Approximate Factoring for A* Search

Aria Haghighi, John DeNero, Dan Klein
Computer Science Division
University of California Berkeley
{aria42, denero, klein}@cs.berkeley.edu

Abstract

We present a novel method for creating A* estimates for structured search problems. In our approach, we project a complex model onto multiple simpler models for which exact inference is efficient. We use an optimization framework to estimate parameters for these projections in a way which bounds the true costs. Similar to Klein and Manning (2003), we then combine completion estimates from the simpler models to guide search in the original complex model. We apply our approach to bitext parsing and lexicalized parsing, demonstrating its effectiveness in these domains.

1 Introduction

Inference tasks in NLP often involve searching for an optimal output from a large set of structured outputs. For many complex models, selecting the highest scoring output for a given observation is slow or even intractable. One general technique to increase efficiency while preserving optimality is A* search (Hart et al., 1968); however, successfully using A* search is challenging in practice. The design of admissible (or nearly admissible) heuristics which are both effective (close to actual completion costs) and also efficient to compute is a difficult, open problem in most domains. As a result, most work on search has focused on non-optimal methods, such as beam search or pruning based on approximate models (Collins, 1999), though in certain cases admissible heuristics are known (Och and Ney, 2000; Zhang and Gildea, 2006). For example, Klein and Manning (2003) show a class of projection-based A* estimates, but their application is limited to models which have a very restrictive kind of score decomposition. In this work, we broaden their projection-based technique to give A* estimates for models which do not factor in this restricted way.

Like Klein and Manning (2003), we focus on search problems where there are multiple projections or "views" of the structure, for example lexical parsing, in which trees can be projected onto either their CFG backbone or their lexical attachments. We use general optimization techniques (Boyd and Vandenberghe, 2005) to approximately factor a model over these projections. Solutions to the projected problems yield heuristics for the original model. This approach is flexible, providing either admissible or nearly admissible heuristics, depending on the details of the optimization problem solved. Furthermore, our approach allows a modeler explicit control over the trade-off between the tightness of a heuristic and its degree of inadmissibility (if any). We describe our technique in general and then apply it to two concrete NLP search tasks: bitext parsing and lexicalized monolingual parsing.

2 General Approach

Many inference problems in NLP can be solved with agenda-based methods, in which we incrementally build hypotheses for larger items by combining smaller ones with some local configurational structure. We can formalize such tasks as graph search problems, where states encapsulate partial hypotheses and edges combine or extend them locally.[1] For example, in HMM decoding, the states are anchored labels, e.g. $VBD[5]$, and edges correspond to hidden transitions, e.g. $VBD[5] \rightarrow DT[6]$.

The search problem is to find a minimal cost path from the start state to a goal state, where the path cost is the sum of the costs of the edges in the path.

---

[1]In most complex tasks, we will in fact have a hypergraph, but the extension is trivial and not worth the added notation.

(a) Local Configurations

	a' → b'	b' → c'
a → b	$\left(\begin{smallmatrix}a\\a'\end{smallmatrix}\right) \to \left(\begin{smallmatrix}b\\b'\end{smallmatrix}\right)$	$\left(\begin{smallmatrix}a\\b'\end{smallmatrix}\right) \to \left(\begin{smallmatrix}b\\c'\end{smallmatrix}\right)$
b → c	$\left(\begin{smallmatrix}b\\a'\end{smallmatrix}\right) \to \left(\begin{smallmatrix}c\\b'\end{smallmatrix}\right)$	$\left(\begin{smallmatrix}b\\b'\end{smallmatrix}\right) \to \left(\begin{smallmatrix}c\\c'\end{smallmatrix}\right)$

(b)

Original Cost Matrix

	a' → b'	b' → c'
a → b	2.0	3.0
b → c	3.0	4.0

Factored Cost Matrix

		c(a' → b') 1.0	c(a' → b') 2.0
c(a → b)	1.0	2.0	3.0
c(b → c)	2.0	3.0	4.0

(c)

