Hierarchical A* Parsing with Bridge Outside Scores

Adam Pauls and **Dan Klein**
Computer Science Division
University of California at Berkeley
{adpauls,klein}@cs.berkeley.edu

Abstract

Hierarchical A* (HA*) uses of a hierarchy of coarse grammars to speed up parsing without sacrificing optimality. HA* prioritizes search in refined grammars using Viterbi outside costs computed in coarser grammars. We present Bridge Hierarchical A* (BHA*), a modified Hierarchial A* algorithm which computes a novel outside cost called a *bridge* outside cost. These bridge costs mix finer outside scores with coarser inside scores, and thus constitute tighter heuristics than entirely coarse scores. We show that BHA* substantially outperforms HA* when the hierarchy contains only very coarse grammars, while achieving comparable performance on more refined hierarchies.

1 Introduction

The Hierarchical A* (HA*) algorithm of Felzenszwalb and McAllester (2007) allows the use of a hierarchy of coarse grammars to speed up parsing without sacrificing optimality. Pauls and Klein (2009) showed that a hierarchy of coarse grammars outperforms standard A* parsing for a range of grammars. HA* operates by computing Viterbi inside and outside scores in an agenda-based way, using outside scores computed under coarse grammars as heuristics which guide the search in finer grammars. The outside scores computed by HA* are auxiliary quantities, useful only because they form admissible heuristics for search in finer grammars.

We show that a modification of the HA* algorithm can compute modified *bridge* outside scores which are tighter bounds on the true outside costs in finer grammars. These bridge outside scores mix inside and outside costs from finer grammars with inside costs from coarser grammars. Because the bridge costs represent tighter estimates of the true outside costs, we expect them to reduce the work of computing inside costs in finer grammars. At the same time, because bridge costs mix computation from coarser and finer levels of the hierarchy, they are more expensive to compute than purely coarse outside costs. Whether the work saved by using tighter estimates outweighs the extra computation needed to compute them is an empirical question.

In this paper, we show that the use of bridge outside costs substantially outperforms the HA* algorithm when the coarsest levels of the hierarchy are very loose approximations of the target grammar. For hierarchies with tighter estimates, we show that BHA* obtains comparable performance to HA*. In other words, BHA* is more robust to poorly constructed hierarchies.

2 Previous Work

In this section, we introduce notation and review HA*. Our presentation closely follows Pauls and Klein (2009), and we refer the reader to that work for a more detailed presentation.

2.1 Notation

Assume we have input sentence $s_0 \dots s_{n-1}$ of length n, and a hierarchy of m weighted context-free grammars $\mathcal{G}_1 \dots \mathcal{G}_m$. We call the most refined grammar $\mathcal{G}_m$ the *target* grammar, and all other (coarser) grammars *auxiliary* grammars. Each grammar $\mathcal{G}_t$ has a set of symbols denoted with capital letters and a subscript indicating the level in the hierarchy, including a distinguished goal (root) symbol G_t. Without loss of generality, we assume Chomsky normal form, so each non-terminal rule r in $\mathcal{G}_t$ has the form $r = A_t \to B_t\ C_t$ with weight w_r.

Edges are labeled spans $e = (A_t, i, j)$. The weight of a derivation is the sum of rule weights in the derivation. The weight of the best (minimum) inside derivation for an edge e is called the Viterbi inside score $\beta(e)$, and the weight of the

Proceedings of the ACL 2010 Conference Short Papers, pages 348–352,
Uppsala, Sweden, 11-16 July 2010. ©2010 Association for Computational Linguistics

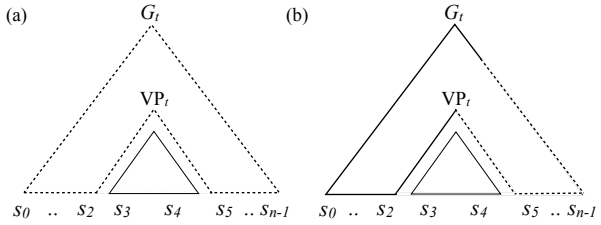

Figure 1: Representations of the different types of items used in parsing and how they depend on each other. (a) In HA*, the inside item $I(\text{VP}_t, 3, 5)$ relies on the coarse outside item $O(\pi_t(\text{VP}_t), 3, 5)$ for outside estimates. (b) In BHA*, the same inside item relies on the bridge outside item $\tilde{O}(\text{VP}_t, 3, 5)$, which mixes coarse and refined outside costs. The coarseness of an item is indicated with dotted lines.

