Arc Minimization in Finite State Decoding Graphs with Cross-Word Acoustic Context

François Yvon [a,1] Geoffrey Zweig [b] George Saon [b]

[a] *GET ENST and CNRS LTCI*
46 rue Barrault, F-75013 Paris

[b] *IBM T.J. Watson Research Center*
P.O. Box Yorktown Height, NY 10958, USA

Abstract

Recent approaches to large vocabulary decoding with weighted finite-state transducers have focused on the use of determinization and minimization algorithms to produce compact decoding graphs.This paper addresses the problem of compiling decoding graphs with long span cross-word context dependency between acoustic models. To this end, we extend the finite-state approach by developing complementary arc factorization techniques that operate on non-deterministic graphs. The use of these techniques allows us to statically compile decoding graphs in which the acoustic models utilize a full word of cross-word context. This is in significant contrast to typical systems which use only a single phone. We show that the particular arc-minimization problem that arises is in fact an NP-complete combinatorial optimization problem. Heuristics for this problem are then presented, and are used in experiments on a Switchboard task, illustrating the moderate sizes and runtimes of the graphs we build.

1 Introduction

In the past, there has been a significant division between the decoding processes used for highly constrained, small vocabulary speech recognition tasks, and those used for large vocabulary unconstrained tasks. In the small vocabulary arena, and in domains where a relatively compact grammar is appropriate, it is common to pre-compile a static state-graph. Given such a graph, a simple and efficient implementation of the Viterbi algorithm can be used for

[1] This work was performed while F. Yvon was visiting the IBM T.J. Watson Research Center

subsequent decoding (Viterbi, 1967). For large vocabulary tasks with n-gram language models (LMs), however, it has traditionally been common to avoid a static search space, and to instead dynamically expand the language model as needed (Jelinek et al., 1975; Odell, 1995; Ney and Ortmanns, 1999). While the latter approach has the advantage of never touching potentially large portions of the search space, it has the important disadvantage that dynamic expansion is significantly more complex, and incurs a run-time overhead of its own.

Remarkably, over the course of the past several years, algorithmic and computational advances have made it possible to handle large vocabulary recognition in essentially the same way as grammar-based tasks. In a recent series of papers (Mohri et al., 1998, 2000; Willett et al., 2001), it has been shown that it is in fact possible to statically compile a state graph that encodes the constraints of both a state-of-the-art language model, and cross-word acoustic context. One of the main algorithmic methods that is used in the process is that of determinization and minimization of the resulting weighted finite-state transducer.

While this previous work (Mohri et al., 1998, 2000) has established Viterbi decoding on statically compiled graphs to be an effective method for large vocabulary decoding, its use with very long-span acoustic models presents problems that have not been previously solved. As an initial step in this process, a cross-word acoustic-context model (typically triphone or quinphone) is encoded as a finite state transducer, and used in subsequent operations. As the amount of acoustic context increases, this transducer grows dramatically in size. Quoting (Mohri et al., 2000) "More generally, when there are n context independent phones, this triphonic construction gives a transducer with $O(n^2)$ states and $O(n^3)$ transitions. A tetraphonic construction would give a transducer with $O(n^3)$ states and $O(n^4)$ transitions." Further evidence of this increase of complexity is given in (Chen, 2003).

The motivation of this paper arises from a desire to utilize acoustic models with very long-span context sensitivity, where simply writing down a reasonably sized transducer that encodes the context sensitivity becomes a significant challenge. In particular, we are interested in utilizing a full-word of cross-word acoustic context, and in this case the number of arcs required to encode the context is very large (proportional to the square of the vocabulary size), and must be minimized.

To this end, we explore a graph building strategy which introduces supplementary states into selected portions of the decoding graph, in return for a large reduction in the number of arcs. The key difference from previous work is that we present a method for introducing *non*-determinism and extra states in return for reducing the number of arcs. Classical minimization operates on deterministic graphs.

A first question we study concerns the existence of tractable algorithms for building a graph with a minimal number of arcs. While it is already known that the related problem of minimizing non-deterministic finite-state automata (NFA) is NP-hard (see, e.g. (Jiang and Ravikumar, 1993)), it is important to note that on the face of it, our problem is significantly more constrained, and therefore perhaps not in the same complexity class. Nonetheless, through a reduction from the known NP-complete optimization problem of Clique Bipartitioning (Feder and Motwani, 1991), we demonstrate that in fact the problem we are faced with is also NP-complete.

We then propose several simple heuristics to reduce the number of arcs in the graph. The key to our factorization methods is the fact that it is relatively straightforward to enumerate - for each word - the sets of predecessor words that give rise to distinct context dependent acoustic realizations. We refer to such sets of predecessor words as *context sets*. By carefully identifying subsets of words that occur in multiple context sets, we will show that it is possible to factor them in such a way as to produce highly compact graphs, with a full word of acoustic context, even for large-vocabulary systems.

This paper is organized as follows. In Section 2.2, we present the basic structure of the decoding graphs and proceed in section 2.3 and 2.4 to illustrate the problem of arc minimization in the case of a unigram language model with various kinds of cross-word contextual dependencies. When larger n-grams are used, this unigram portion occurs as a subgraph, and accounts for the majority of the context-induced arcs.

In Section 3, we cast the problem formally, show that it is NP-complete, and present two simple heuristics for generating compact graphs. In Section 4 we apply these techniques, and present results on graph size, runtime, and word-error rate with various acoustic and linguistic models on the Switchboard and EARS Rich Transcription evaluations.

2 Decoding with static graphs

2.1 Motivations

Over the past few years, the technology of LVCSR systems has significantly evolved, making them able to accommodate increasingly large vocabularies and richer knowledge sources, while keeping computing costs at a reasonable level. In this section, we discuss the benefits of precompiling the search network

into a finite-state graph in the context of single pass search strategies. [2] In fact, we will only be considering single pass algorithms, since we believe that the benefits (in terms of pruning and search efficiency) of including all the available knowledge sources as early as possible largely compensate for the increased complexity of the search space.

(Aubert, 2000) presents a thorough review of the main search algorithms, showing that they basically fall into two major categories:

- those which use a static expansion of the search space, which is then searched using traditional DP techniques and beam pruning: this approach is reminiscent of the traditional approach to small vocabulary speech recognition. General algorithms for weighted finite-state transducer (FST) determinization and minimization (Mohri and Riley, 1998) can additionally help reduce the search space size.
- those which expand the search space on-the-fly; this strategy accommodates arbitrary search strategies such as word synchronous or time synchronous beam search, and depth first search or stack decoding

The benefits of statically precompiling an optimized version the search space are numerous:

- searching a weighted finite-state graph is a well understood task, for which simple, general, yet efficient algorithms can be readily used. Furthermore, the speed vs accuracy tradeoff is controlled by a single parameter: the beam width.
- empirically, when pruning is done, reducing the amount of non-determinism in the searched network reduces the work done by the decoder. (Note, though, that the basic Viterbi HMM recursions make no distinction between deterministic and non-deterministic graphs. Therefore this observation is specific to the kind and amount of pruning that is used.)
- the minimization of weighted FSTs (Mohri, 1994) has the beneficial effect of redistributing transition costs in such a way that language models probabilities are applied as early as possible, in a way similar to language model look-ahead techniques (Ortmanns et al., 1997). This dramatically improves the efficiency of pruning.

