# A Parallel Evolutionary Algorithm for Stochastic Natural Language Parsing

L. Araujo

Dpto. Sistemas Informáticos y Programación.
Universidad Complutense de Madrid.
lurdes@sip.ucm.es

**Abstract.** This paper presents a parallel evolutionary program for natural language parsing. The implementation follows an island model, in which, after a number of generations, demes exchange some individuals in a round-robin manner. The population is composed of potential parsings for a sentence, and the fitness function evaluates the appropriateness of the parsing according to a given stochastic grammar. Both the fitness function and the genetic operators, which require that the result of their application still corresponds to the words in the input sentence, are expensive enough to make the evolutionary program appropriate for a coarse grain parallel model and its distributed implementation. The system has been implemented in a parallel machine using the PVM (Parallel Virtual Machine) software. The paper describes the study of the parameters in the parallel evolutionary program, such as the number of individuals to be exchanged between demes, and the number of generations between exchanges. Different parameters of the algorithm, such as population size, and crossover and mutation rates, have also been tested.

## 1 Introduction

Evolutionary algorithms are an efficient method to deal with hard optimization problems [5, 8, 9]. Statistical methods allow posing different aspects of natural language processing as optimization problems of some kind of measurements given by the statistical model. In this way, evolutionary algorithms are a very useful tool to deal efficiently with different problems of natural language processing. Besides, working with statistical models allows an uncertainty margin on the results that matches properly with the nature of the evolutionary algorithms (which do not guarantee the best solution, but one reasonably good).

Natural language parsing is a search problem that requires exploring a tree of possible parsings. The size of this tree increases exponentially with the length of the sentence or text to be parsed. By scoring every parsing in the tree, such a search can be thought of as an optimization problem, since we are looking for the "best" parsing. Stochastic grammars [4, 1] represent an important part of the statistical methods in computational linguistics, which have allowed real progress on a number of issues including disambiguation, error correction, etc. These grammars give us an evaluation of the tree representing a possible parsing

of a sentence, and we will try to find one of the best parsing according to this measurement.

This work develops a stochastic parallel parser for natural language based on an evolutionary algorithm. The approach adopted herein is based on the evolution programming method, as has been extended by Michalewicz [7] to consider a richer set of data structures for chromosome representation than classic genetic algorithms. In our case individuals are possible parsings, which are evaluated in order to give a measure of how they fit the grammar rules.

Despite the ability of genetic algorithms to find a "good" solution, though perhaps approximate, to optimization problems, when such problems are too hard, they require a very long time to reach the solution. This has led to different efforts to parallelizing these programs in order to accelerate the process. Basically, the approaches to this parallelization can be classified in two groups [2, 6]: *global* parallelization, and *island* of *coarse-grained* parallelization. In the first method there is only one population, as in the sequential model, and it is the evaluation of individuals, and the application of genetic operators what is parallelized. In the island model, the population is divided in subpopulation or *demes*, that usually evolve isolated except for the exchange of some individuals or *migrations* after a number of generations. In this case, we can expect a different behaviour of the parallel model, since this model implies a change in some parameters of the algorithm (such as the population size, which is smaller in each deme), what can result in a faster convergence. Though such a faster convergence may, in principle, reduce the quality of the solutions, results shows [3] that the parallel model with smaller populations but with migration among demes can improve the quality of the sequential solutions, and that there is an optimal number of demes which maximizes the performance. We have adopted the island model, what has allowed us developing a portable implementation valid for both distributed and shared memory platforms.

The rest of the paper proceeds as follows: section 2 describes the main elements in the evolutionary algorithm for parsing; section 3 is devoted to the parallel model; section 4 presents and discusses the experimental results and section 5 draws the main conclusions of this work.

## 2 Evolutionary Algorithm for a Probabilistic Parsing

Discovering the meaning of a sentence or text for any application (machine translation, database interfaces, etc) requires extracting their grammatical structure, usually represented in the form of a labeled tree, which indicates how words relate to each other and form phrases or clauses. However, many words can play several grammatical roles, what leads to a forest of possible parsing trees for a sentence. Probabilistic Context Free Grammars (PCFG)[4] provide a probabilistic model of the language, as well as a mechanism to select one parsing if there is ambiguity. A PCFG is a context free grammar whose rules are assigned a probability. In our case, the PCFG provides a measure of the fitness of the individuals in the evolutionary algorithm.

