Metaheuristics for Natural Language Tagging

Lourdes Araujo[1], Gabriel Luque[2], and Enrique Alba[2]

[1] Dpto. Sistemas Informáticos y Programación,
Facultad de Informática, Univ. Complutense,
28040 Madrid, SPAIN
lurdes@sip.ucm.es
[2] Dpto. de Lenguajes y Ciencias de la Computación,
E.T.S. Ingeniería Informática,
Campus Teatinos, 29071, Málaga, SPAIN
{eat,gabriel}@lcc.uma.es

Abstract. This work compares different metaheuristics techniques applied to an important problem in natural language: tagging. Tagging amounts to assigning to each word in a text one of its possible lexical categories (tags) according to the context in which the word is used (thus it is a disambiguation task). Specifically, we have applied a classic genetic algorithm (GA), a CHC algorithm, and a Simulated Annealing (SA). The aim of the work is to determine which one is the most accurate algorithm (GA, CHC or SA), which one is the most appropriate encoding for the problem (integer or binary) and also to study the impact of parallelism on each considered method. The work has been highly simplified by the use of MALLBA, a library of search techniques which provides generic optimization software skeletons able to run in sequential, LAN and WAN environments. Experiments show that the GA with the integer encoding provides the more accurate results. For the CHC algorithm, the best results are obtained with binary coding and a parallel implementation. SA provides less accurate results than any of the evolutionary algorithms.

1 Introduction

Part of speech tagging is one of the basic tasks in natural language processing. Tagging amounts to assigning to each word of a sentence one of its possible lexical categories according to the context in which the word is used. For instance, the word *can* can be a noun, an auxiliary verb or a transitive verb. The category assigned to the word will determine the structure of the sentence in which it appears and thus its meaning. In fact, tagging is a necessary step for parsing, for information retrieval systems, for speech recognition, etc. Moreover, tagging is a difficult problem since, many words belong to more than one lexical class. To give an idea, according to [7], over 40% of the words appearing in the hand-tagged Brown corpus [11] are ambiguous.

Because of the importance and difficulty of this task, a lot of work has been carried out to produce automatic taggers. Automatic taggers [5, 10, 12], usually based on Hidden Markov Models, rely on statistical information to establish the

probabilities of each scenario. The statistical data are extracted from previously tagged texts, called *corpus*. These stochastic taggers neither require knowledge of the rules of the language nor try to deduce them, and thus they can be applied to texts in any language, provided they can be trained on a corpus for that language previously.

The context in which the word appears helps to decide which is its more appropriate tag, and this idea is the basis for most taggers. For instance, consider the sentence in Figure 1, extracted from the Brown corpus. The word *questioning* can be disambiguated as a common name if the preceding tag is disambiguated as an adjective. But it might happen that the preceding word were ambiguous, so there may be many dependencies which must be resolved simultaneously.

This	the	therapist	may	pursue	in	later	questioning	.
DT	AT	NN	NNP	VB	RP	RP	VB	.
QL			MD	VBP	NNP	RB	NN	
					RB	JJ	JJ	
					NN	JJR		
					FW			
					IN			

Fig. 1. Tags for the words in a sentence extracted from the Brown corpus. Underlined tags are the correct ones, according to the Brown corpus. Tags correspond to the tag set defined in the Brown corpus: DT stands for determiner/pronoun, AT for article, NN for common noun, MD for modal auxiliary, VB for uninflected verb, IN for preposition, JJR for comparative adjective, etc.

The statistical model considered in this work amounts to maximize a global measure of the probability of the set of contexts (a tag and its neighboring tags) corresponding to a given tagging of the sentence. Then, we need a method to perform the search of the tagging which optimizes this measure of probability.

The aim of this work is to check and compare two different variants of evolutionary algorithms to perform such a search: a classic genetic algorithm (GA) and a CHC algorithm. Genetic algorithms have been previously applied to the problem [3, 4], obtaining accuracies as good of those of typical algorithms used for stochastic tagging (such as the widely used of Viterbi [5]) or even better [4]. CHC is a non-traditional genetic algorithm, which presents some particular features. CHC guarantees the survival of the best individuals found by putting the children and parents together and applying selection among them. Similar individuals are not allowed to mate in order to improve diversity. Crossover also differs in such a way that two parents exchange exactly half of the differing parental genes and instead of traditional mutation, CHC re-initializes the population when stagnation is detected.