best derivation of $G \rightarrow s_0 \ldots s_{i-1}\ A_t\ s_j \ldots s_{n-1}$ is called the Viterbi outside score $\alpha(e)$. The goal of a 1-best parsing algorithm is to compute the Viterbi inside score of the edge $(G_m, 0, n)$; the actual best parse can be reconstructed from backpointers in the standard way.

We assume that each auxiliary grammar $\mathcal{G}_{t-1}$ forms a *relaxed projection* of $\mathcal{G}_t$. A grammar $\mathcal{G}_{t-1}$ is a *projection* of $\mathcal{G}_t$ if there exists some many-to-one onto function π_t which maps each symbol in $\mathcal{G}_t$ to a symbol in $\mathcal{G}_{t-1}$; hereafter, we will use A'_t to represent $\pi_t(A_t)$. A projection is *relaxed* if, for every rule $r = A_t \rightarrow B_t\ C_t$ with weight w_r the projection $r' = A'_t \rightarrow B'_t\ C'_t$ has weight $w_{r'} \leq w_r$ in $\mathcal{G}_{t-1}$. In other words, the weight of r' is a lower bound on the weight of all rules r in $\mathcal{G}_t$ which project to r'.

2.2 Deduction Rules

HA* and our modification BHA* can be formulated in terms of prioritized weighted deduction rules (Shieber et al., 1995; Felzenszwalb and McAllester, 2007). A *prioritized weighted deduction rule* has the form

$$\phi_1 : w_1, \ldots, \phi_n : w_n \xrightarrow{p(w_1,\ldots,w_n)} \phi_0 : g(w_1, \ldots, w_n)$$

where $\phi_1, \ldots, \phi_n$ are the *antecedent items* of the deduction rule and ϕ_0 is the *conclusion item*. A deduction rule states that, given the antecedents $\phi_1, \ldots, \phi_n$ with weights $w_1, \ldots, w_n$, the conclusion ϕ_0 can be formed with weight $g(w_1, \ldots, w_n)$ and priority $p(w_1, \ldots, w_n)$.

These deduction rules are "executed" within a generic agenda-driven algorithm, which constructs items in a prioritized fashion. The algorithm maintains an *agenda* (a priority queue of items), as well as a *chart* of items already processed. The fundamental operation of the algorithm is to pop the highest priority item ϕ from the agenda, put it into the chart with its current weight, and form using deduction rules any items which can be built by combining ϕ with items already in the chart. If new or improved, resulting items are put on the agenda with priority given by $p(\cdot)$. Because all antecedents must be constructed before a deduction rule is executed, we sometimes refer to particular conclusion item as "waiting" on an other item(s) before it can be built.

2.3 HA*

HA* can be formulated in terms of two types of items. *Inside* items $I(A_t, i, j)$ represent possible derivations of the edge (A_t, i, j), while *outside* items $O(A_t, i, j)$ represent derivations of $G \rightarrow s_1 \ldots s_{i-1}\ A_t\ s_j \ldots s_n$ rooted at $(G_t, 0, n)$. See Figure 1(a) for a graphical depiction of these edges. Inside items are used to compute Viterbi inside scores under grammar $\mathcal{G}_t$, while outside items are used to compute Viterbi outside scores.

The deduction rules which construct inside and outside items are given in Table 1. The IN deduction rule combines two inside items over smaller spans with a grammar rule to form an inside item over larger spans. The weight of the resulting item is the sum of the weights of the smaller inside items and the grammar rule. However, the IN rule also requires that an outside score in the coarse grammar[1] be computed before an inside item is built. Once constructed, this coarse outside score is added to the weight of the conclusion item to form the priority of the resulting item. In other words, the coarse outside score computed by the algorithm plays the same role as a heuristic in standard A* parsing (Klein and Manning, 2003).