Original Cost Matrix

	a' → b'	b' → c'
a → b	2.0	3.0
b → c	3.0	5.0

Factored Cost Matrix

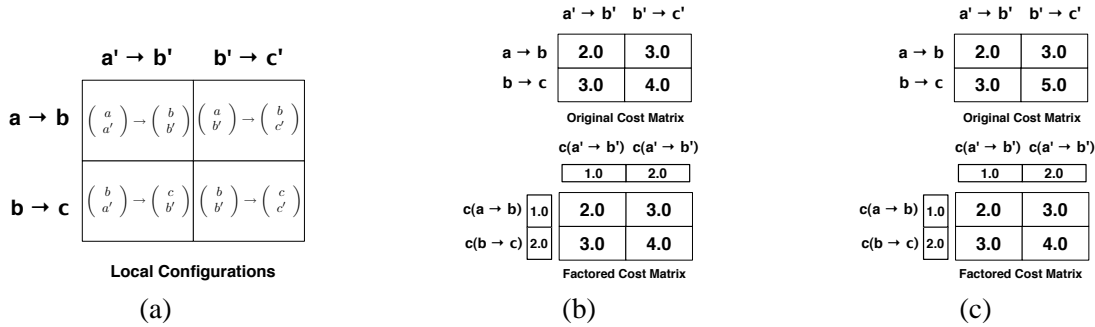|		c(a' → b') 1.0	c(a' → b') 2.0
c(a → b)	1.0	2.0	3.0
c(b → c)	2.0	3.0	4.0

Figure 1: Example cost factoring: In (a), each cell of the matrix is a local configuration composed of two projections (the row and column of the cell). In (b), the top matrix is an example cost matrix, which specifies the cost of each local configuration. The bottom matrix represents our factored estimates, where each entry is the sum of configuration projections. For this example, the actual cost matrix can be decomposed exactly into two projections. In (c), the top cost matrix cannot be exactly decomposed along two dimensions. Our factored cost matrix has the property that each factored cost estimate is below the actual configuration cost. Although our factorization is no longer tight, it still can be used to produce an admissible heuristic.

For probabilistic inference problems, the cost of an edge is typically a negative log probability which depends only on some local configuration type. For instance, in PCFG parsing, the (hyper)edges reference anchored spans $X[i, j]$, but the edge costs depend only on the local rule type $X \to YZ$. We will use a to refer to a local configuration and use $c(a)$ to refer to its cost. Because edge costs are sensitive only to local configurations, the cost of a path is $\sum_a c(a)$. A* search requires a *heuristic function*, which is an estimate $h(s)$ of the *completion cost*, the cost of a best path from state s to a goal.

In this work, following Klein and Manning (2003), we consider problems with *projections* or "views," which define mappings to simpler state and configuration spaces. For instance, suppose that we are using an HMM to jointly model part-of-speech (POS) and named-entity-recognition (NER) tagging. There might be one projection onto the NER component and another onto the POS component. Formally, a projection π is a mapping from states to some coarser domain. A state projection induces projections of edges and of the entire graph $\pi(G)$.

We are particularly interested in search problems with multiple projections $\{\pi_1, \ldots, \pi_\ell\}$ where each projection, π_i, has the following properties: its state projections induce well-defined projections of the local configurations $\pi_i(a)$ used for scoring, *and* the projected search problem admits a simpler inference. For instance, the POS projection in our NER-POS HMM is a simpler HMM, though the gains from this method are greater when inference in the projections have lower asymptotic complexity than

the original problem (see sections 3 and 4).

In defining projections, we have not yet dealt with the projected scoring function. Suppose that the cost of local configurations decomposes along projections as well. In this case,

$$c(a) = \sum_{i=1}^{\ell} c_i(a), \forall a \in \mathcal{A} \qquad (1)$$

where $\mathcal{A}$ is the set of local configurations and $c_i(a)$ represents the cost of configuration a under projection π_i. A toy example of such a cost decomposition in the context of a Markov process over two-part states is shown in figure 1(b), where the costs of the joint transitions equal the sum of costs of their projections. Under the strong assumption of equation (1), Klein and Manning (2003) give an admissible A* bound. They note that the cost of a path decomposes as a sum of projected path costs. Hence, the following is an admissible additive heuristic (Felner et al., 2004),

$$h(s) = \sum_{i=1}^{\ell} h_i^*(s) \qquad (2)$$

where $h_i^*(s)$ denote the optimal completion costs in the projected search graph $\pi_i(G)$. That is, the completion cost of a state bounds the sum of the completion costs in each projection.

In virtually all cases, however, configuration costs will not decompose over projections, nor would we expect them to. For instance, in our joint POS-NER task, this assumption requires that the POS and NER

transitions and observations be generated independently. This independence assumption undermines the motivation for assuming a joint model. In the central contribution of this work, we exploit the projection structure of our search problem without making any assumption about cost decomposition.

