Linear Complexity Context-Free Parsing Pipelines via Chart Constraints

Brian Roark and Kristy Hollingshead
Center for Spoken Language Understanding
Division of Biomedical Computer Science
Oregon Health & Science University
{roark,hollingk}@cslu.ogi.edu

Abstract

In this paper, we extend methods from Roark and Hollingshead (2008) for reducing the worst-case complexity of a context-free parsing pipeline via hard constraints derived from finite-state tagging pre-processing. Methods from our previous paper achieved quadratic worst-case complexity. We prove here that alternate methods for choosing constraints can achieve either linear or $O(N\log^2 N)$ complexity. These worst-case bounds on processing are demonstrated to be achieved without reducing the parsing accuracy, in fact in some cases improving the accuracy. The new methods achieve observed performance comparable to the previously published quadratic complexity method. Finally, we demonstrate improved performance by combining complexity bounding methods with additional high precision constraints.

1 Introduction

Finite-state pre-processing for context-free parsing is very common as a means of reducing the amount of search required in the later stage. For example, the well-known Ratnaparkhi parser (Ratnaparkhi, 1999) used a finite-state POS-tagger and NP-chunker to reduce the search space for his Maximum Entropy parsing model, and achieved linear observed-time performance. Other recent examples of the utility of finite-state constraints for parsing pipelines include Glaysher and Moldovan (2006), Djordjevic et al. (2007), Hollingshead and Roark (2007), and Roark and Hollingshead (2008). Note that by making use of constraints derived from pre-processing, they are no longer performing full exact inference—these are approximate inference methods, as are the methods presented in this paper. Most of these parsing pipeline papers show empirically that these techniques can improve pipeline efficiency for well-known parsing tasks. In contrast, in Roark and Hollingshead (2008), we derived and applied the finite-state constraints so as to *guarantee* a reduction in the worst-case complexity of the context-free parsing pipeline from $O(N^3)$ in the length of the string N to $O(N^2)$ by closing chart cells to entries. We demonstrated the application of such constraints to the well-known Charniak parsing pipeline (Charniak, 2000), which resulted in no accuracy loss when the constraints were applied.

While it is important to demonstrate that these sorts of complexity-reducing chart constraints do not interfere with the operation of high-accuracy, state-of-the-art parsing approaches, existing pruning techniques used within such parsers can obscure the impact of these constraints on search. For example, using the default search parameterization of the Charniak parser, the Roark and Hollingshead (2008) results demonstrated no parser speedup using the techniques, rather an *accuracy* improvement, which we attributed to a better use of the amount of search permitted by that default parameterization. We only demonstrated efficiency improvements by reducing the amount of search via the Charniak search parameterization. There we showed a nice speedup of the parser versus the default, while maintaining accuracy levels. However, internal heuristics of the Charniak search, such as attention shifting (Blaheta and Charniak, 1999; Hall and Johnson, 2004), can make this accuracy/efficiency tradeoff somewhat difficult to interpret.

Furthermore, one might ask whether $O(N^2)$ complexity is as good as can be achieved through the paradigm of using finite-state constraints to close chart cells. What methods of constraint would be required to achieve $O(N \log N)$ or linear complexity? Would such constraints degrade performance, or can the finite-state models be applied with sufficient precision to allow for such constraints without significant loss of accuracy?

In this paper, we adopt the same paradigm pursued in Roark and Hollingshead (2008), but apply it to an exact inference CYK parser (Cocke and Schwartz, 1970; Younger, 1967; Kasami, 1965). We demonstrate that imposing constraints sufficient to achieve quadratic complexity in fact yields observed linear parsing time, suggesting that tighter complexity bounds are possible. We prove that a different method of imposing constraints on words beginning or ending multi-word constituents can give $O(N\log^2 N)$ or $O(N)$ worst-case complexity, and we empirically evaluate the impact of such an approach.

The rest of the paper is structured as follows. We begin with a summary of the chart cell constraint techniques from Roark and Hollingshead (2008), and some initial empirical trials applying these techniques to an exact inference CYK parser. Complexity bounding approaches are contrasted (and combined) with high precision constraint selection methods from that paper. We then present a new approach to making use of the same sort of finite-state tagger output to achieve linear or $N\log^2 N$ complexity. This is followed with an empirical validation of the new approach.