One of the aims of this work is to investigate if the particular mechanism of CHC for diversity can improve the selection of different sets of tags. From previous work, it has been observed that words incorrectly tagged are usually those which require one of their more rare tags, or which appear in an infre-

quent context. The fitness function is based on the probability of the contexts of a sequence of tags assigned to a sentence. Therefore, it is difficult for the GA to change tags within high probability contexts. CHC allows changing simultaneously several tags of the sequence, which can lead to explore combinations of tags very different from those of the ancestors. Thus, it is interesting to study what is more advantageous, the quiet exploration of the GA or the more disruptive one of CHC. We have also compared the results of the GAs with those obtained from Simulated Annealing (SA), in order to ascertain the suitability of the evolutionary approach compared with other optimization methods.

For most tagging applications, the whole process of search is time consuming, what made us to include in the study a parallel version of the algorithms.

The work has been highly simplified by the use of MALLBA [1], a library of search techniques, which provides generic optimization software skeletons that run in sequential, LAN and WAN environments.

The rest of the paper proceeds as follows: Section 2 is devoted to present the MALLBA system, under which the algorithms have been implemented. Sections 3, 4 and 5 describe the GA, CHC, and SA algorithms, and Section 6 discusses the parallel version of these algorithms. Section 7 presents the details of the algorithms as applied to tagging, including the genetic operators. Section 8 describes and discusses the experimental results, and Section 9 draws the main conclusions of this work.

2 MALLBA System

The MALLBA project [1] provides, in an integrated way, a library of skeletons for combinatorial optimization (including exact, heuristic and hybrid methods) which can deal with parallelism in a user-friendly and, at the same time, efficient manner. Its three target environments are sequential computers, LANs of workstations and WANs. Skeletons are generic templates which must be instantiated with the features of the problem to solve. The features related to the selected generic resolution method and its interaction with the particular problem are implemented by the skeleton.

Skeletons are implemented by a set of *required* and *provided* C++ classes that represent an abstraction of the entities participating in the solver method. The *provided* classes implement internal aspects of the skeleton in a problem-independent way. The *required* classes specify information and behavior related to the problem. This conceptual separation allows us to define required classes with a fixed interface but without any implementation, so that provided classes can use required classes in a generic way. MALLBA is publicly available at `http://neo.lcc.uma.es/mallba/easy-mallba/index.html`.

3 Genetic Algorithm

Genetic Algorithms (GAs) [6] are stochastic search methods that have been successfully applied in many real applications of high complexity. A GA is an

iterative technique that applies stochastic operators on a pool of individuals (tentative solutions). An evaluation function associates a value to every individual indicating its suitability to the problem. Traditionally, GAs are associated to the use of a binary representation, but nowadays you can find GAs that use other types of representations. A GA usually applies a recombination operator on two solutions, plus a mutation operator that randomly modifies the individual contents to promote diversity.

```
1   t = 0
2   initialize P(t)
3   evaluate structures in P(t)
4   while not end do
5       t = t + 1
6       select: C(t) = P(t-1)
7       for each pair (p1,p2) in C(t)
8           if 'incest prevention condition'
9               add to C'(t) HUX(p1,p2)
10      evaluate structures in C'(t)
11      replace P(t) from C"(t) and P(t-1)
12      if convergence(P(t))
13          re-start P(t)
```

Fig. 2. Scheme of the CHC algorithm

4 CHC Algorithm

CHC [8] is a variant of genetic algorithm with a particular way of promoting diversity. It uses a highly disruptive crossover operator to produce new individuals maximally different from their parents. It is combined with a conservative selection strategy which introduces a kind of inherent elitism. Figure 2 shows a scheme of the CHC algorithm, whose main features are:

- The mating is not restricted to the best individuals, but parents are randomly paired in a mating pool $C(t)$ (line 6 of Figure 2). However, recombination is only applied if the Hamming distance between the parents is above a certain threshold, a mechanism of *incest prevention* (line 8 of Figure 2).
- CHC uses a *half-uniform crossover* (HUX), which exchanges exactly half of the differing parental genes (line 9 of Figure 2).
- Traditional selection methods do not guarantee the survival of best individuals, though they have a higher probability to survive. On the contrary, CHC guarantees survival of the best individuals selected from the set of parents ($P(t-1)$) and offsprings ($C'(t)$) put together (line 11 of Figure 2).
- Mutation is not applied directly as an operator.
- CHC applies a re-start mechanism if the population remains unchanged for some number of generations (lines 12-13 of Figure 2). The new population includes one copy of the best individual, while the rest of the population is generated by mutating some percentage of bits of such best individual.

