# Efficient Search for Inversion Transduction Grammar

**Hao Zhang** and **Daniel Gildea**
Computer Science Department
University of Rochester
Rochester, NY 14627

## Abstract

We develop admissible A* search heuristics for synchronous parsing with Inversion Transduction Grammar, and present results both for bitext alignment and for machine translation decoding. We also combine the dynamic programming hook trick with A* search for decoding. These techniques make it possible to find optimal alignments much more quickly, and make it possible to find optimal translations for the first time. Even in the presence of pruning, we are able to achieve higher BLEU scores with the same amount of computation.

## 1 Introduction

The Inversion Transduction Grammar (ITG) of Wu (1997) is a syntactically motivated algorithm for producing word-level alignments of pairs of translationally equivalent sentences in two languages. The algorithm builds a synchronous parse tree for both sentences, and assumes that the trees have the same underlying structure but that the ordering of constituents may differ in the two languages. ITG imposes constraints on which alignments are possible, and these constraints have been shown to be a good match for real bitext data (Zens and Ney, 2003).

A major motivation for the introduction of ITG was the existence of polynomial-time algorithms both for alignment and translation. Alignment, whether for training a translation model using EM or for finding the Viterbi alignment of test data, is $O(n^6)$ (Wu, 1997), while translation (decoding) is $O(n^7)$ using a bigram language model, and $O(n^{11})$ with trigrams. While polynomial-time algorithms are a major improvement over the NP-complete problems posed by the alignment models of Brown et al. (1993), the degree of these polynomials is high, making both alignment and decoding infeasible for realistic sentences without very significant pruning. In this paper, we explore use of the "hook trick" (Eisner and Satta, 1999; Huang et al., 2005) to reduce the asymptotic complexity of decoding, and the use of heuristics to guide the search.

Our search heuristics are a conservative estimate of the outside probability of a bitext cell in the complete synchronous parse. Some estimate of this outside probability is a common element of modern statistical (monolingual) parsers (Charniak et al., 1998; Collins, 1999), and recent work has developed heuristics that are *admissible* for A* search, guaranteeing that the optimal parse will be found (Klein and Manning, 2003). We extend this type of outside probability estimate to include both word translation and $n$-gram language model probabilities. These measures have been used to guide search in word- or phrase-based MT systems (Wang and Waibel, 1997; Och et al., 2001), but in such models optimal search is generally not practical even with good heuristics. In this paper, we show that the same assumptions that make ITG polynomial-time can be used to efficiently compute heuristics which guarantee us that we will find the optimal alignment or translation, while significantly speeding the search.

## 2 Inversion Transduction Grammar

An Inversion Transduction Grammar can generate pairs of sentences in two languages by recursively applying context-free bilingual production rules. Most work on ITG has focused on the 2-normal form, which consists of unary production rules that are responsible for generating word pairs:

$$X \rightarrow e/f$$



*Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing (EMNLP 2006)*, pages 224–231,
Sydney, July 2006. 

and binary production rules in two forms that are responsible for generating syntactic subtree pairs:

$$X \rightarrow [Y\,Z]$$

and

$$X \rightarrow \langle Y\,Z \rangle$$

The rules with square brackets enclosing the right hand side expand the left hand side symbol into the two symbols on the right hand side in the same order in the two languages, whereas the rules with pointed brackets expand the left hand side symbol into the two right hand side symbols in reverse order in the two languages.

## 3 A* Viterbi Alignment Selection

A* parsing is a special case of agenda-based chart parsing, where the priority of a node $X[i, j]$ on the agenda, corresponding to nonterminal $X$ spanning positions $i$ through $j$, is the product of the node's current inside probability with an estimate of the outside probability. By the current inside probability, we mean the probability of the so-far-most-probable subtree rooted on the node $X[i, j]$, with leaves being ${}_iw_j$, while the outside probability is the highest probability for a parse with the root being $S[0, N]$ and the sequence ${}_0w_iX_jw_n$ forming the leaves. The node with the highest priority is removed from the agenda and added to the chart, and then explored by combining with all of its neighboring nodes in the chart to update the priorities of the resulting nodes on the agenda. By using estimates close to the actual outside probabilities, A* parsing can effectively reduce the number of nodes to be explored before putting the root node onto the chart. When the outside estimate is both admissible and monotonic, whenever a node is put onto the chart, its current best inside parse is the Viterbi inside parse.