2 Background: Chart Cell Constraints

The basic algorithm from Roark and Hollingshead (2008) is as follows. Let B be the set of words in a string $w_1 \ldots w_k$ that begin a multi-word constituent, and let E be the set of words in the string that end a multi-word constituent. For chart parsing with, say, the CYK algorithm, cells in the chart represent substrings $w_i \ldots w_j$ of the string, and can be indexed with (i, j), the beginning and ending words of the substring. If $w_i \notin B$, then we can *close* any cell (i, j) where $i < j$, i.e., no complete constituents need be stored in that cell. Similarly, if $w_j \notin E$, then we can close any cell (i, j) where $i < j$. A discriminatively trained finite-state tagger can be used to classify words as being in or out of these sets with relatively high tagging accuracy, around 97% for both sets (B and E). The output of the tagger is then used to close cells, thus reducing the work for the chart parser.

An important caveat must be made about these closed cells, related to *incomplete* constituents. For simplicity of exposition, we will describe incomplete constituents in terms of factored categories in a Chomsky Normal Form grammar, e.g., the new non-terminal $Z{:}X{+}W$ that results when the ternary rule production $Z \rightarrow Y \;\; X \;\; W$ is factored into the two binary productions $Z \rightarrow Y \;\; Z{:}X{+}W$ and $Z{:}X{+}W \rightarrow X \;\; W$. A factored category such as $Z{:}X{+}W$ should be permitted in cell (i, j) if $w_j \in E$, even if $w_i \notin B$, because the category could subsequently combine with an Y category to create a Z constituent that begins at some word $w_p \in B$. Hence there are three possible conditions for cell (i, j) in the chart:

1. $w_j \notin E$: closing the cell affects *all* constituents, both complete and incomplete
2. $w_i \notin B$ **and** $w_j \in E$: closing the cell affects only complete constituents
3. $w_i \in B$ and $w_j \in E$: cell is not closed, i.e., it is "open"

In Roark and Hollingshead (2008), we proved that, for the CYK algorithm, there is no work necessary for case 1 cells, a constant amount of work for case 2 cells, and a linear amount of work for case 3 cells. Therefore, if the number of cells allowed to fall in case 3 is linear, the overall complexity of search is $O(N^2)$.

The amount of work for each case is related to how the CYK algorithm performs its search. Each cell in the chart (i, j) represents a substring $w_i \ldots w_j$, and building non-terminal categories in that cell involves combining non-terminal categories (via rules in the context-free grammar) found in cells of adjacent substrings $w_i \ldots w_m$ and $w_{m+1} \ldots w_j$. The length of substrings can be up to order N (length of the whole string), hence there are $O(N)$ *midpoint* words w_m in the standard algorithm, and in the case 3 cells above. This accounts for the linear amount of work for those cells. Case 2 cells have constant work because there is only one possible midpoint, and that is w_i, i.e., the first child of any incomplete constituent placed in a case 2 cell must be span 1, since $w_i \notin B$. This is a very concise recap of the proof, and we refer the reader to our previous paper for more details.

3 Constraining Exact-Inference CYK

Despite referring to the CYK algorithm in the proof, in Roark and Hollingshead (2008) we demonstrated our approach by constraining the Charniak parser (Charniak, 2000), and achieved an improvement in the accuracy/efficiency tradeoff curve. However, as mentioned earlier, the existing complicated system of search heuristics in the Charniak parser makes interpretation of the results more difficult. What can be said from the previous results is that constraining parsers in this way can improve performance of even the highest accuracy parsers. Yet those results do not provide much of an indication of how performance is impacted for general context-free inference.

For this paper, we use an exact inference (exhaustive search) CYK parser, using a simple probabilistic context-free grammar (PCFG) induced from the Penn WSJ Treebank (Marcus et al., 1993). The PCFG is transformed to Chomsky Normal Form through right-factorization, and is smoothed with a Markov (order-2) transform. Thus a production such as $Z \rightarrow Y \;\; X \;\; W \;\; V$ becomes three rules: (1) $Z \rightarrow Y \;\; Z{:}X{+}W$; (2) $Z{:}X{+}W \rightarrow X \;\; Z{:}W{+}V$; and (3) $Z{:}W{+}V \rightarrow W \;\; V$. Note that only two child categories are encoded within the new factored categories, instead of all of the remaining children as in our previous factorization example. This so-called 'Markov' grammar provides some smoothing of the PCFG; the resulting grammar is also smoothed using lower order Markov grammars.

We trained on sections 2-21 of the treebank, and all results except for the final table are on the development section (24). The final table is on the test section (23). All results report F-measure labeled bracketing accuracy for all sentences in the section.

To close cells, we use a discriminatively trained finite-state tagger to tag words as being either in B or not, and also (in a separate pass) either in E or not. Note that the reference tags for each word can be derived directly from the treebank, based on the spans of constituents beginning (or ending) at each word. Note also that these reference tags are based on a non-factored grammar.

