Genetic Triangulation of Graphical Models for Speech and Language Processing

Chris Bartels, Kevin Duh, Jeff Bilmes, Katrin Kirchhoff

Department of Electrical Engineering
University of Washington, U.S.A.
{bartels,duh,bilmes,katrin}@ee.washington.edu

Simon King

Centre for Speech Technology Research
University of Edinburgh, UK
Simon.King@ed.ac.uk

Abstract

Graphical models are an increasingly popular approach for speech and language processing. As researchers design ever more complex models it becomes crucial to find triangulations that make inference problems tractable. This paper presents a genetic algorithm for triangulation search that is well-suited for speech and language graphical models. It is unique in two ways: First, it can find triangulations appropriate for graphs with a mix of stochastic and deterministic dependencies. Second, the search is guided by optimizing the inference speed (CPU runtime) on real data. We show results on 10 real-world speech and language graphs and demonstrate inference speed-ups over standard triangulation methods.

1. Introduction

The predominant statistical model used in automatic speech recognition (ASR) is the Hidden Markov Model (HMM). Although the HMM is quite powerful, it is only one of an infinite number of possible statistical models. Graphical models (GMs) provide a visual abstraction that can represent an enormous family of probability models (including the HMM). With GMs, it is possible to rapidly explore many diverse models for ASR, given an available and computationally efficient software system. There is growing interest in the use of GMs for a variety of speech and language tasks. Examples include multi-stream models for ASR [1], audio-visual speech recognition [2], articulatory modeling [3, 4, 5], pitch tracking [6], and edit distance learning [7].

All exact statistical inference procedures on a graphical model, such as EM training and Viterbi decoding, use a triangulated graph (see Section 2) either explicitly or implicitly [8]. The difference between a good and bad triangulation has a dramatic impact on the computational requirements of inference tasks. Finding the optimal triangulation for a particular graph, unfortunately, is NP-hard [9, 10]. Although many heuristic methods have been proposed, we will show that standard triangulation methods are inadequate for producing inference schemes for many graphs seen in speech and language processing.

This paper addresses this issue by examining some common characteristics of speech/language graphical models (Section 3). We show that standard triangulation methods do not perform well with the large number of deterministic dependencies commonly found in speech/language graphs. Further, we argue that standard heuristics for judging triangulation quality is inferior to directly estimating the inference speed (CPU time) of the triangulation on real speech/language data. We propose two triangulation searches to address these issues. The first generates a large number of triangulations using the heuristics proposed in [10] and ranks them using inference speed. The second triangulation method is based on genetic algorithms. In the following, we first review the basics of triangulation (Section 2) and present the issues specific to triangulating speech/language graphs (Section 3). Section 4 presents a genetic algorithm, and Section 5 reports inference speed-up results on various real-world speech/language graphs.

This work was supported by NSF grant IIS-0093430, NSF/CIA grant IIS-0326276 and an Intel Corporation Grant

2. Background

Graphical models [11] use a general probabilistic inference scheme that requires a graph to be triangulated. A graph is **triangulated** if there are no "chordless cycles," where a **chord** is an edge connecting two non-consecutive vertices in a cycle of length > 3. For instance, the graph in Fig. 1(a) is not triangulated. To triangulate it, we must add one or more edges, as done in Fig, 1(b), 1(c), and 1(d). The resulting graphs have no chordless cycles and are referred to as triangulated graphs or **triangulations**. The edges that are added to create a triangulation are called **fill-in edges**.

Adding fill-in edges creates **cliques**. A clique is a set of variables that are completely connected to each other. The **state space** of a clique is the product of its variables' state space, which is, in turn, the number of values the variable can take on. The state space of a triangulated graph is the sum of the state space of its "maximal cliques," and the magnitude of this correlates with inference speed.

The design of a GM for any task involves three steps: First, the researcher characterizes the speech/language process he wishes to model by specifying a set of hidden/observed variables and stochastic/deterministic dependencies. Then, this GM is triangulated so that inference tasks such as EM training and Viterbi decoding can be performed. Finally, parameter estimation and other statistical inference tasks are run on the distribution defined by the graph. Triangulation is a crucial step in GM design because it transforms any user-specified GM, which can be arbitrary, into a data structure that is efficient for inference tasks. The triangulation quality therefore has a direct effect on the inference speed of practical GM-based systems.