These observations are reinforced by the results of a head-to-head comparison of these two search strategies (Kanthak et al., 2002), which demonstrate that the FST approach permits significantly better operating points than on-the-fly expansion strategies.

[2] In this paper, we use the terms "finite-state decoding graph " and "static decoding graph" to refer to what might more properly be termed a (weighted) finite state automaton. As we associate acoustic emissions with the states and transition probabilities with the arcs, this is also essentially a standard HMM.

Static decoding graph compilation is often performed through the formal composition of several weighted finite-state transducers (Mohri et al., 2000), formally expressed as: $D = H \circ C \circ L \circ G$, where H denotes the HMM FST; C denotes the context-dependency FST, mapping context-dependent acoustic units to phones; L denotes the lexical transducer, mapping phone sequences to word sequences; and G denotes the language model, associating word sequences with probabilities. Further, this compilation involves the additional optimizations of determinization and minimization.

Given the inherent non-determinism present in C (due to the presence of homophones) and L (due to back-off transitions), D is not fully deterministic. Nevertheless, using various tricks, such as determinizing the lexical transducer with additional phonetic symbols, and treating epsilon as a regular symbol in the determinization procedure, very compact graphs can be produced allowing for fast and efficient decoding in many circumstances.

In this context, our goal is to explore the integration of long span contextual dependencies in this architecture. As pointed out in (Aubert, 2000), integrating triphone cross-word acoustic dependencies hardly increases the overall network size. However, as discussed in (Mohri et al., 2000), and more recently in (Chen, 2003), integrating increasingly large contexts make C grow tremendously in size, rendering the compilation of the static graph a serious computational challenge. The problem we address here consists in compiling D in a case where the contextual dependencies modeled in C extend up to *one full word of context* on the left or on the right. It is important to realize that while our experiments only involve specific forms of cross-word dependencies relying on a decision tree to select acoustic context dependent acoustic units, our graph compilation algorithm can in principle accommodate more general patterns of cross words dependencies.

Other work aimed at modeling complex phenomena in the finite-state framework , while keeping the search space reasonably small, includes an a priori simplification of the LM (Mohri et al., 2000) using a compression technique originally proposed in (Seymore and Rosenfeld, 1996); alternative strategies for disambiguating the phonological and lexical transducer (Smaili et al., 2002); a factorization of the LM into a simple LM, which is included in the static graph, and a dynamic LM, expanded on the fly (Dolfing and Hetherington, 2001; Willett and Katagari, 2002); and heuristic state merging strategies (Zheng and Franco, 2002).

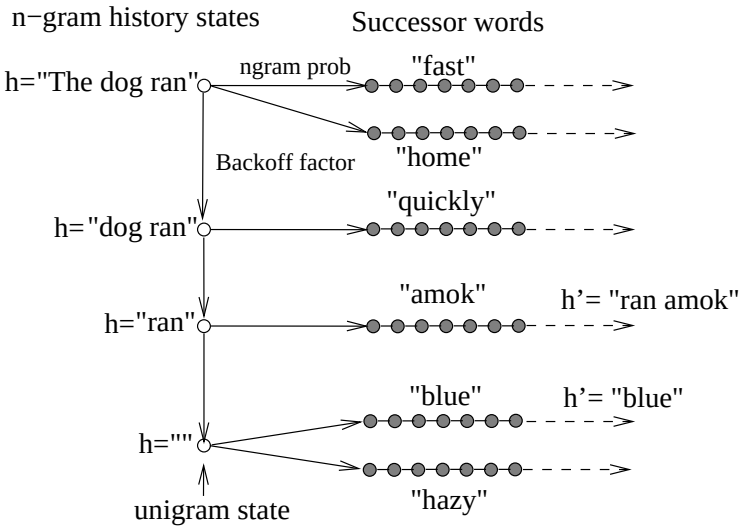

Figure 1. A graph with word internal context only. Arcs emanating from the right-hand side loop back to states on the left.

2.2 Word-Internal Graphs

We begin our discussion of graph structures by illustrating the basic structure of a classical n-gram language model viewed as a stochastic finite-state machine (Jelinek et al., 1975). When LM probabilities are smoothed according to a back-off scheme (Katz, 1987), this automaton can be efficiently factored with the help of non-deterministic transitions (Placeway et al., 1993; Riccardi et al., 1996). Each history h appearing in the LM corresponds to a *history state* N_h in the FST; each word w such that $P(w \mid h)$ occurs in the LM corresponds to a transition weighted with $P(w \mid h)$, labeled with w, between N_h and $N_{h'}$, where h' is the history having the longest common suffix with hw. This basic structure is complemented to account for the back-off model: each history state N_h also has an outgoing epsilon transition to $N_{\overline{h}}$, where $\overline{h}$ is a truncated history. This transition is weighted with the back-off coefficient of history h. The case where $\overline{h}$ is empty is handled via one additional unigram state, which has an outgoing transition for every word in the vocabulary V, weighted with the corresponding unigram probability.

Replacing word labels with the sequence of acoustic states induced by their pronunciation yields the final decoding graph (see on Figure 1). Such graphs can be further processed with general algorithms for weighted FSTs, such as determinization and minimization.

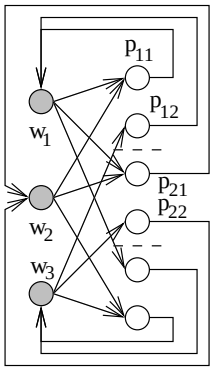

Figure 2. Graph representing left-word acoustic context constraints. There are three vocabulary words, each with two context-sensitive variants. These are separated by dotted lines. Arcs out of the variant-states are labeled with the word value; the others are epsilon arcs.

