

Efficient Geometric Algorithms for Parsing in Two Dimensions

Percy Liang, Mukund Narasimhan, Michael Shilman, and Paul Viola
Microsoft Research
Redmond, WA, 98052

Abstract

Grammars are a powerful technique for modeling and extracting the structure of documents. One large challenge, however, is computational complexity. The computational cost of grammatical parsing is related to both the complexity of the input and the ambiguity of the grammar. For programming languages, where the terminals appear in a linear sequence and the grammar is unambiguous, parsing is $O(N)$. For natural languages, which are linear yet have an ambiguous grammar, parsing is $O(N^3)$. For documents, where the terminals are arranged in two dimensions and the grammar is ambiguous, parsing time can be exponential in the number of terminals. In this paper we introduce (and unify) several types of geometrical data structures which can be used to significantly accelerate parsing time. Each data structure embodies a different geometrical constraint on the set of possible valid parses. These data structures are very general, in that they can be used by any type of grammatical model, and a wide variety of document understanding tasks, to limit the set of hypotheses examined and tested. Assuming a clean design for the parsing software, the same parsing framework can be tested with various geometric constraints to determine the most effective combination.

1 Introduction

Grammatical approaches for document structure analysis have a long history. In comprehensive reviews of the field, a significant percentage of the reported papers have used grammatical approaches [12, 16, 3]. Grammatical document processing research relies on the related work in the field of general grammatical parsing. It is not surprising that these document papers adopt the state of the art in parsing technology at the time of publication. For example, the work of Krishnamoorthy et. al. uses the grammatical and parsing tools available from the programming language community [11] (see also [7, 1]). Similarly the work by Hull uses attributed probabilistic context free grammars [8] (see

also [6, 14, 13]). In the last few years there has been a rapid progress in research on grammars in the natural language community. These advances have led to more powerful grammatical models that can be learned directly from data [15]. Such models are strictly more powerful than the probabilistic context free grammars used in previous document analysis research. Simultaneously there has been progress on accelerating the parsing process in the presence of ambiguity [5, 10]. Motivated by these results a new wave of research on grammatical parsing for documents is likely to result.

A grammar based approach selects a global description of the page from several competing descriptions based on a global figure of merit. The local interpretation which maximizes the global score is selected. This provides a principled technique for globally integrating local measurements and handling local ambiguity. The challenges of grammatical approaches include computational complexity, grammar design, and parameter estimation. The focus of this paper is computational complexity: we introduce a set of general purpose geometric constraints and data structures which can be used to accelerate the parsing of two dimensional documents.

In related work Miller and Viola describe a system for parsing equations which uses geometrical data structures to control the complexity of the parsing process [14]. We improve on their algorithms and present a number of new geometric algorithms as well. Other researchers have used geometric graph data structures to constrain the set of interpretations on the page [13]. While these papers do not discuss grammars directly, the segmentation and recognition problem that they solve can be rephrased as a parsing problem with a simplified grammar. Our work improves on these graph structures as well.

2 Document Grammars

One simple example examined in detail may yield some intuitions regarding the algorithms presented below. Figure 1 shows a page with 4 terminal objects, which depending on the application could be connected components, pen

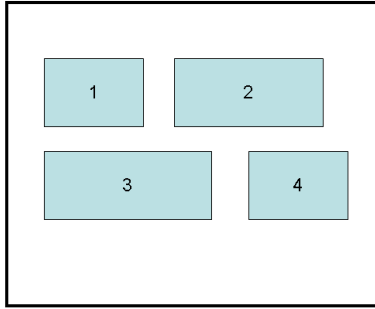


Figure 1. A very simple page with four terminal objects.

strokes, text lines, etc. In this case, let us assume that the objects are words on a simple page and the task is to group the words into lines and lines into paragraphs. A simple grammar that expresses this process is:

(Page \rightarrow ParList)	(LineList \rightarrow Line LineList)
(ParList \rightarrow Par ParList)	(LineList \rightarrow Line)
(ParList \rightarrow Par)	(Line \rightarrow WordList)
(Par \rightarrow LineList)	(WordList \rightarrow Word WordList)
	(WordList \rightarrow Word)

The correct parse in this case is:

```
(Page (ParList
  (Par (LineList
    (Line (WordList (Word 1)
      (WordList (Word 2))))
    (LineList (Line
      (WordList (Word 3)
        (WordList (Word 4))))))))))
```

This parse tree provides a great deal of information about the document structure: there is one paragraph containing two lines; the first line contains word 1 and word 2, etc.

