Parsing Probabilistic Context Free Languages with Multi-Objective Genetic Algorithms

Ramon Lefuel and Brian J. Ross

Brock University
Department of Computer Science,
500 Glenridge Ave.,
St. Catharines, Ontario
Canada L2S 3A1
lera1981@excite.com, bross@cosc.brocku.ca

Abstract. An approach to parsing probabilistic context free languages is presented. Given an input sentence, a genetic algorithm is used to evolve parse trees as defined by a given probabilistic context free grammar. Each chromosome in the population represents a candidate parse tree, using a simple indexed representation. The novelty of the approach is the multi-objective treatment of parse tree fitness. One dimension of the fitness space is the number of contiguous words correctly read by the parse. The other dimension incorporates a measurement equivalent to the probability obtained by complete parse trees, and partial probabilities corresponding to incomplete parses. A number of experiments show that this method is both effective and efficient for parsing natural language sentences.

1 Introduction

Probabilistic models of natural language processing have been proposed [1]. These models have practical merit due to the ambiguity found within natural languages. This is clearly seen when considering sentences with multiple syntactic parses and semantic interpretations. If a candidate parse is "obvious", then the stochastic model ascribes to it a higher probability than its less sensible counterparts. Hence, given a probabilistic context free grammar (PCFG), probabilities are assigned to productions that ideally reflect the distribution of sentences found in their actual use.

Context-free language parsing has a polynomial computational complexity. For example, the CYK algorithm has a complexity of $O(n^3)$ [2]. When searching for a parse tree with the highest probability according to a PCFG, it is necessary to find all the correct parse trees, and select the one with the highest probability. This is a more computationally expensive problem than finding a single correct parse, because the number of parse trees can grow exponentially with respect to the size of the sentence. In degenerate cases, there may be an exponential number of trees to explore. This is unlikely to occur, however, with typical natural language CFG's. In any case, the combinatorial nature of PCFG parsing has lead researchers to explore the use of heuristic techniques. Although heuristic searches do not guarantee optimal solutions, they are very effective for finding reasonable solutions to difficult combinatorial problems.

This paper presents a new technique for parsing sentences. A genetic algorithm is used to evolve a population of parse trees. These parse trees correspond to the rules of a given PCFG, and denote candidate parses for a target sentence. The chromosome representation uses a simple indexed mapping of production rules, which are interpreted in a top-down manner during parse tree construction. A novelty of the approach is the treatment of PCFG parsing as a multi-objective problem. One dimension of the fitness space is the number of consecutive words in a sentence successfully parsed. The other dimension is a measurement of the probability of the parse. This measurement is a true probability if the chromosome transcribes onto a complete parse tree. Otherwise, it is a heuristic value that scales to the probability of the partial parse tree within the chromosome. Pareto ranking is then performed on these fitness dimensions, resulting in a multi-objective perspective of the problem.

Section 2 discusses some background to the problem of PCFG parsing. The experiment design is outlined in Section 3. Results are given in Section 4. Conclusions are given in Section 5.

2 Background

Evolutionary computation has been applied towards CFG parsing. O'Neill and Ryan's grammatical evolution (GE) system uses a context free grammar for defining the target programming language, in which solution programs are written [3]. Each chromosome in the population contains a list of numbers that index the set of productions in the CFG. Once a complete parse tree is evolved, the corresponding program is extracted and interpreted as a potential solution. The translation of chromosomes into parse trees is an instance of the CFG parsing problem. Their GA application is intended to both discover a legal program (parse tree) as well as find a suitable program that solves some given program.

Araujo uses a genetic algorithm to parse probabilistic CFG's (PCFG) [4, 5]. The chromosome representation is fairly sophisticated, as it explicitly denotes aspects of the parse tree, including the portion of the sentence parsed, the production rule used, subsequent nonterminals used on the right-hand side of productions, and the depth of the node within the parse tree. Fitness evaluation measures both the success of the rule in parsing a sentence and the failure rate or incoherence of genes within the rule. Specialized crossover and mutation operators preserve the correctness and performance of offspring chromosomes. Araujo successfully parses several sentences with the system. The best performance is obtained using course-grained parallelism on a multiprocessor computer.

3 Experiment

3.1 Representation

The genetic algorithm is supplied with a sentence to parse, and a PCFG for parsing. Each chromosome represents a partial or complete parse of the sentence with this PCFG. A complete parse tree is a precise and correct parse of the sentence, or portion thereof, with the PCFG. Partial parse tree are incorrect. The right-hand portion is not completely satisfied, because the terminal it is expecting does not match the one found at a given position in the sentence. In addition, partial parse trees can arise when the tree consumes the entire sentence, but still requires further input to complete the parse tree.