2.3 Left-Context Graphs

We now consider the case when the acoustic realization of a given word is a function of the previous word occurrence, as is the case with cross-word contextual models. For this purpose, we now assume that each word w_i has l_i acoustic variants $p_i^1 \dots p_i^{l_i}$, and that each variant p_i^k can only occur if the previous word belongs to the set $Left(p_i^k)$. We denote the total number of acoustic variants by P. As for transitions out of history states, this new situation is straightforwardly taken care of, by making the outgoing transitions of N_h comply with the pronunciation constraints induced by the last word of history h. To illustrate this situation, imagine that "home" has two contextual variants: we would instantiate on the arc from the history node labeled "The dog ran" on Figure 1 the one single variant which can occur in the context of a preceding "ran". However, for the unigram state, the simple factorization described above fails: upon reaching this state, the identity of the previous word is lost, making it no longer possible to apply the contextual constraint.

An obvious solution to this, in which there is a distinct unigram-history state for each word, is illustrated in Figure 2. This kind of brute-force solution is straightforward, and can accommodate any context-sensitivity pattern. However, it is possible to significantly improve on it by carefully grouping words on the basis of the left context variants they induce, as illustrated in Figure 3. This second example uses a four word vocabulary, where each word, except the last, has a single left-context variant. The last word has four variants. The graph on the left indicates which variants are licensed by which words, in a brute-force fashion. It uses 16 arcs, and has a minimal number of vertices. The graph on the right models the very same dependencies by introducing an extra vertex , and in return reduces the number of arcs to 11. Graphs of this form - having n words where the first $n-1$ have a single variant, and the last

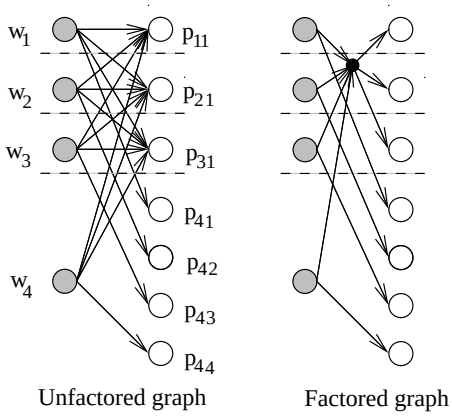

Figure 3. Factoring left contexts. The backwards loops from variants to words have been omitted for clarity. The shared left-context of p_{11}, p_{21}, p_{31}, composed of the words w_1, w_2, w_3, w_4 p is factored out on the right.

has a unique variant for each of the n words - will in general require n^2 arcs if constructed in the straightforward way, but just $3n - 1$ arcs when factored.

By factoring the unigram portion of a left-context decoding graph, we can produce graphs that have many fewer arcs than the theoretical limit of $O(V^2)$. Further, with a small amount of additional effort, the classical techniques of determinization [3] and minimization for weighted FSTs can also be used. In order to avoid an explosion in the size of the determinized graphs, we employ a two-step process:

(1) Label each outgoing edge from a history-set state by a dummy label consisting of the index of the history-set.
(2) Determinize and minimize this graph.
(3) Replace the dummy labels with epsilon
(4) Determinize and minimize the graph

The first step is necessary because without it, there are so many epsilon transitions that the process of determinization uses over 4GB of memory and fails. This is the case even though epsilon is treated as a normal label. It is well known that determinization may exponentially increase the size of a graph (Hopcroft and Ullman, 1979), and in this case the theoretical problem manifests itself in reality.

We have found that the process of determinization and minimization reduces the size of the graph somewhat, and produces a significant improvement in pruning properties. Therefore, all the experimental results presented in Section 4 are for weighted FSTs that have been (pseudo)-determinized and minimized

[3] As is common practice, the determinization procedure does not include full epsilon removal: epsilon transitions are thus treated just like regular (labeled) transitions. Hence the term 'pseudo-determinization.

after our arc-factored construction process.

To summarize, the graph building procedure requires the following steps:

(1) build the unigram graph and compact it using the arc factorization presented in section 3
(2) combine the unigram portion with higher-order components
(3) apply (pseudo) determinization and minimization

In comparison with earlier methods of graph construction (Mohri et al., 2000), we avoid the step of writing down the extremely large context transducer ($O(n^k)$ with n phones and a window size of k) while still being able to apply weighted determinization and minimization. It is also worth noting that our method is actually completely insensitive to the number of phones in the context window, depending only on the fact that there is a single word to the left.

2.4 Right-Context graphs

Dealing with right context dependency patterns, ie. with cases where the pronunciation of a word depend on the successor words, implies a small change from the graph structure presented in Figure 1. Ignoring the special case of the unigram back-off state for a moment, what makes the implementation of left-contextual constraints so straightforward is that each history state encodes the knowledge of the last words, including crucially the last seen one. As a consequence, it is trivial to constrain any arc going out of history state N_h, with $h = \alpha w$, to be labeled with a pronunciation p such that $w \in Left(p)$.

With right context dependencies, the syntactic constraints imposed by the language model no longer operate in a synchronous fashion with the pronunciation restrictions: while the former constraints still apply depending on the past history, the latter now vary according to the future word. Nevertheless, these dependencies can be accommodated in the finite-state framework in the following manner: for each history $h = \alpha w_i$ in the language model, we introduce as many history states N_{h_k} as there are right context variants $p_i^1 \dots p_i^k$ of w_i. For each successor word v of h, G includes an arc from N_{h_k}, labeled with (the sequence of HMM states corresponding to) p_{i_k} if and only if $v \in Right(p_{i_k})$. This arc will point to all the history states $N_{h'_j}$, where h' is, as before, the history having the longest common suffix with hw. Epsilon transition weighted with back-off factors will continue to joint history nodes N_{h_i} with $N_{\overline{h}_i}$, with $\overline{h}$ the truncated history (see Figure 4). As previously, determinization and minimization can be further applied to this graph.

The case of the unigram backoff arc can be dealt with in a manner that is

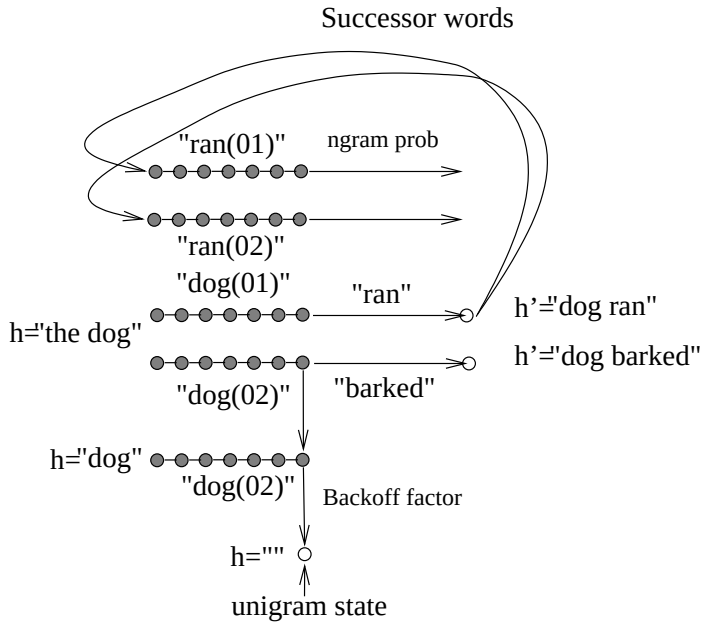

Figure 4. A graph with right word context only. Arcs emanating from the right-hand side loop back to states on the left. Indices (01) and (02) refer to the acoustic realizations of a word.

exactly similar to the left-context case: first by introducing a unigram backoff node for each possible pronunciation variant, then by factoring out the resulting bipartite graph is such a way that the total number of resulting arcs is made minimal.