To relate A* parsing with A* search for finding the lowest cost path from a certain source node to a certain destination node in a graph, we view the forest of all parse trees as a hypergraph. The source node in the hypergraph fans out into the nodes of unit spans that cover the individual words. From each group of children to their parent in the forest, there is a hyperedge. The destination node is the common root node for all the parse trees in the forest. Under the mapping, a parse is a hyperpath from the source node to the destination node. The Viterbi parse selection problem thus becomes finding the lowest-cost hyperpath from the source node to the destination node. The cost in this scenario is thus the negative of log probability. The inside estimate and outside estimate naturally correspond to the $\hat{g}$ and $\hat{h}$ for A* searching, respectively.

A stochastic ITG can be thought of as a stochastic CFG extended to the space of bitext. A node in the ITG chart is a bitext cell that covers a source substring and a target substring. We use the notion of $X[l, m, i, j]$ to represent a tree node in ITG parse. It can potentially be combined with any bitext cells at the four corners, as shown in Figure 1(a).

Unlike CFG parsing where the leaves are fixed, the Viterbi ITG parse selection involves finding the Viterbi alignment under ITG constraint. Good outside estimates have to bound the outside ITG Viterbi alignment probability tightly.

### 3.1 A* Estimates for Alignment

Under the ITG constraints, each source language word can be aligned with at most one target language word and vice versa. An ITG constituent $X[l, m, i, j]$ implies that the words in the source substring in the span $[l, m]$ are aligned with the words in the target substring $[i, j]$. It further implies that the words outside the span $[l, m]$ in the source are aligned with the words outside the span $[i, j]$ in the target language. Figure 1(b) displays the tic-tac-toe pattern for the inside and outside components of a particular cell. To estimate the upper bound of the ITG Viterbi alignment probability for the outside component with acceptable complexity, we need to relax the ITG constraint. Instead of ensuring one-to-one in both directions, we use a many-to-one constraint in one direction, and we relax all constraints on reordering within the outside component.

The many-to-one constraint has the same dynamic programming structure as IBM Model 1, where each target word is supposed to be translated from any of the source words or the NULL symbol. In the Model 1 estimate of the outside probability, source and target words can align using any combination of points from the four outside corners of the tic-tac-toe pattern. Thus in Figure 1(b), there is one solid cell (corresponding to the Model 1 Viterbi alignment) in each column, falling either in the upper or lower outside shaded corner. This can be also be thought of as squeezing together the four outside corners, creat-

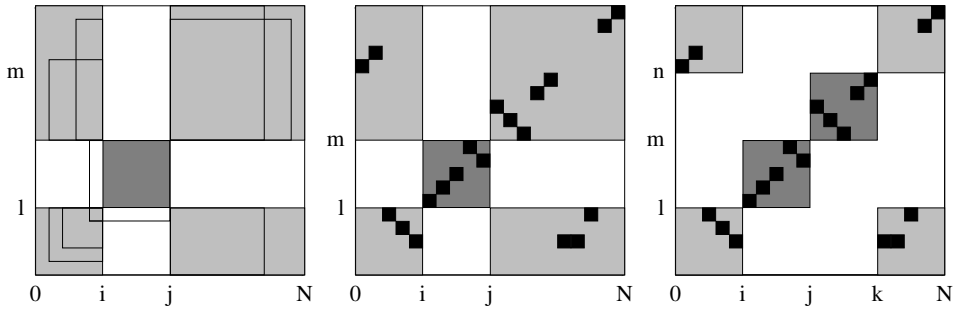



Figure 1: (a) A bitext cell $X[l, m, i, j]$ (shaded) for ITG parsing. The inside cell can be combined with adjacent cells in the four outside corners (lighter shading) to expand into larger cells. One possible expansion to the lower left corner is displayed. (b) The tic-tac-toe pattern of alignments consistent with a given cell. If the inner box is used in the final synchronous parse, all other alignments must come from the four outside corners. (c) Combination of two adjacent cells shown with region for new outside heuristic.