Outside scores are computed by the OUT-L and OUT-R deduction rules. These rules combine an outside item over a large span and inside items over smaller spans to form outside items over smaller spans. Unlike the IN deduction, the OUT deductions only involve items from the same level of the hierarchy. That is, whereas inside scores wait on coarse outside scores to be constructed, outside scores wait on inside scores at the same level in the hierarchy.

Conceptually, these deduction rules operate by

[1]For the coarsest grammar $\mathcal{G}_1$, the IN rule builds rules using 0 as an outside score.

HA*

IN:	$I(B_t,i,l):w_1$	$I(C_t,l,j):w_2$	$O(A'_t,i,j):w_3$	$\xrightarrow{w_1+w_2+w_r+\underline{w_3}}$	$I(A_t,i,j)$	:	$w_1+w_2+w_r$
OUT-L:	$O(A_t,i,j):w_1$	$I(B_t,i,l):w_2$	$I(C_t,l,j):w_3$	$\xrightarrow{w_1+w_3+w_r+w_2}$	$O(B_t,i,l)$	:	$w_1+w_3+w_r$
OUT-R:	$O(A_t,i,j):w_1$	$I(B_t,i,l):w_2$	$I(C_t,l,j):w_3$	$\xrightarrow{w_1+w_2+w_r+w_3}$	$O(C_t,l,j)$	:	$w_1+w_2+w_r$

Table 1: HA* deduction rules. Red underline indicates items constructed under the previous grammar in the hierarchy.

BHA*

B-IN:	$I(B_t,i,l):w_1$	$I(C_t,l,j):w_2$	$\tilde{O}(A_t,i,j):w_3$	$\xrightarrow{w_1+w_2+w_r+w_3}$	$I(A_t,i,j)$	:	$w_1+w_2+w_r$
B-OUT-L:	$\tilde{O}(A_t,i,j):w_1$	$\underline{I(B'_t,i,l):w_2}$	$\underline{I(C'_t,l,j):w_3}$	$\xrightarrow{w_1+w_r+\underline{w_2+w_3}}$	$\tilde{O}(B_t,i,l)$	:	$w_1+w_r+\underline{w_3}$
B-OUT-R:	$\tilde{O}(A_t,i,j):w_1$	$I(B_t,i,l):w_2$	$\underline{I(C'_t,l,j):w_3}$	$\xrightarrow{w_1+w_2+w_r+\underline{w_3}}$	$\tilde{O}(C_t,l,j)$	:	$w_1+w_2+w_r$

Table 2: BHA* deduction rules. Red underline indicates items constructed under the previous grammar in the hierarchy.

first computing inside scores bottom-up in the coarsest grammar, then outside scores top-down in the same grammar, then inside scores in the next finest grammar, and so on. However, the crucial aspect of HA* is that items from all levels of the hierarchy compete on the same queue, interleaving the computation of inside and outside scores at all levels. The HA* deduction rules come with three important guarantees. The first is a *monotonicity* guarantee: each item is popped off the agenda in order of its *intrinsic priority* $\hat{p}(\cdot)$. For inside items $I(e)$ over edge e, this priority $\hat{p}(I(e)) = \beta(e) + \alpha(e')$ where e' is the projection of e. For outside items $O(\cdot)$ over edge e, this priority is $\hat{p}(O(e)) = \beta(e) + \alpha(e)$.

The second is a *correctness* guarantee: when an inside/outside item is popped of the agenda, its weight is its true Viterbi inside/outside cost. Taken together, these two imply an *efficiency* guarantee, which states that only items x whose intrinsic priority $\hat{p}(x)$ is less than or equal to the Viterbi inside score of the goal are removed from the agenda.