Rather than assuming decomposition, we propose to find scores ϕ for the projected configurations which are *pointwise admissible*:

$$\sum_{i=1}^{\ell} \phi_i(a) \leq c(a), \forall a \in \mathcal{A} \qquad (3)$$

Here, $\phi_i(a)$ represents a factored projection cost of $\pi_i(a)$, the π_i projection of configuration a. Given pointwise admissible ϕ_i's we can again apply the heuristic recipe of equation (2). An example of factored projection costs are shown in figure 1(c), where no exact decomposition exists, but a pointwise admissible lower bound is easy to find.

Claim. *If a set of factored projection costs $\{\phi_1, \ldots, \phi_\ell\}$ satisfy pointwise admissibility, then the heuristic from (2) is an admissible A^* heuristic.*

Proof. Assume $a_1, \ldots, a_k$ are configurations used to optimally reach the goal from state s. Then,

$$h^*(s) = \sum_{j=1}^{k} c(a_j) \geq \sum_{j=1}^{k} \sum_{i=1}^{\ell} \phi_i(a_j)$$
$$= \sum_{i=1}^{\ell} \left(\sum_{j=1}^{k} \phi_i(a_j) \right) \geq \sum_{i=1}^{\ell} h_i^*(s) = h(s)$$

$\square$

The first inequality follows from pointwise admissibility. The second inequality follows because each inner sum is a completion cost for projected problem π_i and therefore $h_i^*(s)$ lower bounds it. Intuitively, we can see two sources of slack in such projection heuristics. First, there may be slack in the pointwise admissible scores. Second, the best paths in the projections will be overly optimistic because they have been decoupled (see figure 5 for an example of decoupled best paths in projections).

2.1 Finding Factored Projections for Non-Factored Costs

We can find factored costs $\phi_i(a)$ which are pointwise admissible by solving an optimization problem.

We think of our unknown factored costs as a block vector $\phi = [\phi_1, .., \phi_\ell]$, where vector ϕ_i is composed of the factored costs, $\phi_i(a)$, for each configuration $a \in \mathcal{A}$. We can then find admissible factored costs by solving the following optimization problem,

$$\underset{\phi}{\text{minimize}} \; \|\gamma\| \qquad (4)$$

$$\text{such that, } \gamma_a = c(a) - \sum_{i=1}^{\ell} \phi_i(a), \forall a \in \mathcal{A}$$

$$\gamma_a \geq 0, \forall a \in \mathcal{A}$$

We can think of each γ_a as the amount by which the cost of configuration a exceeds the factored projection estimates (the pointwise A^* gap). Requiring $\gamma_a \geq 0$ insures pointwise admissibility. Minimizing the norm of the γ_a variables encourages tighter bounds; indeed if $\|\gamma\| = 0$, the solution corresponds to an exact factoring of the search problem. In the case where we minimize the 1-norm or ∞-norm, the problem above reduces to a linear program, which can be solved efficiently for a large number of variables and constraints.[2]

Viewing our procedure decision-theoretically, by minimizing the norm of the pointwise gaps we are effectively choosing a loss function which decomposes along configuration types and takes the form of the norm (i.e. linear or squared losses). A complete investigation of the alternatives is beyond the scope of this work, but it is worth pointing out that in the end we will care only about the gap on entire structures, not configurations, and individual configuration factored costs need not even be pointwise admissible for the overall heuristic to be admissible.

Notice that the number of constraints is $|\mathcal{A}|$, the number of possible local configurations. For many search problems, enumerating the possible configurations is not feasible, and therefore neither is solving an optimization problem with all of these constraints. We deal with this situation in applying our technique to lexicalized parsing models (section 4).

Sometimes, we might be willing to trade search optimality for efficiency. In our approach, we can explicitly make this trade-off by designing an alternative optimization problem which allows for slack

---

[2]We used the MOSEK package (Andersen and Andersen, 2000).

in the admissibility constraints. We solve the following soft version of problem (4):

$$\underset{\phi}{\text{minimize}} \ \|\gamma^+\| + C\|\gamma^-\| \tag{5}$$

$$\text{such that, } \gamma_a = c(a) - \sum_{i=1}^{\ell} \phi_i(a), \forall a \in \mathcal{A}$$

where $\gamma^+ = \max\{0, \gamma\}$ and $\gamma^- = \max\{0, -\gamma\}$ represent the componentwise positive and negative elements of γ respectively. Each $\gamma_a^- > 0$ represents a configuration where our factored projection estimate is not pointwise admissible. Since this situation may result in our heuristic becoming inadmissible if used in the projected completion costs, we more heavily penalize overestimating the cost by the constant C.