3 Arc minimization of decoding graphs

3.1 Problem Definition

Before presenting our graph minimization strategies, we introduce some definitions. ($G = (X = (L, R), E)$ is a bipartite graph if the vertices in X can be partitioned as $X = L \cup R$, and all edges in E link a vertex in L with a vertex in R. We denote $n(G)$ the order of G (= the total number of vertices). A biclique $B = (X' = (L', R'), E')$ in G is a complete partial subgraph of G, meaning that E' includes every possible edge from L' to R'. An edge cover of G into biclique is provided by subsets $E_1 \ldots E_k$ of E such that (i) each E_i is a biclique and (ii) $E = \bigcup E_i$. When the E_i are pairwise disjoint, then $E_1 \ldots E_k$ further defines a partition of E. We call *the order of a cover (or partition)* the sum of the orders of all bicliques it contains.

Turning back to our decoding graph, we can see that the unigram backoff portion of it defines such a bipartite graph with $L = V$ (the vocabulary),

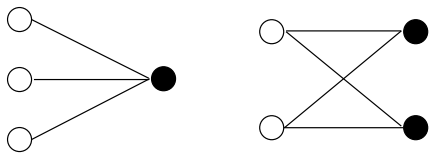

Figure 5. Simple bicliques: 1×3-star and $K_{2,2}$

$R = P$ (the acoustic variants), and E containing one edge between the unigram state for word w_i and each left-context variant p_j^k which can follow w_i (see Figure 3). G has a total of $|V| + |P|$ vertices, and $|V|^2$ arcs. To make the decoding graph smaller, we are looking for ways to factor out dependencies expressed in G by introducing new vertices so as to reduce the number of arcs in the final decoding graph. Each biclique $B = (V', P')$ in G expresses the fact that any variant in P' can follow any word in V': this means that if we introduce an extra state as we did on the graph Figure 3, we can replace the $|V'| * |P'|$ arcs in the decoding graph by just $|V'| + |P'| = n(B)$. Minimizing the number of arcs in the graph expressing contextual dependencies thus amounts to finding a set of bicliques $B_i = (V_i, P_i)$ in G such that:

$$\forall e \in E, \exists i \text{ st. } e \in B_i \tag{1}$$

$$\sum_{i=1}^{i=k} n(B_i) \text{ is minimal} \tag{2}$$

or, in other words, to finding the minimal order cover of edges in G into bicliques.

3.2 NP Completeness

The decision problem associated with this minimization program is clearly in NP. In this section, we show that this minimization problem is, in fact, NP-hard. The proof directly derives from a result of (Feder and Motwani, 1991), which states than finding the minimum order *partition* into bicliques for any given graph is NP-hard. Their argument relies on a variation of a reduction of 3-SAT originally proposed in (Holyer, 1981), and proceeds along the following lines. First, a polynomial transformation of any formula F into a graph $G(F)$ is exhibited, which is such that $G(F)$ only contains 'simple' bicliques: the only bicliques included in $G(F)$ are either $K_{2,2}$, the 2×2 biclique, or stars ($q\times1$ bicliques) (see Figure 5).

(Feder and Motwani, 1991) further shows that F is satisfiable if and only if the edges in $G(F)$ can be partitioned into $K_{2,2}$. Furthermore, any partition of $G(F)$ into bicliques is bound to have an order greater than the total number of edges in $G(F)$, since each biclique type contains at least as many vertices as

edges. This lower bound is only achieved in the case all the bicliques are of the kind $K_{2,2}$, as $1\times k$-stars have indeed more vertices $(k+1)$ than arcs (k). Since the total number of edges in the partition is fixed, (Feder and Motwani, 1991) claim that if we could find in polynomial time the minimum order partition, we could decide whether the edges of G can be partitioned using only $K_{2,2}$s, and thus answer the satisfiability question. Extending their argument to covers is fairly simple, as the minimum order cover of $G(F)$ necessarily contains fewer vertices than the minimal order partition (a partition being a special case of a cover). Since the cover only involves $K_{2,2}$ and stars, we know that the total order of the cover will still be at least equal to the number of edges it covers, which is greater than the number of edges in $G(F)$; furthermore, this bound can only be attained by a partition, as covers can include duplicate edges. We can conclude, by the same argument, that finding the minimal order cover is also a NP-hard problem.

3.3 Alternative formulations

We have focused so far one of the several possible formulations of the arc minimization problem. Before introducing various heuristic solutions for getting through this combinatorial problem, we introduce here two alternative formulations which will help getting a better intuition of our heuristics.

3.3.1 Set theoretic formulation

The relationship $\mathcal{R}$ between words and valid successor pronunciation variants can be represented based on set theoretic concepts only. Any biclique in G defines a pair of sets (X, Y) with $X \in 2^V$ and $Y \in 2^P$, such that $\forall (w,p) \in X \times Y, w \in Left(p)$. The set $\mathcal{B}$ of all bicliques in G can then be naturally ordered with the $\leq$ defined as: $(X,Y) \leq (X',Y')$ if and only if $X \subset X'$ and $Y' \subset Y$. $(\mathcal{B}, \leq)$ defines a lattice structure, known as the Galois lattice of the relationship, and denoted $\mathcal{L}$. The infimum of two elements (X,Y) and (X',Y') is computed as: $(X \cap X', Y \cup Y')$. Given this new definitions, we can reformulate our optimization problem as finding a set of bicliques (V_i, P_i) in $\mathcal{L}$ such that:

$$\bigcup V_i \times P_i = V \times P \tag{3}$$

$$\sum_i |V_i| + |P_i| \text{ is minimum} \tag{4}$$

This formulation of the problem allows the following observation: if (V,P) and (V',P') define a solution, and we have $(V',P') < (V,P)$, then we can improve on this solution by replacing (V',P') with $(V, P' \backslash P)$ which must also

occur in $\mathcal{L}$. This new solution still satisfies the constraint (3) and improve the total cover order of a quantity equal to: $|P \cap P'|$. As a result, any optimal solution will only include pairs of sets which are pairwise incomparable, and will define an anti-chain in $\mathcal{L}$. This fact is used in one of the heuristic for arc minimization (see 3.4).