The chromosome representation is similar to that used in [3]. Chromosomes consist of a fixed-sized list of integers ($1 \leq i \leq maxint$), where *maxint* needs to be at least as large as the maximum number of rules for any set of nonterminals. Each integer is an index that denotes a production rule from the PCFG. For example, the first integer may be taken to refer to a top-level nonterminal (*sentence*). If there are 3 productions for *sentence*, then the gene is interpreted modulo 3 to denote one of these 3 productions. If the right-side of this production is *nounphrase verbphrase*, then the next integer in the sequence will be interpreted modulo K to access one of the K noun phrase productions. When a terminal is reached, it must match with the current word in the sentence, or else the parsing stops with an error condition. This process continues until either a terminal mismatches, or the entire sentence is parsed.

3.2 PCFG

The PCFG used in the experiments is shown in Table 1. Nonterminals are labelled with capital letters, and terminals with lower-case. The corresponding probabilities are shown beside each production. There are two sources of ambiguity in the grammar. The *WH* rules are the primary means by which ambiguous parse trees arise. The *noun phrase* rules also permit multiple parse trees. For example, noun phrase production 7 can be duplicated by productions 4 and 5. Using production 7 entails a probability of 10%, while the use of 4 and 5 has a probability of 4%. Hence production 7 results in stronger parse trees.

3.3 Parsing strategies

Two types of parsing strategies were examined in this research:

- **A**: Fix the root of the parse tree with a top-level nonterminal ("sentence"), and apply the parse from the first word in the target sentence.
- **B**: Root the parse tree with an arbitrary nonterminal determined by the chromosome. Apply the parse iteratively starting at each word in the sentence. Retain the parse that starts from the word that results in the longest substring of consumed words.

Sentence:
1. Noun Phrase, Verb Phrase (95%)
2. Verb Phrase (5%)

Noun Phrase:
1. noun (10%)
2. noun, Preposition Phrase (10%)
3. noun, Noun Phrase (20%)
4. det, Noun Phrase (20%)
5. adj, Noun Phrase (20%)
6. Noun Phrase, WH (10%)
7. det, adj, Noun Phrase (10%)

Verb Phrase:
1. verb (10%)
2. verb, Noun Phrase (20%)
3. verb, Preposition Phrase (20%)
4. verb, Noun Phrase, Preposition Phrase (20%)
5. verb, WH (30%)

WH Phrase:
1. pron, Verb Phrase (50%)
2. pron, Noun Phrase, Verb Phrase (20%)
3. Verb Phrase (30%)

Preposition Phrase:
1. prep, Noun Phrase (100%)

Table 1. Probabilistic Context Free Grammar

Strategy A uses a parser that tries to find a complete parse tree for the entire sentence. Strategy B uses a parser that tries to find partial parse trees for substrings of the sentence. Since trees can be rooted by any nonterminal, the criteria for completeness is relaxed. In both strategies, the parser will record the size of the longest contiguous string of words successfully parsed, along with its corresponding probability. In strategy B, when multiple substrings tie for maximum length of consumed substrings, then the one with the highest probability is recorded.

3.4 Fitness

There are two dimensions of fitness for the PCFG parsing problem. One dimension of the fitness space is the number of consecutive words successfully parsed denoted by the chromosome. The other dimension is the probability for the parse, which is simply the product of the probabilities of all the productions used within the parse tree. We would like to maximize both of these values by parsing the complete sentence with the highest probability possible. A complication with using a linear fitness score for PCFG is that the number of words parsed and the overall probability are in conflict with each other. Longer substrings require more productions, which result in lower overall probabilities. Reconciling these scores into a linear score (weighted sum) is difficult and ultimately arbitrary.

The fact that probabilities are only relevant for complete parse trees poses another problem. Since a partial parse tree denotes a non-member of the language, its probability is zero. This is not a feasible scoring strategy for a genetic algorithm, because until a parse tree is successfully evolved, virtually the entire population will have a zero probability score. Therefore, a *probability distance* heuristic is used instead of pure probabilities. It is defined as follows:

$$ProbDistance = \begin{cases} Pr(tree) & : complete\ \ parse \\ \\ Pr(tree) - 1 & : partial\ \ parse \end{cases}$$

where $Pr(tree)$ is the computed probability for the complete or partial tree that consumes a substring of the sentence. This heuristic assigns a negative score between -1.0 and 0.0 to partial parse trees, and positive values (i.e., true probabilities) between 0.0 and 1.0 to complete trees. Selective pressure is given to partial or complete trees that have higher probabilities.