2.2 Bounding Search Error

In the case where we allow pointwise inadmissibility, i.e. variables γ_a^-, we can bound our search error. Suppose $\gamma_{\max}^- = \max_{a \in \mathcal{A}} \gamma_a^-$ and that L^* is the length of the longest optimal solution for the original problem. Then, $h(s) \leq h^*(s) + L^*\gamma_{\max}^-$, $\forall s \in \mathcal{S}$. This ϵ-admissible heuristic (Ghallab and Allard, 1982) bounds our search error by $L^*\gamma_{\max}^-$.[3]

3 Bitext Parsing

In bitext parsing, one jointly infers a synchronous phrase structure tree over a sentence w_s and its translation w_t (Melamed et al., 2004; Wu, 1997). Bitext parsing is a natural candidate task for our approximate factoring technique. A synchronous tree projects monolingual phrase structure trees onto each sentence. However, the costs assigned by a weighted synchronous grammar (WSG) $\mathcal{G}$ do not typically factor into independent monolingual WCFGs. We can, however, produce a useful surrogate: a pair of monolingual WCFGs with structures projected by $\mathcal{G}$ and weights that, when combined, underestimate the costs of $\mathcal{G}$.

Parsing optimally relative to a synchronous grammar using a dynamic program requires time $O(n^6)$ in the length of the sentence (Wu, 1997). This high degree of complexity makes exhaustive bitext parsing infeasible for all but the shortest sentences. In

contrast, monolingual CFG parsing requires time $O(n^3)$ in the length of the sentence.

3.1 A* Parsing

Alternatively, we can search for an optimal parse guided by a heuristic. The states in A* bitext parsing are rooted bispans, denoted $X\,[i,j] \ :: \ Y\,[k,l]$. States represent a joint parse over subspans $[i,j]$ of w_s and $[k,l]$ of w_t rooted by the nonterminals X and Y respectively.

Given a WSG $\mathcal{G}$, the algorithm prioritizes a state (or edge) e by the sum of its inside cost $\beta_\mathcal{G}(e)$ (the negative log of its inside probability) and its outside estimate $h(e)$, or completion cost.[4] We are guaranteed the optimal parse if our heuristic $h(e)$ is never greater than $\alpha_\mathcal{G}(e)$, the true outside cost of e.

We now consider a heuristic combining the completion costs of the monolingual projections of $\mathcal{G}$, and guarantee admissibility by enforcing point-wise admissibility. Each state $e = X\,[i,j] \ :: \ Y\,[k,l]$ projects a pair of monolingual rooted spans. The heuristic we propose sums independent outside costs of these spans in each monolingual projection.

$$h(e) = \alpha_s(X\,[i,j]) + \alpha_t(Y\,[k,l])$$

These monolingual outside scores are computed relative to a pair of monolingual WCFG grammars $\mathcal{G}_s$ and $\mathcal{G}_t$ given by splitting each synchronous rule

$$r = \begin{pmatrix} \mathbf{X}^{(s)} \\ \mathbf{Y}^{(t)} \end{pmatrix} \rightarrow \begin{pmatrix} \alpha\ \beta \\ \gamma\ \delta \end{pmatrix}$$

into its components $\pi_s(r) = X \rightarrow \alpha\beta$ and $\pi_t(r) = Y \rightarrow \gamma\delta$ and weighting them via optimized $\phi_s(r)$ and $\phi_t(r)$, respectively.[5]

To learn pointwise admissible costs for the monolingual grammars, we formulate the following optimization problem:[6]

$$\underset{\gamma, \phi_s, \phi_t}{\text{minimize}} \ \|\gamma\|_1$$

$$\text{such that, } \gamma_r = c(r) - [\phi_s(r) + \phi_t(r)]$$

$$\text{for all synchronous rules } r \in \mathcal{G}$$

$$\phi_s \geq 0, \phi_t \geq 0, \gamma \geq 0$$

---

[3]This bound may be very loose if L is large.

[4]All inside and outside costs are Viterbi, not summed.

[5]Note that we need only parse each sentence (monolingually) once to compute the outside probabilities for every span.

[6]The stated objective is merely one reasonable choice among many possibilities which require pointwise admissibility and encourage tight estimates.

Figure 2: The gap between the heuristic (left) and true completion cost (right) comes from relaxing the synchronized problem to independent subproblems and slack in the factored models.

