

The Time-Conditioned Approach in Dynamic Programming Search for LVCSR

Stefan Ortmanns and Hermann Ney, *Member, IEEE*

Abstract—This paper presents the time-conditioned approach in dynamic programming search for large-vocabulary continuous-speech recognition. The following topics are presented: the baseline algorithm, a time-synchronous beam search version, a comparison with the word-conditioned approach, a comparison with stack decoding. The approach has been successfully tested on the NAB task using a vocabulary of 64 000 words.

Index Terms—Beam search, dynamic programming, large vocabulary speech recognition, one-pass DP search, search organization, time-conditioned DP search.

I. INTRODUCTION

DURING the last decade, dynamic programming (DP) search has found widespread use in many systems for large-vocabulary continuous-speech recognition (LVCSR). In virtually all these cases, the search space is based on a tree-structured pronunciation lexicon and is structured in a so-called *word-conditioned* fashion. This means that during DP search, a separate set of tree-internal hypotheses is formed and evaluated for each *word history*, which is defined by the type of the language model (LM), e.g., for a bigram LM, we distinguish the hypotheses according to the single predecessor word; for a trigram LM, we distinguish the hypotheses according to the two predecessor words.

In this paper, we present a different approach to the use of DP in the context of a tree-structured pronunciation lexicon. This approach is referred to as *time-conditioned* because the key concept is to distinguish the hypotheses according to the *end time* of the immediate predecessor word hypothesis. The novel contributions of this paper are as follows:

- starting from the Bayes decision rule for speech recognition in the maximum approximation, we derive the baseline algorithm for time-conditioned DP search;
- using a tree-structured pronunciation lexicon, we present the evaluation of the DP recurrence equations in the form of a one-pass time-synchronous left-to-right strategy;
- for practical use in large-vocabulary recognition, we show how pruning can be applied and an efficient implementation can be obtained;
- to analyze the characteristic properties of the time-conditioned DP approach, we present a detailed comparison

Manuscript received April 9, 1998; revised January 26, 2000. This paper was presented in part at ICSLP'96 [17] and ICASSP'97 [14]. This work was supported in part by the Philips GmbH Forschungslaboratorien, Aachen, Germany.

S. Ortmanns was with Vehrstuhl für Informatik VI, Computer Science Department, RWTH Aachen University of Technology, 52056 Aachen, Germany. He is now with Philips Speech Processing, 52072 Aachen, Germany.

H. Ney is with Lehrstuhl für Informatik VI, Computer Science Department, RWTH Aachen University of Technology, 52056 Aachen, Germany.

Publisher Item Identifier S 1063-6676(00)09264-6.

with the word-conditioned DP approach; in addition, we show how the time-conditioned concept leads from a single-best sentence to a word graph in a natural way;

- as an alternative to DP search, stack decoding is used in some systems. We show that the time-conditioned DP approach bears a certain resemblance to stack decoding;
- we report on systematic recognition experiments for the 64 000-word NAB task and give a detailed experimental comparison of the time-conditioned DP approach with the word-conditioned approach.

Before presenting these details of the time-conditioned DP approach in the following sections, we first review the specification of the search problem in large-vocabulary continuous-speech recognition.

II. SPECIFICATION OF THE SEARCH PROBLEM

The acoustic models are given in terms of hidden Markov models (HMMs) [2], [4] representing either context dependent or independent phoneme units. For a hypothesized word sequence $w_1^N = w_1 \cdots w_N$, we imagine a super HMM that is obtained by concatenating the corresponding phoneme HMMs using a pronunciation lexicon. At phoneme and word boundaries, we have to allow for transitions that link the terminal states of any predecessor HMM to the initial states of any successor HMM. Thus, we can compute the joint probability of observing the sequence $x_1^T = x_1 \cdots x_T$ of acoustic input vectors and the state sequence $s_1^T = s_1 \cdots s_T$ through this super HMM:

$$\begin{aligned} \Pr(x_1^T, s_1^T | w_1^N) &= \prod_{t=1}^T p(x_t, s_t | s_{t-1}, w_1^N), \\ &= \prod_{t=1}^T [p(s_t | s_{t-1}, w_1^N) \cdot p(x_t | s_t)] \end{aligned}$$

where $p(x_t, s_t | s_{t-1}, w_1^N)$ denotes the product of the transition and emission probabilities for the super HMM w_1^N . The decomposition has been formulated in such a way that we can distinguish two components of the following approach:

- reference models with the emission probability distributions $p(x_t | s)$ for the acoustic states, e.g. after state tying using decision trees or some other method;
- transition probabilities $p(s_t | s_{t-1}, w_1^N)$ depending on a word sequence hypothesis w_1^N : this implies a huge finite network of states (super HMM) that has to be considered for each word sequence hypothesis w_1^N .

Denoting the language model (LM) probability by $\Pr(w_1^N)$, the Bayes decision rule results in the following optimization

problem:

$$\begin{aligned} [w_1^N]_{opt} &= \operatorname{argmax}_{w_1^N} \left\{ \Pr(w_1^N) \cdot \sum_{s_1^T} \Pr(x_1^T, s_1^T | w_1^N) \right\} \\ &\cong \operatorname{argmax}_{w_1^N} \left\{ \Pr(w_1^N) \cdot \max_{s_1^T} \Pr(x_1^T, s_1^T | w_1^N) \right\}. \end{aligned}$$

Here, we have made use of the so-called maximum approximation. Instead of summing over all paths, we consider only the most probable path. Note that for the maximum approximation to work, we need only the assumption that the resulting optimal word sequences are the same, not necessarily that the maximum provides a good approximation to the sum.

In this maximum approximation, the search space can be described as a huge network through which the best path has to be found. The search has to be performed at two levels: at the state level (s_1^T) and at the word level (w_1^N). As we will see, as a result of the maximum approximation, it will be possible to *recombine* hypotheses at both levels efficiently by DP. Thus the combinatorial explosion of the number of search hypotheses can be limited, which is one of the most important characteristics of DP. At the same time, the search hypotheses are constructed and evaluated in a strictly left-to-right time-synchronous fashion. This property allows an efficient pruning strategy to eliminate unlikely search hypotheses, which is usually referred to as beam search.

III. TIME-CONDITIONED APPROACH IN DP SEARCH

Most DP search strategies for LVCSR can be called a word-conditioned strategy because the hypotheses are conditioned on the predecessor words v for bigram LM's $p(w|v)$ and predecessor word pairs (u, v) for trigram LM's $p(w|u, v)$ (see Section IV). Another approach is to structure the search space by using the end time of the predecessor word or equivalently the start time of the successor word. This concept has already been used in other contexts, namely connected digit recognition [25] and word lattice generation [16]. Without a time-synchronous evaluation, this concept has certain similarities to stack decoding or A^* search as we will discuss in Section V.

A. Principle

To describe the time-conditioned approach and to arrive at the DP formulation, we define the following quantities:

$$\begin{aligned} h(w; \tau, t) &:= \max_{s_{\tau+1}^t} \Pr(x_{\tau+1}^t, s_{\tau+1}^t | w) \\ &= \text{conditional probability that word } w \\ &\quad \text{produces the acoustic vectors } x_{\tau+1}^t. \end{aligned}$$

$$\begin{aligned} G(w_1^n; t) &:= \Pr(w_1^n) \cdot \max_{s_1^t} \Pr(x_1^t, s_1^t | w_1^n) \\ &= \text{joint probability of observing the acoustic} \\ &\quad \text{vectors } x_1^t \text{ and a word sequence } w_1^n \\ &\quad \text{with end time } t. \end{aligned}$$

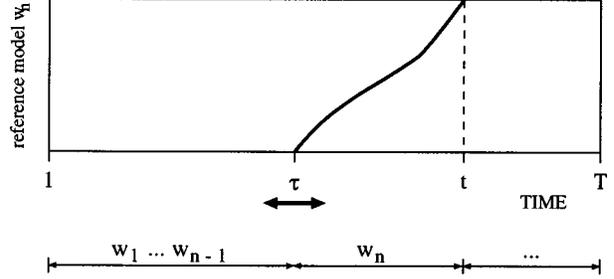


Fig. 1. Illustration of decomposition.