2.4 HA* with Bridge Costs

The outside scores computed by HA* are useful for prioritizing computation in more refined grammars. The key property of these scores is that they form consistent and admissible heuristic costs for more refined grammars, but coarse outside costs are not the only quantity which satisfy this requirement. As an alternative, we propose a novel "bridge" outside cost $\tilde{\alpha}(e)$. Intuitively, this cost represents the cost of the best derivation where rules "above" and "left" of an edge e come from $\mathcal{G}_t$, and rules "below" and "right" of the e come from $\mathcal{G}_{t-1}$; see Figure 2 for a graphical depiction. More formally, let the *spine* of an edge $e = (A_t, i, j)$ for some derivation d be

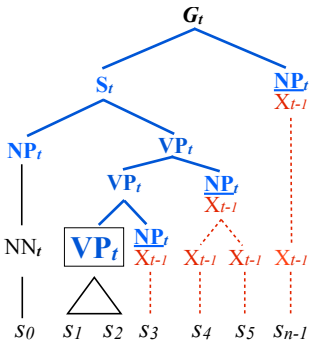

Figure 2: A concrete example of a possible bridge outside derivation for the bridge item $\tilde{O}(\text{VP}_t, 1, 4)$. This edge is boxed for emphasis. The spine of the derivation is shown in bold and colored in blue. Rules from a coarser grammar are shown with dotted lines, and colored in red. Here we have the simple projection $\pi_t(A) = X, \forall A$.

the sequence of rules between e and the root edge $(G_t, 0, n)$. A *bridge outside derivation* of e is a derivation d of $G \rightarrow s_1 \ldots s_i \; A_t \; s_{j+1} \ldots s_n$ such that every rule on or left of the spine comes from $\mathcal{G}_t$, and all other rules come from $\mathcal{G}_{t-1}$. The score of the best such derivation for e is the bridge outside cost $\tilde{\alpha}(e)$.

Like ordinary outside costs, bridge outside costs form consistent and admissible estimates of the true Viterbi outside score $\alpha(e)$ of an edge e. Because bridge costs mix rules from the finer and coarser grammar, bridge costs are at least as good an estimate of the true outside score as entirely coarse outside costs, and will in general be much tighter. That is, we have

$$\alpha(e') \leq \tilde{\alpha}(e) \leq \alpha(e)$$

In particular, note that the bridge costs become better approximations farther right in the sentence, and the bridge cost of the last word in the sentence is equal to the Viterbi outside cost of that word.

To compute bridge outside costs, we introduce

bridge outside items $\tilde{O}(A_t, i, j)$, shown graphically in Figure 1(b). The deduction rules which build both inside items and bridge outside items are shown in Table 2. The rules are very similar to those which define HA*, but there are two important differences. First, inside items wait for bridge outside items *at the same level*, while outside items wait for inside items from the previous level. Second, the left and right outside deductions are no longer symmetric – bridge outside items can extended to the left given two coarse inside items, but can only be extended to the right given an exact inside item on the left and coarse inside item on the right.

2.5 Guarantees

These deduction rules come with guarantees analogous to those of HA*. The monotonicity guarantee ensures that inside and (bridge) outside items are processed in order of:

$$\hat{p}(I(e)) = \beta(e) + \tilde{\alpha}(e)$$
$$\hat{p}(\tilde{O}(e)) = \tilde{\alpha}(e) + \beta(e')$$

The correctness guarantee ensures that when an item is removed from the agenda, its weight will be equal to $\beta(e)$ for inside items and $\tilde{\alpha}(e)$ for bridge items. The efficiency guarantee remains the same, though because the intrinsic priorities are different, the set of items processed will be different from those processed by HA*.

A proof of these guarantees is not possible due to space restrictions. The proof for BHA* follows the proof for HA* in Felzenszwalb and McAllester (2007) with minor modifications. The key property of HA* needed for these proofs is that coarse outside costs form consistent and admissible heuristics for inside items, and exact inside costs form consistent and admissible heuristics for outside items. BHA* also has this property, with bridge outside costs forming admissible and consistent heuristics for inside items, and coarse inside costs forming admissible and consistent heuristics for outside items.

3 Experiments

The performance of BHA* is determined by the efficiency guarantee given in the previous section. However, we cannot determine in advance whether BHA* will be faster than HA*. In fact, BHA* has the potential to be slower – BHA* builds both inside and bridge outside items under the target grammar, where HA* only builds inside items. It is an empirical, grammar- and hierarchy-dependent question whether the increased tightness of the outside estimates outweighs the additional cost needed to compute them. We demonstrate empirically in this section that for hierarchies with very loosely approximating coarse grammars, BHA* can outperform HA*, while for hierarchies with good approximations, performance of the two algorithms is comparable.