For example, consider the chart in Figure 1 for the five symbol string "$abcde$". Each cell in the chart is labeled with the substring that the cell spans, along with the begin and end indices of the substring, e.g., $(3,5)$ spans the third symbol to the fifth symbol:

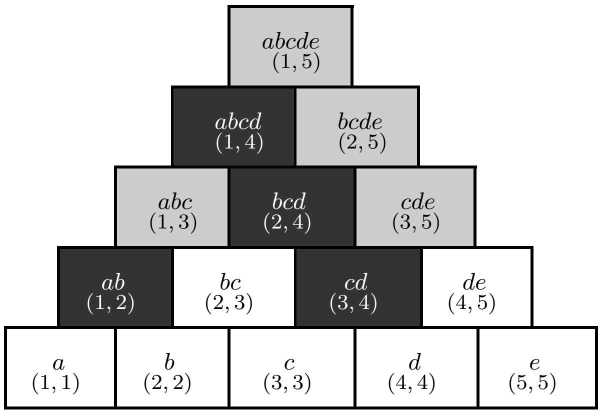


Figure 1: Fragment of a chart structure. Each cell is labeled with the substring spanned by that cell, along with the start and end word indices. Cell shading reflects $b \notin E$ and $d \notin E$ constraints: black denotes "closed" cells; white and gray are "open"; gray cells have "closed" children cells, reducing the number of midpoints requiring processing.

cde. If our tagger output is such that $b \notin E$ and $d \notin E$, then four cells will be closed: $(1,2)$, $(1,4)$, $(2,4)$ and $(3,4)$. The gray shaded cells in the figure have some midpoints that require no work, because they involve closed children cells.

4 Constraint Selection

4.1 High Precision vs Complexity Bounding

The chart constraints that are extracted from the finite-state tagger come in the form of set exclusions, e.g., $d \notin E$. Rather than selecting constraints from the single, best-scoring tag sequence output by the tagger, we instead rely on the whole distribution over possible tag strings to select constraints. We have two separate tagging tasks, each with two possible tags of each word w_i in each string: (1) B or ¬B; and (2) E or ¬E, where ¬X signifies that $w_i \notin X$ for $X \in \{B, E\}$. The tagger (Hollingshead et al., 2005) uses log linear models trained with the perceptron algorithm, and derives, via the forward-backward algorithm, the posterior probability of each of the two tags at each word, so that $\mathrm{Pr(B)} + \mathrm{Pr(\neg B)} = 1$. Then, for every word w_i in the string, the tags B and E are associated with a posterior probability that gives us a score for $w_i \in B$ and $w_i \in E$. All possible set memberships $w_i \in X$ in the string can be ranked by this score. From this ranking, a decision boundary can be set, such that all word/set pairs $w_i \in B$ or $w_j \in E$ with above-threshold probability are accepted, and all pairs below threshold are excluded from the set.

The default decision boundary for this tagging task is 0.5 posterior probability (more likely than not), and tagging performance at that threshold is good (around 97% accuracy, as mentioned previously). However, since this is a pre-processing step, we may want to reduce possible cascading errors by allowing more words into the sets B and E. In other words, we may want more precision in our set exclusion constraints. One method for this is to count the number c of word/set pairs below posterior probability of 0.5, then set the threshold so that only kc word/set pairs fall below threshold, where $0 < k \leq 1$. Note that the closer the parameter k is to 0, the fewer constraints will be applied to the chart. We refer to the resulting constraints as "high precision", since the selected constraints (set exclusions) have high precision. This technique was also used in the previous paper.

We also make use of the ranked list of word/set pairs to impose quadratic bounds on context-free parsing. Starting from the top of the list (highest posterior probability for set inclusion), word/set pairs are selected and the number of open cells (case 3 in Section 2) calculated. When the accumulated number of open cells reaches kN for sentence length N, the decision threshold is set. In such a way, there are only a linear number of open, case 3 cells, hence the parsing has quadratic worst-case complexity.

For both of these methods, the parameter k can vary, allowing for more or less set inclusion. Figure 2 shows parse time versus F-measure parse accuracy on the development set for the baseline (unconstrained) exact-inference CYK parser, and for various parameterizations of both the high precision constraints and the quadratic bound constraints. Note that accuracy actually improves with the imposition of these constraints. This is not surprising, since the finite-state tagger deriving the constraints made use of lexical information that the simple PCFG did not, hence there is complementary information improving the model. The best operating points—fast parsing and relatively high accuracy—are achieved with 90% of the high precision constraints, and $5N$ cells left open. These achieve a roughly 20 times speedup over the baseline unconstrained parser and achieve between 1.5 and 3 percent accuracy gains over the baseline.