5 Simulated Annealing

Simulated Annealing (SA) [9] is a stochastic optimization technique, which has its origin in statistical mechanics. It is based upon a cooling procedure used in industry. This procedure heats the material to a high temperature so that it becomes a liquid and the atoms can move relatively freely. The temperature is then slowly lowered so that at each temperature the atoms can move enough to begin adopting the most stable configuration. In principle, if the material is cooled slowly enough, the atoms are able to reach the most stable (optimum) configuration. This smooth cooling process is known as *annealing*. Figure 3 shows a scheme of SA. First at all, the parameter T, called the temperature, and the solution, are initialized (lines 2-4). The solution $s1$ is accepted as the new current solution if $\delta = f(s1) - f(s0) < 0$. Stagnations in local optimum are prevented by accepting also solutions which increase the objective function value with a probability $exp(-\delta/T)$ if $\delta > 0$. This process is repeated several times to obtain good sampling statistics for the current temperature. The number of such iterations is given by the parameter $Markov\_Chain\_length$, whose name alludes the fact that the sequence of accepted solutions is a Markov chain (a sequence of states in which each state only depends on the previous one). Then the temperature is decremented (line 14) and the entire process repeated until a frozen state is achieved at T_{min} (line 15). The value of T usually varies from a relatively large value to a small value close to zero.

```
1   t = 0
2   initialize(T)
3   s0 = Initial_Solution()
4   v0 = Evaluate(s0)
5   repeat
6       repeat
7           t = t + 1
8           s1 = Generate(s0,T)
9           v1 = Evaluate(s0,T)
10          if Accept(v0,v1,T)
11              s0 = s1
12              v0 = v1
13      until t mod Markov_Chain_length == 0
14      T = Update(T)
15  until 'loop stop criterion' satisfied
```

Fig. 3. Scheme of the Simulated Annealing (SA) algorithm

6 Parallel Heuristics

A parallel EA (PEA) is an algorithm having multiple component EAs, regardless of their population structure. Each component (usually a traditional EA)

subalgorithm includes an additional phase of *communication* with a set of subalgorithms [2]. In this work, we have chosen a *distributed EA* (dEA) because of its popularity and because it can be easily implemented in clusters of machines. In distributed EAs (also known as Island Model) there exists a small number of islands performing separate EAs, and periodically exchanging individuals after a number of isolated steps (*migration frequency*). Concretely, we use a static ring topology in which the best individual is migrated, and asynchronously included in the target populations only if it is better than the local worst-existing solutions.

For the parallel SA (PSA) there also exist multiple asynchronous component SAs. Each component SA, start off from a different random solution, exchanges the best solution found (*cooperation* phase) with its neighbor SA in the ring.

7 The Model for Tagging

The generated tagger must be able to learn from a training corpus so as to produce a table of rules (contexts) called *training table*. Evaluation of tentative solutions is done according to the training table. This table records the different contexts of each tag. The table can be computed by going through the training text and recording the different contexts and the number of occurrences of each of them for every tag in the training text. For example, if we consider contexts with two tags on the left and two tags on the right, the entry in the table for tag *JJ* could have the form:

```
JJ 4557 9519

VBD AT JJ NN IN 37
IN PP$ JJ NNS NULL 20
PPS BEZ JJ TO VB 18
NN IN JJ NN WDT 3
       ...
```

denoting that *JJ* has 4557 different contexts and appears 9519 times in the text, and that in one of those contexts, which appears 37 times, *JJ* is preceded by tags *VBD* and *AT* and followed by *NN* and *IN*, and so on until all the 4557 different contexts have been listed.

The search process is run for each sentence in the text to be tagged. Improvement steps aim to maximize the total probability of the tagging of the sentences in the test corpus. The process finishes either if the fitness deviation lies below a threshold value (convergence) or if the evolutionary process has been running for a maximum number of generations.

