Adaptation, Comparison, and Improvement of Metaheuristic Algorithms to the Part-of-Speech Tagging Problem

Miguel-Alexis Solano-Jiménez; José-Julio Tobar-Cifuentes; Luz-Marina Sierra-Martínez; Carlos-Alberto Cobos-Lozada

Citation: M.-A. Solano-Jiménez, J.-J. Tobar-Cifuentes, L.-M. Sierra-Martínez, C.-A. Cobos-Lozada, "Adaptation, Comparison, and Improvement of Metaheuristic Algorithms to the Part-of-Speech Tagging Problem," *Revista Facultad de Ingeniería*, vol. 29 (54), e11762, 2020.
https://doi.org/10.19053/01211129.v29.n54.2020.11762

Received: July 13, 2020; **Accepted:** September 14, 2020;
Published: September 15, 2020

Conflict of interest: The authors state there is no conflict of interest.

Adaptation, Comparison, and Improvement of Metaheuristic Algorithms to the Part-of-Speech Tagging Problem

Miguel-Alexis Solano-Jiménez[1]
José-Julio Tobar-Cifuentes[2]
Luz-Marina Sierra-Martínez[3]
Carlos-Alberto Cobos-Lozada[4]

Abstract

Part-of-Speech Tagging (POST) is a complex task in the preprocessing of Natural Language Processing applications. Tagging has been tackled from statistical information and rule-based approaches, making use of a range of methods. Most recently, metaheuristic algorithms have gained attention while being used in a wide variety of knowledge areas, with good results. As a result, they were deployed in this research in a POST problem to assign the best sequence of tags (roles) for the words of a sentence based on information statistics. This process was carried out in two cycles, each of them comprised four phases, allowing the adaptation to the tagging problem in metaheuristic algorithms such as Particle Swarm Optimization, Jaya, Random-Restart Hill Climbing, and a memetic algorithm based on Global-Best Harmony Search as a global optimizer, and on Hill Climbing as a local optimizer. In the consolidation of each algorithm, preliminary experiments were carried out (using cross-validation) to adjust the parameters of each algorithm and, thus, evaluate them

[1] Universidad del Cauca (Popayán-Cauca, Colombia). miguelsolano@unicauca.edu.co. ORCID: 0000-0003-1936-3488
[2] Universidad del Cauca (Popayán-Cauca, Colombia). josej@unicauca.edu.co. ORCID: 0000-0002-5436-0816
[3] Ph. D. Universidad del Cauca (Popayán-Cauca, Colombia). lsierra@unicauca.edu.co. ORCID: 0000-0003-3847-3324
[4] Ph. D. Universidad del Cauca (Popayán-Cauca, Colombia). ccobos@unicauca.edu.co. ORCID: 0000-0002-6263-1911

on the datasets of the complete tagged corpus: IULA (Spanish), Brown (English) and Nasa Yuwe (Nasa). The results obtained by the proposed taggers were compared, and the Friedman and Wilcoxon statistical tests were applied, confirming that the proposed memetic, GBHS Tagger, obtained better results in precision. The proposed taggers make an important contribution to POST for traditional languages (English and Spanish), non-traditional languages (Nasa Yuwe), and their application areas.

Keywords: computational intelligence; computational linguistics; evolutionary computing; heuristic algorithms; natural language processing; parts of speech tagging; search methods.

Adaptación, comparación y mejora de algoritmos metaheurísticos al problema de etiquetado de partes del discurso

Resumen

La identificación de partes del discurso (Part-of-Speech Tagging, POST) es una tarea compleja en las aplicaciones de procesamiento de lenguaje natural. Ha sido abordada desde enfoques basados en información estadística y reglas, haciendo uso de distintos métodos y, últimamente, se destacan los algoritmos metaheurísticos obteniendo buenos resultados. Por ello, se involucran en esta investigación para asignar la mejor secuencia de etiquetas (roles) para las palabras de una oración, basándose en información estadística. Este proceso se desarrolló en 2 ciclos, donde cada ciclo tuvo 4 fases para la adaptación al problema de etiquetado en los algoritmos metaheurísticos Particle Swarm Optimization, Jaya, Random-Restart Hill Climbing, y un algoritmo memético basado en Global-Best Harmony Search como optimizador global, y en Hill Climbing como optimizador local. Se realizaron experimentos preliminares (utilizando validación cruzada), para ajustar los parámetros de cada algoritmo y luego ejecutarlos sobre los *datasets* completos de los corpus etiquetados IULA (castellano), Brown (inglés) y Nasa Yuwe (Nasa). Los resultados obtenidos por los etiquetadores propuestos se compararon mediante las pruebas estadísticas no paramétricas de Friedman y Wilcoxon, ratificando que el memético propuesto, GBHS Tagger, obtiene mejores resultados