Given any arbitrary graph, the goal of a triangulation algorithm is to find a triangulation that results in tractable and fast inference. A common technique for this is **vertex elimination**. It proceeds by removing each vertex in the graph in some order and adding fill-in edges that completely connect the removed vertex's neighboring vertices. For a graph of N nodes, there

are N! possible elimination orders; different elimination orders may add different sets of fill-in edges, thereby resulting in different triangulations. Choosing the optimal elimination order is also NP Hard, and various heuristics and stochastic searches [12, 13] have been attempted. It can be shown that vertex elimination always results in a triangulated graph [14]. However, not all triangulations can be created by elimination [10]. An example is Fig. 1(d). This deficiency has a nontrivial effect on the triangulations of speech and language graphs.

3. Triangulation in Speech Graphs

Many graphical models used in speech and language systems use a mixture of stochastic and deterministic variables, where a deterministic variable is a variable in which its value is determined by a function of its parents. These deterministic dependencies are used either to model deterministic relationships in the underlying speech/language process, or to provide ways to implement constraints or parameter tying.

A graph for ASR is shown in Fig. 2. For each frame, five out of ten variables are deterministic. The important thing to learn is that many practical speech/language GMs, such as the ones used in the Experiments section, are large and complex and involve many deterministic variables. The large number of deterministic dependencies creates a problem for standard elimination algorithms. When many variables are deterministic functions of other variables, combinations of certain variable values will have zero probability; this implies that grouping these variables in a clique can result in a significant reduction of the state space. However, standard vertex elimination often cannot achieve the grouping needed for this reduction of state space. When deterministic variables are present, examples exist where the state-space optimal triangulation has an arbitrarily smaller state space than the best elimination based triangulations [10].

Another problem with many convention approaches is that they use heuristics to differentiate between triangulations. State space can be a good heuristic when all of the variable combinations in the graph have non-zero probability, but if many state combinations have zero probability it becomes only a rough upper bound of the amount of computation actually needed. Speech graphs have many zeros for two reasons. The first reason is due to the prevalence of deterministic variables. The second issue is that beam pruning, which is often used in speech/language tasks, adds additional zeros to the distribution. Similarly, triangulation methods which are guided by heuristics, such as choosing nodes in an elimination order, are not always able to find triangulations with fast inference time. These reasons make it difficult to predict the "true" state space of a graph without actually running inference. In the following, we present two triangulation algorithms that address these issues.

4. Timing Based Triangulation Searches

4.1. Multiple Heuristics

Multiple Heuristics Triangulation (MH) is a search technique that generates a large number of triangulations using the heuristics given in [10] and evaluates them based on inference speed. The first step is to run a "boundary search", which is often relevant for speech/language graphs involving variable-length sequences: To model variable-length utterances, the user specifies a template that characterizes the dependencies between variables within a frame. Then, given an input utterance of any length, this template is replicated to fit the whole utterance. The template specified by the user, though intuitive during modeling, may not be optimal for triangulation; thus the templates are re-segmented with new boundaries, and different triangulation searches are run on the results of each boundary search. [15] For each boundary, 205 triangulations are generated and timed. Then, a larger set of 4265 greedy heuristics are tried on the top four boundaries.

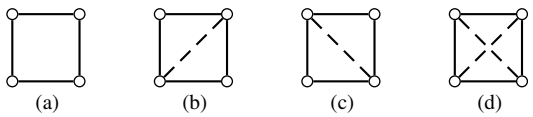


Figure 1: Graph (a) is untriangulated, graphs (b), (c), (d) are triangulated versions of (a). Triangulation in (d) can not be created using elimination on (a)

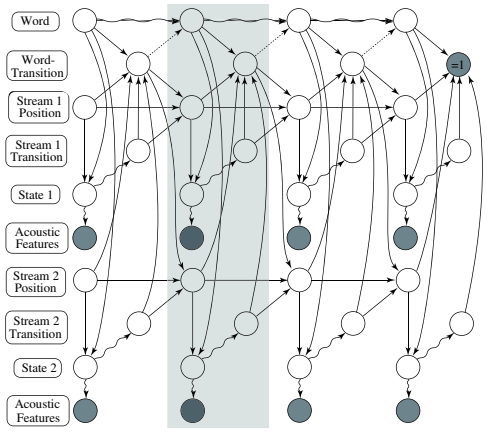


Figure 2: Aynchronous multi-stream graphical model for ASR. Shaded/unshaded nodes are observed/hidden variables; wavy/nonwavy arrows are probabilistic/deterministic dependency. Shaded area indicates one frame.