7.1 Individuals

Tentative solutions are sequences of genes which correspond to each word in the sentence to be tagged. Figure 4 shows some possible individuals for the sentence in Figure 1. Each gene represents a tag and additional information useful in the

Sentence	This	the	therapist	may	pursue	in	later	questioning	.
Ind. 1:	DT	AT	NN	NNP	VBP	IN	JJ	VB	.
Ind. 2:	DT	AT	NN	MD	VB	RB	RB	NN	.
Ind. 3:	QL	AT	NN	NNP	VB	FW	JJ	JJ	.

Fig. 4. Potential individuals for the sentence in Figure 1

evaluation of the chromosome, such as counts of contexts for this tag according to the training table. Each gene's tag is represented by an index to a vector which contains the possible tags of the corresponding word. The composition of the genes depends on the chosen coding, as Figure 5 shows. In the integer coding the gene is just the integer value of the index. In the binary coding the gene is the binary representation of the index. As in the texts we have used for experiments the maximum number of tags per word is 6, we have used both a binary code of 7 and another one of 4 bits.

word	**tag index**							**integer**	**binary(7)**	**binary(4)**
	0	1	2	3	4	5	⋯			
This	DT	QL						0	0000000	0000
the	AT							0	0000000	0000
therapist	NN							0	0000000	0000
may	NNP	MD						1	0000001	0001
pursue	VB	VBP						0	0000000	0000
in	RP	NNP	RB	NN	FW	IN		5	0000101	0101
later	RP	RB	JJ	JJR				3	0000011	0011
questioning	VB	NN	JJ					1	0000001	0001

Fig. 5. Integer and binary codings (7 and 4 bits) of a possible selection of tags chosen for the words of a sentence extracted from the Brown corpus. The selected tags appear underlined.

Initial Population For a given sentence of the test corpus, the chromosomes forming the initial population are created by randomly selecting from a dictionary one of the valid tags for each word, with a bias to the most probable tag. Words not appearing in the dictionary are assigned the most probable for its corresponding context, according to the training text.

7.2 Fitness Evaluation

The fitness of an individual is a measure of the total probability of its sequence of tags, according to the data from the training table. It is computed as the sum of the fitness of its genes, $\sum_i f(g_i)$. The fitness of a gene is defined as

$$f(g) = \log P(T|LC, RC)$$

where $P(T|LC, RC)$ is the probability that the tag of gene g is T, given that its context is formed by the sequence of tags LC to the left and the sequence

RC to the right (the logarithm is taken in order to make fitness additive). This probability is estimated from the training table as

$$P(T|LC, RC) \approx \frac{occ(LC, T, RC)}{\sum_{T' \in \mathcal{T}} occ(LC, T', RC)}$$

where $occ(LC, T, RC)$ is the number of occurrences of the list of tags LC, T, RC in the training table, and $\mathcal{T}$ is the set of all possible tags of g_i. For example, if we are evaluating the individual 1 of Figure 4 and we are considering contexts composed of one tag on the left and one tag on the right of the position evaluated, the fourth gene (word *may*), for which there are two possible tags, NNP (the one chosen in this individual) and MD, will be evaluated as

$$\frac{\#(\text{NN NNP VBP})}{[\#(\text{NN NNP VBP}) + \#(\text{NN MD VBP})]}$$

where # represents the number of occurrences of the context.

A particular sequence LC, T, RC may not be listed in the training table, either because its probability is strictly zero (if the sequence of tags is forbidden for some reason) or, most likely, because there is insufficient statistics. In these cases we proceed by successively reducing the size of the context, alternatively ignoring the rightmost and then the leftmost tag of the remaining sequence (skipping the corresponding step whenever either RC or LC are empty) until one of these shorter sequences matches at least one of the training table entries or until we are left simply with T. In this latter case we take as fitness the logarithm of the frequency with which T appears in the corpus (also contained in the training table).

7.3 Genetic Operators

For the GA, we use a one point crossover, i.e. a crossover point is randomly selected and the first part of each parent is combined with the second part of the other parent thus producing two offsprings. Then, a mutation point is randomly selected and the tag of this point is replaced by another of the valid tags of the corresponding word. The new tag is randomly chosen according to its probability (the frequency it appears in the corpus).

The CHC algorithm applies HUX crossover, randomly taking from each parent half of the tags in which they differ and exchanging them.