By using these definitions, we can isolate the probability contributions of a particular word hypothesis with respect to both the LM and the acoustic model (see Fig. 1)

$$\underbrace{x_1, \dots, \dots, x_\tau}_{G(w_1^{n-1}; \tau)} \underbrace{x_{\tau+1}, \dots, x_t}_{h(w_n; \tau, t)} \underbrace{x_{t+1}, \dots, \dots, x_T}_{\dots}$$

To extend the word sequence hypothesis w_1^{n-1} by one more word w_n , we have to optimize over the unknown word boundary τ

$$\begin{aligned} G(w_1^n; t) &= \max_{\tau} \{ \Pr(w_n | w_1^{n-1}) \cdot G(w_1^{n-1}; \tau) \cdot h(w_n; \tau, t) \} \\ &= \Pr(w_n | w_1^{n-1}) \cdot \max_{\tau} \{ G(w_1^{n-1}; \tau) \cdot h(w_n; \tau, t) \} \quad (1) \end{aligned}$$

where we have incorporated the conditional LM probability $\Pr(w_n | w_1^{n-1})$. By definition, we have then for the overall recognition score

$$G(w_1^N; T) = \Pr(w_1^N) \cdot \max_{s_1^T} \Pr(x_1^T, s_1^T | w_1^N).$$

So far, we have not constrained the LM in any way, and no recombination of hypotheses is possible. This, however, is different for an m -gram LM $p(v_m | v_1^{m-1})$. Here it is sufficient for each time t to distinguish partial word sequences w_1^n only if their final $(m-1)$ predecessor words v_2^m are different. Therefore, we define

$$\begin{aligned} H(v_2^m; t) &:= \max_{w_1^n: w_{n-m+2}^n = v_2^m} \{ \Pr(w_1^n) \cdot \max_{s_1^t} \Pr(x_1^t, s_1^t | w_1^n) \} \\ &= \text{joint probability of observing the acoustic} \\ &\quad \text{vectors } x_1^t \text{ and a word sequence} \\ &\quad \text{with end sequence } v_2^m \text{ and end time } t. \end{aligned}$$

Thus, we have the DP recursion for the quantities $H(v_2^m; t)$

$$\begin{aligned} H(v_2^m; t) &= \max_{v_1} \left\{ p(v_m | v_1^{m-1}) \cdot \max_{\tau} \{ H(v_1^{m-1}; \tau) \cdot h(v_m; \tau, t) \} \right\}. \quad (2) \end{aligned}$$

The computation of the word scores $h(v_m; \tau, t)$ will be considered next.

B. Time-Synchronous One-Pass Organization

To compute the scores $h(v_m; \tau, t)$ efficiently, we will use a prefix tree for the pronunciation lexicon and organize all evalu-

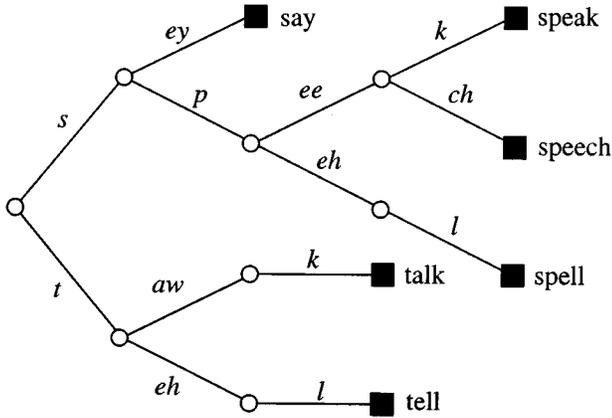


Fig. 2. Example of a tree-organized pronunciation lexicon (lexical prefix tree).

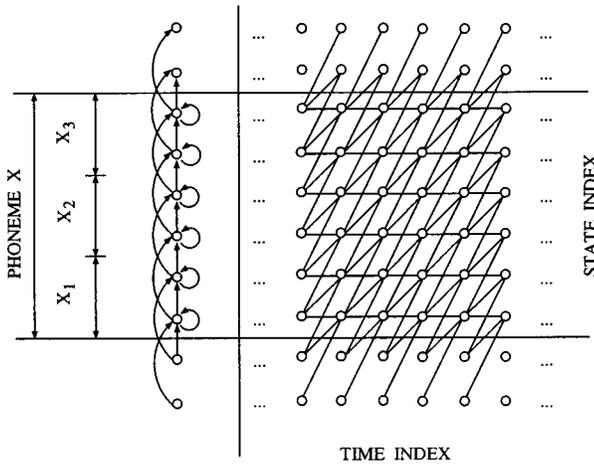


Fig. 3. Structure of a phoneme model and search space.

ation steps in a left-to-right time-synchronous strategy. An example of such a lexical prefix tree is shown in Fig. 2. We will often simply refer to it as a tree. Each phoneme arc of this lexical prefix tree stands for a HMM. The HMM used in this paper has a tripartite structure with a total of six states. Fig. 3 shows such an HMM along its time—state trellis or network.

For notational convenience, throughout the paper, we will always consider the state index s of an arc rather than the arc itself and assume that the lexical structure is fully captured by the transition probabilities of the super HMM representing the lexical tree. To formulate the DP approach, we introduce the following quantity to denote tree-internal search hypotheses with respect to the start time τ [17]:

$$\begin{aligned}
 Q_\tau(t, s) &:= \\
 &= \max_{w_1^n} \Pr(w_1^n) \cdot \left\{ \max_{s_1^t: s_\tau = S_{w_n}, s_t = s} \Pr(x_1^t, s_1^t | w_1^n) \right\} \\
 &= \max_{w_1^n} \left\{ \Pr(w_1^n) \cdot \max_{s_1^t: s_\tau = S_{w_n}} \Pr(x_1^t, s_1^t | w_1^n) \right\} \\
 &\quad \cdot \max_{s_{\tau+1}^t: s_t = s} \Pr(x_{\tau+1}^t, s_{\tau+1}^t | \dots) \\
 &= \text{overall score of the best partial} \\
 &\quad \text{(state, word)-sequence that at time } t \text{ ends} \\
 &\quad \text{in state } s \text{ of the lexical tree with start time } \tau
 \end{aligned}$$

where we have used the symbol S_w to denote the terminal state of word w . In this definition, we have already taken into account that the optimization over the state sequence s_1^t can be done in two steps, namely for the first part s_1^τ and the second part $s_{\tau+1}^t$ of the state sequence. The justification is that we can always decompose such a path to (t, s) into two parts:

$$[(-, -) \rightarrow (t, s)] = [(-, -) \rightarrow (\tau, S_{w_n})] \circ [(\tau + 1, 0) \rightarrow (t, s)].$$

We have used the notation

$$\Pr(x_{\tau+1}^t, s_{\tau+1}^t | \dots)$$

to express the scores of tree-internal hypotheses that start at time $\tau + 1$, but have not yet reached a terminal state so that no word index is available for them. Before moving from time $(t - 1)$ to time t , we have to allow for a word boundary hypothesis. To this purpose, we start up a separate copy of the lexical prefix tree for each start time hypothesis $\tau = t - 1$

$$Q_{t-1}(t - 1, s) = \begin{cases} H_{\max}(t - 1) & \text{if } s = 0 \\ 0 & \text{if } s > 0 \end{cases}$$

where the fictitious state $s = 0$ (for the root of the tree) is used for initialization and $H_{\max}(t - 1)$ is defined to be the best existing hypothesis at time $(t - 1)$

$$H_{\max}(t - 1) := \max_{v_1^{m-1}} H(v_1^{m-1}; t - 1).$$

Note that there are only hypotheses $Q_\tau(t, s)$ for $\tau \leq t$. In the tree interior, we have the usual DP recursion:

$$Q_\tau(t, s) = \max_{\sigma} \{p(x_t, s | \sigma) \cdot Q_\tau(t - 1, \sigma)\}.$$

Here, the symbol $p(x_t, s | \sigma)$ denotes the product of emission and transition probabilities of the HMM as before. Back pointers as used in the word-conditioned search method are not needed since each tree copy is directly conditioned on its start time τ . When a path hypothesis has reached a terminal state with word label v_m , i. e., when we have a hypothesis $Q_\tau(t, S_{v_m})$, we can compute the acoustic word score $h(v_m; \tau, t)$

$$h(v_m; \tau, t) = \frac{Q_\tau(t, S_{v_m})}{H_{\max}(\tau)}.$$

By inserting this equation into (2), we obtain the DP recursion

$$\begin{aligned}
 H(v_2^m; t) &= \max_{v_1^{m-1}} \left\{ p(v_m | v_1^{m-1}) \cdot \max_{\tau} \left\{ H(v_1^{m-1}; \tau) \cdot \frac{Q_\tau(t, S_{v_m})}{H_{\max}(\tau)} \right\} \right\}. \quad (3)
 \end{aligned}$$

The recombinations captured by this equation are illustrated in Fig. 4. For each word end hypothesis v_m at time t , we collect all possible start times τ with associated scores $Q_\tau(t, S_{v_m})$, and for each start time hypothesis τ , there is a separate list of predecessor hypotheses v_{m-1} with scores $H(v_1^{m-1}; t)$.