We can get a better picture of what is going on by considering the scatter plots in Figure 3, which plot

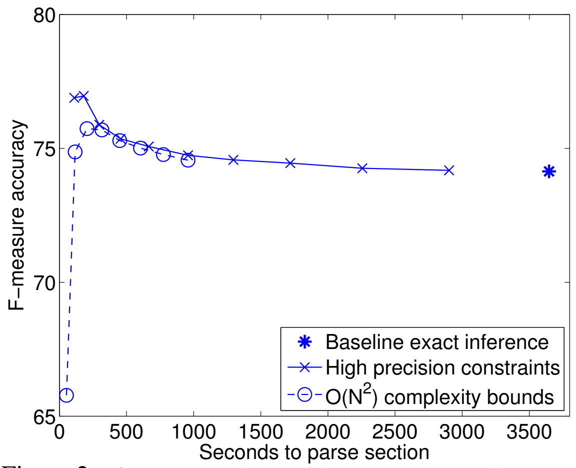
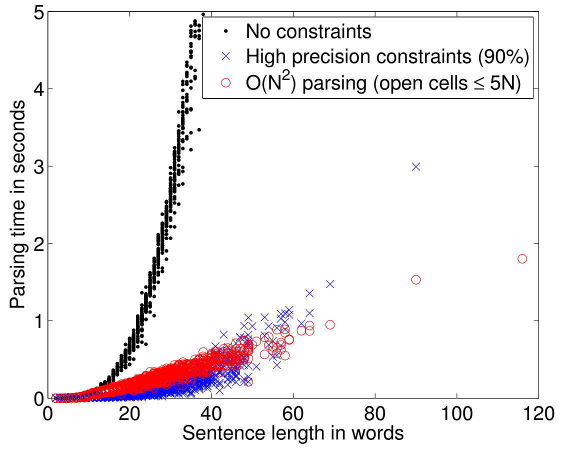
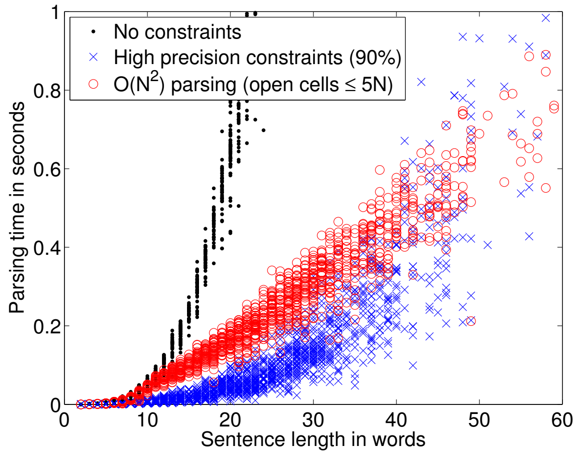


Figure 2: Time to parse (seconds) versus accuracy (F-measure) for the baseline of exact inference (no constraints) versus two methods of imposing constraints with varying parameters: (1) High precision constraints; (2) Sufficient constraints to impose $O(N^2)$ complexity (the number of open cells $\leq kN$).

each sentence according to its length versus the parsing time for that sentence at three operating points: baseline (unconstrained); high precision at 90%; and quadratic with $5N$ open cells. The top plot shows up to 120 words in the sentence, and up to 5 seconds of parsing time. The middle graph zooms in to under 1 second and up to 60 words; and the lowest graph zooms in further to under 0.1 seconds and up to 20 words. It can be seen in each graph that the unconstrained CYK parsing quickly leaves the graph via a steep cubic curve.

Three points can be taken away from these plots. First, the high precision constraints are better for the shorter strings than the quadratic bound constraints (see bottom plot); yet with the longer strings, the quadratic constraints better control parsing time than the high precision constraints (see top plot). Second, the quadratic bound constraints appear to actually result in roughly linear parsing time, not quadratic. Finally, at the "crossover" point, where quadratic constraints start out-performing the high precision constraints (roughly 40-60 words, see middle plot), there is quite high variance in high precision constraints versus the quadratic bounds: some sentences process more quickly than the quadratic bounds, some quite a bit worse. This illustrates the difference between the two methods of selecting constraints: the high precision constraints can provide very strong gains, but there is no guarantee for the worst case. In such a way, the high precision constraints are similar to other tagging-derived