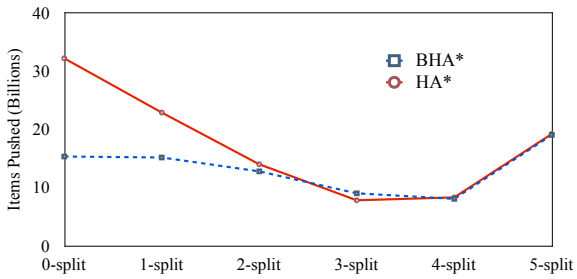

Figure 3: Performance of HA* and BHA* as a function of increasing refinement of the coarse grammar. Lower is faster.

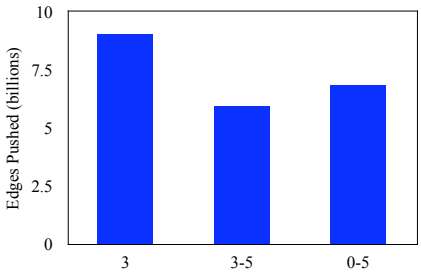

Figure 4: Performance of BHA* on hierarchies of varying size. Lower is faster. Along the x-axis, we show which coarse grammars were used in the hierarchy. For example, 3-5 indicates the 3-,4-, and 5-split grammars were used as coarse grammars.

We performed experiments with the grammars of Petrov et al. (2006). The training procedure for these grammars produces a hierarchy of increasingly refined grammars through state-splitting, so a natural projection function π_t is given. We used the Berkeley Parser[2] to learn such grammars from Sections 2-21 of the Penn Treebank (Marcus et al., 1993). We trained with 6 split-merge cycles, producing 7 grammars. We tested these grammars on 300 sentences of length ≤ 25 of Section 23 of the Treebank. Our "target grammar" was in all cases the most split grammar.

[2]http://berkeleyparser.googlecode.com

In our first experiment, we construct 2-level hierarchies consisting of one coarse grammar and the target grammar. By varying the coarse grammar from the 0-split (X-bar) through 5-split grammars, we can investigate the performance of each algorithm as a function of the coarseness of the coarse grammar. We follow Pauls and Klein (2009) in using the number of items pushed as a machine- and implementation-independent measure of speed. In Figure 3, we show the performance of HA* and BHA* as a function of the total number of items pushed onto the agenda. We see that for very coarse approximating grammars, BHA* substantially outperforms HA*, but for more refined approximating grammars the performance is comparable, with HA* slightly outperforming BHA* on the 3-split grammar.

Finally, we verify that BHA* can benefit from multi-level hierarchies as HA* can. We constructed two multi-level hierarchies: a 4-level hierarchy consisting of the 3-,4-,5-, and 6- split grammars, and 7-level hierarchy consisting of all grammars. In Figure 4, we show the performance of BHA* on these multi-level hierarchies, as well as the best 2-level hierarchy from the previous experiment. Our results echo the results of Pauls and Klein (2009): although the addition of the reasonably refined 4- and 5-split grammars produces modest performance gains, the addition of coarser grammars can actually hurt overall performance.

Acknowledgements

This project is funded in part by the NSF under grant 0643742 and an NSERC Postgraduate Fellowship.

References

P. Felzenszwalb and D. McAllester. 2007. The generalized A* architecture. *Journal of Artificial Intelligence Research*.

Dan Klein and Christopher D. Manning. 2003. A* parsing: Fast exact Viterbi parse selection. In *Proceedings of the Human Language Technology Conference and the North American Association for Computational Linguistics (HLT-NAACL)*, pages 119–126.

M. Marcus, B. Santorini, and M. Marcinkiewicz. 1993. Building a large annotated corpus of English: The Penn Treebank. In *Computational Linguistics*.

Adam Pauls and Dan Klein. 2009. Hierarchical search for parsing. In *Proceedings of The Annual Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*.

Slav Petrov, Leon Barrett, Romain Thibaux, and Dan Klein. 2006. Learning accurate, compact, and interpretable tree annotation. In *Proccedings of the Association for Computational Linguistics (ACL)*.

Stuart M. Shieber, Yves Schabes, and Fernando C. N. Pereira. 1995. Principles and implementation of deductive parsing. *Journal of Logic Programming*, 24:3–36.