Figure 3: (a) A tree-to-tree transducer rule. (b) An example training sentence pair that yields rule (a).

Figure 2 diagrams the two bounds that enforce the admissibility of $h(e)$. For any outside cost $\alpha_{\mathcal{G}}(e)$, there is a corresponding optimal completion structure o under $\mathcal{G}$, which is an outer shell of a synchronous tree. o projects monolingual completions o_s and o_t which have well-defined costs $c_s(o_s)$ and $c_t(o_t)$ under $\mathcal{G}_s$ and $\mathcal{G}_t$ respectively. Their sum $c_s(o_s) + c_t(o_t)$ will underestimate $\alpha_{\mathcal{G}}(e)$ by pointwise admissibility.

Furthermore, the heuristic we compute underestimates this sum. Recall that the monolingual outside score $\alpha_s(X\,[i,j])$ is the minimal costs for any completion of the edge. Hence, $\alpha_s(X\,[i,j]) \leq c_s(o_s)$ and $\alpha_t(X\,[k,l]) \leq c_t(o_t)$. Admissibility follows.

3.2 Experiments

We demonstrate our technique using the synchronous grammar formalism of tree-to-tree transducers (Knight and Graehl, 2004). In each weighted rule, an aligned pair of nonterminals generates two ordered lists of children. The non-terminals in each list must align one-to-one to the non-terminals in the other, while the terminals are placed freely on either side. Figure 3(a) shows an example rule.

Following Galley et al. (2004), we learn a grammar by projecting English syntax onto a foreign language via word-level alignments, as in figure 3(b).[7]

We parsed 1200 English-Spanish sentences using a grammar learned from 40,000 sentence pairs of the English-Spanish Europarl corpus.[8] Figure 4(a) shows that A* expands substantially fewer states while searching for the optimal parse with our *op-*

---

[7] The bilingual corpus consists of translation pairs with fixed English parses and word alignments. Rules were scored by their relative frequencies.

[8] Rare words were replaced with their parts of speech to limit the memory consumption of the parser.

timization heuristic. The *exhaustive* curve shows edge expansions using the null heuristic. The intermediate result, labeled *English only*, used only the English monolingual outside score as a heuristic. Similar results using only Spanish demonstrate that both projections contribute to parsing efficiency. All three curves in figure 4 represent running times for finding the optimal parse.

Zhang and Gildea (2006) offer a different heuristic for A* parsing of ITG grammars that provides a forward estimate of the cost of aligning the unparsed *words* in both sentences. We cannot directly apply this technique to our grammar because tree-to-tree transducers only align non-terminals. Instead, we can augment our synchronous grammar model to include a lexical alignment component, then employ both heuristics. We learned the following two-stage generative model: a tree-to-tree transducer generates trees whose leaves are parts of speech. Then, the words of each sentence are generated, either jointly from aligned parts of speech or independently given a null alignment. The cost of a complete parse under this new model decomposes into the cost of the synchronous tree over parts of speech and the cost of generating the lexical items.

Given such a model, both our optimization heuristic and the lexical heuristic of Zhang and Gildea (2006) can be computed independently. Crucially, the sum of these heuristics is still admissible. Results appear in figure 4(b). Both heuristics (*lexical* and *optimization*) alone improve parsing performance, but their sum *opt+lex* substantially improves upon either one.
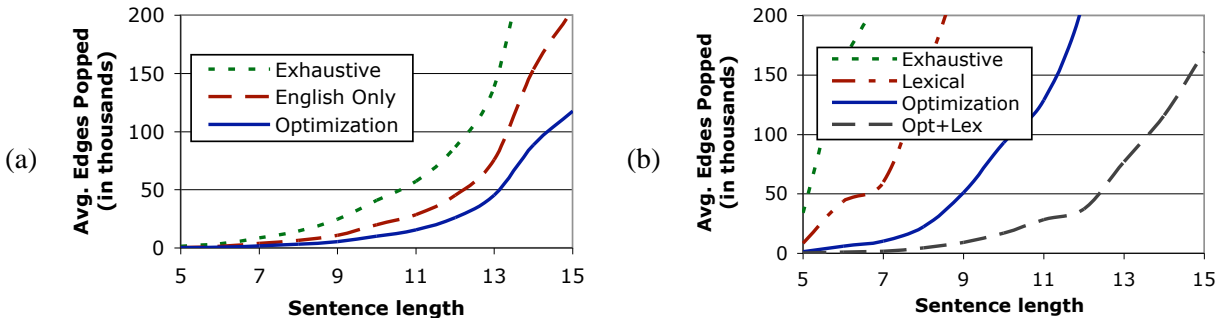
Figure 4: (a) Parsing efficiency results with optimization heuristics show that both component projections constrain the problem. (b) Including a lexical model and corresponding heuristic further increases parsing efficiency.