**Individuals** The chromosomes of our evolutionary algorithm represent potential parsings for an input sentence according to a given probabilistic context free grammar (PCFG). The input sentence is given as a sequence of words. The set of categories for each word is searched in a dictionary (lexicon). A chromosome is represented as a data structure containing the following information:

- Fitness of the chromosome.
- A list of *genes*, each representing the parsing of a different set of words in the sentence.
- The number of genes in the chromosome.
- The depth of the parsing tree.

Each gene represents the parsing of a consecutive set of words in the sentence. If this parsing involves no-terminal symbols, the parsing of the subsequent partitions of the set of words is given in later genes. Accordingly, the information contained in a gene is the following:

- The sequence of words in the sentence to be analyzed by the gene.
- The rule of the grammar used to parse the words in the gene.
- If the right hand side of the rule contains no terminal symbols, the gene also stores the list of references to the genes in which the analysis of these symbols proceeds.
- The depth of the node corresponding to the gene in the parsing tree. It will be used in the evaluation function.

The data structure of Figure 1 represents one possible chromosome for the sentence: "the man sings a song".

**Initial Population** In order to create the chromosomes in the initial population, the set of words in the sentence is randomly partitioned, making sure that there is at least one verb in the second part, which corresponds to the main $VP$. The set of words corresponding to the $NP$ is parsed by randomly generating any of the possible $NP$ rules. The same is done for generating the parsing of the $VP$ with the $VP$ rules. The process is improved by enforcing the application of those rules able to parse the right number of words of the gene. The process continues until there are no terminal symbols left pending to be parsed.

### 2.1 Individuals Evaluation

The opportunities of an individual to survive depend on the measurements of their adaptation or fitness. In our case, this measurement is given by a couple of values, $f_{coher}$, which measures the ability of a chromosome to parse the objective sentence, and $f_{prob}$, which measures the probability of the rules employed in the parsing.

$f_{coher}$ is based on the relative number of *coherent* genes. A gene will be considered *coherent* if

| (fitness): X | | |
|---|---|---|
| (number of genes): 4 | | |
| (genes): | | |
| (gene number) | (rule) | (gene decomposition): (first word, number of words, gene): |
| (1) | $S \rightarrow NP, VP$ | NP:(1, 2, 2) |
| | | VP:(3, 3, 3) |
| (2) | $NP \rightarrow Det, Noun$ | Det: *The* |
| | | Noun: *man* |
| (3) | $VP \rightarrow Verb, NP$ | Verb: *sings* |
| | | NP:(4, 2, 4) |
| (4) | $NP \rightarrow Det, Noun$ | Det: *a* |
| | | Noun: *song* |

**Fig. 1.** Data structure of a chromosome. The first gene tells us that the set of words in the sentence has been partitioned by the third word, i.e., the first two words correspond to the main $NP$, and the following three words to the main VP. The gene also tells us that the parsing of the $NP$ is given by the gene 2, and the parsing of the VP is given by the gene 3. Since the rule in the gene 2 has only terminal symbols in its right hand side, there is no gene decomposition. On the contrary, the rule for gene 3 presents an $NP$ symbol in its right hand side, whose parsing is done in gene 4.

a) it corresponds to a rule whose right hand side is only composed of terminal symbols, and they correspond to the categories of the words to be parsed by the rule.
b) it corresponds to a rule with non-terminal symbols in its right hand side and each of them is parsed by a coherent gene.

Accordingly, $f_{coher}$ is computed as

$$f_{coher} = \frac{\text{number of coherent genes} - \sum_{i \in \text{incoherent genes}} \frac{\text{penalization}}{\text{depth}(i)}}{\text{total number of genes}}.$$

The formula takes into account the relative relevance of the genes: the higher in the parsing tree is the node corresponding to an incoherent gene, the worse is the parsing. Thus the fitness formula introduces a penalization factor which decreases with the depth of the gene.

$f_{prob}$ is computed as

$$\prod_{i=1}^{n} \text{Prob}(g_i)$$

where $\text{Prob}(g_i)$ is the probability of the grammatical rule of gene $g_i$ in the chromosome.

Fitness is then computed as a linear combination of both:

$$Fitness = w_{\text{coher}} f_{\text{coher}} + w_{\text{prob}} f_{\text{prob}}$$

where $w_{coher}$ and $w_{prob}$ are parameters that allow tuning the computation along the evolution process. In the first generations $w_{coher}$ is higher in order to produce individuals corresponding to possible parsing trees, while later, $w_{prob}$ becomes higher in order to select the most probable individuals.

### 2.2 Reproduction

Each generation finishes with the creation of new individuals that will substitute other individuals in the population. These new individuals are created by means of the *crossover* and *mutation* operators.

