ARC MINIMIZATION IN FINITE STATE DECODING GRAPHS WITH CROSS-WORD ACOUSTIC CONTEXT

G. Zweig, G. Saon

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

F. Yvon[*]

ENST and CNRS URA 820
46 rue Barrault, F-75013 Paris

ABSTRACT

Recent approaches to large vocabulary decoding with finite state graphs have focused on the use of state minimization algorithms to produce relatively compact graphs. This paper extends the finite state approach by developing complementary arc-minimization techniques. The use of these techniques in concert with state minimization 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, and describe the reduction from 3-SAT. We present experimental results that illustrate the moderate sizes and runtimes of graphs for the Switchboard task.

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 subsequent decoding [1]. For large vocabulary tasks with n-gram language models, however, it has traditionally been common to avoid a static search space, and to instead dynamically expand the language model as needed [2, 3, 4]. 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 [5, 6], 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 state-minimization of the resulting weighted FSM.

While this previous work [5, 6] has established Viterbi decoding on statically compiled graphs to be an effective method for large vocabulary decoding, it has focused on the minimization of states alone, and can therefore result in graphs that are sub-optimal from the point of view of storage and runtime - both of which are proportional to the number of edges in a graph. In particular, when one attempts to move beyond triphone cross-word acoustic context, the number of edges can become quite large.

In this paper, we significantly enhance the technology of static graph compilation by presenting methods for not only minimizing the number of states in the decoding graph, but the number of arcs as well. The motivation for this is the use of cross-word acoustic context that extends an entire word to the left or right of the current word. In this case, there are potentially as many acoustic variants for a particular word as there are words in the vocabulary. In practice, the number of variants is much smaller, but even here a straightforward implementation results in a prohibitively large graph. More critically, even the application of state-based minimization techniques leaves a tremendous number of unnecessary arcs in the graphs.

Since there are well-known algorithms for state minimization, it is reasonable to ask if similar procedures exist for arc minimization. However, through a reduction from the known NP-complete optimization problem of Clique Bipartitioning [7], we demonstrate that in fact the problem we are faced with is NP-complete. In this respect it is similar to state minimization - which in the worst case requires exponential time in the determinization step - but it differs in the sense that we do not know of previous algorithms that work well in average cases.

The remainder of this paper is organized as follows. In Section 2.1, we present the basic structure of the decoding graphs and proceed in Section 2.2 to illustrate the problem of arc minimization in the simplest case of a unigram language model with left-context acoustic models. In Section 2.3, we cast the problem formally, show that it is NP-complete, and present two simple heuristics for generating compact graphs. In Section 3 we apply these techniques, and present results on graph size, runtime, and word-error rate with bigram and trigram language models on Switchboard.

2. MAKING LARGE GRAPHS SMALL

2.1. Word-Internal Graphs

We begin our discussion of graph structures by considering the basic graph structure induced by a n-gram language model with backoff smoothing. At first, we do not consider any cross-word acoustic context. When LM probabilities are smoothed according to a back-off scheme [8], this automaton can be efficiently factored [9] as follows. Each history h appearing in the LM corresponds to a *history state* N_h in the FSM; 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'

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

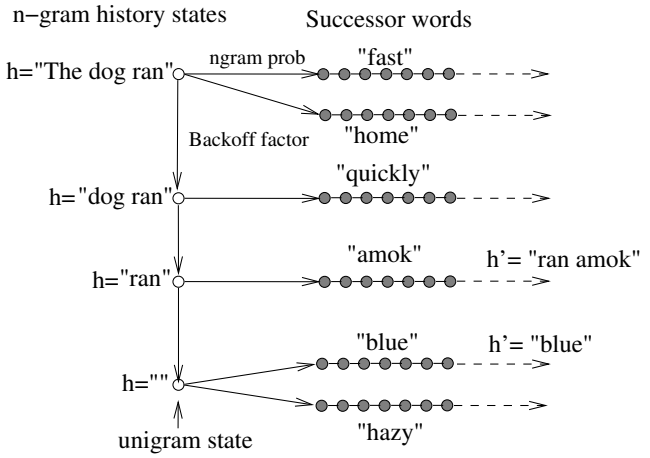

Fig. 1. A graph with word internal context only. Arcs emanating from the right-hand side loop back to states on the left.

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 spontaneous 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 acoustic states induced by their pronunciation yields the final decoding graph (see on on Figure 1). Such graphs can be further processed with general algorithms for weighted FSMs, such as the removal of epsilon transitions, determinization, and minimization [6]. Our own graph construction algorithm simply factors out in a tree identical sequences of emitting states which occur in variants of the same word .

