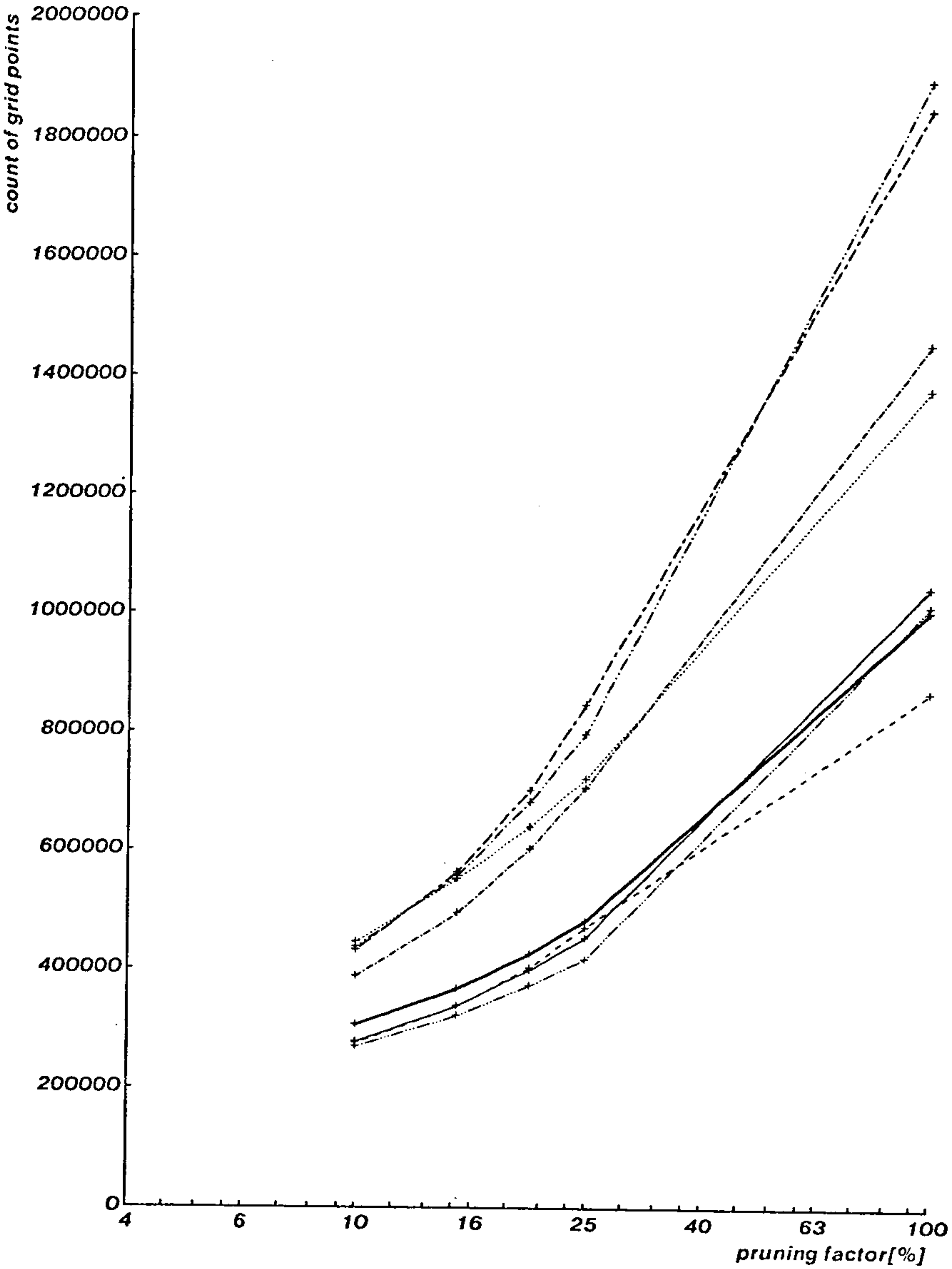


Figure 3-4: Number of Grid Points [180 Recognitions] for 8 Speakers vs. Pruning Factor P