3.3.2 Algebraic formulation

Denote by $\mathbf{A} \in \{0,1\}^{m \times n}$ the adjacency matrix of the graph, i.e. $a_{ij} = 1$ if and only if there is an edge between node $i \in L$ and $j \in R$ and $a_{ij} = 0$ otherwise. Suppose first that we wish to find an edge partition of the graph. This amounts to finding a factorization of $\mathbf{A}$ as

$$\mathbf{A} = \mathbf{BC}, \qquad \mathbf{B} \in \{0,1\}^{m \times l}, \mathbf{C} \in \{0,1\}^{l \times n}$$

In the case of a cover, the previous equality turns into the following set of inequalities

$$\begin{cases} \sum_{k=1}^{l} b_{ik} c_{kj} \leq a_{ij}, \; a_{ij} = 0 \\ \sum_{k=1}^{l} b_{ik} c_{kj} \geq a_{ij}, \; a_{ij} = 1 \end{cases}$$

Indeed, the cover should not introduce additional edges and the same edge can be accounted for in at least one way (i.e. at least one k for which $b_{ik} = c_{kj} = 1$). For a given l, the objective function to be minimized is the sum of elements of B and C

$$\min_{b_{ik}, c_{kj} \in \{0,1\}} \sum_{i,k} b_{ik} + \sum_{k,j} c_{kj}$$

subject to the previous constraints. In order to find the minimum order cover, the above quadratic $(0,1)$-integer programming problem has to be solved for every $l = 1 \ldots mn$ and the best solution has to be retained.

3.4 Two Heuristics for Arc Minimization

In this section, we present two simple heuristics that have proven both to perform reasonably well and to remain tractable even for large vocabulary tasks. Both methods begin by identifying the variants $p_i^1 \ldots p_i^{l_i}$ of each word w. For the type of cross-word dependencies we have considered, this can be

done in a brute-force fashion by enumerating all possible predecessor words and using a decision tree to identify the corresponding sequence of context dependent states. Concurrently, for each word, we obtain a partitioning of V into the sets $Left(p_i^k)$ that induce the different variants. Since $|\bigcup_k Left(p_i^k)| = |V|$, the space required just to store all these sets is proportional to the square of the number of vocabulary words. For vocabularies over about 10,000 words, this is impractical, and we had to resort to simple local heuristics, which do not require the knowledge of the entire graph. Our algorithms thus work in an online fashion, examining each $Left(p_i^k)$ as it is enumerated, and then moving on.

The first algorithm is presented in Figure 6. It maintains a collection of history sets, each with an associated set of acoustic variants. It proceeds word-by-word computing the Cartesian intersection between the current set of history sets and the sets $Left(p_i^k)$ of word w_i. Thus, the members of each history set are guaranteed to induce the same behavior with respect to all the words seen so far. This strategy basically amounts to proceeding greedily upwards in the lattice $\mathcal{L}$, repeatedly computing the supremum of sets of pairs all having the form $(V = Left(p), P = \{p\})$, while trying to avoid reaching nodes where V would be empty.

If run to completion (i.e. over all words $w_1 \ldots w_n$), one gets sets of words that behave identically with respect to their successors. However, this tends to result in overly small sets, and in practice, after a given number of sets (e.g. 100) have been generated, it is better to store them, and "reset" the algorithm (Figure 6).

Our second heuristic (see Figure 7) relies on the following observation: a word set S, corresponding to the left context set of a pronunciation p_j^k, must group words having some similarity with respect to their right acoustic environment.

In the limit, observing a two-word history set suggests that these two words must be very similar, being the only pair in the lexicon to trigger a specific left-context variant. It is therefore reasonable to assume that this pair will in fact co-occur in a large number of bigger history sets. Extending the argument to larger sets, we can see reasons why medium sized context sets should thus provide an advantageous basis for decomposing the remaining history sets. The heuristic of Figure 7 exploits this observation.

The main parameter of the basis set decomposition algorithm is θ_1. To appreciate qualitatively why, consider the extreme case of $\theta_1 = 1$: in that case, we end up with $|V|$ $1\times$ $|V|$ stars, corresponding to the simple-minded one-unigram-state-per-word solution of Figure 3. Each time we increase θ_1 and include a new attested basis set $B = (H, P)$ of size $|H|$, we actually add $|H|$ supplementary vertices to the total cover order while potentially removing up

Input: Sequential presentation of the sets $Left(p_i^k)$.
Output: Set of history sets $\mathcal{H}$ and licensed variants $\mathcal{P}$.
Data Structures:
$\mathcal{H} : \{H_1, H_2, \ldots H_n\}$. Each H_i is a set of words.
$\mathcal{P} : \{P_1, P_2, \ldots P_n\}$. Each P_i is a set of word variants.
(1) $\mathcal{H}_{final} \leftarrow \emptyset \quad \mathcal{P}_{final} \leftarrow \emptyset \quad i \leftarrow 1$
(2) $\mathcal{H} \leftarrow \{\{V\}\} \quad \mathcal{P} \leftarrow \{\emptyset\}$
(3) Repeat until $|\mathcal{H}| > threshold$
- Process w_i:
  - $H'_{jk} \leftarrow H_j \cap Left(p_i^k)$
  - $P'_{jk} \leftarrow P_j \cup p_i^k$
- $\mathcal{H} \leftarrow \mathcal{H}'$
- $\mathcal{P} \leftarrow \mathcal{P}'$
- $i \leftarrow i + 1$

(4) $\mathcal{H}_{final} \leftarrow \mathcal{H}_{final} \cup \mathcal{H}$
(5) $\mathcal{P}_{final} \leftarrow \mathcal{P}_{final} \cup \mathcal{P}$
(6) if $i = |V|$ output $\mathcal{H}_{final}$ and $\mathcal{P}_{final}$ and end
(7) else goto step 3.

Figure 6. Cartesian intersection algorithm for computing history sets and licensed acoustic variants.

to $|H|$ vertices from existing bicliques. With increasing values of $|H|$, $|P|$ actually gets smaller up to a point where increasing θ_1 actually hurts the performances. This is illustrated on Figure 8 for different values of θ_1: while the steady increase of $\Sigma_i\ |H_i|$ is initially more than rewarded by the decrease of $\Sigma_i\ |P_i|$, substantially reducing the total order of the cover, an optimum seems to be reached around $3,500$, after which the total decomposition order starts increasing.

It is important to realize that Figure 8 only shows a small portion of the curve: for smaller values of θ_1, the total order in fact increases substantially. For instance, it already attains $2.6e+6$ for $\theta_1 = 2000$, demonstrating that the choice of large enough values for θ_1 is crucial for building small graphs.