The PCFG parsing problem is effectively characterized as a multi-objective optimization problem (MOP). The maximum substring length and probability distance are used as the MOP fitness vector. Pareto ranking is then performed on individuals using this vector [6, 7]. Pareto ranking is based upon the following notion of multi-objective domination.

Definition 1. *Given a problem defined by a vector of objectives* $\boldsymbol{f} = (f_1, ..., f_k)$ *subject to appropriate problem constraints. Then vector* $\boldsymbol{u}$ **dominates** $\boldsymbol{v}$ *iff* $\forall i \in (1, ..., k) : u_i \geq v_i \wedge \exists i \in (1, ..., k) : u_i > v_i$.

A vector is dominated if another vector exists that is better in at least 1 objective, and at least as good in the remaining objectives.

With respect to PCFG parsing, individuals are assigned a Pareto rank according to the following criteria. The rank 1 set are individuals that are not dominated in the population. Each successive rank are dominated by all individuals of lesser rank, and are not dominated by fellow rank members. The overall advantage of Pareto ranking is that substring length and and probability distance no longer must be combined together in some *ad hoc* manner.

3.5 Other experimental parameters

The experiments were run using the Mr P Gamp system [8]. This is a C++ based system that supports multiple representations, multiple populations, and a variety of reproduction and selection strategies. For this application, the reproduction operators preserve genome values (integers between 1 and 7). One specialized reproduction operator used is *modified 2-point crossover*. Here, a point denoting the length of the crossover is selected randomly. Then, two starting points for each chromosome are selected. The genes ranging from the first starting point to the length of the crossover in the first chromosome are then crossed over with the genes ranging from the second starting point to the length of the crossover in the second chromosome.

Another specialized operator is *shuffle swapping mutation*. A random number ranging from one to the number of genes in the chromosome is selected to indicate the number of genes to be mutated. Every gene that is selected for mutation gets swapped with another gene.

The implementation of multiple population evolution uses a *universal list* that redistributes subpopulation members to one another at a specified interbreeding frequency. When an interbreeding generation is reached, each subpopulation copies a specified number of its elite chromosomes to the universal list. An equal number of members from the universal list are randomly copied back to each subpopulation, replacing the worst in that population.

The parameters that are common for most of the experiments are listed in Table 2. Parameter differences in specific experiments are reported in Section 4.

Parameter	*Value*	
Evolution paradigm	generational	
Max generations	60 to 100	
Runs/experiment	5	
Sub-populations	2	
Interbreeding rate	every 10 generations	
Migration strategy	50% elite	
	Sub-population 1	*Sub-population 2*
Population sizes	270	180
Crossover rate	70%	80%
Mutation rate	20%	10%
Replication rate	10%	10%
Tournament size	3	4
Crossover operator	2-point	modified 2-point
Mutation operator	shuffle swapping	gene replacement

Table 2. Common experiment parameters

4 Results

4.1 Trees rooted at *sentence* nonterminal

The first set of experiments use parsing strategy A discussed in Section 3.3. This strategy assumes that the tree is rooted at the *sentence* rule in Table 1, and the parse begins at the first word of the input sentence. The runs use the sentences

1. Jack regretted that he ate the whole thing.
2. The man who gave Bill the money drives a big car.
3. The man who lives in the red house saw the thieves in the bank.
4. Jack likes visiting kids.
5. The man who lives in the bank likes visiting kids in the red house.

Table 3. Example sentences

Sentence	*Chromosome size*	*Max. generations*	*Complete parses*	*Approx. time per run (sec)*
1	10	60	5	1
2	15	60	4	2
3	20	60	4	2
4	10	60	5	1
5	30	80	5	5

Table 4. Overall results

in Table 3. The first 3 sentences are from [4]. Other experimental parameters, as well as overall results, are shown in Table 4. Since longer sentences require larger parse trees, their chromosomes are longer. A total of 5 runs are performed per sentence, each using a different random number seed.

The first 3 sentences in Table 3 have unambiguous parse trees. All 3 sentences were successfully parsed with strategy A. As is shown in Table 4, of the total of 15 runs for these sentences, 13 runs found complete parse trees.