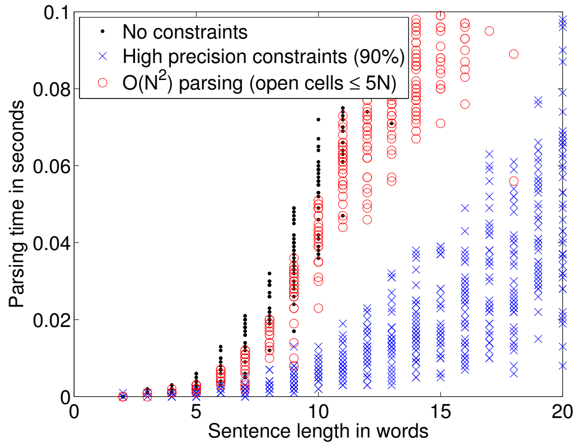


Figure 3: Scatter plots of sentence length versus parsing time for (1) baseline exact inference (no constraints); (2) high precision begin- and end-constituent constraints (90% level); and (3) $O(N^2)$ constraints (5N open cells).

constraints like POS-tags or chunks.

4.2 Combining Constraints

Depending on the length of the string, the quadratic constraints may close more or fewer chart cells than the high precision constraints—more for long strings, fewer for short strings. We can achieve

Constraints	F-measure accuracy	time (seconds)
None (baseline CYK)	74.1	3646
High Precision (90%)	77.0	181
Quadratic ($5N$)	75.7	317
Quad ($5N$) + HiPrec (90%)	76.9	166

Table 1: Speed and accuracy of exact-inference CYK parser on WSJ section 24 under various constraint conditions, including combining quadratic bound constraints and high precision constraints.

worst-case bounds, along with superior typical case speedups, by combining both methods as follows: first apply the quadratic bounds; then, if there are any high precision constraints that remain unapplied, add them. Table 1 shows F-measure accuracy and parsing time (in seconds) for four trials on the development set: the baseline CYK with no constraints; high precision constraints at the 90% level; quadratic bound constraints at the $5N$ level; and a combination of the quadratic bound and high precision constraints. We can see that, indeed, the combination of the two yield speedups over both independently, with no significant drop in accuracy from the high precision constraints alone. Further results with worst-case complexity bounds will be combined with high precision constraints in this way.

The observed linear parsing time in Figure 3 with the quadratic constraints raises the following question: can we apply these constraints in a way that guarantees linear complexity? The answer is yes, and this is the subject of the next section.

5 Linear and $N\log^2 N$ Complexity Bounds

Given the two sets B and E, recall the three cases of chart cells (i, j) presented in Section 2: 1) $w_j \notin E$ (cell completely closed); 2) $w_j \in E$ and $w_i \notin B$ (cell open only for incomplete constituents); and 3) $w_i \in B$ and $w_j \in E$ (cell open for all constituents). Quadratic worst-case complexity is achieved with these sets by limiting case 3 to hold for only $O(N)$ cells—each with linear work—and the remaining $O(N^2)$ cells (cases 1 and 2) have none or constant work, hence overall quadratic (Roark and Hollingshead, 2008).

One might ask: why would imposing constraints to achieve a quadratic bound give us linear observed parsing time? One possibility is that the linear number of case 3 cells don't have a linear amount of work, but rather a constant bounded amount of work.

If there were a constant bounded number of midpoints, then the amount of work associated with case 3 would be linear. Note that a linear complexity bound would have to guarantee a linear number of case 2 cells as well since there is a constant amount of work associated with case 2 cells.

To provide some intuition as to why the quadratic bound method resulted in linear observed parsing time, consider again the chart structure in Figure 1. The black cells in the chart represent the cells that have been closed when $w_j \notin E$ (case 1 cells). In our example, $w_2 \notin E$ caused the cell spanning *ab* to be closed, and $w_4 \notin E$ caused the cells spanning *abcd*, *bcd* and *cd* to be closed. Since there is no work required for these cells, the amount of work required to parse the sentence is reduced. However, the quadratic bound does not include any potential reduced work in the remaining open cells. The gray cells in the chart are cells with a reduced number of possible midpoints, as effected by the closed cells in the chart. For example, categories populating the cell spanning *abc* in position $(1, 3)$ can be built in two ways: either by combining entries in cell $(1, 1)$ with entries in $(2, 3)$ at midpoint $m = 1$; or by combining entries in $(1, 2)$ and $(3, 3)$ at midpoint $m = 2$. However, cell $(1, 2)$ is closed, hence there is only one midpoint at which $(1, 3)$ can be built ($m = 1$). Thus the amount of work to parse the sentence will be less than the worst-case quadratic bound based on this processing savings in open cells.

While imposition of the quadratic bound may have resulted (fortuitously) in constant bounded work for case 3 cells and a linear number of case 2 cells, there is no guarantee that this will be the case. One method to guarantee that both conditions are met is the following: if $|E| \leq k$ for some constant k, then both conditions will be met and parsing complexity will be linear. We prove here that constraining E to contain a constant number of words results in linear complexity.