The grammatical approach can be adopted for many types of document analysis tasks, including the parsing of mathematical expressions, text information extraction, and table extraction.

In general, productions in a grammar have the form ($A \rightarrow B C$) which states that the non-terminal symbol A can be replaced by the non-terminal B followed by the non-terminal C . Following the general conventions for grammars, non-terminals are written in upper case and terminals in lower case. We will restrict our discussion to a binarized grammar in which each non-terminal has either one or two elements on the right hand side.¹ A simple weighted grammar, or equivalently a Probabilistic Context Free Grammar (PCFG), additionally assigns a cost to every production. Productions which are applicable expand the non-terminals with a log probability proportional to cost.

¹Any more general grammar can be converted to a binary grammar easily.

Most practitioners of computer science are familiar with the notion of a programming language grammar. These grammars are specially designed to be both globally and locally unambiguous. For such grammars very efficient linear time parsing algorithms can be used. In the case of English language grammars, where there are unavoidable ambiguities, there are often hundreds or thousands of valid parses for any sentence [4]. For a linear sequence of terminals parsing requires $O(PN^3)$ time, where P is the number of productions in the grammar and N are the number of terminals.

While there are a number of competing parsing algorithms, one simple yet generic framework is called Chart Parsing [9]. Chart parsing attempts to fill in the entries of a chart $C(A, R)$ which is the best score of a non-terminal A as an interpretation of the sub-sequence of terminals R . The cost of any non-terminal can be expressed as the following recurrence:

$$C(A, R) = \min_{\substack{A \rightarrow BC \\ R_1 \cap R_2 = \emptyset \\ R_1 \cup R_2 = R}} C(B, R_1) + C(C, R_2) + l(A \rightarrow BC) \quad (1)$$

where $\{BC\}$ ranges over all productions for A , and R is a subsequence of terminals (what we will call a region), and R_1 and R_2 are subsequences which are disjoint and whose union is R (i.e. they form a partition). Essentially the recurrence states that the score for A is computed by finding a low cost decomposition of the terminals into two disjoint sets. Each production is assigned a cost (or loss or negative log probability) in a table, $l(A \rightarrow BC)$. The entries in the chart (sometimes called edges) can be filled in any order, either top down or bottom up. The complexity of the parsing process arises from the number of chart entries that must be filled and the work required to fill each entry. When analyzing a linear sequence of terminals there are $O(PN^2)$ entries (since there are $\frac{1}{2} \binom{N}{2} \in O(N^2)$ pairs $\langle i, j \rangle$). Since the work required to fill each entry is $O(N)$, the overall complexity is $O(PN^3)$, where P is the number of productions.

2.1 Geometric Parsing Is Exponential

In this paper we will study algorithms for parsing terminals arranged on a two dimensional page. Unfortunately a direct application of chart parsing to two dimensional arrangements of terminals requires exponential time. The key problem is that the terminals no longer have a linear sequential order. Returning to (1), the region R is now a subset, and R_1 and R_2 are subsets which are disjoint and whose union is R (i.e. they form a partition). As before we can analyze the size of the chart, which is $O(P|\mathcal{P}(N)|)$ where $\mathcal{P}(N)$ is set of all subsets of N terminals. In general there are an exponential number of subsets, and hence the algorithm is exponential.

Others have certainly observed this dilemma and have either assumed that the 2D terminals are linearly ordered (by some earlier process) or have worked with other linearizations (for example [11] parses linear projections of the input).

Miller and Viola introduced an effective heuristic which significantly improved performance. They select subsets R_1, R_2 such that either $\text{CHULL}(R_1) \cap R_2 = \emptyset$ or $\text{CHULL}(R_2) \cap R_1 = \emptyset$. Calling these sets *regions* is now appropriate, since each set lies within a convex region of the page. It is worth noting that if the terminals lie along a line (and therefore have a strict linear ordering) the convex hull criterion yields the $O(N^2)$ regions and is equivalent to the contiguous subsequence constraint used in conventional parsing. While this constraint is effective in practice, the worst case complexity is still exponential and the computation of the convex hulls themselves is somewhat expensive.