Individuals resulting from the application of genetic operators along with the old population are used to create the new one.

8 Experiments

We have used as the set of training texts for our taggers the Brown corpus [11], one of the most widespread in linguistics. The tag set of this corpus is not too large, what favours the accuracy of the system. Moreover, this tag set has been reduced by grouping some related tags under a unique name tag, what

improves statistics. For instance, different kinds of adjectives (JJ, $JJ + JJ$, JJR, $JJR + CS$, JJS, JJT) distinguished in the corpus have been grouped under the common tag JJ.

The CHC algorithm has been run with a crossover rate of 50%, without mutation. Whenever convergence is achieved, 90% of population is renewed. The GA applies the recombination operator with a rate of 50%, and the mutation operator with a rate of 5%. In the parallel version, the migration occurs every 10 generations. We made several tests with different parameter settings for determining the best values for each algorithms. The analysis of other specific operators is defered for a future work.

Table 1. Tagging accuracy obtained with the CHC algorithm for a test text of 2500 words. PS stands for Population Size.

Context	**CHC-Int**				**CHC-Bin(7)**				**CHC-Bin(4)**			
	PS = 20		PS = 56		PS = 20		PS = 56		PS = 20		PS = 56	
	Seq.	**Par.**	**Seq.**	**Par.**	**Seq.**	**Par.**	**Seq.**	**Par.**	**Seq.**	**Par.**	**Seq.**	**Par.**
1-0	89.96	90.15	89.34	89.34	92.08	92.17	91.04	90.95	91.94	**92.53**	91.18	91.35
2-0	91.41	91.68	90.91	91.32	93.34	93.43	92.35	91.90	93.38	**93.52**	92.04	92.40
3-0	92.58	92.89	91.68	91.72	93.74	93.97	92.98	93.07	93.92	**93.97**	93.16	93.25
1-1	93.12	93.48	92.39	92.58	94.78	94.97	93.88	94.06	**95.14**	94.82	94.06	94.10
2-1	93.56	93.70	93.07	93.21	94.51	94.51	93.83	94.06	94.47	**94.65**	93.83	94.15
2-2	94.51	94.11	94.29	93.98	94.61	94.73	94.78	94.78	95.01	**95.23**	94.06	94.87

Tables 1 and 2 show the results obtained with the CHC and GA algorithms, using both, integer and binary codings. In order to study the impact of the length of the code for the binary representation, we have used 7 bits and a 4 bits codes (which are enough, because the maximum number of possible tags of a word is 6). Each row in tables corresponds to a kind of context: 1-0 is a context which considers only the tag of the preceding word, 1-1 considers the tag of the preceding and succeeding words, etc. Figures represent the best result out of twenty independent runs. The globally best result for each row appears in boldface. **Int** stands for the integer representation, **Bin(7)** for the binary representation with a code of 7 bits, and **Bin(4)** for binary with a code of 4 bits. Measures have been taken for different population sizes. Furthermore, sequential and parallel versions with 4 islands are analyzed. The population size of each island is the global population size divided by the number of islands.

Looking at Table 1, the first conclusion is that the binary coding always achieves a higher accuracy than the integer one. Moreover, the shorter the binary code to represent the tag, the better. This suggests that the integer representation is not appropriate for CHC, probably because the low number of genes of the latter limits the CHC mechanism to avoid crossover between similar individuals. Regarding the parallel executions, we can observe that the parallel version usually provides more accurate results, particularly for the population of 20 individuals, because for such a small population the higher diversity introduced by parallelism is beneficial.

Table 2. Accuracy obtained with the GA for a test text of 2500 words. PS stands for Population Size.

Context	**GA-Int**				**GA-Bin(7)**				**GA-Bin(4)**			
	PS = 20		PS = 56		PS = 20		PS = 56		PS = 20		PS = 56	
	Seq.	**Par.**	**Seq.**	**Par.**	**Seq.**	**Par.**	**Seq.**	**Par.**	**Seq.**	**Par.**	**Seq.**	**Par.**
1-0	**93.30**	93.12	92.94	92.67	92.84	92.48	92.53	92.39	93.07	92.44	92.35	92.21
2-0	**93.97**	93.70	93.43	93.88	93.83	93.56	93.61	93.16	93.43	92.89	93.61	93.07
3-0	**94.47**	94.38	93.88	93.79	94.19	94.01	94.28	93.65	93.97	93.52	93.83	93.70
1-1	94.78	94.87	94.92	94.42	**95.14**	94.37	94.64	94.42	94.64	94.01	94.64	94.06
2-1	94.69	**95.19**	94.69	94.87	94.87	94.69	94.55	94.78	94.64	94.37	94.69	94.28
2-2	95.19	95.23	95.14	94.83	**95.54**	94.91	94.96	95.14	95.41	94.64	94.73	95.00