de precisión. Los etiquetadores propuestos se convierten en un aporte muy importante para el POST, tanto para lenguas tradicionales (Inglés y Castellano), no tradicionales (Nasa Yuwe), y sus áreas de aplicación.

Palabras clave: algoritmos heurísticos; computación evolutiva; etiquetado de partes del discurso; inteligencia computacional; lingüística computacional; métodos de búsqueda; procesamiento de lenguaje natural.

Adaptação, comparação e melhora de algoritmos metaheurísticos ao problema de etiquetado de partes do discurso

Resumo

A identificação de partes do discurso (Part-of-Speech Tagging, POST) é uma tarefa complexa nas aplicações de processamento de linguagem natural. Tem sido abordada desde enfoques baseados em informação estatística e regras, fazendo uso de distintos métodos e, ultimamente, destacam-se os algoritmos metaheurísticos obtendo bons resultados. Por isso, envolvem-se nesta pesquisa para assignar a melhor sequência de etiquetas (papéis) para as palavras de uma oração, baseando-se em informação estatística. Este processo desenvolveu-se em 2 ciclos, onde cada ciclo teve 4 fases para a adaptação ao problema de etiquetado nos algoritmos metaheurísticos Particle Swarm Optimization, Jaya, Random-Restart Hill Climbing, e um algoritmo mimético baseado em Global-Best Harmony Search como otimizador global, e em Hill Climbing como otimizador local. Realizaram-se experimentos preliminares (utilizando validação cruzada), para ajustar os parâmetros de cada algoritmo e depois executá-los sobre os datasets completos dos corpus etiquetados IULA (castelhano), Brown (inglês) e Nasa Yuwe (Nasa). Os resultados obtidos pelos etiquetadores propostos compararam-se mediante as provas estatísticas não paramétricas de Friedman e Wilcoxon, ratificando que o mimético proposto, GBHS Tagger, obtém melhores resultados de precisão. Os etiquetadores propostos convertem-se em um aporte muito importante para o POST, tanto para línguas tradicionais (Inglês e Castelhano), não tradicionais (Nasa Yuwe), e suas áreas de aplicação.

Palavras chave: algoritmos heurísticos; computação evolutiva; etiquetado de partes do discurso; inteligência computacional; linguística computacional; métodos de busca; processamento de linguagem natural.

I. INTRODUCTION

Metaheuristic algorithms are being applied every day in a variety of areas of knowledge. It is not unusual, therefore, to use them in the problem of Part-of-speech Tagging (POST) or Identification. This is a complex task of great importance in Natural Language, given the challenges it faces, such as: the ambiguity of words, the size of the tag set, and the tagging of unknown words [1, 2].

Metaheuristic algorithms in the tagging problem (POST) have been used to assign the best sequence of tags (roles) for the words of a sentence, based on both statistical information and rules of transformation to solve this problem, obtaining outstanding results in contrast to traditional approaches. Related work includes: 1) Alhasan and Al-taani [3], who represented the tagging problem as a graph, the nodes are the possible tags of a sentence and use the optimization algorithm by Bee Colony Optimization (BCO), which finds the best solution path. 2) Sierra Martínez, Cobos and Corrales [4] proposed a memetic algorithm for tagging based on Global-Best Harmony Search (GBHS) that includes knowledge of the problem through a local optimization strategy based on the Hill Climbing algorithm. 3) Forsati & Shamsfard [5] presented two improvements to the HSTAgger tagger based on the Harmony Search metaheuristic, called HSTAgger (I) and HSTAgger (II), which increase search efficiency and improve the selection of new solutions for harmony memory. 4) Ekbal and Saha [6] addressed the tagging problem using single-objective and multiobjective optimization based on the Simulated Annealing-Based Multiobjective Optimization Algorithm proposed in [7], exploiting the search capacity of the simulated annealing algorithm.