As we illustrate in the following section, these heuristics do a good job in generating compact graphs. They are, however, heuristics, and consistent with the NP-hardness of the problem are not guaranteed to find an optimal answer.

Input: Sequential presentation of the sets $Left(p_i^k)$.
Output: Set of history sets $\mathcal{H}$ and licensed variants $\mathcal{P}$.
Data Structures:
$\mathcal{B}$: *Basis set.*$\{(H_1, P_1), (H_2, P_w) \ldots (H_n, P_n)\}$. Each H_i is a set of words, each P_i is a set of word variants.
(1) $\mathcal{B} \leftarrow \{(w_1, \emptyset), (w_2, \emptyset) ... (w_n, \emptyset)\}$
(2) for each word variant p_i^k, if $|Left(p_i^k)| < \theta_1$
$\mathcal{B} \leftarrow \mathcal{B} \cup \{(Left(p_i^k), \emptyset)\}$
(3) For each word variant p_i^k:
- find $\{j_1 \ldots j_l\}$ st.
  - $\bigcup_{j=j_1}^{j_l} H_j = Left(p_i^k)$
  - $\Sigma_{j=j_1}^{j_l} |H_j|$ is maximum
- for each $j = j_1 \ldots j_k, P_j \leftarrow P_j \cup \{p_i^k\}$

(4) for each $(H_i, P_i) \in \mathcal{B}$ if $|P_i| < \theta_2$ and $|H_i| > 1$
$\mathcal{B} \leftarrow \mathcal{B} \setminus (H_i, P_i)$
(5) if $\mathcal{B}$ was changed during 4 goto 3
(6) output $\mathcal{B}$

Figure 7. Basis-set algorithm for computing history sets and licensed acoustic variants.

Step 1 initializes the basis set with singleton words. This ensures that the decomposition process in step 3 always succeeds. Step 2 simply accumulates word sets no larger than θ_1 as potential basis sets. Based on the basis sets collected during 1 and 2, step 3 actually greedily computes a cover. An additional pruning step (4) removes basis sets, based on their actual contribution to the cover, discarding those which don't cover enough arcs. The condition ($|H_i| > 1$) guarantees that singletons aren't pruned out during this stage: as mentioned earlier, having all the singletons in the basis set is a sufficient condition for the decomposition step (3) to succeed. This algorithm terminates after exactly two iterations: if a basis set is not pruned at the first iteration, it cannot be pruned at the second, as its contribution may only increase as other basis sets get discarded.

4 Experimental Results

4.1 System description

For the experiments reported in Section 4.2, we used a Switchboard system based on a 18K vocabulary, with more than 300K left-context variants. Speech features are derived from 24-dimensional MFCCs, further transformed into a canonical space through the application of normalization techniques (VTLN and FMLLR), and then projected onto a discriminative 60-dimensional space using heteroscedastic discriminant analysis (HDA) (Saon et al., 2000). Acoustic modeling uses cross-word context-dependent HMMs: the context dependent states of any given phone are determined through the application of a

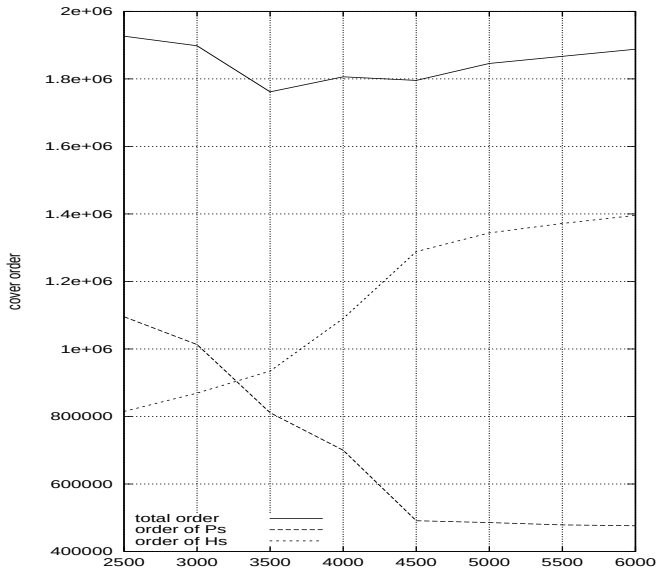

Figure 8. Choosing the right basis set size

This figure plots the total order of the cover for various value of θ_1 (see Figure 7). For these experiments, the value of θ_2 is fixed and equals to 5.

decision-tree making its decision based on a 11-phone window. This window can potentially look at all the surrounding phones up to one full word of context *on the left*; the right context can extend up to the current word boundary.

The complete acoustic model set includes 142K Gaussian models and about 3.7K HMM states for the word internal system, and about 160K Gaussian models and 4.6K HMM states for the cross-word system. The acoustic models were trained on a combination of Switchboard and Callhome data using discriminative training (MMI) techniques (Nadàs et al., 1988; Woodland and Povey, 2000). Language models (bigram and trigram) were trained on Switchboard transcripts and were smoothed with a modified version of Kneser-Ney backoff (Chen and Goodman, 1996); the cut-off counts were set to 1 for trigram. n-gram counts are given in Table 1.

Table 1
Language model size

1-gram	2-gram	3-gram
17.5K	388K	327K

4.2 Results

In this section, we present experimental results obtained using the Switchboard'00 evaluation set (Switchboard part).

The first results we report here concern the comparative efficiency of the graph

factorization procedures presented in Section 3.4. These results, which only concern the unigram back-off section of the graph, are displayed in Table 2.

Table 2
Arc factorization results: the number of vertices and edges in the unigram backoff subgraph.

	# M vertices	# M edges
Un-factored graph	0.300	306
Cartesian Product	0.310	5.5
Basis Set	0.320	1.8

As can be seen from the figures in Table 2, arc factorization dramatically reduces the graph size, trimming it down by more that two orders of magnitude. The basis set decomposition significantly outperforms the Cartesian product heuristic, reflecting the fact that the multi-pass strategy it implements allows it to reach a much better covering. The number of arcs in the factored bipartite graph has 2 orders of magnitude fewer arcs than the unfactored graph, with only a marginal increase of the number of states.

The primary goal of our approach is to reduce the size of decoding graphs to the point where it becomes feasible to employ an entire word of cross-word acoustic context, and our factoring procedures have allowed us achieve this goal. Using the basis set algorithm, we were able to build left-context graphs which were only twice the size of a graph with purely word-internal contexts, while providing us with a substantial reduction in error rate.