Table 2 shows the results obtained with the GA. In this case, the integer representation provides the best results. The parallel version for a population size of 20 individuals is not able to improve the sequential results, because of the small size of the islands. However, for the population size of 56 individuals the parallel results improve the sequential ones for some contexts.

Comparing both tables, 1 and 2, we can observe that the GA has reached the globally best results for most kinds of contexts (1-0, 2-0, 3-0, 2-1 and 2-2), though the differences are small. This shows that the exploration of the search space given by the classical crossover and mutation operators are enough for this specific problem.

Table 3. Accuracy obtained with the SA algorithm for a test text of 2500 words (best result out of twenty independent runs)

	Context type					
	1-0	2-0	3-0	1-1	2-1	2-2
Seq.	91.32	92.40	92.98	94.24	94.06	94.60
Par.	91.00	92.53	92.67	93.47	94.15	94.33

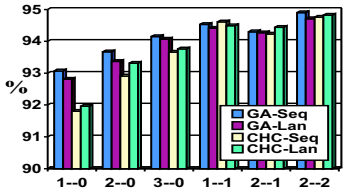
Table 3 presents the data obtained with the SA algorithm. The SA algorithm performs 5656 iterations using a Markov chain of length 800 and with a decreasing factor of 0.99. In the parallel version, each SA component exchanges the best solution found with its neighbor SA in the ring, every 100 iterations. We can observe that SA provides worse results than any of the evolutionary algorithms, thus proving the advantages of the evolutionary approach.

Table 4 presents the average and standard deviation for the codings which provide the best results of each algorithm (integer for GA and binary of 4 bits for CHC), in both the sequential and the parallel implementations. We can observe that fluctuations in the accuracy of different runs are within a 1% interval, so we can claim that the algorithm is very robust.

Another feature of the results that is worth mentioning is that the accuracy obtained, around 95%, is a very good result [5] according to the statistical model used. To our knowledge, the results reported here outperforms in accuracy and efficiency to any existing work on tagging English texts with heuristics methods.

Table 4. Average and standard deviation of the accuracy for a population of 20 individuals and a test text of 2500 words

Context	GA-Int		CHC-Bin(4)	
	Seq.	Par.	Seq.	Par.
1-0	93.07 ± 0.24	92.81 ± 0.23	91.79 ± 0.16	91.96 ± 0.30
2-0	93.68 ± 0.23	93.37 ± 0.18	92.91 ± 0.28	93.31 ± 0.16
3-0	94.15 ± 0.32	94.07 ± 0.20	93.66 ± 0.16	93.76 ± 0.18
1-1	94.54 ± 0.13	94.41 ± 0.23	94.61 ± 0.31	94.49 ± 0.24
2-1	94.31 ± 0.26	94.27 ± 0.25	94.23 ± 0.21	94.44 ± 0.17
2-2	94.91 ± 0.26	94.72 ± 0.35	94.76 ± 0.15	94.82 ± 0.32



We must take into account that the accuracy is limited by the statistical data provided to the search algorithm. Moreover, the goal of the model is to maximize the probability of the context composed by the tags assigned to a sentence, but it is only an approximate model. The correct tag for a word is not always (though most times) the most probable one, and the algorithm captures this fact, but sometimes it is not the one which provides the most probable context either, and it is just in these cases when the tagger fails.

Table 5. Ratios of the execution times of the different versions of the algorithms with respect to the execution time of the sequential GA with integer representation and 1-0 context (17.2805 s.)