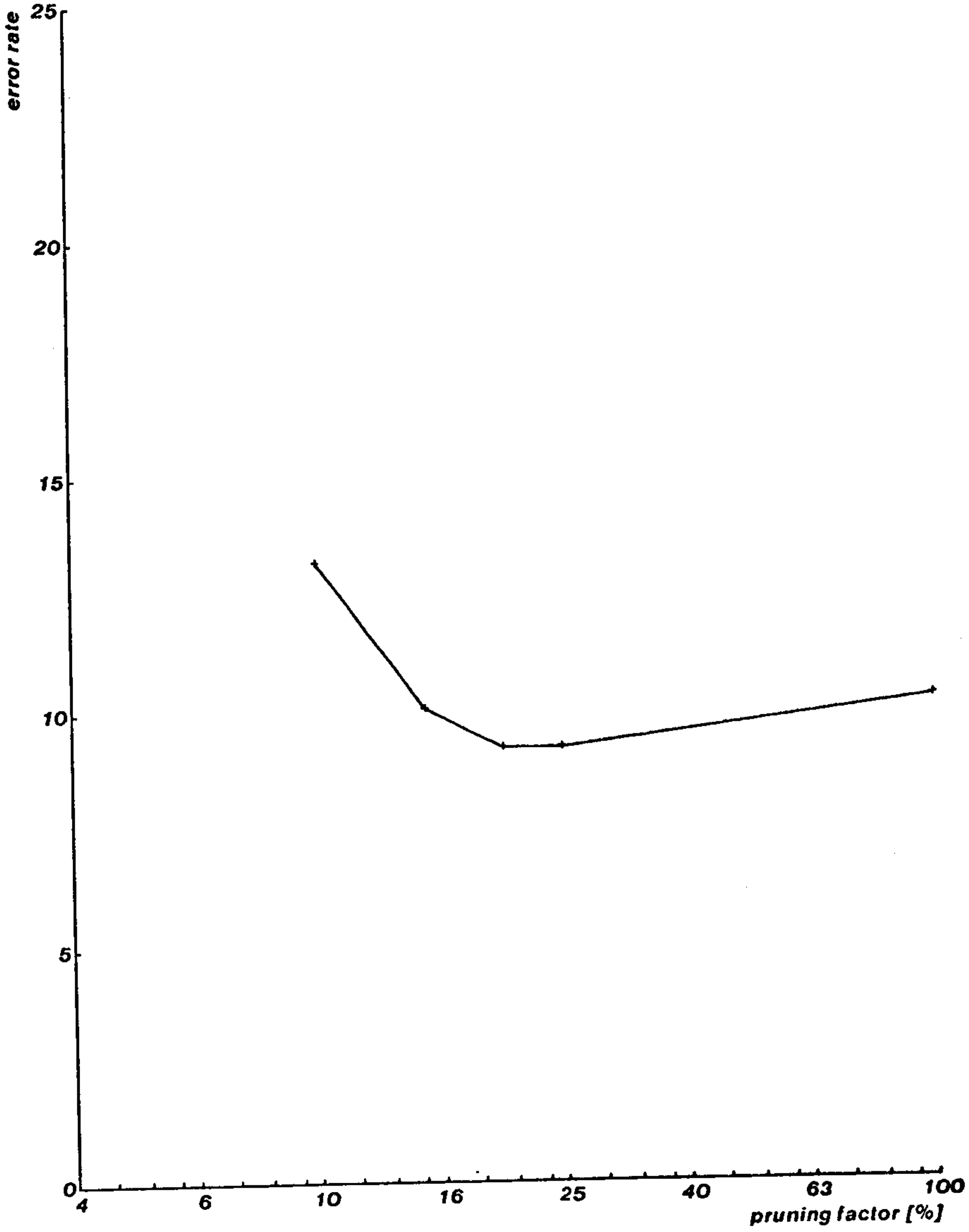


Figure 3-5: Average Error Rate vs. Pruning Factor P

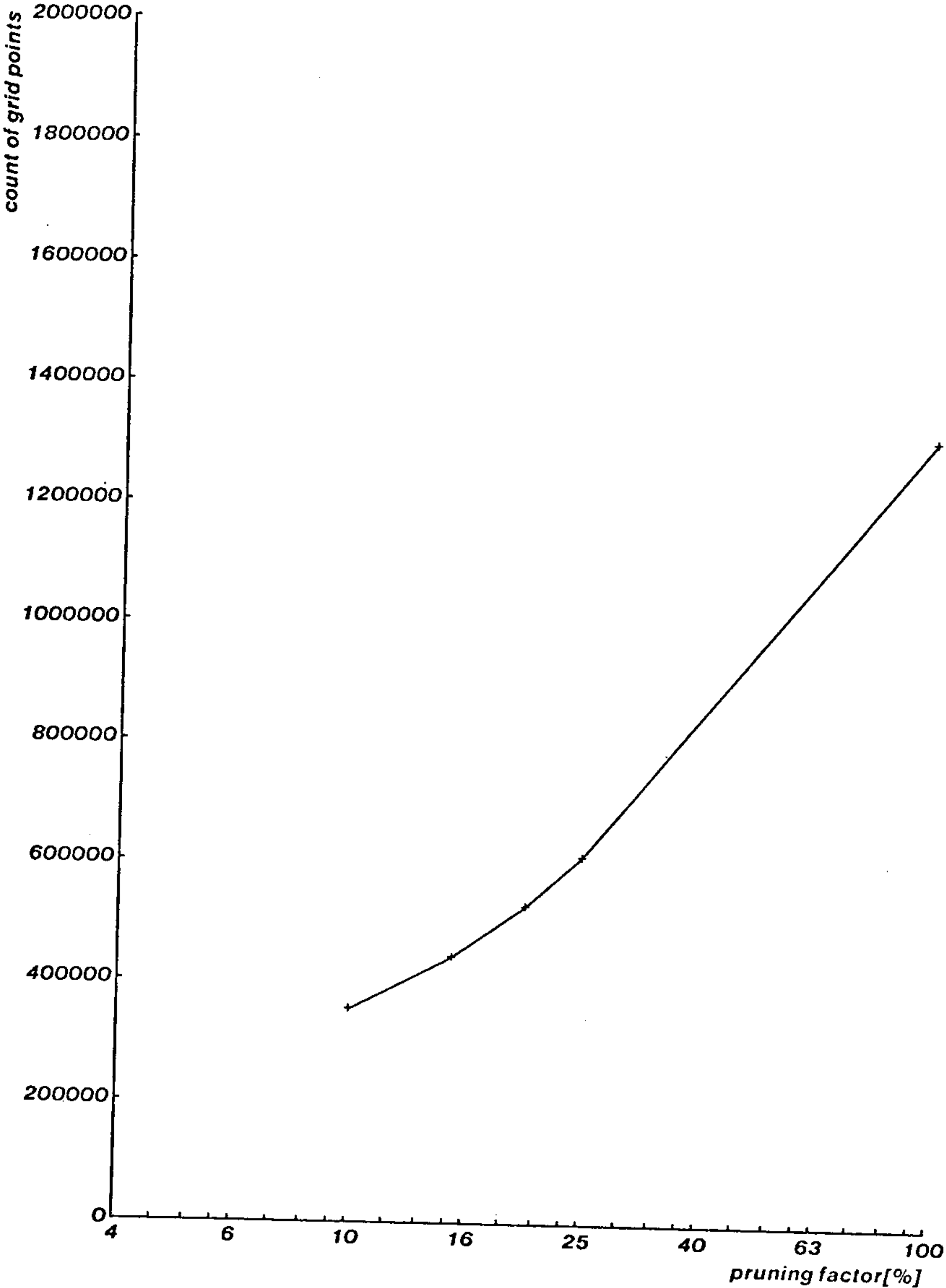


Figure 3-6: Number of Grid Points for 180 Recognitions vs. Pruning Factor P

Comparing Fig.3-1 with Fig.3-2, we also see that the run time improvements as given by the branch and bound method with pruning are not as substantial as indicated by the saving of grid points in Fig.3-1. This behavior is due to the larger overhead needed to perform the branch and bound search (comparing local minima among the references). This indicates that for vocabularies substantially larger than 40 words an alternate search strategy might result in faster operation.

In figure 3-3 the recognition results for the branch and bound based pruned algorithm are shown for various values of the pruning factor P. Results are plotted separately for the eight speakers in our data base and are given in terms of error rate (percent confused). Notice that P=100 means that no pruning is performed and the algorithm operates based on the branch and bound technique only. From figure 3-3 it can be seen that pruning does not necessarily have to be associated with an increase in error rate. In fact for P=15 or 20, improvements of up to 3% or 4% can be observed for some speakers. This is true, since in many cases two initially badly matching utterances such as a "B" and a "C", will be prevented from leading to confusion due to the pruning operation. Notice also -independently of the pruning and in agreement with earlier results[4] and other studies- the relatively high speaker dependency of the recognition results.