ing a new cell whose probability is estimated using IBM Model 1. In contrast, the inside Viterbi alignment satisfies the ITG constraint, implying only one solid cell in each column and each row. Mathematically, our Model 1 estimate for the outside component is:

$$h_{\mathrm{M1}}(l, m, i, j) = \prod_{\substack{t<i,\\ t>j}} \max_{\substack{s<l,\\ s>m}} P(f_t, e_s)$$

This Model 1 estimate is admissible. Maximizing over each column ensures that the translation probability for each target word is greater than or equal to the corresponding word translation probability under the ITG constraint. Model 1 virtually assigns a probability of 1 for deleting any source word. As a product of word-to-word translation probabilities including deletions and insertions, the ITG Viterbi alignment probability cannot be higher than the product of maximal word-to-word translation probabilities using the Model 1 estimate.

The Model 1 estimate is also monotonic, a property which is best understood geometrically. A successor state to cell $(l, m, i, j)$ in the search is formed by combining the cell with a cell which is adjacent at one of the four corners, as shown in Figure 1(c). Of the four outside corner regions used in calculating the search heuristic, one will be the same for the successor state, and three will be a subset of the old corner region. Without loss of generality, assume we are combining a cell $(m, n, j, k)$ that is adjacent to $(l, m, i, j)$ to the upper right. We define

$$H_{\mathrm{M1}}(l, m, i, j) = -\log h_{\mathrm{M1}}(l, m, i, j)$$

as the negative log of the heuristic in order to correspond to an estimated cost or distance in search terminology. Similarly, we speak of the cost of a chart entry $c(X[l, m, i, j])$ as its negative log probability, and the cost of a cell $c(l, m, i, j)$ as the cost of the best chart entry with the boundaries $(l, m, i, j)$. The cost of the cell $(m, n, j, k)$ which is being combined with the old cell is guaranteed to be greater than the contribution of the columns $j$ through $k$ to the heuristic $H_{\mathrm{M1}}(l, m, i, j)$. The contribution of the columns $k$ through $N$ to the new heuristic $H_{\mathrm{M1}}(l, n, i, k)$ is guaranteed to be greater in cost than their contribution to the old heuristic. Thus,

$$H_{\mathrm{M1}}(l, m, i, j) \leq c(m, n, j, k) + c(X \rightarrow YZ) + H_{\mathrm{M1}}(l, n, i, k)$$

meaning that the heuristic is monotonic or consistent.

The Model 1 estimate can be applied in both translation directions. The estimates from both directions are an upper bound of the actual ITG Viterbi probability. By taking the minimum of the two, we can get a tighter upper bound.

We can precompute the Model 1 outside estimate for all bitext cells before parsing starts. A naïve implementation would take $O(n^6)$ steps of computation, because there are $O(n^4)$ cells, each of which takes $O(n^2)$ steps to compute its Model 1 probability. Fortunately, exploiting the recursive

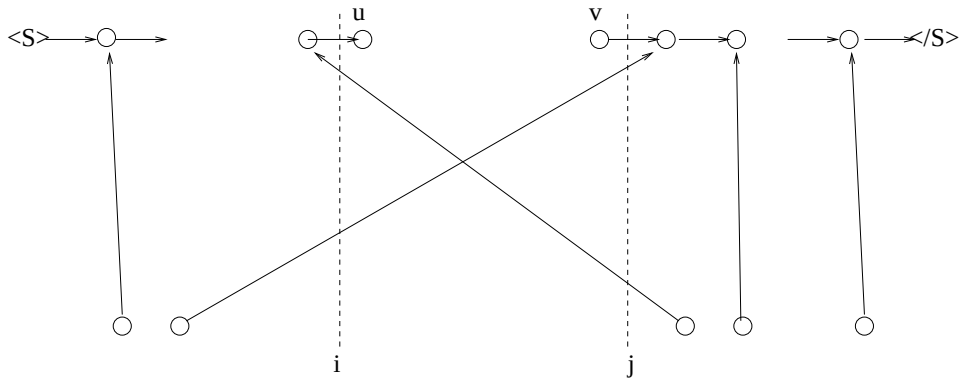



Figure 2: The region within the dashed lines is the translation hypothesis $X[i, j, u, v]$. The word sequence on the top is the Viterbi translation of the sentence on the bottom. Wide range word order change may happen.