The crossover operator combines two parsings to generate a new one. The part of one parent after a point randomly selected is exchanged with the corresponding part of the other parent to produce two offsprings, under the constraint that the genes exchanged correspond to the same type of parsing symbol (NP, VP, etc) in order to avoid wrong references of previous genes in the chromosome. The operator selects two parent chromosomes. Then, a word is randomly selected from the input sentence, and the inner most gene to which the selected word corresponds in each parent chromosome is identified. If the genes correspond to different sets of words, the next gene in the inner most order is selected. This process continues until the sequences of words whose parsings are to be exchanged are the same, or until the main NP or VP are reached. If the two selected genes parse the same sequence of words, they are exchanged. Otherwise, genes appropriated for the exchange are randomly generated. Mutation is applied to the chromosome resulting of the crossover operation with a probability given by the mutation rate, an input parameter. In this case, a new parsing is generated for a randomly selected gene. At each generation a number of chromosomes equal to the number of offsprings is selected to be replaced. The selection of chromosomes to be replaced in each generation is performed with respect to the relative fitness of the individuals: a chromosome with a worse than average fitness has higher chances to be selected for replacement.

## 3 Parallel Model and Implementation

We have adopted an island model, which may be implemented in both shared or distributed memory architectures, thus making the model portable. The design of the parallel model is intended to reduce the communications. To this purpose, the following options have been chosen:

- *System components*: The system is composed of a number of processes, called *cooperative parsers*, each of which performs, by evolutionary programming, a parsing for the same input sentence. The initial condition in each deme (random number generation) is different in order to obtain different individuals in different populations. There is a special process, known as *main selector*, which selects the best individual among the best ones of each deme.

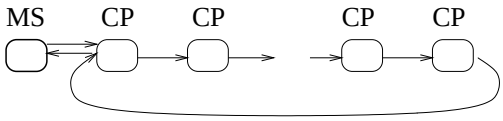


**Fig. 2.** Migration Policy. CP stands for Cooperative parser, MS for Main Selector.

- *Migration policy*: To reduce communications, migrations do not take place from one deme to any other, but only to the next one (Figure 2). The $N$ cooperative parsers thus form a ring. Nevertheless, this policy has been compared with an all-to-all policy so as to make sure that this choice does not imply a significant reduction in the quality of the solutions.
- *Synchronism*: We have adopted an asynchronous model, in which a cooperative parser, after a fixed number of generations, sends a fixed number of individuals to the next *cooperative parser* and then continues the evolution, checking in each generation the arrival of the same number of individuals from the previous parser.
- *Convergence policy*: Again with the aim of reducing communications, a *cooperative parser* which reaches convergence sends its best individual to the *main selector*. The main selector takes this solution as the absolute best, giving it to the user and finishing after killing all cooperative parsers. If no parser which reaches the convergence, all of them finish after a number of generations fixed given by the user. Then each one sends its best solution to the selector, which chooses the best among them for the user.
- *Criteria for selection of individuals to migrate*: They are randomly chosen with a probability proportional to their fitness.
- *Criteria for selection of individuals to be replaced by the 'immigrants'*: They are randomly chosen with equal probability.

## 4 Experiments

The algorithm has been implemented on C++ language with the software PVM on a SGI-Cray ORIGIN 2000. In order to evaluate the performance we have considered the parsing of the sentences appearing in Table 1. The average length of the sentences is around 10 words. However, they present different complexities for the parsing, mainly the length and the number of subordinate phrases.

| | |
|---|---|
| 1 | Jack(noun) regretted(verb) that(wh) he(pro) ate(verb) the(det) whole(adj) thing(noun) |
| 2 | The(det) man(noun) who(wh) gave(verb) Bill(noun) the(det) money(noun) drives(verb) a(det) big(adj) car(noun) |
| 3 | The(det) man(noun) who(wh) lives(verb) in(prep) the(det) red(adj) house(noun) saw(verb) the(det) thieves(noun) in(prep) the(det) bank(noun) |

**Table 1.** Sentences used in the parsing experiments.

### 4.1 Study of the Evolutionary Algorithm Parameters

The parameters of the algorithm determine the influence of two fundamental factors in the success of an evolutionary algorithm: population diversity and selective pressure. The most relevant of them, population size and crossover and mutation rates, have been studied in detail.

Figure 3 shows the number of generations required to reach a correct parsing for each sentence versus the population size, in a sequential execution. In general, the higher the "sentence complexity", the larger the population size required to reach the correct parsing in a reasonable number of steps. The sentence complexity depends on its length and on the number of subordinate phrases it contains. However, large populations lead to replication of chromosomes and slow evolutions, so high percentages of genetic operators are required in order to accelerate the process.