The notion of region (and subregion) is quite general. A region of the input is a set of the terminals, coupled with a constraint on the set of subsets which are valid for parsing. For the case of classical parsing the regions are sequential terminals and are represented by a pair of integers $\langle i, j \rangle$. The subregions are sequential subsets of terminals $\langle i, k \rangle$ and $\langle k, j \rangle$.

In this paper we propose several new kinds of geometric regions. We define and motivate each region and we also present an efficient algorithm for enumerating valid subsets. Each can lead to a significant speed up of parsing.

2.2 Rectangle Hull Region

The first region is called a *Rectangle Hull Region*. Subregions are constrained so that $\text{RHULL}(R_1) \cap \text{RHULL}(R_2) = \emptyset$, where $\text{RHULL}(X)$ is smallest axis aligned rectangle which contains X (known as a rectangle hull). The motivation for this region comes from the convex hulls used by Miller and Viola, but it is much easier to compute. For printed pages which have been deskewed, the constraint implied by the Rectangle Hull Region is essentially equivalent to the convex hull constraint.

The number of valid rectangle regions is polynomial (as in a conventional sequence region). To see this, notice that the left boundary of $\text{RHULL}(X)$ is defined by the leftmost point in the subset of terminals X . This point must be the leftmost point of some terminal in X . The same argument applies to the top, bottom, and right boundaries. Every potential rectangle hull region $\langle t, b, l, r \rangle$ can be enumerated by selecting 4 terminals. The left boundary of the first terminal defines the left boundary of the region, l ; the top of the second defines the top, t ; the bottom of the third defines the bottom, b ; the right boundary of the final terminal defines the right, r . There are no more than $\binom{N}{4} = O(N^4)$ valid subregions. Note that in practice there are many fewer

regions, since many of the regions may fail the disjoint test above (since a terminal may be split by the boundary and will neither lie inside of $\text{RHULL}(R_1)$ nor entirely outside). In experiments it is often the case that approximately $O(N^2)$ rectangular subregions are admissible.

For top down parsing the following efficient algorithm is used to enumerate all valid pairs of rectangle hull subregions (these pairs are used to drive the recursion in Equation 1). Insert the top, t_i , and bottom, b_i , of each terminal, i , into a single sorted list. Given a region $\langle t, b, l, r \rangle$, it can be split into two valid subregions as $\langle t, s, l, r \rangle$ and $\langle s, b, l, r \rangle$, if $t < s < b$ and the boundary does not intersect any terminal in the current region. All admissible s can be found in a single pass over the sorted list from top to bottom. As the list is traversed, we keep track both of the current element, s' and the set of all terminals which are split: $t_i < s' < b_i$ (the top of terminal i is above the separator and the bottom is below the separator). The value s' is a valid separator if the list of split terminals is empty. Note that the list of split terminals can be updated rapidly. If the previous element is a top, t_i , add i to the list. If the previous element is a bottom, b_i , remove i from the list. Vertical splits are computed by repeating the entire process for the left and right bounds of the terminals. Note that the work of sorting can be done once for the entire page. Thereafter sublists of the full sorted list are used. Also note that there can be at most $4N$ pairs of admissible subsets, though many of these are often pruned away since they are inadmissible.

3 Convex Hull Region

The convex hull criteria of Miller and Viola is a clear candidate for use as a type of region. The key challenge is to enumerate all admissible subsets efficiently. Recall that subsets R_1 and R_2 must satisfy the criteria that $R_1 \cap \text{CHULL}(R_2) = \emptyset$ or $R_2 \cap \text{CHULL}(R_1) = \emptyset$. We have found that a closely related criteria is equally valuable, but has the advantage that it is very efficiently enumerable. This new criteria requires that $\text{CHULL}(R_1) \cap \text{CHULL}(R_2) = \emptyset$. Note that this new criteria always implies the first.