Context	GA-Int		GA-Bin(4)		CHC-Int		CHC-Bin(4)	
	Seq	**Par.**	**Seq**	**Par.**	**Seq**	**Par.**	**Seq**	**Par.**
1-0	1	0.899	1.999	1.349	1.002	0.603	1.849	1.308
2-0	4.085	2.160	11.043	6.258	3.749	1.832	10.027	5.436
3-0	18.695	6.138	55.394	18.998	17.003	5.384	47.328	16.490
1-1	4.493	1.866	13.152	5.502	4.450	1.857	11.850	5.480
2-1	21.250	7.206	67.692	23.467	21.014	5.566	60.878	20.655
2-2	96.434	25.344	268.559	52.860	95.742	26.349	247.645	68.255

Table 5 shows the average execution time for the integer and binary (4 bits) codings of the GA and CHC algorithms, which respectively provided the best results for each of them. We can observe that the execution time increases with the size of the context. We can also observe that CHC is slightly faster than GA when using the same codification. Probably, the lack of mutation in CHC compensates its additional computations. Binary codings are slower than integer ones, because they require a decodification step to apply the fitness function. The table also shows that the parallel implementation reduces the execution time, and this reduction is increasingly beneficial with the size of the context.

9 Conclusions

This work compares different optimization methods to solve an important natural language task: the selection for each word in a text of one of its possible lexical categories. The optimization methods considered here have been a classic genetic algorithm (GA), a CHC algorithm and a simulated annealing (SA). The implementation of each of them has been carried out with MALLBA.

Results obtained allow extracting a number of conclusions, such as that the integer coding performs better than the binary one for the GA, while the binary one is the best for the CHC algorithm. Furthermore, the shorter the binary code for the tag of each word, the better the performance. Parallelism has also proven useful, allowing to obtain the best results even with small populations in the case of the CHC, and to reduce the execution time in any algorithm. The GA has been found to be slightly better than CHC, indicating that the exploration of the search space achieved by the classical genetic operators is enough for this problem. The two evolutionary algorithms have outperformed SA.

For the future, we plan to extend this study to additional corpus such as the Susanne and Penn Treebank.

Acknowledgments

The first author has been supported by the project TIC2003-09481-C04. The two last authors have been partially funded by the Spanish MCyT and FEDER under contracts TIC2002-04498-C05-02 (the TRACER project).

References

1. E. Alba, F. Almeida, M. J. Blesa, J. Cabeza, C. Cotta, M. Díaz, I. Dorta, J. Gabarró, C. León, J. Luna, L. M. Moreno, C. Pablos, J. Petit, A. Rojas, and F. Xhafa. MALLBA: A library of skeletons for combinatorial optimisation. In *Euro-Par*, pages 927–932. Springer, 2002.
2. E. Alba and M. Tomassini. Parallelism and evolutionary algorithms. *IEEE Transactions on Evolutionary Computation*, 6(5):443–462, 2002.
3. L. Araujo. Part-of-speech tagging with evolutionary algorithms. In *Proc. of the Int. Conf. on Intelligent Text Processing and Computational Linguistics (CICLing-2002), LNCS 2276*, pages 230–239. Springer-Verlag, 2002.
4. L. Araujo. Studying the advantages of a messy evolutionary algorithm for natural language tagging. In *Proc. of the Int. Genetic and Evolutionary Computation Conference (GECCO), LNCS 2724*, pages 1951–1962. Springer-Verlag, 2003.
5. E. Charniak. *Statistical Language Learning*. MIT press, 1993.
6. L. Davis. *Handbook of Genetic Algorithms*. Van Nostrand Reinhold,1991.
7. S. J. DeRose. Grammatical category disambiguation by statistical optimization. *Computational Linguistics*, 14:31–39, 1988.
8. L. J. Eshelman. The CHC adaptive search algorithm: How to have safe search when engaging in nontraditional genetic recombination. In *Proceedings of the First Workshop on FOGA*, pages 265–283, San Mateo, CA, 1991. Morgan Kauffman.
9. S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. Optimization by simulated annealing. *Science, Number 4598, 13 May 1983*, 220, 4598:671–680, 1983.
10. B. Merialdo. Tagging English text with a probabilistic model. *Computational Linguistics*, 20(2):155–172, 1994.
11. F. W. Nelson and H. Kucera. Manual of information to accompany a standard corpus of present-day edited American English, for use with digital computers. Technical report, Department of Linguistics, Brown University., 1979.
12. H. Schutze and Y. Singer. Part of speech tagging using a variable memory Markov model. In *Proc. of the 1994 of the Association for Computational Linguistics*. Association for Computational Linguistics, 1994.