4.2. Genetic Algorithms for Triangulation

Genetic Algorithms (GAs) [16] are a class of evolution-inspired search/optimization techniques that perform particularly well in problems with complex, poorly understood search spaces. The fundamental idea is to encode problem solutions as genes, and to evolve successive populations of solutions through the use of genetic operators (selection, crossover, and mutation). Potential solutions are evaluated according to a *task-specific* fitness function which represents the desired optimization criterion. The individual steps are as follows:

Initialize: Randomly generate a population of genes.
While fitness improves by a certain threshold:
Evaluate fitness: calculate each gene's fitness
Apply operators: apply the selection, crossover, and mutation operators to create a new population.

The selection operator probabilistically chooses genes for the next generation so that fitter genes have higher chances of survival. Crossover combines two genes to create new genes, in effect exploring new regions of the search space. Mutation creates genetic diversity by randomly altering parts of an existing gene. GAs provide no guarantee of finding the optimal solution, but often finds good solutions quickly and are relatively robust against premature convergence to local optima.

We chose to apply GAs to triangulation since GAs can ad-

	Variables		Sec/100frame		Utterance Time (sec)			Speedup	
	total	det	MH	GA	Base	MH	GA	ΔMH	ΔGA
FeatureDetect	9	6	8.94	6.77	*fail*	21.8	16.2	∞	∞
MultiStream	14	9	2.92	2.03	63.9	2.42	1.68	26.4	38.0
CTS Decode	9	6	16.39	11.7	24.7	8.80	5.20	2.80	4.75
PhoneFree 1	50	35	1.49	1.56	0.34	0.35	0.34	0.97	1.00
PhoneFree 2	40	28	0.909	0.896	0.24	0.22	0.22	1.09	1.09
Mandarin	10	6	7.39	7.89	41.92	41.14	45.43	1.02	0.923
Edit D. 1 training	10	9	0.847	0.831	11.84	9.19	9.37	1.29	1.26
Edit D. 1 decoding	11	10	0.226	0.222	2.62	2.58	2.67	1.02	0.98
Edit D. 2 training	10	9	0.288	0.286	2.16	1.87	1.89	1.16	1.14
Aurora Decode	6	4	0.163	0.163	27.28	27.14	27.18	1.00	1.01

Table 1: Experimental results. 1st column shows total number of variables per frame and number of deterministic variables. Base:Baseline, MH:multiple heuristics, GA:genetic algorithm, all times in seconds. The speedups, ΔMH, ΔGA are "Utterance Time" ratios Base/MH, Base/GA. *fail* indicates that all Baseline triangulations were not decodable within the available memory

dress the two problems common to speech/language graphs in a principled manner. First, to allow for search on all possible triangulations, each gene in the GA is defined to represent some triangulation. All potential fill-in edges are encoded, so the space of *all* possible triangulations is available to the GA. This stands in contrast to vertex elimination, which cannot search a certain subspace of triangulations. Since the optimal triangulation for graphs with deterministic dependencies often exist in this subspace, GAs have the potential to find better triangulations of real-world graphs.

Second, we define the GA fitness function as the triangulations' inference speed on real data. Specifically, the fitness is calculated by running inference on a triangulation and counting the number of frames processed in a fixed number of seconds. (Alternatively, we can count sec/frame, as done in the Results section.) If each evaluation is allowed to run for a large number of seconds, the fitness will more closely approximate the true time required to perform inference on a large dataset. There is a tradeoff, however, as setting evaluation time small enables more triangulations to be evaluated and searched. The advantage of fitness functions is that it avoids the use of heuristics to evaluate a triangulation's quality, and allows for direct optimization of the quantity desired in real-world situation. It should be noted that fitness function can also be modified to account for memory usage and other practical considerations, such that multiple ways to judge triangulations can be jointly evaluated.

The selection operators used here are standard GA operators such as roulette wheel selection or tournament selection. Crossover of two parent triangulations is done as follows: First, each parent graph replicates itself to produce a child graph. Then, edges between the children graphs are swapped with some probability, in effect creating triangulations that are combinations of both parents.

We extend this crossover approach to allow for "boundary search", as was done in MH. We allow triangulations based on different boundaries but restrict crossover to genes that come from the same boundary. This is the idea of selective crossover, and has the effect of creating several "species" of genes. This technique improves robustness against local optimum and enlarges the search space. All of the boundaries are placed in the initial population, but one boundary eventually dominates.