Recall that valid rectangle hull subregions are enumerated by finding separating lines which are defined by the boundaries of the terminals. Convex hull subregions can be generated similarly. First note that every pair of admissible subsets $\text{CHULL}(R_1) \cap \text{CHULL}(R_2) = \emptyset$ is separated by a line which is tangent to both $\text{CHULL}(R_1)$ and $\text{CHULL}(R_2)$ (see Figure 2). The proof omitted due to space. Recall that the goal is to enumerate subsets that satisfy the convex hull criteria efficiently, *not to compute convex hulls*. We do so by enumerating all potential co-tangent lines. Since every co-tangent line is defined by a pair of terminals, all potential co-tangent lines can be enumerated in $O(\binom{N}{2})$.

Miller and Viola do not discuss the algorithm used to

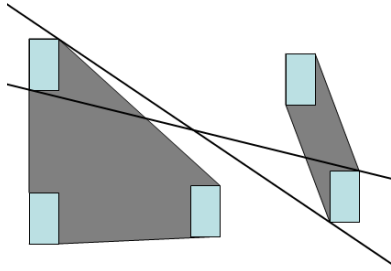


Figure 2. The co-tangent lines for a pair of terminals. The green rectangles represent the bounding boxes of the terminals while the grey polygons are a pair of implicitly defined convex hulls.

enumerate convex hull admissible subsets. It is possible that their algorithm, while very fast for the smaller problems they encountered, may well have been exponential. We believe this is the first description of an algorithm for enumerating convex hull compatible subregions efficiently.

4 Graph Region

In his work on the parsing of mathematical expressions, Matsakis proposes the use of a Minimum Spanning Tree(MST) on the set of terminals [13]. Given some distance measure between pairs of terminals on the page (perhaps centroid distance or nearest point distance) one can compute the MST in $O(N^2 \log N^2)$. Valid regions are subsets of the terminals which are connected in this graph. This criteria is based on the observation that when two terminals are near neighbors on the page they are often near neighbors in the parse tree. Conversely, if a pair of subsets are far apart they are rarely combined into a single subtree. Disappointingly, the MST criterion is worse case exponential, though it is quite effective in practice.

Perhaps a more serious flaw is the fragility of the MST. One type of problem arises from expressions containing fractions, since the fraction bar sometimes lies closer to the symbols in the numerator than other symbols in the numerator (see Figure 3). Additionally for handwritten inputs there are many examples where the MST contradicts the parse tree.

The fragility of the MST can be addressed by adding additional edges to the graph. A simple proposal is to connect the k nearest neighbors on the page. This increases the number of subsets significantly, but tends to fix many of the problems with the MST.

One can directly enumerate all connected subsets of a graph using a variant of breadth first search, being careful

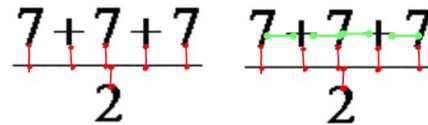


Figure 3. (Left) An arrangement of symbols for which the MST does not yield the correct subgroups. Note this type of arrangement also admits an exponential (though manageable) number of parses. (Right) Additional edges added to the MST which allows the correct grouping.

to use a hash table to exclude subsets which are encountered twice. The search proceeds using a queue of connected subsets. A subset of size n is removed from the queue and is extended by finding all nodes which are directly connected to some element of the subset. If there are k such connected nodes, a collection of k connected subsets of size $n + 1$ are created and placed back in the queue. Before insertion, a hash table is consulted to determine if the subset is a duplicate, and if so is discarded.

For graphs where there are an exponential number of connected subgraphs, all enumeration algorithms are intractable. In practice the constraint that a subregion be connected is used to prune regions generated by one of the other region types. In this case the complexity is that of the enumerating region (though the final number of regions is often significantly reduced).

5 Partial Order Region

In his work on determining reading order, Breuel define a partial order on the lines of the pages [2]. Each pair of lines, a and b , is examined to determine if there is an ordering between them: (Condition 1), line a comes before line b if their ranges of horizontal coordinates overlap and if a is above b on the page; (Condition 2), line a comes before b if a is entirely to the left of b and if there does not exist a line c whose vertical coordinates are between a and b and whose range of horizontal coordinates overlaps both a and b . Surprisingly these two simple rules yield sufficient constraint so that the topological sort of the lines frequently matches the page reading order.