Sentences 4 and 5 in Table 3 have ambiguous parse trees, primarily caused by the *WH* rules in Table 1. Sentence 4 was successfully parsed in all these runs. Of the two legal parses for this sentence, the preferred higher probability parse tree was found in all 5 runs (where "visiting" is an adjective describing "kids"). The alternative parse was also found in 2 of these runs.

Sentence 5 is a considerably more complex problem than all the other sentences, given its size and the ambiguity within it. To handle this sentence, the sizes of sub-population 1 and 2 were increased to 300 and 290 respectively, and a maximum of 80 generations were performed. 4 runs found the preferred parse ("visiting" is an adjective), while 1 run found the lower-probability alternate parse. It should be noted that of the 4 preferred parse trees, one of them was actually a more obscure variation of the tree, in which the *noun phrase* productions 4 and 5 were used instead of production 7. This variant has a lower probability than the alternate parse tree.

Figure 1 shows the Pareto distribution of one subpopulation from one run of sentence 5. The horizontal axis is the number of words consumed from the beginning of the sentence. Positive probability distances denote true probabilities for complete parse trees, while negative probability distances are for incomplete parse trees. Probability distances typically have very small magnitudes, which

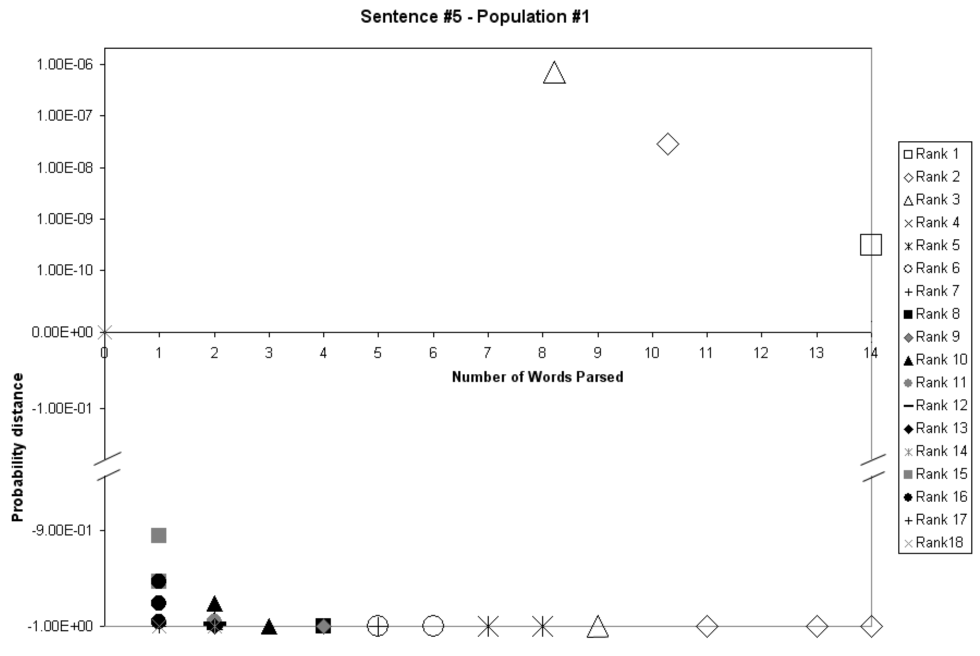

Fig. 1. Pareto distribution of subpopulation 1

makes them difficult to distinguish on the graph for some of the negative valued instances. The Pareto distribution of parse trees take the form of columns on the chart. This is because each parse tree parses a discrete number of words in the input sentence. The parse trees will have a variety of probability distances within each column. Higher column values dominate lower ones. Note that there are complete parse trees for partial initial substrings of the sentence, including possibly null strings. For example, "Jack regretted" is a complete parse for nonterminal *sentence*, although it is still a substring of sentence 1.

4.2 Arbitrarily rooted trees for unambiguous sentences

A second set of experiments used strategy B, where trees can be rooted at any nonterminal, and parsing starts at each successive word in the sentence. The longest substring consumed is used as the result for the parse. A total of 5 runs were tried for each of sentences 1 through 3 in Table 3, and 100 generations were performed per run.

Permitting a variety of rooted parse trees resulted in a greater number of complete parse trees in the populations. Complete parse trees rooted at nonterminals such as *verb phrase* or *WH* are common. Surprisingly, this greater variety of rooted trees was detrimental to the discovery of complete parse trees for the entire sentence. None of the 15 runs found complete *sentence* parse trees. Obviously, the distribution of parse trees in the population hindered finding parse trees for the entire sentence. A conjecture for this is that too many nonterminal parse trees dominate others, at the expense of constructing a *sentence* tree.