nature of the cells, we can compute values for the inside and outside components of each cell using dynamic programming in $O(n^4)$ time (Zhang and Gildea, 2005).

## 4 A* Decoding

The of ITG decoding algorithm of Wu (1996) can be viewed as a variant of the Viterbi parsing algorithm for alignment selection. The task of standard alignment is to find word level links between two fixed-order strings. In the decoding situation, while the input side is a fixed sequence of words, the output side is a bag of words to be linked with the input words and then reordered. Under the ITG constraint, if the target language substring $[i, j]$ is translated into $s_1$ in the source language and the target substring $[j, k]$ is translated into $s_2$, then $s_1$ and $s_2$ must be consecutive in the source language as well and two possible orderings, $s_1 s_2$ and $s_2 s_1$, are allowed. Finding the best translation of the substring of $[i, k]$ involves searching over all possible split points $j$ and two possible reorderings for each split. In theory, the inversion probabilities associated with the ITG rules can do the job of reordering. However, a language model as simple as bigram is generally stronger. Using an $n$-gram language model implies keeping at least $n-1$ boundary words in the dynamic programming table for a hypothetical translation of a source language substring. In the case of a bigram ITG decoder, a translation hypothesis for the source language substring $[i, j]$ is denoted as $X[i, j, u, v]$, where $u$ and $v$ are the left boundary word and right boundary word of the target language counterpart.

As indicated by the similarity of parsing item notation, the dynamic programming property of the Viterbi decoder is essentially the same as the bitext parsing for finding the underlying Viterbi alignment. By permitting translation from the null target string of $[i, i]$ into source language words as many times as necessary, the decoder can translate an input sentence into a longer output sentence. When there is the null symbol in the bag of candidate words, the decoder can choose to translate a word into null to decrease the output length. Both insertions and deletions are special cases of the bitext parsing items.

Given the similarity of the dynamic programming framework to the alignment problem, it is not surprising that A* search can also be applied in a similar way. The initial parsing items on the agenda are the basic translation units: $X[i, i+1, u, u]$, for normal word-for-word translations and deletions (translations into nothing), and also $X[i, i, u, u]$, for insertions (translations from nothing). The goal item is $S[0, N, \langle s\rangle, \langle /s\rangle]$, where $\langle s\rangle$ stands for the beginning-of-sentence symbol and $\langle /s\rangle$ stands for the end-of-sentence symbol. The exploration step of the A* search is to expand the translation hypothesis of a substring by combining with neighboring translation hypotheses. When the outside estimate is admissible and monotonic, the exploration is optimal in the sense that whenever a hypothesis is taken from the top of the agenda, it is a Viterbi translation of the corresponding target substring. Thus, when $S[0, N, \langle s\rangle, \langle /s\rangle]$ is added to the chart, we have found the Viterbi translation for the entire sentence.

$$\beta(X[i,j,u,v]) = \max\left\{\beta_{\langle\rangle}(X[i,j,u,v]), \beta_{[]}(X[i,j,u,v])\right\}$$

$$\beta_{[]}(X[i,j,u,v]) = \max_{k,v_1,u_2,Y,Z}\left[\beta(Y[i,k,u,v_1]) \cdot \beta(Z[k,j,u_2,v]) \cdot P(X \rightarrow [YZ]) \cdot P_{lm}(u_2 \mid v_1)\right]$$

$$= \max_{k,u_2,Y,Z}\left[\max_{v_1}\left[\beta(Y[i,k,u,v_1]) \cdot P_{lm}(u_2 \mid v_1)\right] \cdot P(X \rightarrow [YZ]) \cdot \beta(Z[k,j,u_2,v])\right]$$

Figure 3: Top: An ITG decoding constituent can be built with either a straight or an inverted rule. Bottom: An efficient factorization for straight rules.