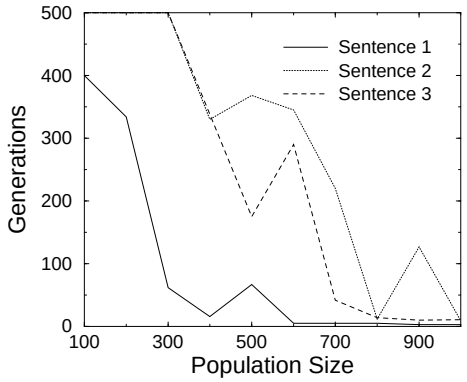


**Fig. 3.** Number of generations required to reach the correct parsing for different input sentences, when using crossover rate of 50% and mutation rate of 20%.

### 4.2 Evaluating the Parallel Model

| Sentence | Sequential | Parallel | | | | |
|---|---|---|---|---|---|---|
| | | 2 Proc. | 4 Proc. | 6 Proc. | 8 Proc. | 10 Proc. |
| sentence1 | 16.55 | 10.48 | 3.08 | 3.09 | 2.09 | 2.09 |
| sentence2 | 50.03 | 19.12 | 15.02 | 10.64 | 3.48 | 3.49 |
| sentence3 | 52.70 | 25.40 | 22.71 | 19.34 | 14.93 | 14.79 |

**Table 2.** Time in seconds required to reach a correct parsing when processing sequentially and in parallel.

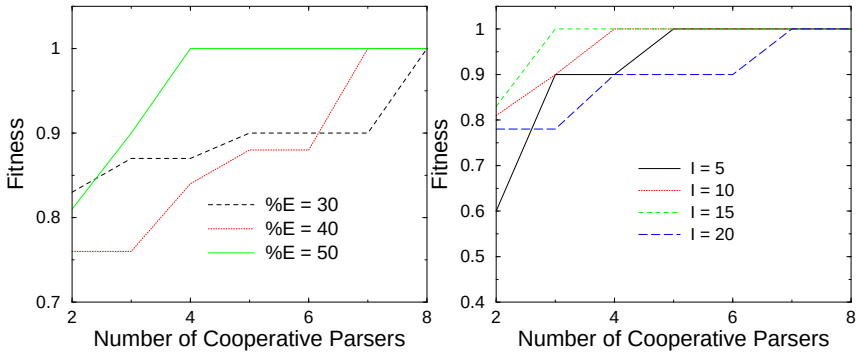


**Fig. 4.** Performance of the sentence 2 for different sizes of the migrating population, with a population of 50 individual per processor, a crossover rate of 40%, a mutation rate of 20% and an interval between migrations of 10 generations.

**Fig. 5.** Performance of the sentence 2 for different intervals of migration, with a population of 50 individual per processor, a crossover rate of 40%, a mutation rate of 20% and a migrating population of 50%.

Table 2 shows the improvement in performance obtained by increasing the number of processors. This experiment has been carried out with a population size of 200 individuals (the minimum required for the sequential version to reach the correct parsing), a crossover rate of 50%, a mutation rate of 20%, a migrating population of 40 individuals and an interval of migration of 15 generations. We can observe that the parallel execution achieves a significant improvement even with just 2 processors. We can also observe that saturation is reached for some number of processors. However, this number is expected to increase with the complexity of the sentences to be parsed.

### 4.3 Tuning the Parallel Model

There is a number of options that have to be fixed in order to completely specify the model: rate of the population migrated, number of generations between migrations, parameters of each cooperative parser (size of the deme, crossover and mutation rates, and maximum number of generations), which determine the behaviour of each isolated parser.

**Size of the Migrating Population and Migration Interval** Figure 4 shows the results obtained for different sizes of the migrating population for the sentence 2. Results show that a small population, such as that of 50 individuals used in this experiment, requires a large size of the migrating population (50%), in order to reach a high fitness.