Tables 3 and 4 summarize our main results, giving graph sizes, run-times, and word error rates for increasingly complex language models. Since it is straightforward to build decoding graphs that use only within-word context, these are the baseline for improvement. Runtimes cited are inclusive of the Gaussian computation and were obtained using a relatively narrow beam parameter: at each time frame, only the best 5,000 state hypotheses are kept. All the experiments used a 4GHz pentium IV PC with 512 Megabytes of memory.

Table 3
Graph sizes after minimization and determinization. For left-word acoustic context, these operations cannot complete without our graph factorization method.

	2-gram		3-gram	
	# M states	# M arcs	# M states	# M arcs
Word Internal Baseline	0.822	2.009	1.85	4.26
Unfactored Left Context	-	-	-	-
Factored Left Context	0.934	2.464	2.03	4.95

It is remarkable that the graphs obtained using cross-word left-context de-

pendencies are only marginally bigger that the corresponding baseline word internal graphs, demonstrating the effectiveness of our arc minimization strategy. We have observed that this is the case for fourgram as well as trigram language models, and are able to produce fourgram graphs (Zweig et al., 2002).

Table 4
Word Error and run times. Without graph factorization, it is not possible to construct the left-context graph and decode.

	2-gram		3-gram	
	WER	xRT	WER	xRT
Word Internal Baseline	26.5	0.9	24.7	1
Unfactored Left Context	-	-	-	-
Factored Left Context	24.5	1.1	22.6	1.1

The results in Table 4 were obtained using a beam large enough to ensure that there were virtually no search errors; identical word error rates were achieved using 100,000 active states. In fact, we have observed that the word error rate remains relatively constant as long as the number of live states exceeds the number of context-dependent units in the acoustic model.

In a novel application of static decoding graph technology, we have used our graphs to perform MMI training without the need for creating lattices. The basic procedure we follow is that of (Woodland and Povey, 2000), except that instead of constructing separate denominator lattices for each training utterance, we simply use the entire unigram decoding graph. This dispenses with the extra step of creating lattices, and removes the approximation present in using them to represent all possible word sequences. As the runtimes presented here indicate, we can achieve real-time performance thus allowing for large-scale training. In the context of MMI training, it is interesting to note that it is in fact unnecessary to store word labels in the decoding graph, as the training procedure merely accumulates posterior occupancy counts for HMM states. Removing word labels does not change the probability distribution over state sequences (all that matters for MMI), but it does make the determinization / minimization of the decoding graph much more effective, speeding the training procedure by a significant margin. For the unigram case, for instance, we observed a reduction in size of the word internal graph by factor of 2; for the cross-word graph the reduction was close to 30% (see Table 5).

4.3 Results in the 2003 DARPA EARS Evaluation

In a further application of this technology, factored left-context decoding graphs were constructed and used in the 2003 DARPA EARS competition (Extensible, Affordable, Reusable Speech-to-Text). This resulted in the winning

Table 5
Unigram graphs for MMI training

	Word internal		Left context	
	# states	# edges	# states	# edges
Unigram with word labels	170 K	358 K	227 K	700 K
Unigram without word labels	74.5 K	162 K	142 K	524 K

Table 6
Number of n-grams, states and arcs for the speaker independent and speaker adapted decoding graphs. Runtime is on the bottom line.

Number of	SI	SA
ngrams	0.2M	3.3M
states	0.6M	9.6M
arcs	1.7M	23.9M
Runtime	0.11xRT	0.63xRT

entry in the sub-realtime category, with a word error rate of 29.0%, compared with the next closest competitor at 34.4% (Le, 2003). For this task, a two-pass decoding strategy was used, the first pass using a smaller decoding graph and unadapted acoustic model, and the second pass using speaker-adapted models (Saon et al., 2003). The main difference lies in the choice of the language model: for the speaker independent decoding we opted for a bigram LM whereas the final decoding step uses a 4-gram LM. The latter was trained on the following corpora: 3M words of Switchboard, 58M words from web scripts publicly available from the University of Washington, 3M Broadcast news words relevant to Switchboard topics and 7M words of the English Gigaword corpus. The acoustic-context model for the speaker adapted system was the same as that discussed in the preceding sections. The speaker-independent system was similar but trained on unadapted feature vectors. The language model and graph sizes are presented in Table 6, along with the runtime on a 3GHz pentium.

The number of n-grams used in the speaker-adapted decoding graph is an order-of-mangnitude greater than than the graphs for which results are presented in the previous section. Further, the n-gram context is four rather than three. Thus, these results indicate that our method scales well to larger systems, and that the method produces graphs that outperform other methods in the literature.

5 Conclusion and perspectives

In this paper, we have proposed a new methodology for building static decoding graphs for LVCSR systems which can accommodate acoustic models exhibiting patterns of long distance contextual dependencies (up to one word on the left). We have shown that the same minimization problem occurs for both left and right cross-word dependencies, and that this problem is in fact NP-hard. By developing heuristics for decomposing the connectivity in the unigram portion of the graph, we are able now to generate static decoding graphs for state-of-the-art Switchboard systems. The resulting graphs are compact enough to allow real-time decoding, and MMI training where the entire language model is used as the "denominator lattice."

While the paper has been concerned with the application of arc minimization in a very specific case - where the acoustic realizations of words are sensitive to the identity of the preceding words - we believe that there are a number of related problems that make the paradigm of general interest, for example:

- Language model factorization. The arcs that encode the bigram portion of a language model can be thought of as a bipartite graph going from words to words. Minimizing the size of this graph can be seen as a natural extension of this work to the case of weighted graphs.
- Introducing non-determinism in a more general way. Our work has focused on introducing non-determinism in the unigram portion of a decoding graph. Our results prove that this additional non-determinism does not seem to hurt the experimental performance. This suggests that trading off a small amount of non-determinism for a large reduction in the number of arcs - anywhere in a graph - might be worth considering.
- Decoding without pruning. The runtime of a Viterbi decoding without pruning is proportional to AT, where A is the number of arcs and T the number of time frames. Using the technique of (Zweig and Padmanabhan, 2000) for reducing the space complexity to $O(\log T)$, this is easily attainable in practice and guaranteed to eliminate all search errors. In this case, arc minimization is the most relevant optimization criterion.

Acknowledgements

The authors thank Brian Kingsbury, Lidia Mangu, and Stanley Chen for useful comments, insights, and portions of the language and acoustic models. The authors further wish to thank the two anonymous reviewers for helping to sharpen the presentation.

References

Aubert, X., 2000. A brief overview of decoding techniques for large vocabulary continuous speech recognition. In: Proceedings of the ISCA Tutorial and Research Workshop, Automatic Speech Recognition: Challenges for the new Millenium (ASR2000). Paris, France, September 2000. Paris, France, pp. 91–96.