### 4.1 A* Estimates for Translation

The key to the success of A* decoding is an outside estimate that combines word-for-word translation probabilities and $n$-gram probabilities. Figure 2 is the picture of the outside translations and bigrams of a particular translation hypothesis $X[i,j,u,v]$.

Our heuristic involves precomputing two values for each word in the input string, involving forward- and backward-looking language model probabilities. For the forward looking value $h_f$ at input position $n$, we take a maximum over the set of words $S_n$ that the input word $t_n$ can be translated as:

$$h_f(n) = \max_{s \in S_n}\left[P_t(s \mid t_n) \max_{s' \in S} P_{lm}(s' \mid s)\right]$$

where:

$$S = \bigcup_n S_n$$

is the set of all possible translations for all words in the input string. While $h_f$ considers language model probabilities for words following $s$, the backward-looking value $h_b$ considers language model probabilities for $s$ given possible preceding words:

$$h_b(n) = \max_{s \in S_n}\left[P_t(s \mid t_n) \max_{s' \in S} P_{lm}(s \mid s')\right]$$

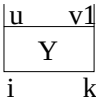
Our overall heuristic for a partial translation hypothesis $X[i,j,u,v]$ combines language model probabilities at the boundaries of the input substring with backward-looking values for the preceding words, and forward-looking values for the following words:

$$h(i,j,u,v) = \left[\max_{s \in S} P_{lm}(u \mid s)\right]\left[\max_{s \in S} P_{lm}(s \mid v)\right] \cdot \prod_{\substack{n<i,\\ n>j}} \max\left[h_b(n), h_f(n)\right]$$

Because we don't know whether a given input word will appear before or after the partial hypothesis in the final translation, we take the maximum of the forward and backward values for words outside the span $[i,j]$.

### 4.2 Combining the Hook Trick with A*

The hook trick is a factorization technique for dynamic programming. For bilexical parsing, Eisner and Satta (1999) pointed out we can reduce the complexity of parsing from $O(n^5)$ to $O(n^4)$ by combining the non-head constituents with the bilexical rules first, and then combining the resultant hook constituents with the head constituents. By doing so, the maximal number of interactive variables ranging over $n$ is reduced from 5 to 4. For ITG decoding, we can apply a similar factorization trick. We describe the bigram-integrated decoding case here, and refer to Huang et al. (2005) for more detailed discussion. Figure 3 shows how to decompose the expression for the case of straight rules; the same method applies to inverted rules. The number of free variables on the right hand side of the second equation is 7: $i$, $j$, $k$, $u$, $v$, $v_1$, and $u_2$.[1] After factorization, counting the free variables enclosed in the innermost $\max$ operator, we get five: $i$, $k$, $u$, $v_1$, and $u_2$. The decomposition eliminates one free variable, $v_1$. In the outermost level, there are six free variables left. The maximum number of interacting variables is six overall. So, we reduced the complexity of ITG decoding using bigram language model from $O(n^7)$ to $O(n^6)$. If we visualize an ITG decoding constituent $Y$ extending from source language position $i$ to $k$ and target language boundary words $u$ and $v_1$ with a diagram:




[1] $X$, $Y$, and $Z$ range over grammar nonterminals, of which there are a constant number.