While Breuel's technique is quite beautiful due to its simplicity, we have found two types of common failures: ambiguity and incorrectness. Ambiguity arises whenever the topological sort is not unique. In this case one of the valid orderings is correct but many others are not. Incorrectness results when the true ordering is inconsistent with the partial order. The second rule is the primary cause of

incorrectness, since it sometimes fails in situations where there are two different two column zones on the page. In addition, these heuristics often fail for the headers and footers of pages. It is tempting to modify the second condition: (Condition 2') line a comes before b if a overlaps b vertically and a is to the left of b . This new condition is consistent with almost all pages of text, but it often yields a partial order which is ambiguous. Note that condition 2' yields a strictly weaker partial order, since if condition 2' is satisfied this necessarily implies condition 2.

Breuel's partial order suggests a new type of region, which we call the Partial Order Region. In this case regions are selected so that no element in R_2 comes before an element in R_1 which is denoted $R_1 \leq R_2$. An efficient algorithm for enumerating such regions is based on a search of the directed graph implied by the partial order. The test for an admissible subregion is simply to check that all children of nodes in R_2 lie in R_2 , or equivalently all parents of nodes in R_1 lie in R_1 . Begin by performing a topological sort on the nodes to form a total order. Nodes are then each assigned either to R_1 or R_2 using depth first search. Before a node is assigned a label it is checked for admissibility.

```

Recurse(Node[] n, int i)
{
    if (n.Length == i)
        AddSubsets();
    else {
        if (NoParentR2(n[i]) {
            n[i].SetR1();
            Recurse(n, i+1);
        }
        if (NoChildR1(n[i]) {
            n[i].SetR2();
            Recurse(n, i+1);
        }
        n[i].Unset();
    }
}

```

If there are no links in the partial order graph the above algorithm is exponential. This is the worse case. Breuel's graphs are quite dense and he reports that these partial orders often yield a unique topological ordering. In this case the algorithm is quite efficient and is computationally equivalent to parsing a linear sequence.

6 A Simple Example

The simple example in Figure 1 can be used to demonstrate the various geometric regions. Given that there are four terminals, there are 16 proper subsets. A region is considered compatible with the correct interpretation if it admits all the subgroups in that parse tree. In this case the

correct regions are [1], [2], [3], [4], [1, 2], [3, 4].

Table 1 shows the admissible subsets for each type of region. The sequence region is assumed to have perfect knowledge of the correct order. The other regions observe only the geometry of the terminals. The partial order region is given the following directed edges: $1 < 2$, $1 < 3$, $2 < 4$, $3 < 4$. The graph region is given the following neighborhood edges: 1-2, 1-3, 2-3, 2-4, 3-4. Since all regions enumerate the subsets of size one, the first differences appears among the 10 larger subsets. Both the "hull" based regions perform perfectly. They *only* admit subsets which are compatible the correct answer. All other subsets are rejected. The Graph Region performs worst, since in a tight collection of four symbols almost every subset is connected. The Partial Order Region is a bit better, but it suffers from the tight placement of the terminals as well. Both the "graph" related regions perform much better when the terminals are strung out in a more linear arrangement. Note that the sequence region, which has access to the terminal ordering of this page, performs worse than either hull region. In this case the geometric region constraint has additional value beyond the ordering of the nodes.

Interestingly, it is the tight placement of the terminals which leads to the success of the "hull" regions. There are applications, like the parsing of mathematics, where the symbols are placed very tightly and the hull regions are not compatible with the correct parse (i.e. they reject valid subsets). One simple solution is to erode the terminals slightly, which essentially reduces the tightness of the placement. In the limit one can erode the terminal to a single point, for example the centroid. In this case the hulls are maximally compatible but still provide significant constraint.

7 Experiments

In order to support some of the claims of this paper we have analyzed pages from the UWIII document image database. The UWIII database gives ground truth for words, lines, and zones, as well as the reading order. In other work we are constructing a grammatical approach for the extraction of this and additional information such as sections and columns. Classification performance on this problem will be described elsewhere. Here we report the number of subregions produced for several pages from the database. The goal here is to show that the geometric regions produce a reasonable number of subregions. One valid point of comparison is to compare the number of regions produced by a sequence region given the ground truth linear ordering of the page. The other regions operate with only geometric information. Another point of comparison is the number of subset in the power set. Note that the Rectangle Hull Region

	Seq	Graph	Partial Order	Rect Hull	Convex Hull
1	X	X	X	X	X
2	X	X	X	X	X
3	X	X	X	X	X
4	X	X	X	X	X
1 2	X	X	X	X	X
1 3		X	X		
1 4					
2 3	X	X	X		
2 4		X	X		
3 4	X	X	X	X	X
1 2 3	X	X	X		
1 2 4		X			
1 3 4		X			
2 3 4	X	X	X		
Totals:	14	9	13	6	6

Table 1. A table of the valid regions enumerated by the various types of geometric regions.