In mutation, a random edge from the set of all possible fill-in edges is chosen. If the chosen edge exists in the gene, then it is deleted; otherwise, it is added. However, if deletion/addition causes an untriangulated graph, the change is undone and another random edge is chosen. We show that the space of all possible triangulations can be traversed using this single edge mutation:

Lemma 1. *Let $G = (V, E)$ be a triangulated graph and let $G' = (V, E')$ be a spanning triangulated subgraph of G with $|E \setminus E'| = k$. Then there is an increasing sequence $G' = G_0 \subset ... \subset G_k = G$ of triangulated graphs that differ by exactly one edge. [17, Lemma 2.21, page 20]*

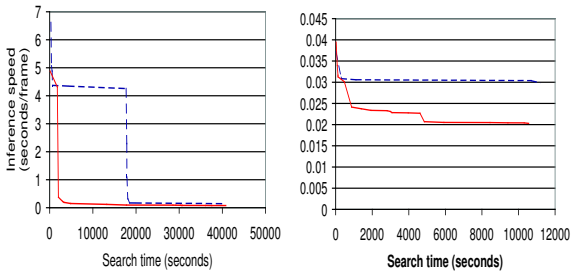
Theorem 2. *Given any two triangulations of a graph, $T_a(G)$ and $T_b(G)$ there is a sequence of triangulated graphs $T_a(G), T_1(G), T_2(G), ..., T_b(G)$ with only a single edge difference between subsequent graphs in the sequence.*

Proof. Suppose both $T_a(G)$ and $T_b(G)$ are spanning subgraphs of some $T_s(G)$. From lemma 1 one can create a series of graphs from $T_a(G)$ to $T_s(G)$, and likewise a there exists a series from $T_b(G)$ to $T_s(G)$ which can be followed in reverse to get from $T_s(G)$ to $T_g(G)$. One can always choose $T_s(G)$ to be the complete graph. □

This is an important property, as it shows that the GA is able to search *all* possible triangulations from any initial condition. Finally, we compare our work to previous work in genetic triangulation [13] and note significant differences. First, [13] searches for elimination orders, whereas our method searches all possible triangulations. Second, we evaluate genes using inference speed (CPU time) on real data, rather than heuristics, which is more practical for users of graphical models.

5. Experiments and Results

For our experiments, we evaluated the MH and GA on a diverse set of speech/language graphical models. The goal is to see whether either of them consistently finds better triangulations in real-world situations, as compared to standard triangulation techniques. We collected 10 real-world graphical models from various speech/language researchers: **Aurora Decoding** - whole word model for digit recognition, **CTS Decoding** - continuous speech recognition graph using monophones and bigram, **Edit Distance training 1, 2, decoding** - graphs for learning edit distance parameters from data [7], **Feature Detect** - for extracting phonetic features, courtesy of Simon King, **PhoneFree 1, 2** - isolated-word scoring using a phone-free model, from Karen Livescu, **Mandarin** - graphs for model Mandarin



(a) (b)

Figure 3: Comparison of GA and MH search on (a) **FeatureDetect** and (b) **MultiStream** graphs. The plots show the inference speed (seconds/frame) of the current best triangulation found as a function of search time (seconds). Triangulations found by the MH and GA are shown as dashed and solid lines, respectively. Note that GA improves inference speed at a much faster rate.

Chinese tonal phones with asynchronous spectral and pitch feature streams [18], **MultiStream** - asynchronous multi-stream training graph, based on graphs used in [1] and [2].

We evaluate our triangulation algorithm with two criteria: first is the inference speed performance of the resulting triangulation, and second is the search time required find a good triangulation. For the first criteria, the performance of MH and GA are compared with a baseline of standard triangulation methods: 6 methods are used to create 600 triangulations and for each method the triangulation with the best state space is chosen. These 6 triangulations plus a triangulation created by one single clique of all variables are timed, and the best is given in the results and referred to in the table by "*Base*". Table 1 compares the time performance of each method. The column labeled "Sec/100frames" gives the amount of time required for evaluating a short sequence of frames, as reported during the search process. This is the objective function optimized by both MH and GA (Baseline does not optimize on inference speed, so it does not have this number). A result more indicative of inference time as seen by the end-user of the graphical model is reported in the column labeled "Utterance Time", which lists the number of seconds required for performing inference over a number of complete utterances (or sentences). The speed improvement of MH and GA over instant triangulation techniques are shown in the column labeled "Speedup". It is defined as the ratio of utterance times (e.g. ΔMH=$\frac{Base}{MH}$, ΔGA=$\frac{Base}{GA}$), and shows the relative improvement in inference speed one gets after spending some time to acquire a triangulation by MH or GA methods. As seen, both MH and GA speedups are very high and quite comparable.