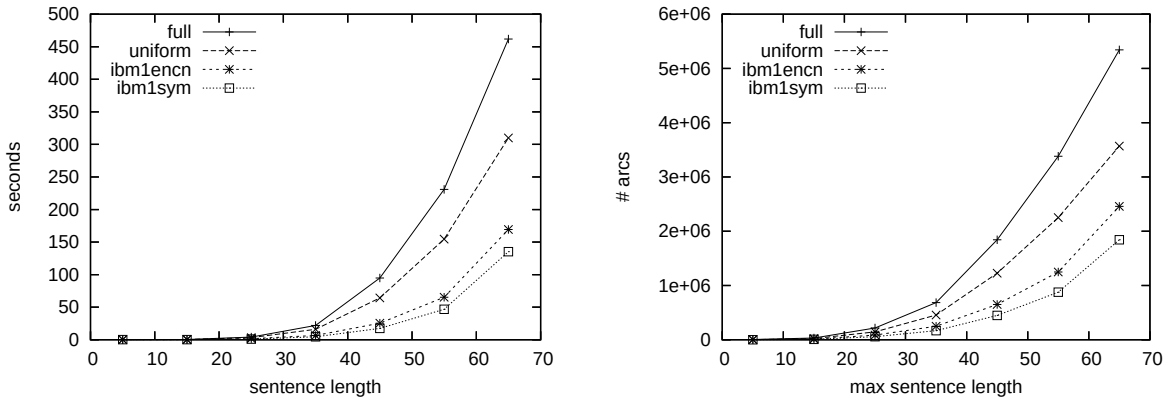
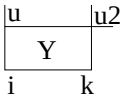



Figure 4: Speed of various techniques for finding the optimal alignment.

the hook corresponding to the innermost $\max$ operator in the equation can be visualized as follows:




with the expected language model state $u_2$ "hanging" outside the target language string.

The trick is generic to the control strategies of actual parsing, because the hooks can be treated as just another type of constituent. Building hooks is like applying special unary rules on top of non-hooks. In terms of of outside heuristic for hooks, there is a slight difference from that for non-hooks:

$$h(i,j,u,v) = \left[\max_{s \in S} P_{lm}(s \mid v)\right] \cdot \prod_{\substack{n<i,\\ n>j}} \max\left[h_b(n), h_f(n)\right]$$

That is, we do not need the backward-looking estimate for the left boundary word $u$.

## 5 Experiments

We tested the performance of our heuristics for alignment on a Chinese-English newswire corpus. Probabilities for the ITG model were trained using Expectation Maximization on a corpus of 18,773 sentence pairs with a total of 276,113 Chinese words and 315,415 English words. For EM training, we limited the data to sentences of no more than 25 words in either language. Here we present timing results for finding the Viterbi alignment of longer sentences using this fixed translation model with different heuristics. We compute alignments on a total of 117 test sentences, which are broken down by length as shown in Table 1.

| *Length* | # sentences |
|---|---|
| 0-9 | 5 |
| 10–19 | 26 |
| 20–29 | 29 |
| 30–39 | 22 |
| 40–49 | 24 |
| 50–59 | 10 |
| 60 | 1 |

Table 1: Length of longer sentence in each pair from test data.

| *method* | *time* | *speedup* |
|---|---|---|
| full | 815s | – |
| uniform | 547s | 1.4 |
| ibm1encn | 269s | 3.0 |
| ibm1sym | 205s | 3.9 |

Table 2: Total time for each alignment method.

Results are presented both in terms of time and the number of arcs added to the chart before the optimal parse is found. *Full* refers to exhaustive parsing, that is, building a complete chart with all $n^4$ arcs. *Uniform* refers to a best-first parsing strategy that expands the arcs with the highest inside probability at each step, but does not incorporate an estimate of the outside probability. *Ibm1encn* denotes our heuristic based on IBM model 1, applied to translations from English to Chinese, while *ibm1sym* applies the Model 1 heuristic in both translation directions and takes the minimum. The factor by which times were decreased was found to be roughly constant across different length sentences. The alignment times for the entire test set are shown in Table 2, the best heuristic is 3.9 times faster than exhaustive