	A04GBIN	A04FBIN	A004BIN	A009BIN
Terminals:	30	49	44	47
Sequence:	465	1225	990	1128
Rect:	316	648	948	1040
Partial Order:	1599	2143	990	1128
PowerSet	1.0E+009	5.6E+014	1.8E+013	1.4E+014

Table 2. Number of regions generated. First row is the number of lines in the file (the two right most files are two column files, the two leftmost are one column files).

consistent produces the fewest subregions.

8 Conclusion

Grammatical parsing has proven valuable in a number of document analysis problems. With recent advances in grammar learning and parsing, we are likely to see an increase in interest in this area. In this paper we have presented geometric algorithms which can be used to accelerate parsing for a wide variety of two dimensional parsing problems. Examples include document structure extraction, parsing of mathematical expressions (both printed and handwritten), and document information extraction.

References

- [1] D. Blostein, J. R. Cordy, and R. Zanibbi. Applying compiler techniques to diagram recognition. In *Proceedings of the Sixteenth International Conference on Pattern Recognition*, volume 3, pages 123–136, 2002.
- [2] T. Breuel. High performance document layout analysis. In *2003 Symposium on Document Image Understanding Technology*, Greenbelt Maryland, 2003.
- [3] K.-F. Chan and D.-Y. Yeung. Mathematical expression recognition: a survey. *International Journal on Document Analysis and Recognition*, 3:3–15, 2000.
- [4] E. Charniak. Statistical techniques for natural language parsing. *AI Magazine*, 1997.
- [5] E. Charniak, S. Goldwater, and M. Johnson. Edge-based best-first chart parsing. In *Proceedings of the Fourteenth National Conference on Artificial Intelligence*, pages 127–133, 1998.
- [6] P. Chou. Recognition of equations using a two-dimensional stochastic context-free grammar. In *SPIE Conference on Visual Communications and Image Processing*, Philadelphia, PA, 1989.
- [7] A. Conway. Page grammars and page parsing, a syntactic approach to document layout recognition. In *Proceedings of the Second International Conference on Document Analysis and Recognition*, pages 761–764, Tsukuba Science City, Japan, 1993.
- [8] J. F. Hull. Recognition of mathematics using a two-dimensional trainable context-free grammar. Master’s thesis, MIT, June 1996.
- [9] M. Kay. Chart generation. In *Proceedings of the 34th conference on Association for Computational Linguistics*, pages 200–204. Association for Computational Linguistics, 1996.
- [10] D. Klein and C. D. Manning. A* parsing: Fast exact viterbi parse selection. Technical Report dbpubs/2002-16, Stanford University, 2001.
- [11] M. Krishnamoorthy, G. Nagy, S. Seth, and M. Viswanathan. Syntactic segmentation and labeling of digitized pages from technical journals. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15:737–747, 1993.
- [12] S. Mao, A. Rosenfeld, and T. Kanungo. Document structure analysis algorithms: A literature survey. In *Proc. SPIE Electronic Imaging*, volume 5010, pages 197–207, January 2003.
- [13] N. Matsakis. Recognition of handwritten mathematical expressions. Master’s thesis, Massachusetts Institute of Technology, Cambridge, MA, May 1999.
- [14] E. G. Miller and P. A. Viola. Ambiguity and constraint in mathematical expression recognition. In *Proceedings of the National Conference of Artificial Intelligence*. American Association of Artificial Intelligence, 1998.
- [15] B. Taskar, D. Klein, M. Collins, D. Koller, and C. Manning. Max-margin parsing. In *Empirical Methods in Natural Language Processing (EMNLP04)*, 2004.
- [16] R. Zanibbi, D. Blostein, and J. R. Cordy. A survey of table recognition: Models, observations, transformations, and inferences. *International Journal of Document Analysis and Recognition*, 7(1):1–16, 2004.