Said metaheuristic approaches have been applied to corpus tagged in English, the Brown Corpus [8], the Penn Treebank Corpus [9], and other non-traditional languages such as Arabic with the KALIMAT corpus [10], Bengali (Bangladesh) [11], Hindi (India) [12], Telugu (India) [13], and Nasa Yuwe (an indigenous language of Colombia) [14]. Generally, these proposals use the Petrov tag set [15].

Metaheuristic algorithms solve problems using a search process (exploration and exploitation) of optimal solutions for a particular problem [16]. Thus, memetic algorithms [17] use population-based search to explore solutions, and local search

based on neighborhood for the exploitation of promising solutions [18, 19]. They also add knowledge of the problem to solve it. Table 1 describes the metaheuristics studied in this research for their subsequent adaptation to the tagging problem.

Table 1. Metaheuristic algorithms studied to adapt to the tagging problem.

Metaheuristic algorithm	Description
Random-Restart Hill Climbing (RRHC) [20]	Simple state metaheuristic that improves Hill Climbing (HC) [17]. It seeks to prevent HC from being trapped in local optimum by performing repetitive explorations in the problem space, which are generated randomly until the stop criterion occurs or a better solution is not found.
Particle Swarm Optimization (PSO) [21]	Population metaheuristic motivated by the intelligent collective behavior of swarms in nature. Each potential solution is called a particle, the set of particles is known as a swarm, and the position of each particle changes depending on its own experience and the experience of the swarm [22].
Jaya [23]	Population metaheuristic that seeks to find the best solution in the shortest possible time, but is also always trying to get away from failure, generating an optimal balance between exploration and exploitation. Jaya is a novel, simple and efficient algorithm for optimization problem solving with and without restrictions.
GBHS Tagger [4]	Memetic algorithm adapted to the tagging problem [4], based on the Global-Best Harmony Search (GBHS) metaheuristic, which has the following parameters [17]: HMS (Harmonic Memory Size), NI (number of improvisations), HCMR (Harmonic Memory Consideration Rate), and PARMin, PARMax (Tone Adjustment Rate). GBHS Tagger includes knowledge of the tagging problem using a local optimizer, adapted from the Hill Climb (HC) metaheuristic [17], which is applied to the best harmony in harmonic memory (HM). In addition to the GBHS parameters, three more parameters are defined: ProbOpt, which controls the percentage of times the local optimization process is carried out; MaxNeighbors, which defines the number of neighbors used in the local optimization process, and the parameter Alpha, which controls whether the components of each harmony in the population are randomly generated from their possible tags or taken from the tag with higher probability.

Figure 1 shows the representation of the solution used for this investigation, which consists of: 1) a first vector of the size of the number of words in a sentence (one position per word), which contains the tags assigned to each word, from position 0 to position n-1 ($T_0, T_1, \ldots, T_{n-1}$); 2) a second vector containing the cumulative probability of each tagged word, and its relationship with its predecessor and successor, and 3) a field that stores the value of the fitness function, calculated as shown in Figure 1, adapted from [5]. In GBHS Tagger, the selected context for the word to be tagged is a trigram (predecessor, word to tag, successor).

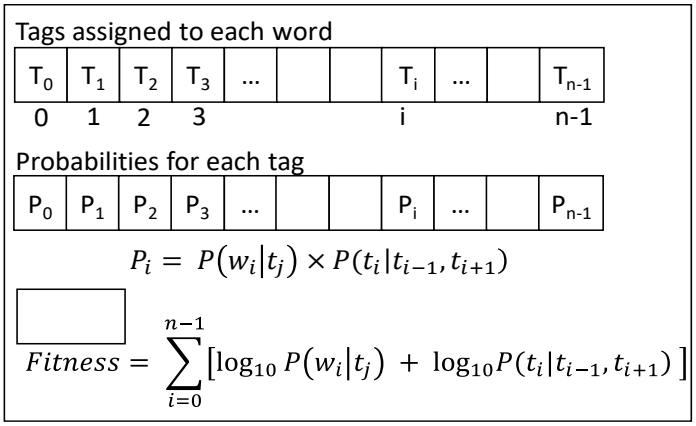

Fig. 1. Representation of the solution [4].