The algorithm described is most efficient in connection with the time-synchronous beam search concept. Then considering all index pairs (t, s) , we can ask how many different start times τ might be present in the set of state hypotheses $Q_\tau(t, s)$. As can be expected and will be confirmed by the experimental results, this number is 30–50, which is roughly the order of the average duration of a word.

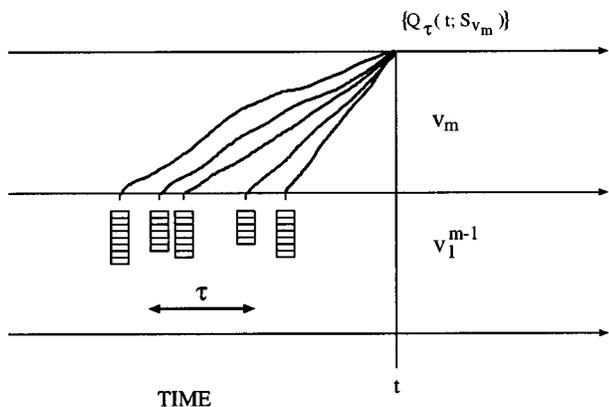


Fig. 4. Word boundary optimization in the time-conditioned one-pass approach.

We summarize the characteristic features of DP [5] as they are encountered here as in many discrete optimization problems:

- optimization criterion can be decomposed such that the cost of each edge in the network under consideration depends only on the identity of the edge and nothing else;
- carrying out the DP recursion amounts to filling in tables in a recursive way. In the case considered here, the tables to be filled are the scores $H(v_2^m; t)$ at the word level and the score $Q_\tau(t, s)$ at the state level.

C. Beam Search and LM Look-Ahead

For large vocabulary speech recognition, the search method described cannot be used without somehow limiting the possible search space. As an example, we estimate the size of the search space for the 64 000-word WSJ system (see experimental results) and a sentence of $T = 20000$ 10-ms frames:

$$20\,000 \text{ trees} \cdot 200\,000 \text{ arcs/tree} \cdot 6 \text{ HMM states/arc} \\ = 2.4 \cdot 10^{10} \text{ HMM states.}$$

Therefore, in full search, there is this number of HMM states for which the DP recursions have to be evaluated every 10-ms time frame of the input signal. In contrast with this astronomic number, experimental tests will show that, by using the beam search method, this number can be reduced to less than an average of 50 000 HMM states per time frame without loss in performance. The principal concept of beam search is to retain only the most promising hypotheses $Q_\tau(t, s)$ for further evaluation at each time frame t ; the other hypotheses are removed [11].

The efficiency of the pruning method is improved by including the LM probabilities as early as possible into the pruning process. This operation, referred to as LM look-ahead, works as follows [1], [15], [18], [27]. Considering a state s in the lexical tree, we know that the extensions of this state hypothesis cannot produce all words w of the vocabulary, but only a certain subset, which we denote by $\mathcal{W}(s)$. Thus to take this effect into account, we compute the probability of the most likely word that can be generated for each state s

$$\pi(s) := \max_{w \in \mathcal{W}(s)} p(w)$$

where we have use the *unigram* LM $p(w)$. By using a bigram or trigram LM, a better focusing of the beam search can be ex-

pected [15], [18], but it is not evident how to incorporate such a LM look-ahead into the time-conditioned approach in an *efficient* way.

As in word-conditioned approach, there are three different types of pruning during the search that are performed every time frame [27].

- *Acoustic pruning* is the pruning operation that is typically meant by beam search. To incorporate the look-ahead LM probabilities $\pi(s)$ into the acoustic pruning, we modify the scores $Q_\tau(t, s)$ and define

$$\tilde{Q}_\tau(t, s) = \pi(s) \cdot Q_\tau(t, s).$$

At time t , we compare each hypothesis $\tilde{Q}_\tau(t, s)$ with the *best* hypothesis at that time. We prune a hypothesis (τ, s, t) with score $Q_\tau(t, s)$ if

$$\tilde{Q}_\tau(t, s) < f_{AC} \cdot \max_{(\tau', s')} \left\{ \tilde{Q}_{\tau'}(t, s') \right\}$$

where the threshold $f_{AC} < 1$ is used to control the number of surviving state hypotheses.

- *Language model pruning* is applied only to word end hypotheses, i. e. hypotheses (v_1^{m-1}, t) with scores $H(v_1^{m-1}; t)$. A word end hypothesis $H(v_1^{m-1}; t)$ is removed if

$$H(v_1^{m-1}; t) < f_{LM} \cdot H_{\max}(t)$$

where f_{LM} is the so-called language model pruning threshold. In other words, this pruning operation refers to the bookkeeping lists used for word end hypotheses (v_1^{m-1}, t) rather than the tree-internal hypotheses (τ, s, t) . Its goal is to reduce the memory and CPU cost of performing the LM recombinations as required by (3).

- *Histogram pruning* limits the number of surviving state hypotheses (τ, s, t) to a maximum number M_{Sta} . If the number of active states is larger than M_{Sta} , only the best M_{Sta} hypotheses are retained and the other hypotheses are removed. This pruning method is also called histogram pruning because we use a histogram over the scores of the active states [27].

Of these three pruning operations, the acoustic pruning is by far the most important one.

D. Implementation

To fully exploit the advantages of beam search, a dynamic construction of the search space is necessary as in the implementations of other DP beam search variants [12], [13]. The implementation is based on an array organization so that a direct and thus efficient access to all active search hypotheses for states, arcs and trees is possible. Using these type of organization, the search space is created dynamically during the recognition process. The operations of the time-conditioned search algorithm are summarized in Table I. There are two levels: the acoustic level at which the tree-internal hypotheses are evaluated and the word level at which the LM recombination is performed. Note that for simplification purposes it is tacitly assumed that all operations apply to active hypotheses only, i. e. hypotheses surviving in the beam search concept.

TABLE I
TIME-CONDITIONED LEXICAL TREE SEARCH ALGORITHM FOR AN
 m -GRAM LM

proceed over time t from left to right	
ACOUSTIC LEVEL: process state hypotheses $\{Q_\tau(t, s)\}$	
- initialization:	
$Q_{t-1}(t-1, s) = \begin{cases} H_{\max}(t-1) & \text{if } s = 0 \\ 0.0 & \text{if } s > 0 \end{cases}$	
- time alignment: $Q_\tau(t, s)$ using DP recursion	
- prune unlikely hypotheses	
- purge bookkeeping lists	
WORD LEVEL: process word end hypotheses $\{Q_\tau(t, S_{v_m})\}$	
for each word end v_m do	
for each boundary τ do	
- word score: $h(v_m; \tau, t) = Q_\tau(t, S_{v_m})/H_{\max}(\tau)$	
for each word $(m-1)$ -gram v_1^m do	
$H(v_2^m; t) = \max_{(v_1, \tau)} \{ p(v_m v_1^{m-1}) \cdot H(v_1^{m-1}; \tau) \cdot h(v_m; \tau, t) \}$	
$(\tilde{v}_1, \tilde{\tau})(v_2^m; t) = \operatorname{argmax}_{(v_1, \tau)} \{ p(v_m v_1^{m-1}) \cdot H(v_1^{m-1}; \tau) \cdot h(v_m; \tau, t) \}$	
- store best predecessor $\tilde{v}_1(v_2^m; t)$	
- store best boundary $\tilde{\tau}(v_2^m; t)$	
- cache best score $H(v_2^m; t)$	
$H_{\max}(t) = \max_{v_2^m} \{ H(v_2^m; t) \}$	

Unlike the word-conditioned method, back pointers [13] are not needed since the hypotheses (τ, t, s) with the score $Q_\tau(t, s)$ are conditioned directly on the start times τ . However, the time-conditioned approach results in a more complex LM recombination. Before the LM recombination itself, we first collect all possible start times τ of v_m for each time t and for each word end hypothesis v_m and calculate the acoustic score $h(v_m; \tau, t)$. Second, we perform the LM recombination. Here we have to use the partial word sequence hypotheses (v_1^{m-1}, τ) with scores $H(v_1^{m-1}; \tau)$. For each previous time frame τ , these hypotheses must have been cached for later re-use.