4 Lexicalized Parsing

We next apply our technique to lexicalized parsing (Charniak, 1997; Collins, 1999). In lexicalized parsing, the local configurations are lexicalized rules of the form $X[h,t] \rightarrow Y[h',t'] \, Z[h,t]$, where h, t, h', and t' are the head word, head tag, argument word, and argument tag, respectively. We will use $r = X \rightarrow YZ$ to refer to the CFG backbone of a lexicalized rule. As in Klein and Manning (2003), we view each lexicalized rule, ℓ, as having a CFG projection, $\pi_c(\ell) = r$, and a dependency projection, $\pi_d(\ell) = (h,t,h',t')$ (see figure 5).[9] Broadly, the CFG projection encodes constituency structure, while the dependency projection encodes lexical selection, and both projections are asymptotically more efficient than the original problem. Klein and Manning (2003) present a factored model where the CFG and dependency projections are generated independently (though with compatible bracketing):

$$P(Y[h,t]Z[h',t'] \mid X[h,t]) = \qquad (6)$$
$$P(YZ|X)P(h',t'|t,h)$$

In this work, we explore the following non-factored model, which allows correlations between the CFG and dependency projections:

$$P(Y[h,t]Z[h',t'] \mid X[h,t]) = P(YZ|X,t,h) \quad (7)$$
$$P(t'|t,Z,h',h) \, P(h'|t',t,Z,h',h)$$

This model is broadly representative of the successful lexicalized models of Charniak (1997) and

Collins (1999), though simpler.[10]

4.1 Choosing Constraints and Handling Unseen Dependencies

Ideally we would like to be able to solve the optimization problem in (4) for this task. Unfortunately, exhaustively listing all possible configurations (lexical rules) yields an impractical number of constraints. We therefore solve a relaxed problem in which we enforce the constraints for only a subset of the possible configurations, $\mathcal{A}' \subseteq \mathcal{A}$. Once we start dropping constraints, we can no longer guarantee pointwise admissibility, and therefore there is no reason not to also allow penalized violations of the constraints we do list, so we solve (5) instead.

To generate the set of enforced constraints, we first include all configurations observed in the gold training trees. We then sample novel configurations by choosing (X,h,t) from the training distribution and then using the model to generate the rest of the configuration. In our experiments, we ended up with 434,329 observed configurations, and sampled the same number of novel configurations. Our penalty multiplier C was 10.

Even if we supplement our training set with many sample configurations, we will still see new projected dependency configurations at test time. It is therefore necessary to generalize scores from training configurations to unseen ones. We enrich our procedure by expressing the projected configuration costs as linear functions of features. Specifically, we define feature vectors $f_c(r)$ and $f_d(h,t,h't')$ over the CFG and dependency projections, and intro-

---

[9]We assume information about the distance and direction of the dependency is encoded in the dependency tuple, but we omit it from the notation for compactness.

[10]All probability distributions for the non-factored model are estimated by Witten-Bell smoothing (Witten and Bell, 1991) where conditioning lexical items are backed off first.
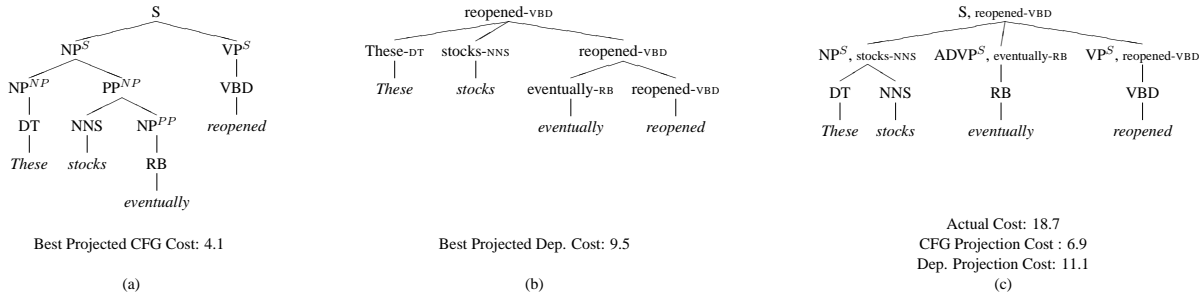
S