Lemma 1: *If $|E| \leq k$ for some k, then the amount of work for any cell is bounded by ck for some constant c (grammar constant).*

Proof: Recall from Section 2 that for each cell (i, j), there are $j-i$ midpoints m that require combining entries in cells (i, m) and $(m+1, j)$ to create entries in cell (i, j). If $m > i$, then cell (i, m) is empty unless $w_m \in E$. If cell (i, m) is empty, there is no work to be done at that midpoint. If $|E| \leq k$, then there are a maximum of k midpoints for any cell, hence the amount of work is bounded by *ck* for some constant c.□

Lemma 2: *If $|E| \leq k$ for some k, then the number of cells (i, j) such that $w_j \in E$ is no more than kN where N is the length of the string.*

Proof: For a string of length N, each word w_j in the string has at most N cells such that w_j is the last word in the substring spanned by that cell, since each such cell must begin with a distinct word w_i in the string where $i \leq j$, of which there are at most N. Therefore, if $|E| \leq k$ for some k, then the number of cells (i, j) such that $w_j \in E$ would be no more than kN.□

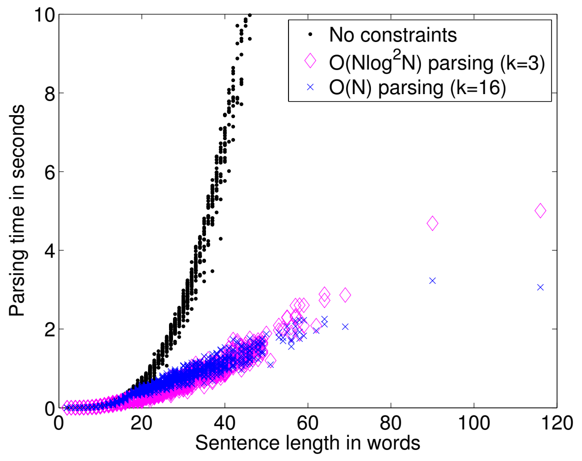
Theorem: *If $|E| \leq k$, then the parsing complexity is $O(k^2N)$.*

Proof: As stated earlier, each cell (i, j) falls in one of three cases: 1) $w_j \notin E$; 2) $w_j \in E$ and $w_i \notin B$; and 3) $w_i \in B$ and $w_j \in E$. Case 1 cells are completely closed, there is no work to be done in those cells. By Lemma 2, there are at maximum kN cells that fall in either case 2 or case 3. By Lemma 1, the amount of work for each of these cells is bounded by *ck* for some constant c. Therefore, the theorem is proved.□

If $|E| \leq k$ for a constant k, the theorem proves the complexity will be $O(N)$. If $|E| \leq k \log N$, then parsing complexity will be $O(N\log^2 N)$. Figure 4 shows sentence length versus parsing time under three different conditions[1]: baseline (unconstrained); $O(N\log^2 N)$ at $|E| \leq 3 \log N$; and linear at $|E| \leq 16$. The bottom graph zooms in to demonstrate that the $O(N\log^2 N)$ constraints can outperform the linear constraints for shorter strings (see around 20 words). As the length of the string increases, though, the performance lines cross, and the linear constraints demonstrate higher efficiency for the longer strings, as expected.

Unlike the method for imposing quadratic bounds, this method only makes use of set E, not B. To select the constraints, we rank the word/E posterior probabilities, and choose the top k (either constant or scaled with a $\log N$ factor); the rest of the words fall outside of the set. In this approach,

[1]Selection of these particular operating points for the $N\log^2 N$ and linear methods is discussed in Section 6.



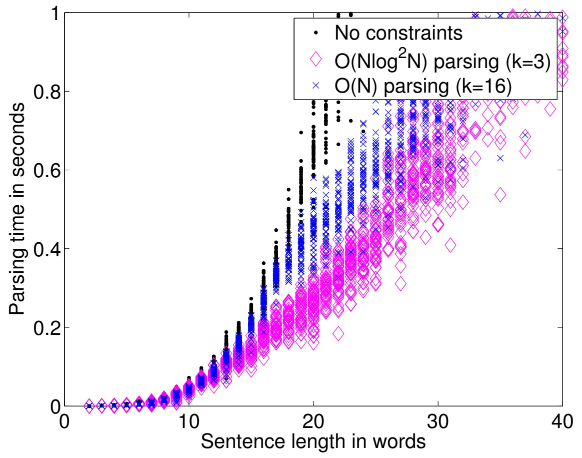


Figure 4: Scatter plots of sentence length versus parsing time for (1) baseline exact inference (no constraints); (2) $O(N\log^2 N)$ constraints; and (3) $O(N)$ constraints.

every word falls in the B set, hence no constraints on words beginning multi-word constituents are imposed.