The computationally most expensive step in the LM recombination is the access to the LM probabilities. Therefore, we re-organize the evaluation in such a way that the number of LM accesses is a minimum. We perform the optimization over the word boundary τ first *without* LM probabilities. To this purpose, we introduce the auxiliary function $\hat{H}(v_1^m; t)$

$$\hat{H}(v_1^m; t) := \max_{\tau} \{ H(v_1^{m-1}; \tau) \cdot h(v_m; \tau, t) \}$$

and rewrite (3) as

$$H(v_2^m; t) = \max_{v_1} \left\{ p(v_m|v_1^{m-1}) \cdot \hat{H}(v_1^m; t) \right\}.$$

To obtain a fast access to the hypotheses $H(v_1^{m-1}; \tau)$, we use hashing.

In Table I, we distinguish between *storing* and *caching*. At each time frame t and for each word $(m-1)$ -gram v_2^m , the best predecessor word $\tilde{v}_1(v_2^m; t)$ and best word boundary $\tilde{\tau}(v_2^m; t)$ must be *stored* to recover the recognized word sequence; this information must be preserved until the end of the sentence has

been reached. In contrast, the scores $H(v_2^m; t)$ are only *cached*, i. e. this information is typically needed only for the immediate subsequent frames, say 100 frames.

E. Across-Word Phoneme Models

In the preceding paragraphs, we have described the time-conditioned lexical tree search in the context of word-internal phoneme models. We will now extend the search strategy to across-word phoneme models. To simplify the presentation, we confine ourselves to across-word triphone contexts.

For each word boundary hypothesis along with the corresponding word pair, we have to allow for all possible across-word triphone contexts. To this purpose, the lexical prefix tree is modified in the following way. For each word end hypothesis, we have to introduce fan-out arcs of the last phoneme in the lexical tree so that there is a separate *fan-out arc* for each possible across-word *successor* phoneme. Similarly, each first-generation phoneme of the lexical tree is replaced by a set of *fan-in arcs* depending on the possible across-word predecessor phonemes. Strictly speaking, as a result of these fan-in arcs for word start hypotheses, we do not have a single prefix tree any more, but a set of *lexical subtrees* so that there is a separate subtree for each across-word predecessor phoneme.

The modifications considered so far refer only to the static situation, i.e. the organization of the lexical prefix tree. When applying this lexical structure in search, we have to distinguish the hypotheses according to the across-word contexts. This means that the state hypotheses $Q(t, s)$ must be made dependent on the possible across-word predecessor phonemes and the word end hypotheses $H(v_2^m; t)$ must be made dependent on the across-word successor phonemes. When performing DP recombination at the word level, we have to take into account that each word pair hypothesis implies a unique across-word phoneme context. In the rest of the paper, we will only consider the case of word-internal phoneme models.

IV. COMPARISON WITH WORD-CONDITIONED SEARCH

A. Review: Word-Conditioned Approach in DP Search

We review the word-conditioned approach for a general m -gram LM. Stretching notation, we use the symbol $Q_{v_1^{m-1}}(t, s)$ to denote tree-internal search hypotheses for time t and state s of the lexical tree with respect to predecessor word sequence v_1^{m-1}

$$\begin{aligned} Q_{v_1^{m-1}}(t, s) &:= \\ &= \max_{w_1^n: w_{n-m+1}^{n-1} = v_1^{m-1}} \Pr(w_1^{n-1}) \cdot \max_{s_1^t: s_t = s} \Pr(x_1^t, s_1^t | w_1^{n-1}) \\ &= \text{overall score of the best partial (state, word)-} \\ &\quad \text{sequence that at time } t \text{ ends in state } s \\ &\quad \text{of the lexical tree for predecessor words } v_1^{m-1}. \end{aligned}$$

At potential word boundaries, we have the DP recursion

$$Q_{v_2^m}(t, s = 0) = \max_{v_1} \left\{ p(v_m|v_1^{m-1}) \cdot Q_{v_1^{m-1}}(t, S_{v_m}) \right\}. \quad (4)$$

TABLE II
WORD-CONDITIONED LEXICAL TREE SEARCH FOR AN m -GRAM LM

proceed over time t from left to right	
ACOUSTIC LEVEL: process state hypotheses $\{Q_{v_1^{m-1}}(t, s)\}$	
- initialization: $Q_{v_1^{m-1}}(t-1, s=0) = H(v_1^{m-1}; t-1)$ $B_{v_1^{m-1}}(t-1, s=0) = t-1$	
- time alignment: $Q_{v_1^{m-1}}(t, s)$ using DP	
- propagate back pointers $B_{v_1^{m-1}}(t, s)$	
- prune unlikely hypotheses	
- purge bookkeeping lists	
WORD LEVEL: process word end hypotheses $\{Q_{v_1^{m-1}}(t, S_{v_m})\}$	
for each word $(m-1)$ -gram v_2^m do	
$H(v_2^m; t) = \max_{v_1} \{p(v_m v_1^{m-1}) \cdot Q_{v_1^{m-1}}(t, S_{v_m})\}$	
$\tilde{v}_1(v_2^m; t) = \operatorname{argmax}_{v_1} \{p(v_m v_1^{m-1}) \cdot Q_{v_1^{m-1}}(t, S_{v_m})\}$	
- store best predecessor $\tilde{v}_1 := \tilde{v}_1(v_2^m; t)$	
- store best boundary $\tilde{\tau} := B_{\tilde{v}_1 v_2^m}(t, S_{v_m})$	

In the word or tree interior, we have the usual DP recursion for time alignment

$$Q_{v_2^m}(t, s) = \max_{\sigma} \{p(x_t, s|\sigma) \cdot Q_{v_2^m}(t-1, \sigma)\}.$$

The operations of the word-conditioned search method are summarized in Table II. Comparing it with the time-conditioned variant as shown in Table I, we can see that the optimization over the word boundaries is fully incorporated into the time alignment, i.e. the handling of the tree-internal hypotheses. In this table, we make use of the quantities $H(v_2^m; t)$ and $H(v_1^{m-1}; t-1)$ which, in the context of this table, the reader may regard as pure auxiliary ones. However, in the next subsection, we will show that they are identical with the quantities $H(v_2^m; t)$ introduced for the time-conditioned method.

B. Equivalence of Time-Conditioned and Word-Conditioned Search

We will show the equivalence of time-conditioned and word-conditioned search. Although formally the two search variants must be equivalent by construction, it is instructive to see exactly how the equivalence can be established. We will consider the equivalence for a general m -gram LM. The two search approaches are compared in Fig. 5. For a fixed time t , this figure shows the set of active state hypotheses along with the associated paths for both the time-conditioned (on the left) and the word-conditioned approach (on the right). For each of the two approaches, there are several *copies* of state hypotheses; for the sake of simplicity, we have shown only three copies. For the time-conditioned approach, there are three start times $\tau = \tau_1, \tau_2, \tau_3$ with associated lists of predecessor word sequences v_1^{m-1} . For the word-conditioned approach, there are three predecessor histories $v_1^{m-1} = \varphi_1, \varphi_2, \varphi_3$ each of which again has various word boundaries (=end times of v_{m-1}).