In the present work, the adaptation of several metaheuristic algorithms to the tagging problem was carried out, using the representation of the solution proposed in [4], in order to propose improvements to the memetic presented in the same work, at the same time that it was sought to evaluate its performance on the corpus in Castilian IULA [24], English Brown [8] and Nasa Yuwe [14].

The rest of the article is organized as follows: Section 2 presents the methodology used; Section 3 details the adaptation of the selected metaheuristics to the tagging problem; Section 4 shows the results of the experiments carried out, and, finally, Section 5 presents the discussion, conclusions and future work.

II. METHODOLOGY

This section describes the dataset used for the evaluation of the algorithms, the activities carried out in each phase of the cycles of the Iterative Research Pattern (IRP) methodology [25], used for carrying out this work, and how the experiments were set up.

A. Used Method

Two cycles were used for this research. The first cycle focused on the adaptation of the metaheuristic algorithms to the tagging problem and the selection of the best one; the second cycle focused on the adaptation of the selected metaheuristic

algorithms to the tagging problem and the proposal of a new version of the memetic algorithm. Table 2 describes the activities carried out in each phase.

Table 2. Description of the methodology.

<table>
<tr><th></th><th>Observation</th><th>Identification</th><th>Development</th><th>Testing</th></tr>
<tr><td rowspan="3">Cycle 1</td><td rowspan="3">Review of metaheuristics and investigation of corpus and set of tags.</td><td>Selection of corpus</td><td>Design and implementation of the database</td><td>Execution of experiments with the complete dataset.</td></tr>
<tr><td>Mapping of tags to universal tagging.</td><td>Adaptation and implementation of the metaheuristic algorithms to the POST.</td><td>Execution of non-parametric statistical tests.</td></tr>
<tr><td>Study of selected metaheuristic algorithms.</td><td>Configuration and execution of experiments on small datasets.</td><td>Analysis and discussion of results</td></tr>
<tr><td rowspan="3">Cycle 2</td><td>Research into memetic algorithms and discrete implementations.</td><td rowspan="2">Adaptation and implementation of the PSO, Jaya, Random-Restart Hill Climbing (RRHC), and HC algorithms to the tagging problem.</td><td>Implementation of the final versions of the proposed taggers.</td><td>Execution of full experiments and tests.</td></tr>
<tr><td rowspan="2">Study and understanding of the memetic GBHS Tagger.</td><td>Integration of HC Tagger to the memetic algorithm.</td><td rowspan="2">Analysis of results and issuance of conclusions.</td></tr>
<tr><td>Experiment setup on small datasets and parameter tuning.</td><td>Execution of the experiments and development of the new version of the memetic.</td></tr>
</table>

B. Dataset and Experimental Setup

As part of this work, the IULA (Spanish), Brown (English) and Nasa Yuwe tagged corpus were integrated into a single database designed and developed in SQL Server. The experiments were carried out on this database and, for their execution (both preliminary and complete), a client-server model was used, in which the clients (machines) request the tasks to be carried out. Each task receives the phrase and the algorithm that it must run and evaluate. Likewise, each task is executed 30 times (repetitions of the experiment) on the local machine. Once the task is finished, the results are recorded in the cloud database.

III. RESULTS

In the first instance of this section, the adaptation of the algorithms to the tagging problem and a new version of the memetic GBHS Tagger (GBHS4Tagger) are presented. In the second instance, the experiments and the results obtained with the

proposed taggers are shown. It is highlighted that all the adapted algorithms use the representation of the solution presented in [4], described in Figure 1.

A. Proposed JayaTagger

A discrete version of Jaya, called DJaya and proposed by [27], was used, it is free of parameters. The adaptation consisted in moving towards the best-known solution and moving away from the worst solution. Handling of the worst solution parameter was varied. The JayaTagger algorithm only handles three parameters: $PopulationSize$, MaxGenerations and $P4$. The latter controls the new solution from selecting a tag of the worst solution X_W, making the algorithm simple to implement and evaluate. In Figure 2, the proposed JayaTagger pseudocode is presented.