4.3 Performance

All the experiments were undertaken on a IBM/PC compatible computer with an Intel 4 2.0 GHz CPU, 256 MB 800 MHz memory, and running Windows XP. The GA system is implemented in C++. Approximate wall clock measurements for each run is shown in Table 4. This performance is very good, considering that different experiments test anywhere between 28000 and 47000 parse trees. The multiple populations are executed by a single CPU, and thus true parallelism is not being exploited.

5 Conclusion

A new technique for discovering parse trees for probabilistic context free grammars has been presented. As is illustrated in [3], the linear representation of parse trees is ideally suited for representing parse trees for CFG's. Besides its simplicity for transcribing trees, it also permits the use of standard crossover and mutation operators. Treating word consumption and probability as separate dimensions in the MOP fitness space is advantageous, eliminating the need for arbitrary, *ad hoc* combinations of these separate aspects of stochastic parsing. A number of sentences of varying degrees of complexity and ambiguity were successfully parsed. Performance wise, a complete run usually takes less than 3 seconds to complete, demonstrating the advantages of the straight-forward representation and evaluation schemes.

This research is inspired by Araujo's use of a genetic algorithm to parse probabilistic CFG's [4, 5]. He successfully parses 3 of the same sentences parsed here. A number of differences can be drawn between his approach and ours. He uses a much more complex chromosomal representation of parse trees, which incorporates explicit information of the parse tree morphology. This kind of representation requires specialized reproduction operators, which must preserve chromosome correctness. Our linear representation permits the use of generic reproduction operators. Araujo's fitness evaluation is quite complex, as it merges both probability and word consumption scores together, which is unnecessary with the MOP Pareto approach we use. Furthermore, his fitness scheme evaluates gene performance by evaluating gene "coherence" and "incoherence". A chromosome with dysfunctional genes will be suitably penalized. Although there can be erroneous genes in our chromosomes, we do not explicitly detect and penalize them, but instead allow evolution to manage their survival.

It is difficult to benchmark performance between different platforms. Araujo runs his experiments on a multi-processor SGI-CRAY Origin 2000. Using 10 processors, his implementation requires 15 seconds to successfully parse sentence 3 from Table 3. Our implementation takes approximately 2 seconds to parse the same sentence using a single Pentium 4 CPU. Platform differences not withstanding, our faster performance is likely due to the simpler representation, reproduction, and fitness evaluation.

There are a number of future directions for this research. The use of real-world data would be worth considering, since our sentences use artificial probability distributions. The study of more complex grammars with additional forms of ambiguity is worth exploring. Our multi-objective scoring strategy would also benefit with the inclusion of population diveristy heuristics, as done in [9]. Variable length chromosomes are also worth consideration, since chromosome size strongly influences evolution effectiveness.

Acknowledgements: This work was partially supported by NSERC Operating Grant 138467.

References

1. Charniak, E.: Statistical Language Learning. MIT Press (1993)
2. Hopcroft, J., Motwani, R., Ullman, J.: Introduction to Automata Theory, Languages, and Computation. second edn. Addison Wesley (2001)
3. O'Neill, M., Ryan, C.: Grammatical Evolution. IEEE Transactions on Evolutionary Computation **5** (2001) 349–358
4. Araujo, L.: A Parallel Evolutionary Algorithm for Stochastic Natural Language Parsing. In: PPSN VII. (2002) 700–709
5. Araujo, L.: Symbiosis of Evolutionary Techniques and Statistical Natural Language Processing. IEEE Transactions on Evolutionary Computation **8** (2004) 14–27
6. Goldberg, D.: Genetic Algorithms in Search, Optimization, and Machine Learning. Addison Wesley (1989)
7. Coello, C.C., Veldhuizen, D.V., Lamont, G.: Evolutionary Algorithms for Solving Multi-Objective Problems. Kluwer Academic Publishers (2002)
8. Lefuel, R.: The Effects of GA's with Multiple Populations on Various Representations and its Implementation with Pareto Ranking when Evolving PCFG's (2004) Hons. Thesis, Dept of Computer Science, Brock U.
9. Ross, B., Zhu, H.: Procedural Texture Evolution Using Multiobjective Optimization. New Generation Computing (2004) In press.