6 Results

Figure 5 plots F-measure accuracy versus time to parse the development set for four methods of imposing constraints: the previously plotted high precision and quadratic bound constraints, along with $O(N\log^2 N)$ and linear bound constraints using methods described in this paper. All methods are employed at various parameterizations, from very lightly constrained to very heavily constrained. The complexity-bound constraints are not combined with the high-precision constraints for this plot.

As can be seen from the plot, the linear and $O(N\log^2 N)$ methods do not, as applied, achieve as favorable of an accuracy/efficiency tradeoff curve as the quadratic bound method. This is not surprising,

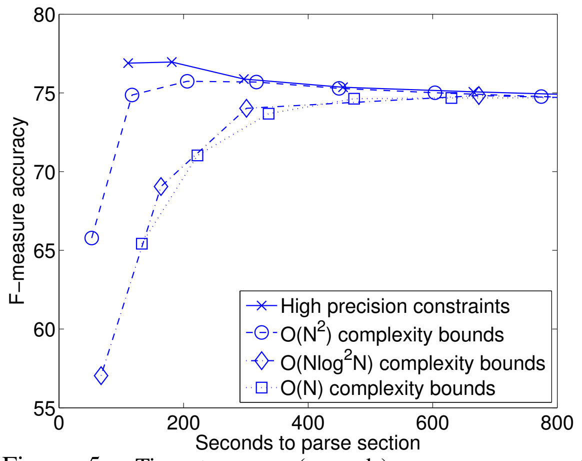


Figure 5: Time to parse (seconds) versus accuracy (F-measure) for high precision constraints of various thresholds versus three methods of imposing constraints with complexity bounds: (1) $O(N^2)$ complexity (number of open cells $\leq kN$); (2) $O(N\log^2 N)$ complexity ($|E| \leq k\log N$); and (3) linear complexity ($|E| \leq k$).

given that no words are excluded from the set B for these methods, hence far fewer constraints overall are applied with the new method than with the quadratic bound method.

Of course, the high precision constraints can be applied together with the complexity bound constraints, as described in Section 4.2. For combining complexity-bound constraints with high-precision constraints, we first chose operating points for both the linear and $O(N\log^2 N)$ complexity bound methods at the points before accuracy begins to degrade with over-constraint. For the linear complexity method, the operating point is to constrain the set size of E to a maximum of 16 members, i.e., $|E| \leq 16$. For the $N\log^2 N$ complexity method, $|E| \leq 3\log N$.

Table 2 presents results for these operating points used in conjunction with the 90% high precision constraints. For these methods, this combination is particularly important, since it includes all of the high precision constraints from the set B, which are completely ignored by both of the new methods. We can see from the results in the table that the combination brings the new constraint methods to very similar accuracy levels as the quadratic constraints, yet with the guarantee of scaling linearly to longer and longer sentences.

The efficiency benefits of combining constraints, shown in Table 2, are relatively small here because the dataset contains mostly shorter sentences. Space

Constraints	F-measure accuracy	time (seconds)
None (baseline CYK)	74.1	3646
High Precision (90%)	77.0	181
Quad ($5N$) + HiPrec (90%)	76.9	166
$N\log^2N$ ($3\log N$) + HP (90)	76.9	170
Linear (16) + HiPrec (90%)	76.8	167

Table 2: Speed and accuracy of exact-inference CYK parser on WSJ section 24 under various constraint conditions, including combining various complexity bound constraints and high precision constraints.

Constraints	F-measure accuracy	time (seconds)
None (baseline CYK)	73.8	5122
High Precision (90%)	76.8	272
Quad ($5N$) + HiPrec (90%)	76.8	263
$N\log^2N$ ($3\log N$) + HP (90)	76.8	266
Linear (16) + HiPrec (90%)	76.8	264

Table 3: Speed and accuracy of exact-inference CYK parser on WSJ section 23 under various constraint conditions, including combining various complexity bound constraints and high precision constraints.

limitations prevent us from including scatter plots similar to those in Figure 3 for the constraint combination trials, which show that the observed parsing time of shorter sentences is typically identical under each constraint set, while the parsing time of longer sentences tends to differ more under each condition and exhibit characteristics of the complexity bounds. Thus by combining high-precision and complexity constraints, we combine typical-case efficiency benefits with worst-case complexity bounds.