NP^S VP^S

NP^{NP} PP^{NP} VBD

DT NNS NP^{PP} *reopened*

These *stocks* RB

eventually

Best Projected CFG Cost: 4.1

(a)

reopened-VBD

These-DT stocks-NNS reopened-VBD

These *stocks* eventually-RB reopened-VBD

eventually *reopened*

Best Projected Dep. Cost: 9.5

(b)

S, reopened-VBD

NP^S, stocks-NNS ADVP^S, eventually-RB VP^S, reopened-VBD

DT NNS RB VBD

These *stocks* *eventually* *reopened*

Actual Cost: 18.7
CFG Projection Cost : 6.9
Dep. Projection Cost: 11.1

(c)

Figure 5: Lexicalized parsing projections. The figure in (a) is the optimal CFG projection solution and the figure in (b) is the optimal dependency projection solution. The tree in (c) is the optimal solution for the original problem. Note that the sum of the CFG and dependency projections is a lower bound (albeit a fairly tight one) on actual solution cost.

duce corresponding weight vectors w_c and w_d. The weight vectors are learned by solving the following optimization problem:

$$\operatorname*{minimize}_{\gamma, w_c, w_d} \|\gamma^+\|^2 + C\|\gamma^-\|^2 \qquad (8)$$

$$\text{such that,} \quad w_c \geq 0, w_d \geq 0$$

$$\gamma_\ell = c(\ell) - [w_c^T f_c(r) + w_d^T f_d(h, t, h', t')]$$

$$\text{for } \ell = (r, h, t, h', t') \in \mathcal{A}'$$

Our CFG feature vector has only indicator features for the specific rule. However, our dependency feature vector consists of an indicator feature of the tuple (h, t, h', t') (including direction), an indicator of the part-of-speech type (t, t') (also including direction), as well as a bias feature.

4.2 Experimental Results

We tested our approximate projection heuristic on two lexicalized parsing models. The first is the factored model of Klein and Manning (2003), given by equation (6), and the second is the non-factored model described in equation (7). Both models use the same parent-annotated head-binarized CFG backbone and a basic dependency projection which models direction, but not distance or valence.[11]

In each case, we compared A* using our approximate projection heuristics to exhaustive search. We measure efficiency in terms of the number of expanded hypotheses (edges popped); see figure 6.[12] In both settings, the factored A* approach substantially outperforms exhaustive search. For the fac-

---

[11]The CFG and dependency projections correspond to the PCFG-PA and DEP-BASIC settings in Klein and Manning (2003).

[12]All models are trained on section 2 through 21 of the English Penn treebank, and tested on section 23.

tored model of Klein and Manning (2003), we can also compare our reconstructed bound to the known tight bound which would result from solving the pointwise admissible problem in (4) with all constraints. As figure 6 shows, the exact factored heuristic does outperform our approximate factored heuristic, primarily because of many looser, backed-off cost estimates for unseen dependency tuples. For the non-factored model, we compared our approximate factored heuristic to one which only bounds the CFG projection as suggested by Klein and Manning (2003). They suggest,

$$\phi_c(r) = \min_{\ell \in \mathcal{A}: \pi_c(\ell) = r} c(\ell)$$

where we obtain factored CFG costs by minimizing over dependency projections. As figure 6 illustrates, this CFG only heuristic is substantially less efficient than our heuristic which bounds both projections.