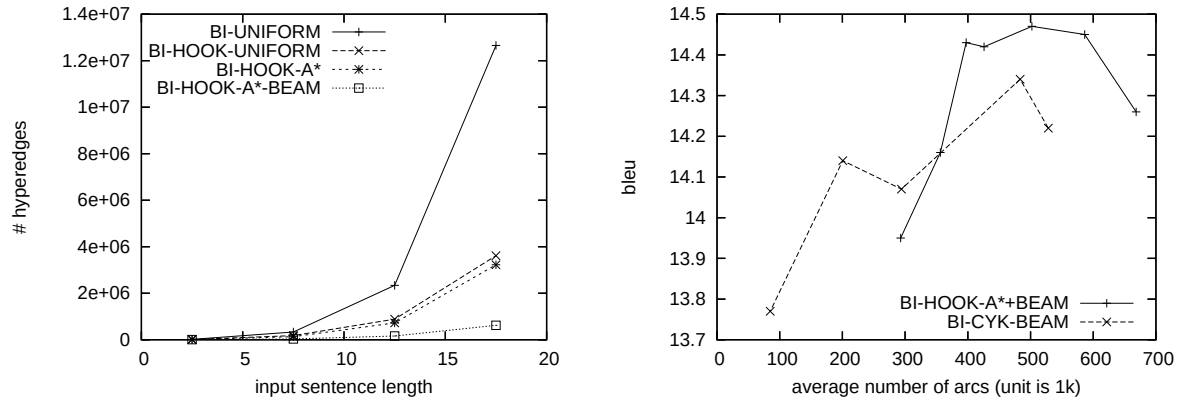



Figure 5: On the left, we compare decoding speed for uniform outside estimate best-first decoding with and without the hook trick, as well as results using our heuristic (labeled A*) and with beam pruning (which no longer produces optimal results). On the right, we show the relationship between computation time and BLEU scores as the pruning threshold is varied for both A* search and bottom-up CYK parsing.

dynamic programming.

We did our ITG decoding experiments on the LDC 2002 MT evaluation data set for translation of Chinese newswire sentences into English. The evaluation data set has 10 human translation references for each sentence. There are a total of 371 Chinese sentences of no more than 20 words in the data set. These sentences are the test set for our different versions of ITG decoders using both a bigram language model and a trigram language model. We evaluate the translation results by comparing them against the reference translations using the BLEU metric. The word-for-word translation probabilities are from the translation model of IBM Model 4 trained on a 160-million-word English-Chinese parallel corpus using GIZA++. The language model is trained on a 30-million-word English corpus. The rule probabilities for ITG are from the same training as in the alignment experiments described above.

We compared the BLEU scores of the A* decoder and the ITG decoder that uses beam ratio pruning at each stage of bottom-up parsing. In the case of bigram-integrated decoding, for each input word, the best 2 translations are put into the bag of output words. In the case of trigram-integrated decoding, top 5 candidate words are chosen. The A* decoder is guaranteed to find the Viterbi translation that maximizes the product of $n$-grams probabilities, translation probabilities (including insertions and deletions) and inversion rule probabilities by choosing the right words and the right word order subject to the ITG constraint.

Figure 5 (left) demonstrates the speedup obtained through the hook trick, the heuristic, and pruning, all based on A* search. Table 3 shows the improvement of BLEU score after applying the A* algorithm to find the optimal translation under the model. Figure 5 (right) investigates the relationship between the search effort and BLEU score for A* and bottom-up CYK parsing, both with pruning. Pruning for A* works in such a way that we never explore a low probability hypothesis falling out of a certain beam ratio of the best hypothesis within the bucket of $X[i, j, *, *]$, where $*$ means any word. Table 4 shows results for trigram-integrated decoding. However, due to time constraint, we have not explored time/performance tradeoff as we did for bigram decoding.

The number of combinations in the table is the average number of hyperedges to be explored in searching, proportional to the total number of

| *Decoder* | Combinations | BLEU |
|---|---|---|
| BI-UNIFORM | 8.02M | 14.26 |
| BI-HOOK-A* | 2.10M | 14.26 |
| BI-HOOK-A*-BEAM | 0.40M | 14.43 |
| BI-CYK-BEAM | 0.20M | 14.14 |

Table 3: Decoder speed and BLEU scores for bigram decoding.

| *Decoder* | Cbns | BLEU |
|---|---|---|
| TRI-A*-BEAM($10^{-3}$) | 213.4M | 17.83 |
| TRI-A*-BEAM($10^{-2}$) | 20.7M | 17.09 |
| TRI-CYK-BEAM($10^{-3}$) | 21.2M | 16.86 |

Table 4: Results for trigram decoding.

computation steps.