In principle, the equivalence between the two search approaches is obtained as follows. For a hypothesis (t, s) with a predecessor word sequence v_1^{m-1} , we know that the best path must have passed the state $S_{v_{m-1}}$ at some time τ , where, of

course, the unknown boundary τ is obtained by maximization. Using this decomposition, we have for any state s

$$\begin{aligned} Q_{v_1^{m-1}}(t, s) &= \max_{w_1^n: w_{n-m+1}^{n-1}=v_1^{m-1}} \Pr(w_1^{n-1}) \cdot \max_{s_1^t: s_t=s} \Pr(x_1^t, s_1^t | w_1^{n-1}) \\ &= \max_{\tau} \left\{ \max_{w_1^n: w_{n-m+1}^{n-1}=v_1^{m-1}} \Pr(w_1^{n-1}) \cdot \max_{s_1^\tau: s_\tau=S_{v_{m-1}}} \Pr(x_1^\tau, s_1^\tau | w_1^{n-1}) \right. \\ &\quad \left. \cdot \max_{s_{\tau+1}^t: s_t=s} \Pr(x_{\tau+1}^t, s_{\tau+1}^t | -) \right\} \end{aligned}$$

with the probability of the best path through the lexical tree that starts at time τ and reaches state s at time t

$$\max_{s_{\tau+1}^t: s_t=s} \{\Pr(x_{\tau+1}^t, s_{\tau+1}^t | -)\} = \frac{Q_\tau(t, s)}{H_{\max}(\tau)}.$$

Thus we have the identity, which is to be interpreted as mathematical identity, *not* as recurrence equation

$$Q_{v_1^{m-1}}(t, s) = \max_{\tau} \left\{ H(v_1^{m-1}; \tau) \cdot \frac{Q_\tau(t, s)}{H_{\max}(\tau)} \right\}.$$

This identity summarizes the link between the word-conditioned search approach, i.e. the hypotheses $Q_{v_1^{m-1}}(t, s)$, and the time-conditioned search approach, i.e. the hypotheses $Q_\tau(t, s)$. This identity can be applied to word boundary hypotheses by setting $s = S_{v_m}$

$$Q_{v_1^{m-1}}(t, S_{v_m}) = \max_{\tau} \{H(v_1^{m-1}; \tau) \cdot h(v_m; \tau, t)\}.$$

Thus, we have

$$\begin{aligned} Q_{v_2^m}(t, s=0) &= \max_{v_1} \left\{ p(v_m | v_1^{m-1}) \cdot Q_{v_1^{m-1}}(t, S_{v_m}) \right\} \\ &= \max_{v_1} \left\{ p(v_m | v_1^{m-1}) \cdot \max_{\tau} \{H(v_1^{m-1}; \tau) \cdot h(v_m; \tau, t)\} \right\} \\ &= H(v_2^m; t). \end{aligned}$$

In the light of the above considerations, we can compare the computational effort for time-conditioned and word-conditioned search strategies: considering tree-internal hypotheses, we can see that one more optimization step is needed to get from time-conditioned search hypotheses to word-conditioned search hypotheses. Therefore, in summary, there are conflicting expectations as follows.

- The word-conditioned hypotheses seem to be more condensed since we have already got rid of the word boundary information which is not directly relevant for the ultimate task of speech recognition.
- The number of time-conditioned tree hypotheses might be smaller because the number of possible start times $\tau = 1, \dots, T$ is always smaller than the number of possible word histories in the word-conditioned search. The total number of possible time-conditioned tree copies is much smaller than in the word-conditioned variant. To be more

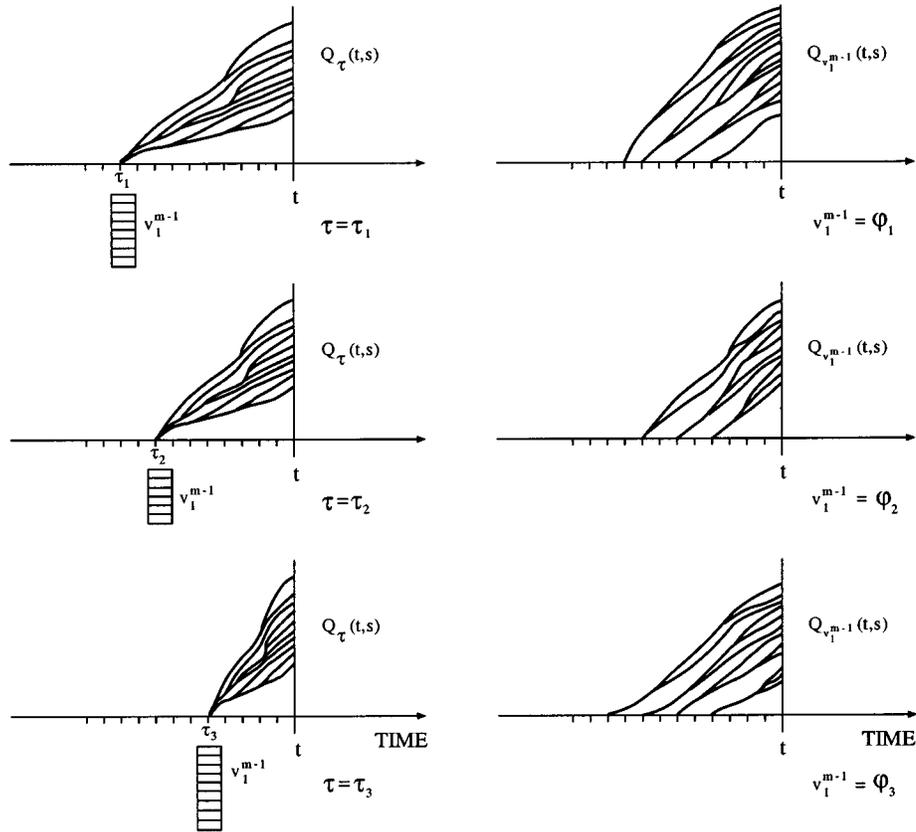


Fig. 5. Comparison of time-conditioned and word-conditioned search.

specific, we have to make some assumptions about the maximum duration of a sentence. As a worst case independent of the recognition task, we consider a sentence of length $T = 200 \text{ sec} = 20\,000$ 10-ms time frames and a trigram LM for $W = 60\,000$ words

$$T = 2 \cdot 10^4 \ll W^2 = 3.6 \cdot 10^9.$$

In particular, for larger vocabularies and more complex language models, this advantage might become even larger for the time-conditioned approach.

Ultimately, the answer to these conflicting views can only be given by the experimental test. Table III summarizes the main differences between both search variants.

C. Time-Conditioned Approach and Word Graphs

The time-conditioned DP approach leads in a straightforward way to word graphs. First, we recapitulate the problem of word graph construction.

Hypothesizing a word and its end time, how can we find a limited number of “most likely” predecessor words? This task is difficult since the start time of each word may very well depend on the predecessor word under consideration, which results in an interdependence of start times and predecessor words.

Considering the success of the one-pass beam search strategy, what we want to achieve conceptually, is to keep track of word sequence hypotheses whose scores are very close to the locally optimal hypothesis, but that do not survive due to the recombination process. The basic idea is to represent all these word sequences by a word graph, in which each edge represents a word

TABLE III
COMPARISON OF THE CHARACTERISTIC FEATURES BETWEEN THE TIME-CONDITIONED (TC) AND THE WORD-CONDITIONED (WC) SEARCH METHOD

characteristic features	search method	
	WC	TC
time synchronous processing of most promising hypotheses	yes	yes
optimization over word boundaries	integrated	separate
caching of intermediate results for LM recombination	no	yes
total number of state hyp. for more complex LMs	increases	constant

hypothesis. Each word sequence contained in the word graph should be close (in terms of scoring) to the single best sentence produced by the one-pass algorithm.

To construct a word graph, we introduce an explicit definition of the word boundary $\tau(w_1^n; t)$ between the word hypothesis w_n ending at time t and the predecessor sequence hypothesis w_1^{n-1}

$$\tau(t; w_1^n) := \underset{\tau}{\operatorname{argmax}} \{ G(w_1^{n-1}; \tau) \cdot h(w_n; \tau, t) \}.$$

It should be emphasized that the LM probability $\Pr(w_n | w_1^{n-1})$ does *not* affect the optimal word boundary according to (1) and is therefore omitted in the definition of the word boundary function $\tau(w_1^n; t)$. The crucial assumption now is that the dependence of the word boundary $\tau(t; w_1^n)$ can be confined to the final word pair w_{n-1}^n . The justification is that the other words have virtually no effect on the position of the word boundary between words w_{n-1} and w_n [26]. In general, this boundary,

i.e., the start time of word w_n as given by time alignment, will depend on the immediate predecessor word w_{n-1} : if the predecessor word w_{n-1} is sufficiently long, all time alignment paths are recombined before they reach the final state of the predecessor word. In formulae, we express this word pair approximation by the equation

$$\tau(t; w_1^n) = \text{const}(w_1^{n-2}) \quad \text{or} \quad \tau(t; w_1^n) = \tau(t; w_{n-1}^n)$$

i.e., the word boundary between words w_{n-1} and w_n does *not* depend on the predecessor words w_1^{n-2} . Under the assumption of this word pair approximation, it turns out that only a small modification in the bookkeeping of the word-conditioned tree search is needed [19]: when hypothesizing a word boundary as expressed by (4), the algorithm must keep track of the score $Q_{w_{n-1}}(t, S_{w_n})$ of *every* word pair (w_{n-1}, w_n) to generate a word graph rather than only the *best* predecessor word for each successor word w_n .