Chen, S. F., 2003. Compiling large context phonetic decision trees into finite-state transducers. In: Proceedings of Eurospeech/Interspeech 2003. Vol. to appear. Geneva, Switzerland.

Chen, S. F., Goodman, J., 1996. An empirical study of smoothing techniques for language modeling. In: Proceedings of the 34th Annual Meeting of the Association for Computational Linguistics. Santa Cruz, pp. 310–318.

Dolfing, H. J., Hetherington, I. L., 2001. Incremental language models for speech recognition using finite-state transducers. In: Proceeding of the IEEE workshop on Automatic Speech Recognition and Understanding (ASRU'01). Madonna di Campiglio Trento, Italy, pp. 58–62.

Feder, T., Motwani, R., 1991. Clique partitions, graph compression and speeding-up algorithms. In: Proceedings of the 23rd Annual ACM Symposium on Theory of Computing, New Orleans, Louisiana, 1991. pp. 123–133.

Holyer, I., 1981. The NP-completeness of some edge partition problems. Siam Journal of Computer Science 10 (4), 713–717.

Hopcroft, J. E., Ullman, J. D., 1979. Introduction to automata theory, languages and computation. Addison-Wesley.

Jelinek, F., Bahl, L. R., Mercer, R. L., 1975. Design of a linguistic statistical decoder for the recognition of continuous speech. IEEE Transactions on Information Theory 21, 250–256.

Jiang, T., Ravikumar, B., 1993. Minimal nfa problems are hard. SIAM Journal on Computing 22 (6), 1117–1141.

Kanthak, S., Ney, H., Riley, M., Mohri, M., 2002. A comparison of two LVR search optimization techniques. In: Proceedings of the International Conference on Spoken Langage Processing (ICSLP). Denver, CO, pp. 1309–1312.

Katz, S. M., 1987. Estimation of probabilities from sparse data for the language model component of a speech recognizer. IEEE Transactions on Acoustics, Speech, and Signal Processing 35 (3), 400–401.

Le, A., 2003. Rich transcription 2003: Spring stt evaluation results. Presentation at the RT 2003 Spring Workshop, http://www.nist.gov/speech/tests/rt/rt03/spring/.

Mohri, M., 1994. Minimization of sequential transducers. Lecture Notes in Computer Science 807, 151–163.

Mohri, M., Riley, M., 1998. Network optimisation for large vocabulary speech recognition. Speech Communication 25 (3), 1–12.

Mohri, M., Riley, M., Hindle, D., Ljolje, A., Pereira, F., 1998. Full expansion of context-dependent networks in large vocabulary speech recognition. In: Proceedings of the International Conference on Acoustics, Speech and Signal

Processing (ICASSP). Vol. 2. Seattle, pp. 665–668.
Mohri, M., Riley, M., Pereira, F. C. N., 2000. Weighted finite-state transducers in speech recognition. In: Proceedings of the ISCA Tutorial and Research Workshop, Automatic Speech Recognition: Challenges for the new Millenium (ASR2000). Paris, France, September 2000. Paris, France, pp. 97–106.
Nadàs, A., Nahamoo, D., Picheny, M., 1988. On model-robust training algorithm for speech recognition. IEEE Transaction on Audio, Signal and Speech Processing 36, 1432–1435.
Ney, H., Ortmanns, S., 1999. Dynamic programming search for continuous speech recognition. IEEE Signal Processing Magazine , 64–83.
Odell, J., 1995. The use of context in large vocabulary speech recognition. Ph.D. thesis, University of Cambridge.
Ortmanns, S., Eiden, A., Ney, H., Coenen, N., 1997. Look-ahead techniques for fast beam search. In: Proceedings of the International Conference on Acoustics, Speech and Signal Processing (ICASSP). pp. 1783–1786.
Placeway, P., Schwartz, R., Fung, P., Nguyen, L., 1993. The estimation of powerful language models from small and large corpora. In: Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). Minneapolis, MN, pp. 33–36.
Riccardi, G., Pierraccini, R., Bocchieri, E., 1996. Stochastic automata for language modeling. Computer, Speech and Language 10 (265–293).
Saon, G., Padmanabhan, M., Gopinath, R., Chen, S., 2000. Maximum likelihood discriminative feature spaces. In: Proceedings of the International Conference on Acoustics, Speech and Signal Processing (ICASSP). Istanbul, pp. 1747–1751.
Saon, G., Zweig, G., Kingsbury, B., Mangu, L., Chaudhari, U., 2003. An architecture for rapid decoding of large vocabulary conversational speech. In: Proceedings of Eurospeech/Interspeech 2003. Geneva.
Seymore, K., Rosenfeld, R., 1996. Scalable trigram backoff language models. In: Proceedings of the International Conference on Spoken Langage Processing (ICSLP). Philadelphia, Pennsylvania, pp. 232–235.
Smaili, N., Cardinal, P., Boulianne, G., Dumouchel, P., 2002. Disambiguation of finite-state transducers. In: Proceedings of COLING'02. Taipeh.
Viterbi, A., 1967. Error bounds for convolutional codes and an asymptotically optimal decoding algorithm. IEEE Transactions on Information Theory 13, 260–269.
Willett, D., Katagari, S., 2002. Recent advances in efficient decoding combining on-line transducer composition and smoothed language model incorporation. In: Proceedings of the International Conference on Acoustics, Speech and Signal Processing (ICASSP). Vol. I. Orlando, FL, pp. 713–716.
Willett, D., McDermott, E., Minami, Y., Katagari, S., 2001. Time and memory efficient viterbi decoding for LVCSR using a precompiled search network. In: Proceedings of the European Conference on Speech Communication and Technology. Aalborg, DK, pp. 847–851.

Woodland, P., Povey, D., 2000. Large scale discriminative training for speech recognition. In: Proceedings of the ISCA Tutorial and Research Workshop, Automatic Speech Recognition: Challenges for the new Millenium (ASR2000). Paris, France, September 2000. Paris, France, pp. 7–16.

Zheng, J., Franco, H., 2002. Fast hierarchical grammar optimization algorithm toward time and space efficiency. In: Proceedings of the International Conference on Spoken Langage Processing (ICSLP). Vol. 1. Denver, CO, pp. 393–396.

Zweig, G., Padmanabhan, M., 2000. Exact alpha-beta computation in logarithmic space with application to map graph construction. In: Proceedings of the International Conference on Spoken Langage Processing (ICSLP). Beijing, China.

Zweig, G., Yvon, F., Saon, G., 2002. Arc minimization in finite state decoding graphs with cross-word acoustic context. In: Proceedings of the International Conference on Spoken Langage Processing (ICSLP). Vol. 1. Denver, CO, pp. 389–392.