JayaTagger

1. Define of parameter: $TamPopulation$, MaxGenerations, $P4$
2. Random initialization of the population
3. Define the best solution in the population (X_B)
4. Define the worst solution in the population (X_W)
5. **for** $i = 1$ to MaxGenerations **do**
6. **for** $j = 1$ *to* $TamPopulation$ **do** /* for each particle */
7. X_{new} /* generating a new solution */
8. **for** $h = 1$ *to* n **do** /* for each Word in the sentence */
9. **if** ($Activo[h] = true$) **then** /* does the word have more than one tag? */
10. $random \leftarrow r(1, P4)$
11. **In case of** ($random$)
12. **case 1**: /*To preserve the tag in the solution*/
13. $X_{new(h)} = P_j(h)$
14. **case 2**: /* To select tag of the best solution */
15. $X_{new(h)} = X_B(h)$
16. **case 3**: /*To select tag randomly*/
17. $X_{new(h)} = LB_h + r \times (UB_h - LB_h)$
18. **case 4**: /*To select tag of the worst solution*/
19. $X_{new(h)} = X_W(h)$
20. **end_if**
21. **end_for**
22. Evaluate the fitness function of the new solution (X_{new}) (Figure **1**)
23. **if** (fitness (X_{new}) > fitness (P_j))
24. $P_j \leftarrow X_{new}$
25. **end_if**
26. **end_for**
27. **end_for**
28. **return** X_B /*Return the solution of the population */

Fig. 2. JayaTagger pseudocode

B. Proposed PSOTagger

The adaptation proposed is done according to the following parameters (a discrete version of PSO [26] was used): W, that selects a random tag for each dimension of a particle; $C1$, that selects the tag of the best particle history for that word; $C2$, that

selects the tag of the best global of the swarm for each dimension of the particle, and P, that maintains the components of the current particle. Additionally, PSOTagger involves the parameters $PopulationSize$ and $MaxGenerations$ from its original version. The tuning of the $W, C1, C2$, and P parameters in PSO was carried out experimentally with cross validation of 5 folders and a small dataset as a sample of the evaluation dataset. The PSOTagger pseudocode is presented in Figure 3.

PSOTagger

```
1.  Define parameters: TamPopulation, MaxGenerations, W, C1, C2, P
2.  Initialize random Population
3.  Define the best Global Particle (G_best)
4.  Define  P_best for particle in the swarm
5.  for i = 1 to MaxGeneraciones do
6.     for j = 1 to TamPopulation do /* for each particle */
7.        Evaluate fitness function of the particle (P_j) through Figure 1
8.        if (fitness (P_j) > fitness (P_best))
9.           P_best ← P_j
10.       end_if
11.    end_for
12.    Define G'_best ← Population^best /* the best of the current population is defined */
13.    if (fitness (G'_best) > fitness (G_best)) /* best current is greater than previous best */
14.       G_best ← G'_best
15.    end_if
16.    for j = 1 to TamPopulation do /* for each particle */
17.       for k = 1 to n do /* for each Word in sentence */
18.          if (Activo[j] = true) then  /*does the word have more than one tag? */
19.             random← r /* U (0,1) */
20.             if (random < W ) then
21.                P_k ← LB_k + r × (UB_k − LB_k) /*random tag selection */
22.             end_if
23.             else if (random< W + C1)  then
24.             P_k ← P_best(k) /* the tag is selected from the best historical */
25.             end_if
26.             else if (random< W + C1 + C2) then
27.             P_k ← G_best(k) /* the tag is selected from the Global Best */
28.             end_if
29.             else
30.                P_k ← P_k /* the tag is kept */
31.             end_if
32.          end_if
33.       end_for
34.    end_for
35. return G_best
```

Fig. 3. PSOTagger pseudocode.

C. Proposed Random-Restart Hill Climbing (RRHC) Tagger

The adaptation of the RRHCTagger algorithm to the tagging problem was carried out as follows. 1) The parameters: $n\_restart$ controls the number of restarts of solution S; $Prob$, list that stores the probabilities of the possible tags of a word; $ActiveIndex$, a list that stores the positions of words that have more than one tag,

and $StatusTrigram$, a list that stores the words selected to make a stochastic improvement. 2) The solution is stochastically improved, after a certain number of iterations without obtaining improvements, the algorithm saves the current result and the solution is restarted again (n_restart parameter), selecting another word from all the possibilities. 3) A tabu memory was implemented, which saves the words that were selected in the solution restart. In Figure 4, the proposed RRHCTagger pseudocode is presented.