V. COMPARISON WITH STACK DECODING AND A^* SEARCH

In this section, we will present and discuss the similarities between the time-conditioned DP search and the so-called stack decoding or A^* search. Like DP, stack decoding [3] was already considered and used right after the introduction for HMM in acoustic modeling. As we will show, the time-conditioned variant of DP has some aspects bearing interesting analogies to stack decoding.

Over time, both stack decoding and DP went through several developments. For the following comparison, we consider primarily the variant of stack decoding as described in [8], [21], and [23]. In particular, we primarily use the implementation in [23] as reference for our comparison. First, we reformulate the main characteristics of this variant of stack decoding in the terminology and notation of this paper as follows.

- The recognition is performed in the maximum approximation.
- The partial word sequence hypotheses (w_1^n, τ) and the associated scores $G(w_1^n, \tau)$ play the central role in the formulation of stack decoding. For each end time τ , a separate priority queue or so-called stack is used to rank the partial word sequence hypotheses w_1^n . The full set of these priority queues is referred to as *multistack*.
- Strictly speaking, the concept of stack decoding requires a conservative estimate of the probability score (of both the acoustic model and the LM) for extending an arbitrary partial word sequence hypothesis w_1^n from time τ to the end of the speech signal. However, this is difficult without looking ahead of time τ to the end of the speech signal. Therefore, an approximation is applied: rather than looking ahead of time, the scores $G(w_1^n, \tau)$ are normalized with respect to the best score $\max_{w_1^n} G(w_1^n, \tau)$ and these *normalized* scores are then used to rank the partial word sequence hypotheses w_1^n .
- The key operation in stack decoding is the selection of the most promising partial word sequence hypothesis (w_1^n, τ) , its extension by one word w_{n+1} and the maintenance (reordering, updating) of the relevant priority queues. In particular the just extended partial word

sequence hypothesis w_1^{n+1} with end time t has to be included in this operation. In the spirit of the multi-stack concept, the shortest partial word hypotheses, i.e., those with shortest end time τ , are extended first. In [23], the extension is carried out using a lexical prefix tree for the pronunciation lexicon. The tree-internal state hypotheses are computed in a beam-like DP fashion. When a word-end state is reached, the LM probability is incorporated into the score and the relevant priority queues are updated.

- There are certain pruning steps [8], [22], [24] to limit the number of word sequence hypotheses. In [8], a so-called fast match is used to produce of short list of word candidates so that the acoustic matching, i.e., the computation of the score $h(w; \tau, t)$, needs to be carried out only for this set of words and not the *whole* vocabulary.
- In the case of an m -gram LM, the same LM recombination as in DP is performed to keep the number of potential word sequence hypotheses as small as possible.

Ignoring details, we can see that, in both stack decoding and time-conditioned DP search, two levels of hypotheses are defined and used: the level of partial word sequence hypotheses (w_1^n, τ) and the level of acoustic word hypotheses $(w_{n+1}; \tau, t)$. The main differences between the two approaches are related to the way in which the hypotheses are built up over the time axis as follows.

- In the time-conditioned DP search presented here, *all* operations are carried out in a strictly time-synchronous fashion; this applies to the scores at both the state level $Q_\tau(t, s)$ and the word level (w_1^n, t) . According to the concept of DP, the strategy is to look backward and compute the best *predecessor* hypothesis. There is never any sorting or explicit backtracking along the time axis.
- In stack decoding, the key concept is to look forward, i.e. to ask which of the partial word sequences is the most promising one and to extend that. As a result, some sorting or ranking of the partial word sequence hypotheses is required, and the extension of the best partial word sequence hypothesis with time τ is performed separately for each end time τ . By definition, this operation is *time-asynchronous*. Nevertheless, as reported in the variant [23], for each end time τ , we may compute the extension for *all* successor words (or some subset) in a *time-synchronous* way; in particular, a tree-structured lexicon of the vocabulary in connection with a beam search fashion can be used. Then the remaining difference to the time-conditioned DP search is that here *both* the extension of partial word sequence hypotheses *and* the extension of tree-internal state hypotheses are carried out in a strictly left-to-right time-synchronous fashion.

VI. EXPERIMENTAL RESULTS

A. Database and Definition of the Search Effort

To test the time-conditioned approach and to compare it with the word-conditioned approach, recognition experiments were carried out on the ARPA North American Business (NAB'94) H1 development corpus [10]. The test set comprises ten female

TABLE IV
SEARCH EFFORT AND WORD ERROR RATES FOR THE *TIME*-CONDITIONED
SEARCH METHOD ON THE NAB'94 H1 DEVELOPMENT CORPUS
FOR A 64 000-WORD VOCABULARY

LM type	F_{AC}	acoustic search space				comparisons in LM recomb.	word errors [%]	
		states	arcs	trees	words		DEL/INS	WER
bigram LM $PP = 237.1$	80	7474	2141	28	160	3392	2.8 / 1.8	14.8
	90	13644	3782	32	281	6959	2.8 / 1.6	14.0
	100	23349	6347	35	477	12942	2.8 / 1.5	14.0
	110	36945	9948	39	755	21727	2.7 / 1.5	13.9
	120	53583	14361	42	1099	30817	2.7 / 1.5	13.9
	130	71758	19158	44	1472	40644	2.7 / 1.5	13.8
	150	154028	40106	49	3264	92373	2.7 / 1.5	13.8
trigram LM $PP = 172.0$	80	7301	2092	28	155	8006	1.9 / 2.4	13.6
	90	13372	3708	31	272	15656	1.7 / 2.0	12.2
	100	22864	6217	35	461	28622	1.7 / 1.9	12.0
	110	36191	9749	38	731	46429	1.7 / 1.9	11.9
	120	52593	14099	41	1066	66667	1.7 / 1.9	11.9
	130	70616	18859	43	1434	86887	1.7 / 1.9	11.8
	150	151711	39507	48	3184	140530	1.7 / 1.8	11.8

and ten male speakers resulting in 310 sentences with 7387 spoken words. In all recognition tests, a 69 969-word vocabulary including 5234 pronunciation variants was used. The lexical prefix tree consists of 205 682 phoneme arcs where each arc corresponds to a six-state HMM as shown in Fig. 3. To allow intraphrase silence, a special arc with a one-state HMM is assigned to the root of the tree. Thirty-nine of the spoken words ($=0.5\%$) were not part of the vocabulary. A language model with a test set perplexity of $PP = 237.1$ and $PP = 172.0$ for the bigram and for the trigram case, respectively was applied [28].

The training of the emission probability distributions was performed on WSJ 0 and WSJ 1 training corpora. In this task, context dependent phoneme models (triphones, diphones, monophones) were used sharing 4699 emission probability distributions [7]. For this experiment we used about 270 000 Laplacian mixture densities (with a single pooled vector of absolute deviations) for each gender.

To give a quantitative measurement of the search effort, we report the following quantities in addition to the word error rate (WER):

- The *average size of the acoustic search space* is defined as the average number of state hypotheses, arc hypotheses, tree hypotheses and word end hypotheses per time frame after pruning. In the tables of the experimental results, these numbers will be referred to simply as *states*, *arcs*, *trees*, and *words*.
- The *average effort of the LM recombination* is measured as the average number of comparisons per time frame that have to be performed in the LM recombination as specified by (2) and (4) for the time-conditioned and word-conditioned approach, respectively.

The averaging is always performed over time frames of all test sentences. The LM pruning threshold was kept fixed for all recognition experiments. In the experiments, the most important parameter to be varied was the acoustic pruning threshold.

B. Results for Time-Conditioned Search

In a first series of experiments, we measured the search effort and the word error rate for the time-conditioned search method.

The recognition results are shown in Table IV for a bigram LM and a trigram LM. In both cases, the table reports the size of the search space and the word error rate (WER[%]) as a function of F_{AC} which is defined as a scale factor times the logarithm of the acoustic pruning threshold f_{AC} . The threshold F_{AC} was increased from 80 in steps of ten up to 130; in these experiments, a maximum number of $M_{Sta} = 150\,000$ states per time frame was used. In addition, in order to check for potential search errors, a recognition experiment with $F_{AC} = 150$ and $M_{Sta} = 300\,000$ was performed.