For the second criteria, we compare the search time for MH and GA: Fig. 3 shows the improvement of triangulations as a function of search time. For both plots, it's important to note that the GA achieves good triangulations at a much faster rate than the MH. This means that a researcher can easily use GA search within the development cycle of a speech/language graphical model, like one does with the baseline triangulation techniques.

6. Conclusions

We have addressed the important issue of making inference tasks tractable on graphical models for speech and language processing. The proposed genetic algorithm searches the space of all possible triangulations and optimizes the triangulation based on inference speed on real data. We demonstrate on 10 real-world speech/language graphs that our method outperforms conventional triangulation techniques. All the above algorithms have been implemented in the publicly-available Graphical Models Toolkit (GMTK) [19]. It is our hope that our work in triangulation search will encourage more graphical model research in speech and language processing.

7. References

[1] Y. Zhang, Q. Diao, S. Huang, W. Hu, C. Bartels, and J. Bilmes, "DBN based multi-stream models for speech," in *Proc. ICASSP*, 2003.

[2] A. Subramanya, J. Gowdy, C. Bartels, and J. Bilmes, "DBN based multi-stream models for audio-visual speech recognition," in *Proc. ICASSP*, Montreal, Canada, 2004.

[3] G. Zweig, "Speech recognition with dynamic Bayesian networks," Ph.D. dissertation, UC Berkeley, Spring 1998.

[4] K. Livescu and J. Glass, "Feature-based pronunciation modeling for speech recognition," in *HLT*, 2004.

[5] J. Frankel, M. Wester, and S. King, "Articulatory feature recognition using dynamic Bayesian networks," in *Proc. ICSLP*, Sept. 2004.

[6] X. Li, J. Malkin, and J. Bilmes, "Graphical model approach to pitch tracking," in *ICSLP*, 2004.

[7] K. Filali and J. Bilmes, "A dynamic bayesian framework to model context and memory in edit distance learning: An application to pronunciation classification," in *Proc. of Conf. Assoc. of Computational Lingustics*, 2005.

[8] F. Jensen and F. Jensen, "Optimal junction trees," in *Proc. 10th Conf. Uncertainty in AI*, 1994.

[9] S. Arnborg, D. J. Corneil, and A. P. Proskurowski, "Complexity of finding embeddings in a k-tree," *SIAM J. Alg. Dis. Meth.*, vol. 8, pp. 227–284, 1987.

[10] C. Bartels and J. Bilmes, "Elimination is not enough: Non-minimal triangulations for graphical models," U. of Washington, Tech. Rep. UWEETR-2004-0010, 2004.

[11] M. Jordan, "Graphical models," *Statistical Science*, no. 19, pp. 140–155, 2004.

[12] U. Kjaerulff, "Triangulation of graphs-algorithms giving small total state space," Aalborg University, Tech. Rep. R-90-09, March 1990.

[13] P. Larrañaga, C. Kuijpers, M. Poza, and R. Murga, "Decomposing Bayesian networks by genetic algorithms," *Statistics and Computing*, vol. 7, no. 1, pp. 19–34, 1997.

[14] D. J. Rose, "Triangulated graphs and the elimination process," *J. Math. Anal. Appl.*, vol. 32, pp. 597–609, 1970.

[15] J. Bilmes and C. Bartels, "On triangulating dynamic graphical models," in *Proc. 19th Conf. Uncertainty in AI*, Acapulco, Mexico, 2003.

[16] J. Holland, *Adaptation in Natural and Artificial Systems*. University of Michigan Press, 1975.

[17] S. Lauritzen, *Graphical Models*. Oxford Science Publications, 1996.

[18] L. Xin, G. Ji, J. Bilmes, and M. Ostendorf, "DBN multi-stream models for Mandarin toneme recognition," in *Proc. ICASSP*, 2005.

[19] J. Bilmes and G. Zweig, "The graphical models toolkit: An open source software system for speech and time-series processing," in *ICASSP*, June 2002.