RRHCTagger

```
1.  Definition of parameter S, iterations_HC, i_restart y n_restart
2.  i_restart = 0
3.  Prob     /* List of the probabilities of a word according to all the options */
4.  Random initialization of S
5.  ActiveIndex      /* This list stores the positions of words that have more than one tag */
6.  StateTrigam      /* This list stores the position of the words that were selected */
7.  j = Random(ActiveIndex.lenght)      /* j is going to be a random position of activeIndex*/
8.  Prob = possibleProbabilities (j)     /* Prob gets possible probabilities of j*/
9.  X_new ← Copy(S)
10. for i = 1 to iterations_HC do
11.         if (i_restart > n_restart )          /*if i_restart outperforms to n_restart */
12.             StateTrigram.add (j)
13.             if (fitness(X_new) > fitness(S))
14.                 S ← X_new
15.                 if (StateTrigram.contains(j − 1)) /* If the neighbor of j is found */
16.                     StateTrigram.remove(j − 1)/*It removes of the list */
17.                 end_if
18.                 if (StateTrigram.contains(j + 1)) /* If the successor neighbor is found */
19.                     StateTrigrama.remove(j + 1)  /* It removes of the list */
20.                 end_if
21.                 X_new ← Copy(S)  /* A new copy is taken to apply stochastic search */
22.                 pos ← Random(ActiveIndex.lenght) /* a new position is selected
                                                        from activeIndex */
23.                 j = ActiveIndex(pos)   /* The new word should not be in the list of
                                              StateTrigram */
24.                 Prob = possibleProbabilities (j)     /* The probabilities of the new word
                                                            are obtained */
25.                 i_restart = 0
26.             end_if
27.             Prob.sort() /*It is sorted in ascending order, with the best chance in the first
                              position*/
28.             X2_new ← Copy(X_new)  /* A new copy is created for the modifications to be
                                         made */
29.             pos2 =Prob [0]
30.             X2_new(j) = pos2   /* the tag is changed in position j*/
31.             if (fitness(X2_new) > fitness(X_new) >))
32.                 X_new ← X2_new
33.                 Prob.remove(Prob [0]) /* This tag is removed, to not be used again */
34.             else
35.                 Prob.remove(Prob [0])
36.                 i_restart + +
37.             end_if
38. end_for
39. return S
```

Fig. 4. RRHCTagger pseudocode.

D. Proposed GBHS4Tagger

The GBHS4Tagger algorithm is based on the GBHSTagger memetic algorithm proposed in [4] and its improvement consists of the following steps. 1) The Hill Climbing (HC) algorithm was adapted to the tagging problem involving two

neighborhoods. The first one selects a random word, regardless of the condition, and the second selects the word with the lowest probability. These neighborhoods are controlled with the $Prob$ parameter. 2) The proposed HCTagger was incorporated into GBHS Tagger 2 [4] as a local optimizer and, thus, the new memetic version called GBHS4Tagger. In Figure 5, the proposed HCTagger pseudocode is shown and in Figure 6, the proposed GBHS4Tagger is shown.

HCTagger

1. Define parameter $Prob$(Probability to choose the selection strategy of the word to tag)
2. Define parameter N (Maximum number of evaluations of the fitness function)
3. Random Initialization of the $olution$
4. ListPossibilitiesTags /* It contains all the possibilities of the words to tag */
5. **for** $i = 1$ to N **do**
6. $random \leftarrow r(0,1)$ /* a random number is generated (0,1) */
7. $X_{new} \leftarrow$ Copy$(Solution)$
8. **if** (random$\leq Prob$) **then** /*if random number is smaller than $Prob$ defined in the experiment setup */
9. $pos \leftarrow$ r(tags) /* To select the position of the Word randomly */
10. **else**
11. $pos \leftarrow$ index of the word with the lowest probability /* The position of the word is selected based on the lowest probability */
12. **end_if**
13. $j \leftarrow LB_{pos} + r \times (UB_{pos} - LB_{pos})$ /* A random index is selected from the possibilities of the word selected (pos) */
14. $X_{new(pos)} = 4.$ ListPossibilitiesTags[pos][j] /* The tag is assigned to the new solution */
15. **if** (fitness (X_{new}) > fitness $(Solucion)$) **then**
16. $Solucion \leftarrow X_{new}$ /* Si la nueva solución supera a la actual se reemplaza */
17. **else**
18. ListPossibilitiesTags (pos). remove(j) /* the possibility is eliminated from the list */
19. **end_if**
20. **end_for**
21. **return** $Solution$

Fig. 5. HCTagger pseudocode

E. Experiments with the proposed taggers

To carry out the experiments, in the first instance, an adjustment (fine-tuning) of the tagging parameters was carried out using a small dataset (sample) of 5000 sentences, in order to select the best combinations of parameters of each algorithm. Table 3 shows the distribution of the sentences in the test and training datasets for each folder, with which the experiments were carried out on each complete corpus, as seen in Table 5. All the experiments were executed using cross-validation of 5 folders, except for the Nasa Yuwe corpus, with which Leave-One-Out was used, since the dataset has only 175 sentences.