Note that these speedups are achieved with no additional techniques for speeding up search, i.e., modulo the cell closing mechanism, the CYK parsing is exhaustive—it explores all possible category combinations from the open cells. Techniques such as coarse-to-fine or A* parsing, the use of an agenda, or setting of probability thresholds on entries in cells—these are all orthogonal to the current approach, and could be applied together with them to achieve additional speedups. However, none of these other techniques provide what the current methods do: a complexity bound that will hold even in the worst case.

To validate the selected operating points on a different section, Table 3 presents speed and accuracy results on the test set (WSJ section 23) for the exact-inference CYK parser.

We also conducted similar preliminary trials for parsing the Penn Chinese Treebank (Xue et al., 2004), which contains longer sentences and different branching characteristics in the induced grammar. Results are similar to those shown here, with chart constraints providing both efficiency and accuracy gains.

7 Conclusion

We have presented a method for constraining a context-free parsing pipeline that provably achieves linear worst case complexity. Our method achieves comparable observed performance to the quadratic complexity method previously published in Roark and Hollingshead (2008). We were motivated to pursue this method by the observed linear parsing time achieved with the quadratic bound constraints, which suggested that a tighter complexity bound could be achieved without hurting performance.

We have also shown that combining methods for achieving complexity bounds—which are of primary utility for longer strings—with methods for achieving strong observed typical case speedups can be profitable, even for shorter strings. The resulting combination achieves both typical speedups and worst-case bounds on processing.

The presented methods may not be the only way to achieve these bounds using tagger pre-processing of this sort, though they do have the virtue of very simple constraint selection. More complicated methods that track, in fine detail, how many cells are open versus closed, run the risk of a constraint selection process that is itself quadratic in the length of the string, given that there are a quadratic number of chart cells. Even so, the presented methods critically control midpoints for all cells only via the set E (words that can end a multi-word constituent) and ignore B. More complicated methods for using both sets that also achieve linear complexity (perhaps with a smaller constant), or that achieve $O(N\log N)$ complexity rather than $O(N\log^2 N)$, may exist.

Acknowledgments

This research was supported in part by NSF Grant #IIS-0447214 and DARPA grant #HR0011-08-1-0016. Any opinions, findings, conclusions or recommendations expressed in this publication are those of the authors and do not necessarily reflect the views of the NSF or DARPA.

References

D. Blaheta and E. Charniak. 1999. Automatic compensation for parser figure-of-merit flaws. In *Proceedings of the 37th annual meeting of the Association for Computational Linguistics (ACL)*, pages 513–518.

E. Charniak. 2000. A maximum-entropy-inspired parser. In *Proceedings of the 1st Conference of the North American Chapter of the Association for Computational Linguistics*, pages 132–139.

J. Cocke and J.T. Schwartz. 1970. Programming languages and their compilers: Preliminary notes. Technical report, Courant Institute of Mathematical Sciences, NYU.

B. Djordjevic, J.R. Curran, and S. Clark. 2007. Improving the efficiency of a wide-coverage CCG parser. In *Proceedings of the 10th International Workshop on Parsing Technologies (IWPT)*, pages 39–47.

E. Glaysher and D. Moldovan. 2006. Speeding up full syntactic parsing by leveraging partial parsing decisions. In *Proceedings of the COLING/ACL 2006 Main Conference Poster Sessions*, pages 295–300.

K. Hall and M. Johnson. 2004. Attention shifting for parsing speech. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 40–46.

K. Hollingshead and B. Roark. 2007. Pipeline iteration. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 952–959.

K. Hollingshead, S. Fisher, and B. Roark. 2005. Comparing and combining finite-state and context-free parsers. In *Proceedings of the Human Language Technology Conference and the Conference on Empirical Methods in Natural Language Processing (HLT/EMNLP)*, pages 787–794.

T. Kasami. 1965. An efficient recognition and syntax analysis algorithm for context-free languages. Technical report, AFCRL-65-758, Air Force Cambridge Research Lab., Bedford, MA.

M.P. Marcus, B. Santorini, and M.A. Marcinkiewicz. 1993. Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2):313–330.

A. Ratnaparkhi. 1999. Learning to parse natural language with maximum entropy models. *Machine Learning*, 34(1-3):151–175.

B. Roark and K. Hollingshead. 2008. Classifying chart cells for quadratic complexity context-free inference. In *Proceedings of the 22nd International Conference on Computational Linguistics (COLING)*, pages 745–752.

N. Xue, F. Xia, F. Chiou, and M. Palmer. 2004. The Penn Chinese treebank: Phrase structure annotation of a large corpus. *Natural Language Engineering*, 10(4):1–30.

D.H. Younger. 1967. Recognition and parsing of context-free languages in time n^3. *Information and Control*, 10(2):189–208.