2.2. Left-Context Graphs

We now consider the case when the acoustic realization of a given word is triggered by 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. 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. In general, 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 on Figure 3. This second example uses a four words 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

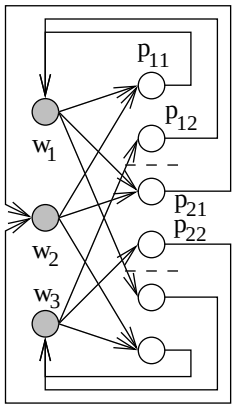

Fig. 2. State-minimal graph representing left-word acoustic context constraints. There are three vocabulary words, each with two context-sensitive variants.

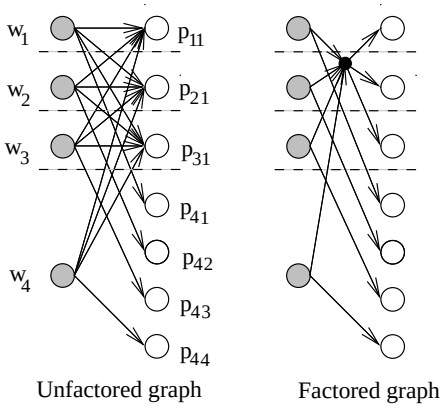

Fig. 3. Factoring left contexts. The backwards loops from variants to words have been omitted for clarity. The biclique amongst $w_1, w_2, w_3, w_4, p_{11}, p_{21}, p_{31}$ is factored out on the right.

variants are licensed by which words, in a brute-force fashion. It uses 16 arcs, and is minimal with respect to state minimization. The graph on the right models the same dependencies by introducing an extra state, 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 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.

2.3. Minimizing Left-Context Graphs

2.3.1. 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 $E_1 \dots E_k$ 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.

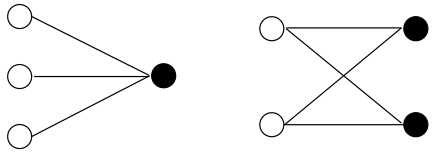

Fig. 4. Simple bicliques: 1×3 star and $K_{2,2}$

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), $R = P$ (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. 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 states so as to minimize the number of arcs in the final graph. This amounts to finding a set of bicliques $B_i = (V_i, P_i)$ in G such that:

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

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

or, in other words, to finding the minimal order cover of edges in G into bicliques. This is because each biclique B_i incurs an extra state, for which there will be one incoming edge from each state in L_i, and one outgoing edge for each state in R_i (see Figure 3).

2.3.2. NP Completeness

In this section, we show that this minimization problem is, in fact, NP-hard. The proof derives directly from a result of [7], which proves than finding the minimum order *partition* into bicliques for any given graph is NP-hard. This proof relies on a variation of a reduction of 3-SAT originally proposed in [10], and proceeds along the following lines. They first exhibit a polynomial transformation of any formula F into a graph $G(F)$ 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×1 bicliques) (see Figure 4).

[7] 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 stars have indeed more vertices than arcs. Since the total number of edges in the partition is fixed, [7] 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 an NP-hard problem.

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:
     - $\forall j, k, H'_{jk} \leftarrow H_j \cap Left(p_i^k)$
     - $\forall j, k, P'_{jk} \leftarrow P_j \cup p_i^k$
   - $\mathcal{H} \leftarrow \mathcal{H}'$
   - $\mathcal{P} \leftarrow \mathcal{P}'$
   - $i \leftarrow i + 1$
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 2.

Fig. 5. Cartesian intersection algorithm for computing history sets and licensed acoustic variants.

2.3.3. Two Algorithms for Arc Minimization

In this section, we present two heuristics that have proven to perform well. Both methods begin by identifying the acoustic variants $p_i^1 \ldots p_i^{l_i}$ of each word w_i. This can be done in a brute-force fashion by enumerating all possible predecessor words of w_i and using a decision tree to identify the corresponding sequences 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 so we have designed our algorithms to 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 5. 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. 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) has been generated, it is better to store them, and "reset" the algorithm (Figure 5).

Our second heuristic 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. Frequently occurring, medium sized context-sets have empirically proven to provide an advantageous basis for decomposing the remaining lot of context sets. The heuristic of Figure 6 exploits this observation.