The interval of migration is another parameter to fix in the parallel model, closely connected with the size of the migrating population. Experiments have been carried out in order to determine the best values, once the size of the

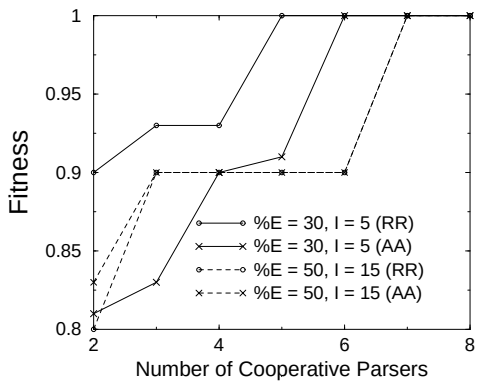


**Fig. 6.** Comparison of two different communication policies: round-robin (RR) and all-to-all (AA). Results correspond to a population of 50 individual per processor, a limit of 100 generations, a crossover rate 40% and a mutation rate of 20. In the graphics, E stands for Exchange rate and I for migration Interval. Fitness corresponds to the $f_$coher value.

migrating population has been fixed. Figure 5 shows the results obtained for the sentence 2. We can observe that the best results are obtained with an interval of 15 generations. On the contrary, the worst results are obtained for too frequent exchanges (I=5) and also for too spread exchanges (I=20).

**Round-Robin or All-to-all** We have investigated two different policies migration:

- *Round-Robin* policy: cooperative parsers are organized in a ring sequence, each of them sending the migrating population to the next one in the sequence and receiving it from the preceding one.
- *All-to-all* policy: Each process sends the migrating population to all the other processes and receives population from all of them. In this case, each process sends a migrating population equal to the migrating population size divided by the number of processes in the system.

Figure 6 shows the results obtained with both policies. We can observe that results obtained with both policies are comparable, the round-robin policy slightly outperforming the all-to-all one. Therefore, the round-robin policy is more sensible because with a similar performance considerably reduces communications.

## 5 Conclusions

The complexity of the parsing process for sentences of natural languages makes it appropriate to use optimization methods such as evolutionary algorithms which provide an approximate solution in a reasonable time. This work presents an evolutionary algorithm which works with a population of potential parsings for a

given Probabilistic Context Free Grammar and an input sentence. Probabilistic grammars allow weighting the possible parsings, thus providing a method to evaluate individuals. In this work, the evaluation is given by a combination of the coherence of the parsing, i.e. its ability to match every word and syntactic tag with the grammar rules, and the probability computed from the probabilities of the rules applied.

Results from a number of tests indicate that the evolutionary approach is robust enough to deal with the parsing problem. The tests indicate that the GA parameters need to be suitable for the input sentence complexity. The more complex the sentence (length and subordination degree), the larger the population size required to quickly reach a correct parsing.

The evolutionary algorithm has been parallelized in the form of an island model, in which processors exchange migrating populations asynchronously and in a round-robin sequence. Experiments on the size of the migrating population show that it is necessary to exchange a significant rate of the population. Experiments on the migration intervals show that exchanges should be neither too frequent nor too sparse. Results obtained for these experiments exhibit a clear improvement in the performance, thus showing that the problem has enough granularity for the parallelization, even when applied to artificial sentences. It is expected this improvement to be greater when applied to real sentences as those extracted from a linguistics corpus.

## References


1. L. Araujo. Evolutionary parsing for a probabilistic context free grammar. In *Proc. of the Int. Conf. on on Rough Sets and Current Trends in Computing (RSCTC-2000)*, 2000.
2. E. Cantú-Paz. A survey of parallel genetic algorithms. Technical report, Illinois Genetic Algoritms Laboratory, IlliGAL Report No. 97003, 1997.
3. E. Cantu-Paz and D. E. Goldberg. Predicting speedups of idealized bounding cases of parallel genetic algorithms. In T. Back, editor, *Proceedings of the Seventh International Conference on Genetic Algorithms*, pages pp. 113–120. CA: Morgan Kaufmann, 1997.
4. E. Charniak. *Statistical Language Learning.* MIT press, 1993.
5. David E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning.* Addison-Wesly, 1989.
6. R. Poli M. Nowostawski. Review and taxonomy of parallel genetic algorithms. Technical report, School of Computer Science, The University of Birmingham, UK, Technical Report CSRP-99-11, 1999.
7. Z. Michalewicz. *Genetic algorithms + Data Structures = Evolution Programs.* Springer-Verlag, 2nd edition, 1994.
8. A. Ruiz-Andino, L. Araujo, J. Ruz, and F. Sáenz. Parallel evolutionary optimization with constraint propagation. In *Proc. of the Int. Conf. on Parallel Problem Solving from Nature (PPSN)*, volume 1498 of *Lecture Notes in Computer Science*, pages 270–279. Springer-Verlag, 1998.
9. A. Ruiz-Andino, L. Araujo, F. Sáenz, and J. Ruz. A hybrid evolutionary approach for solving constrained optimization problems over finite domains. *IEEE Transactions on Evolutionary Computation*, 4(4):353–372, 2000.