Looking at the numbers of the acoustic search effort, we can see that the results are very similar for both the bigram LM and the trigram LM. For both types of LM, there is a saturation effect for the word error rate: beyond 36 000 state hypotheses ($F_{AC} = 110$), the word error rate goes down only by 0.1% absolute, even if the number of states is increased to 150 000. What is remarkable for both type of LM is that the average number of tree hypotheses varies only between 30–50. In other words, the average time span of a state hypothesis, i. e. the average difference $t - \tau$ of a tree-internal hypothesis $Q_\tau(t, s)$, covers only 30–50 time frame which seems to reflect the average word duration.

As to the computational effort for the LM recombination, there is a clear difference between bigram and trigram LM as can be expected: the number of comparisons is higher for the trigram LM, namely by a factor of two. To illustrate the meaning of such a number, we analyzed in more detail the recognition experiment with $F_{AC} = 110$ for the trigram LM. Here the number of comparisons in the LM recombination of (3) averaged over all time frames is 46 429 and can be broken down as follows.

- There are 731 word end hypotheses $Q_\tau(S_{v_m}, t)$. However, there are only 133 distinct words v_m (not shown in the table) so that each word has about 5.6 different start time hypotheses τ .
- Since, for all 731 word end hypotheses, there is a total of 46 429 comparisons, we can compute the average number of predecessor hypotheses v_1^{m-1} in the lists $H(v_1^{m-1}, \tau)$ (with $m = 3$ for the trigram LM): $46\,429/731 = 63.5$.

As we will see in the following comparison, the effort in the LM recombination is an important difference between the time-conditioned and the word-conditioned search variants.

TABLE V
SEARCH EFFORT AND WORD ERROR RATES FOR THE WORD-CONDITIONED
SEARCH METHOD ON THE NAB'94 H1 DEVELOPMENT CORPUS
FOR A 64 000-WORD VOCABULARY

LM type	F_{AC}	acoustic search space				comparisons in LM recomb.	word errors [%]	
		states	arcs	trees	words		DEL/INS	WER
bigram LM $PP = 237.1$	90	5578	1544	15	58	58	6.1 / 1.2	22.1
	100	10823	2974	24	112	112	3.8 / 1.2	16.0
	110	20154	5535	37	211	211	3.2 / 1.4	14.5
	120	33661	9226	51	356	356	2.9 / 1.4	13.9
	130	50135	13664	65	532	532	2.9 / 1.5	13.9
	150	116482	30889	99	1302	1302	2.8 / 1.5	13.8
trigram LM $PP = 172.0$	70	1807	558	17	44	44	2.4 / 2.6	17.1
	80	3928	1157	29	90	90	2.0 / 2.2	14.0
	90	8153	2345	48	180	180	1.8 / 2.0	12.8
	100	15812	4516	73	332	332	1.8 / 1.9	12.1
	110	27599	7860	100	551	551	1.7 / 1.9	11.9
	120	42842	12134	125	817	817	1.7 / 1.9	11.9
	130	59642	16736	145	1101	1101	1.7 / 1.9	11.9
	150	133645	36250	208	2433	2433	1.7 / 1.9	11.9

C. Comparison with Results for Word-Conditioned Search

A second series of recognition experiments was run to test the word-conditioned search method and compare it with the time-conditioned search method. The experiments were carried out in the same way as before. The results, i.e., the search effort and the word error rate, are shown in Table V, again as a function of the acoustic pruning threshold F_{AC} . As in the time-conditioned search variant, there are three similar pruning operations; for more details, see [19].

The overall results of Table V are similar to those of Table IV. First of all, the best word error rates are the same as in the time-conditioned method: 13.8% for the bigram LM and 11.9% for the trigram LM. In addition, the acoustic search effort is roughly comparable in both cases although the word-conditioned method tends to require a somewhat smaller effort, typically by 10%–20%. What is different from the time-conditioned method is the result that the acoustic search effort goes up by a factor of 1.3–1.5 when switching from the bigram LM to the trigram LM. Of course, this effect can be expected due to the structuring of the search space. Furthermore, for higher numbers of state hypotheses, the number of tree hypotheses tends to increase significantly more for the word-conditioned method than for the time-conditioned method.

As can be expected, the most important difference between the two search variants is the computational effort for the LM recombination. For the word-conditioned search variant, the number of comparisons in the LM recombination of (4) is identical to the number of word end hypotheses. In general, this effort is drastically smaller than the corresponding effort in the time-conditioned method.

In addition, to further analyze the results with respect to the effect of potential search errors, we compared the results sentence by sentence for a bigram LM experiment with a word error rate of 13.9% and a trigram experiment with a word error rate of 11.9%. In each of these cases, we selected the recognition test with the smallest search effort. For each sentence, we checked whether the time-conditioned method produced an identical, a better or a worse score in comparison with the word-conditioned method. The results are summarized in Table VI. The table reports also the associated word errors. By looking at the com-

TABLE VI
COMPARISON OF THE TIME-CONDITIONED (TC) AND THE WORD-CONDITIONED
(WC)SEARCH METHOD (NAB'94 H1 DEVELOPMENT SET: 310
SENTENCES = 7387 WORDS)

LM type	number of sentences with:	word errors	
		TC	WC
bigram	identical score	259	692 / 692
	better score	46	310 / 309
	worse score	5	22 / 22
	total	310	1024 / 1023
trigram	identical score	292	794 / 794
	better score	18	83 / 85
	worse score	0	0 / 0
	total	310	877 / 879

parison of the sentence scores as reported in the table, a small advantage in favor of the time-conditioned method can be seen. The time-conditioned search seems to produce more sentences with better scores. Nevertheless, this does not seem to pay off in terms of word errors: a significant difference in the total number of word errors cannot be observed.

D. CPU Time Measurements

Finally, we run a third series of experiments to produce a detailed breakdown of the computational cost for the two search methods. These measurements were performed on an ALPHA 5000 PC (SpecInt'95: 15.4) using a subset of the NAB'94 H1 development corpus. This subset contains every first sentence of each speaker. Table VII shows the recognition cost measured by the real time factor (RTF) for each of the four major operations: acoustic likelihood calculation, acoustic search, LM recombination and LM access. In addition, the average search space and the word error rate are given. It can be observed that there is no significant difference in the overall CPU time for both methods. Due to the increased effort for LM recombination and LM access, the overall real-time factor is somewhat higher for the time-conditioned method than for the word-conditioned method: it goes up from 25.5 to 26.5 for the bigram LM and from 28.2 to 29.1 for the trigram LM. These recognition experiments indicate that both search methods are suitable for large vocabulary recognition and are roughly comparable in terms of search efficiency.

TABLE VII

BREAKDOWN OF THE SEARCH COST FOR BOTH SEARCH VARIANTS (SUBSET OF THE NAB'94 H1 DEVELOPMENT CORPUS: TEN FEMALE AND TEN MALE SPEAKER; 20 SENTENCES = 519 SPOKEN WORDS = 211.24 sec; ALPHA 5000 PROCESSOR WITH 15.4 SPECINT'95)

Type of LM	bigram PP = 259.5		trigram P = 191.8	
	TC	WC	TC	WC
search method				
search space: states:	35406	32579	34491	27323
arcs:	9552	8947	9307	7818
trees:	38	49	38	98
word error rate [%]	13.3	13.3	10.6	10.6
CPU time [RTF]				
acoustic likelihood	15.97	17.89	15.98	16.23
acoustic search	7.68	7.07	7.66	10.38
LM recombination	0.85	0.06	1.91	0.47
LM access	1.22	0.10	2.97	0.57
other operations	0.56	0.53	0.54	0.56
overall RTF	26.28	25.50	29.06	28.21

For both search variants, the lion's share of the computational effort is required for the calculation of the acoustic likelihoods. This calculation can be speeded up by a factor of about eight using a preselection method as proposed in [6], [20]. An acceleration of the pure acoustic search can be achieved by using a *bigram* LM rather than a *unigram* LM in the look-ahead in the word-conditioned method [15], [18]. However, for the time-conditioned search variant, it is not evident how to efficiently apply a bigram LM in the look-ahead. Therefore, in this paper, only the unigram LM look-ahead has been used.

VII. SUMMARY

This paper has presented the time-conditioned DP search method for large-vocabulary continuous-speech recognition. In terms of efficiency and recognition accuracy, the recognition experiments performed on the NAB'94 64 000-word task have demonstrated that this search variant is competitive with the widely-used word-conditioned search variant.