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.
     - $\bigcup_j H_{j_k} = Left(p_i^k)$
     - $\Sigma_j \,|\, H_{j_k} \,|$ 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 $|\, P_i \,| \ < \theta_2$
   $\mathcal{B} \leftarrow \mathcal{B} \setminus (H_i, P_i)$
5. if $\mathcal{B}$ was changed during step 4 goto 3
6. output $\mathcal{B}$

Fig. 6. Basis-set decomposition algorithm for computing history sets and licensed acoustic variants.

3. EXPERIMENTS

3.1. System description

For these experiments, we have used a Switchboard system based on a 18K vocabulary, with about 300K left context variants. Speech features are derived from 24-dimensional MFCCs, further transformed into a canonical space through the application of VTLN and FMLLR, and then projected onto a discriminative 60-dimensional space with HDA. Acoustic modeling uses cross-word context-dependent HMMs: the context dependent states of any given phone are determined through the application of a decision-tree making its decision based on a 11-phone window. The acoustic model set includes 150K Gaussian models and about 3.7 K HMM states, estimation was perfored using MMI training. Acoustic and language models (bigram and trigram smoothed with Knesser-Ney Backoff) were trained on a combination of Switchboard and Callhome data.

3.2. Results

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. In Table 1, we show that this is indeed possible, with the factored left-context graphs being only about twice the size of a graph with purely word-internal contexts, while providing us with a consistent WER reduction. As a direct result of the compact size of our graphs, the decoding is relatively fast, as illustrated in Table 2, along with corresponding word-error rates obtained on Switchboard'00 evaluation set (switchboard part). Runtimes cited are exclusive of the Gaussian computation and were obtained on a 1.5 Ghz pentium IV PC.

4. 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). In spite of the NP-hardness of the arc minimization problem, we devised heuristic approaches which provided us with reasonably small graphs.

	2-gram		3-gram	
	# K states	# K arcs	# K states	# K arcs
Word Internal	376	1,654	877	2,311
Basis Set	713	4,152	1,503	5,288
Cartesian	1,125	11,570	2,545	15,967

Table 1. Graph sizes

	2-gram		3-gram	
	WER	xRT	WER	xRT
Word Internal	27.1	4.2	26.0	10.2
Left Context (BSD)	26.3	6.9	25.3	14.3

Table 2. Word Error and run times

Using this graph compilation technique, we were able to achieve state-of-the-art recognition on a large vocabulary task. Although not reported in this paper, the case were pronunciation variants vary according to the word on the right can be handled analogously. The extension to bi-directional (left and right) constraints is underway.

5. REFERENCES

[1] A.J. Viterbi, "Error bounds for convolutional codes and an asymptotically optimal decoding algorithm," *IEEE Transactions on Information Theory*, vol. 13, pp. 260–269, 1967.

[2] Frederik Jelinek, Lalit R. Bahl, and Robert L. Mercer, "Design of a linguistic statistical decoder for the recognition of continuous speech," *IEEE Trans. Inf. Thy.*, vol. 21, pp. 250–256, 1975.

[3] Julian Odell, *The use of context in large vocabulary speech recognition*, Ph.D. thesis, University of Cambridge, 1995.

[4] Hermann Ney and Stefan Ortmanns, "Dynamic programming search for continuous speech recognition," *IEEE Signal Processing Magazine*, pp. 64–83, 1999.

[5] Mehryar Mohri, Michael Riley, Don Hindle, Andrej Ljolje, and Fernando Pereira, "Full expansion of context-dependent networks in large vocabulary speech recognition," in *Proceedings of ICASSP '98*, Seattle, 1998.

[6] Mehryar Mohri, Michael Riley, and Fernando C. N. Pereira, "Weighted finite-state transducers in speech recognition," in *Proceedings of ASR2000*, Paris, France, 2000.

[7] Tomás Feder and Rajeev Motwani, "Clique partitions, graph compression and speeding-up algorithms," in *Proceedings of the 23rd Annual ACM Symposium on Theory of Computing, New Orleans, Louisiana, 1991*, 1991, pp. 123–133.

[8] Slava M. Katz, "Estimation of probabilities from sparse data for the language model component of a speech recognizer," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 35, no. 3, pp. 400–401, 1987.

[9] Guiseppe Riccardi, Roberto Pierraccini, and Enrico Bocchieri, "Stochastic automata for language modeling," *Computer, Speech and Language*, vol. 10, no. 265–293, 1996.

[10] Ian Holyer, "The NP-completeness of some edge partition problems," *Siam Journal of Computer Science*, vol. 10, no. 4, pp. 713–717, 1981.