## 6 Conclusion

A* search for Viterbi alignment selection under ITG is efficient using IBM Model 1 as an outside estimate. The experimental results indicate that despite being a more relaxed word-for-word alignment model than ITG, IBM Model 1 can serve as an efficient and reliable approximation of ITG in terms of Viterbi alignment probability. This is more true when we apply Model 1 to both translation directions and take the minimum of both. We have also tried to incorporate estimates of binary rule probabilities to make the outside estimate even sharper. However, the further improvement was marginal.

We are able to find the ITG Viterbi translation using our A* decoding algorithm with an outside estimate that combines outside bigrams and translation probabilities for outside words. The hook trick gave us a significant further speedup; we believe this to be the first demonstrated practical application of this technique.

Interestingly, the BLEU score for the optimal translations under the probabilistic model is lower than we achieve with our best bigram-based system using pruning. However, this system makes use of the A* heuristic, and our speed/performance curve shows that the heuristic allows us to achieve higher BLEU scores with the same amount of computation. In the case of trigram integrated decoding, there is 1 point of BLEU score improvement by moving from a typical CYK plus beam search decoder to a decoder using A* plus beam search.

However, without knowing what words will appear in the output language, a very sharp outside estimate to further bring down the number of combinations is difficult to achieve.

The brighter side of the move towards optimal decoding is that the A* search strategy leads us to the region of the search space that is close to the optimal result, where we can more easily find good translations.

**Acknowledgments** This work was supported by NSF ITR IIS-09325646 and NSF ITR IIS-0428020.

## References

Peter F. Brown, Stephen A. Della Pietra, Vincent J. Della Pietra, and Robert L. Mercer. 1993. The mathematics of statistical machine translation: Parameter estimation. *Computational Linguistics*, 19(2):263–311.

Eugene Charniak, Sharon Goldwater, and Mark Johnson. 1998. Edge-based best-first chart parsing. In *Proceedings of the Sixth Workshop on Very Large Corpora*.

Michael John Collins. 1999. *Head-driven Statistical Models for Natural Language Parsing*. Ph.D. thesis, University of Pennsylvania, Philadelphia.

Jason Eisner and Giorgio Satta. 1999. Efficient parsing for bilexical context-free grammars and head automaton grammars. In *37th Annual Meeting of the Association for Computational Linguistics*.

Liang Huang, Hao Zhang, and Daniel Gildea. 2005. Machine translation as lexicalized parsing with hooks. In *International Workshop on Parsing Technologies (IWPT05)*, Vancouver, BC.

Dan Klein and Christopher D. Manning. 2003. A* parsing: Fast exact viterbi parse selection. In *Proceedings of the 2003 Meeting of the North American chapter of the Association for Computational Linguistics (NAACL-03)*.

Franz Josef Och, Nicola Ueffing, and Herman Ney. 2001. An efficient a* search algorithm for statistical machine translation. In *Proceedings of the ACL Workshop on Data-Driven Machine Translation*, pages 55–62, Toulouse, France.

Ye-Yi Wang and Alex Waibel. 1997. Decoding algorithm in statistical machine translation. In *35th Annual Meeting of the Association for Computational Linguistics*.

Dekai Wu. 1996. A polynomial-time algorithm for statistical machine translation. In *34th Annual Meeting of the Association for Computational Linguistics*.

Dekai Wu. 1997. Stochastic inversion transduction grammars and bilingual parsing of parallel corpora. *Computational Linguistics*, 23(3):377–403.

Richard Zens and Hermann Ney. 2003. A comparative study on reordering constraints in statistical machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, Sapporo, Japan.

Hao Zhang and Daniel Gildea. 2005. Stochastic lexicalized inversion transduction grammar for alignment. In *Proceedings of the 43rd Annual Conference of the Association for Computational Linguistics (ACL-05)*, Ann Arbor, MI.