REFERENCES

- [1] F. Alleva, X. Huang, and M.-Y. Hwang, "Improvements on the pronunciation prefix tree search organization," in *Proc. IEEE Int. Conf. Acoustics, Speech, Signal Processing*, Atlanta, GA, May 1996, pp. 133–136.
- [2] L. R. Bahl, F. Jelinek, and R. L. Mercer, "A maximum likelihood approach to continuous speech recognition," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. PAMI-5, pp. 179–190, Mar. 1983.
- [3] L. R. Bahl and F. Jelinek, "Decoding for channels with insertions, deletions and substitutions with applications to speech recognition," *IEEE Trans. Inform. Theory*, vol. IT-21, pp. 404–411, July 1975.
- [4] J. K. Baker, "Stochastic modeling for automatic speech understanding," in *Speech Recognition*, D. R. Reddy, Ed. New York: Academic, 1975, pp. 512–542.
- [5] R. E. Bellman, *Dynamic Programming*. Princeton, NJ: Princeton Univ. Press, 1957.
- [6] E. Bocchieri, "Vector quantization for the efficient computation of continuous density likelihoods," in *Proc. IEEE Int. Conf. Acoustics, Speech, Signal Processing*, Minneapolis, MN, Apr. 1993, pp. 692–695.
- [7] C. Dugast, R. Kneser, X. Aubert, S. Ortmanms, K. Beulen, and H. Ney, "Continuous speech recognition tests and results for the NAB'94 corpus," in *Proc. ARPA Spoken Language Technology Workshop*, Austin, TX, Jan. 1995, pp. 156–161.
- [8] P. S. Gopalakrishnan, L. R. Bahl, and R. L. Mercer, "A tree-search strategy for large vocabulary continuous speech recognition," in *Proc. IEEE Int. Conf. Acoustics, Speech, Signal Processing*, Detroit, MI, May 1995, pp. 572–575.

- [9] R. Haeb-Umbach and H. Ney, "Improvements in time-synchronous beam search for 10000-word continuous speech recognition," *IEEE Trans. Speech Audio Processing*, vol. 2, pp. 353–356, Apr. 1994.
- [10] F. Kubala, "Design of the 1994 CSR benchmark tests," in *Proc. ARPA Spoken Language Technology Workshop*, Austin, TX, Jan. 1995, pp. 41–46.
- [11] B. T. Lowerre and R. Reddy, "The HARP Y speech understanding system," in *Trends in Speech Recognition*, W. A. Lea, Ed. Englewood Cliffs, NJ: Prentice-Hall, 1980, pp. 340–360.
- [12] H. Ney, R. Haeb-Umbach, B.-H. Tran, and M. Oerder, "Improvements in beam search for 10000-word continuous speech recognition," in *Proc. IEEE Int. Conf. Acoustics, Speech, Signal Processing*, San Francisco, CA, Mar. 1992, pp. 13–16.
- [13] H. Ney, D. Mergel, A. Noll, and A. Paeseler, "Data driven organization of the dynamic programming beam search for continuous speech recognition," *IEEE Trans. Signal Processing*, vol. 40, pp. 272–281, Feb. 1992.
- [14] H. Ney and S. Ortmanms, "Extensions to the word graph method for large vocabulary continuous speech recognition," *Proc. IEEE Int. Conf. Acoustics, Speech, Signal Processing*, vol. 3, pp. 1787–1790, Apr. 1997.
- [15] J. J. Odell, V. Valtchev, P. C. Woodland, and S. J. Young, "A one-pass decoder design for large vocabulary recognition," in *ARPA Spoken Language Technology Workshop*, Plainsboro, NJ, Mar. 1994, pp. 405–410.
- [16] M. Oerder and H. Ney, "Word graphs: An efficient interface between continuous speech recognition and language understanding," in *Proc. IEEE Int. Conf. Acoustics, Speech, Signal Processing*, vol. 2, Minneapolis, MN, Apr. 1993, pp. 119–122.
- [17] S. Ortmanms, H. Ney, F. Seide, and I. Lindam, "A comparison of time conditioned and word conditioned search techniques for large vocabulary speech recognition," in *Proc. IEEE Int. Conf. Spoken Language Processing*, Philadelphia, PA, Oct. 1996, pp. 2091–2094.
- [18] S. Ortmanms, A. Eiden, H. Ney, and N. Coenen, "Look-ahead techniques for fast beam search," in *Proc. IEEE Int. Conf. Acoustics, Speech, Signal Processing*, vol. 3, Munich, Germany, Apr. 1997, pp. 1783–1786.
- [19] S. Ortmanms, H. Ney, and X. Aubert, "A word graph algorithm for large vocabulary continuous speech recognition," *Computer, Speech, Language*, vol. 11, pp. 43–72, Jan. 1997.
- [20] S. Ortmanms, H. Ney, and T. Firzlauff, "Fast likelihood computation methods for continuous mixture densities in large vocabulary speech recognition," in *Proc. 5th Eur. Conf. Speech Communication Technology*, Rhodes, Greece, Sept. 1997, pp. 143–146.
- [21] D. B. Paul, "Algorithms for an optimal A* search and linearizing the search in the stack decoder," in *Proc. IEEE Int. Conf. Acoustics, Speech, Signal Processing*, Toronto, ON, Canada, May 1991, pp. 693–696.
- [22] —, "An efficient A* stack decoder algorithm for continuous speech recognition with a stochastic language model," in *Proc. IEEE Int. Conf. Acoustics, Speech, Signal Processing*, San Francisco, CA, Mar. 1992, pp. 25–28.
- [23] S. Renals and M. Hochberg, "Efficient search using posterior phone probability estimates," in *Proc. IEEE Int. Conf. Acoustics, Speech, Signal Processing*, Detroit, MI, May 1995, pp. 596–599.
- [24] —, "Efficient evaluation of the LVCSR search space using the NOWAY decoder," in *Proc. IEEE Int. Conf. Acoustics, Speech, Signal Processing*, Atlanta, GA, May 1996, pp. 149–152.
- [25] H. Sakoe, "Two-level DP matching—A dynamic programming-based pattern matching algorithm for connected word recognition," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-27, pp. 588–595, Dec. 1979.
- [26] R. Schwartz and S. Austin, "A comparison of several approximate algorithms for finding multiple (N-best) sentence hypotheses," in *Proc. IEEE Int. Conf. Acoustics, Speech, Signal Processing*, Toronto, ON, Canada, May 1991, pp. 701–704.
- [27] V. Steinbiss, B.-H. Tran, and H. Ney, "Improvements in beam search," in *Proc. Int. Conf. Spoken Language Processing*, Yokohama, Japan, Sept. 1994, pp. 1355–1358.
- [28] F. Wessel, S. Ortmanms, and H. Ney, "Implementation of word based statistical language models," in *Proc. SQEL Workshop on Multi-Lingual Information Retrieval Dialogs*, Pilsen, Czech Republic, April 1997, pp. 55–59.

ACKNOWLEDGMENT

Philips GmbH Forschungslaboratorien, Aachen, Germany, provided the 64 000-word pronunciation dictionary.



Stefan Ortmanns received the Dipl.-Ing. degree in mechanical engineering from Aachen University of Applied Sciences, Aachen, Germany, in 1987, and the Dipl.-Inform. and Ph.D. degrees in computer science from Aachen University of Technology in 1993 and 1998, respectively.

From 1993 to 1998, he was a Research Assistant with the Computer Science Department, Aachen University of Technology. In July 1998, he joined the Dialogue Research Department, Bell Labs–Lucent Technologies, Murray Hill, NJ. He is now with

Philips Speech Processing, Aachen, Germany.



Hermann Ney (M'86) received the Diplom degree in physics from the University of Goettingen, Germany, in 1977 and the Dr.-Ing. degree in electrical engineering from the Technical University of Braunschweig, Braunschweig, Germany, in 1982.

In 1977, he joined Philips Research Laboratories, Aachen, Germany, where he worked on various aspects of speaker verification, isolated and connected word recognition and large-vocabulary continuous-speech recognition. In 1985, he was appointed Head of the Speech and Pattern Recognition Group.

In 1988–1989, he was a Visiting Scientist with AT&T Bell Laboratories, Murray Hill, NJ. In July 1993, he joined the RWTH Aachen University of Technology, Germany, as a Professor of computer science. His work has concentrated on the application of dynamic programming and statistical techniques for decision-making in context. His current interests cover all aspects of pattern and speech recognition, such as signal processing, search strategies, language modeling, automatic learning, and language translation.