Since our heuristic is no longer guaranteed to be admissible, we evaluated its effect on search in several ways. The first is to check for search errors, where the model-optimal parse is not found. In the case of the factored model, we can find the optimal parse using the exact factored heuristic and compare it to the parse found by our learned heuristic. In our test set, the approximate projection heuristic failed to return the model optimal parse in less than 1% of sentences. Of these search errors, none of the costs were more than 0.1% greater than the model optimal cost in negative log-likelihood. For the non-factored model, the model optimal parse is known only for shorter sentences which can be parsed exhaustively. For these sentences up to length 15, there were no search errors. We can also check for violations of pointwise admissibility for configurations encoun-
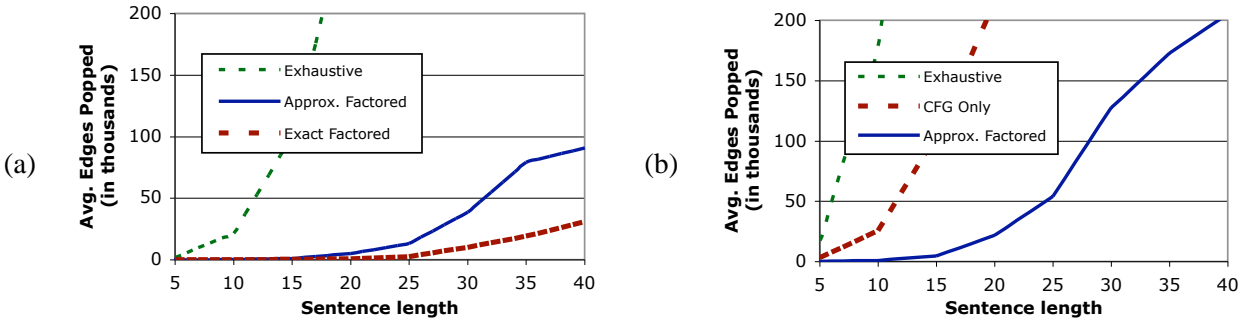
Figure 6: Edges popped by exhaustive versus factored A* search. The chart in (a) is using the factored lexicalized model from Klein and Manning (2003). The chart in (b) is using the non-factored lexicalized model described in section 4.

tered during search. For both the factored and non-factored model, less than 2% of the configurations scored by the approximate projection heuristic during search violated pointwise admissibility.

While this is a paper about inference, we also measured the accuracy in the standard way, on sentences of length up to 40, using EVALB. The factored model with the approximate projection heuristic achieves an F_1 of 82.2, matching the performance with the exact factored heuristic, though slower. The non-factored model, using the approximate projection heuristic, achieves an F_1 of 83.8 on the test set, which is slightly better than the factored model.[13] We note that the CFG and dependency projections are as similar as possible across models, so the increase in accuracy is likely due in part to the non-factored model's coupling of CFG and dependency projections.

5 Conclusion

We have presented a technique for creating A* estimates for inference in complex models. Our technique can be used to generate provably admissible estimates when all search transitions can be enumerated, and an effective heuristic even for problems where all transitions cannot be efficiently enumerated. In the future, we plan to investigate alternative objective functions and error-driven methods for learning heuristic bounds.

References

E. D. Andersen and K. D. Andersen. 2000. The MOSEK interior point optimizer for linear programming. In H. Frenk *et al.*, editor, *High Performance Optimization*. Kluwer Academic Publishers.

Stephen Boyd and Lieven Vandenberghe. 2005. *Convex Optimization*. Cambridge University Press.

Eugene Charniak. 1997. Statistical parsing with a context-free grammar and word statistics. In *National Conference on Artificial Intelligence*.

Michael Collins. 1999. Head-driven statistical models for natural language parsing.

Ariel Felner, Richard Korf, and Sarit Hanan. 2004. Additive pattern database heuristics. *JAIR*.

Michel Galley, Mark Hopkins, Kevin Knight, and Daniel Marcu. 2004. What's in a translation rule? In *HLT-NAACL*.

Malik Ghallab and Dennis G. Allard. 1982. A$^*_\epsilon$ - an efficient near admissible heuristic search algorithm. In *IJCAI*.

P. Hart, N. Nilsson, and B. Raphael. 1968. A formal basis for the heuristic determination of minimum cost paths. In *IEEE Transactions on Systems Science and Cybernetics*. IEEE.

Dan Klein and Christopher D. Manning. 2003. Factored A* search for models over sequences and trees. In *IJCAI*.

Kevin Knight and Jonathan Graehl. 2004. Training tree transducers. In *HLT-NAACL*.

I. Dan Melamed, Giorgio Satta, and Ben Wellington. 2004. Generalized multitext grammars. In *ACL*.

F. J. Och and H. Ney. 2000. Improved statistical alignment models. In *ACL*.

Ian H. Witten and Timothy C. Bell. 1991. The zero-frequency problem: Estimating the probabilities of novel events in adaptive text compression. *IEEE*.

Dekai Wu. 1997. Stochastic inversion transduction grammars and bilingual parsing of parallel corpora. *Comput. Linguist.*

Hao Zhang and Daniel Gildea. 2006. Efficient search for inversion transduction grammar. In *EMNLP*.

---

[13]Since we cannot exhaustively parse with this model, we cannot compare our F_1 to an exact search method.