Figure 3-4 depicts the corresponding computational cost in terms of the number of grid points in the search space, i.e., the number of times the innermost loop of the algorithm has to be executed for the 180 recognitions of the testing corpus. As could be expected, the number of grid points that need to be computed decreases monotonically with decreasing pruning factor. A pruning factor of 15 or 20 (which yields acceptable recognition performance) will achieve a reduction of grid points by a factor of 2 to 3. Notice also, that while the curves for the different speakers behave similarly as a function of the pruning factor in a qualitative way, their actual quantitative values do differ strongly across speakers. This speaker dependency in speed (up to a factor of two) has to be considered should certain run times be required in a practical recognition system.

To summarize these observations in a very crude way we have taken the freedom for the purpose of illustration to average our results over the eight speakers as shown in figures 3-5 and 3-6. A value of P=20 can be seen to yield lowest error rates while a value of P=15 still leads to equivalent performance. This suggests that enough discriminatory confidence is accumulated when the path of an inappropriate reference candidate falls behind the best path by more than 20 percent of the length of the test token. This result shows that a search algorithm with pruning, i.e., an algorithm that does NOT perform an exhaustive search for the optimal score often times improves performance by virtue of imposing additional constraint on the search. This observation is consistent with previous results concerning the optimal choice of an adjustment window[4].

The cost (in terms of grid points) averaged across speakers obtained by such pruning can be inferred from figure 3-6. Note, that if one attempts to meet certain performance goals, it is better to use the data obtained for a speaker with the highest run times, rather than the average across speakers. For the purpose of comparison, however, we have chosen to present the data in this way.

To obtain optimal performance data for the preset thresholds algorithm, two thresholds were determined empirically, providing an error rate equivalent to an algorithm where no pruning was performed, while minimizing for computational cost.

4. Summary and Conclusion

We have shown that the branch-and-bound-with-pruning-method is the fastest algorithm of all the methods we have investigated. More importantly it is insensitive to changes in parametric representation or in vocabulary. This insensitivity to changes proves to be very beneficial in research systems when almost all aspects of the system are changing continuously, since optimization and tuning of thresholds is usually time consuming and cumbersome.

More specifically, the advantages are:

- This method yields 5 times faster operation for our recognition system than performing a conventional exhaustive search. (Using a lead threshold of 15% of the length of the test token (P=15) as pruning factor in addition to a branch and bound based search method (which yields approximately the same error rate as without pruning) and using a search space window of ±50 msec[4])

- Substantial cost reductions were achieved due to the insensitivity of the algorithm in face of system changes such as changes in parametric representation or vocabulary, thus eliminating the need for costly retuning.

- Flexible pruning thresholds (from no pruning at all up to rigid pruning) allow to manually trade off efficiency and recognition performance, if so desired.

- If no pruning is performed, the algorithm reduces to the branch and bound search guaranteeing optimality. This provides identical results as exhaustive search, while reducing the computational cost by about 60%.

- It is also possible to compute the guaranteed n-best candidates while obtaining more efficient operation.

The disadvantage of this technique is its higher requirements for primary memory storage, since several matches are operated on "in parallel". For systems with insufficient local memory, fast software implementations of such a technique and VLSI-implementations might therefore be faced more severely by the problem of performing fast I/O than by doing the computation necessary for recognition.

Acknowledgement

The authors would like to thank Jon Weissman and Jan Asbury for their invaluable help with programming, running tests and preparation of the final document.

References

1. H.Sakoe, S.Chiba, "Dynamic Programming Optimization for Spoken Word Recognition," *IEEE Transactions on Acoustics, Speech, Signal Processing*, Vol. ASSP-26, No. 1, February 1978, pp. 43-49.

2. F.Itakura, "Minimum Prediction Residual Principle Applied to Speech Recognition," *IEEE Transactions on Acoustics, Speech, Signal Processing*, Vol. ASSP-23, No. 1, February 1975, pp. 67-72.

3. L.R.Rabiner,S.E.Levinson,A.E.Rosenberg,J.G.Wilpon, "Speaker-Independent Recognition of Isolated Words Using Clustering Techniques," *IEEE Transactions on Acoustics, Speech, Signal Processing*, Vol. ASSP-27, No. 4, August 1979, pp. 336-349.

4. A.Waibel, B.Yegnanarayana, "Optimization of Nonlinear Time Warping Techniques in Isolated Word Recognition Systems," *J.Acoust.Soc.Am.*, Vol. 68, Suppl.1, November 1980, , soon to appear as CMU tech-report

5. Z.Li, F.Alleva, R.Reddy, "Effect of Reference Set Selection on Speaker Dependent Speech Recognition," *J.Acoust.Soc.Am.*, Vol. 69, Suppl.1, May 1981, , soon to appear as CMU tech-report