GBHS adapted to the Tagging problem (GBHS4Tagger)

1. Define parameters: HMS, NI, HCMR, PARMin, ParMax, ProbOpt, Alpha, MaxNeighbors, Prob
2. HM random initialization or improved initialization using the Alpha parameter
3. **for** $i = 1$ to NI **do**
4. $PAR \leftarrow PARMin + (PARMax - PARMin) \times \left({}^{i}/_{NI}\right)$ /* definition of PAR */
5. **for** $j = 1$ to n **do** /* for each word in the sentence */
6. **if** (Active[j] == true) **then** /* Does the current word have more than one possible tag? */
7. **if** $(U(0,1) \leq HMCR)$ **then** /* memory consideration */
8. $x_j' \leftarrow x_j^p$, Where $p \sim U(1, \ldots, HMS)$
9. **if** $(U(0,1) \leq PAR$ **then**
10. $x_j' \leftarrow x_k^{best}$, Where best is the index of best harmony in HM and $k \sim U(1,n)$
end_if
11. **else** /* random selection */
12. $x_j' \leftarrow LB_j + r \times (UB_j - LB_j)$
13. **end_if**
14. **else**
15. $x_j' \leftarrow$ UniqueTagWord(j)
16. **end_if**
17. **end_for**
18. **while** (visited (x_j')) do /* If the solution has been visited before */
19. **if** (Active[j] == true) **then** /* mutate the new harmony (x_j')
20. A different one than the current one is randomly selected */
21. **end_while**
22. Evaluate fitness function of the new harmony (x_j')
23. **if** (fitness (x_j') > fitness of the worst harmony in HS) **then**
24. HM [pos_worst] $\leftarrow x_j'$ /* replace worst in memory */
25. **end_if**
26. **if** $(U(0,1) < ProbOpt)$ **then**
27. Apply Local Optimization to the best harmony in HM
28. **end_if**
29. **end_for**
30. **return** best harmony in HM

Fig. 6. GBHS4Tagger pseudocode. Adapted from [4].

Table 3. Test and training data set for the experiments.

Data sets		IULA corpus			Brown corpus		
Test Data	**Folder with Training data**	**Sentences in the Test data**	**Words in the Test data**	**Words in the Training data**	**Sentences in the Test data**	**Words in the Test data**	**Words in the Training data**
1	2, 3, 4, 5	8416	16316	65521	10595	23105	45113
2	1, 3, 4, 5	8416	16357	65480	10600	22852	45199
3	1, 2, 4, 5	8416	16466	65371	10600	23130	45009
4	1, 2, 3, 5	8416	16290	65547	10600	22929	45130
5	1, 2, 3, 4	8415	16408	65429	10603	23111	45025
Nasa Yuwe corpus		The Nasa Yuwe corpus used Leave-One-Out, a method that takes one phrase as test data and the remaining 174 phrases as training data. This process is repeated for all the sentences in the corpus. It is a commonly used evaluation method when the dataset is small.					

In Table 4, following, the configuration of the algorithms for the experiments carried out with each corpus is presented.

Table 4. Configuration of the tagging algorithms for the experiments.

Algorithm	Parameters defined for the IULA corpus	Parameters defined for the Brown corpus	Parameters defined for the Nasa Yuwe corpus
PSOTagger	W = 0.3, C1 = 0.15, C2 = 0.45 and P = 0.1	W = 0.3, C1 = 0.15, C2 = 0.45 and P = 0.1	W = 0.3, C1 = 0.2, C2 = 0.4 and P = 0.1.
JayaTagger	P4 = 4	P4 = 3	P4 = 4.
RRHCTagger	n_restart = 6	n_restart = 4.	n_restart = 4.
GBHS4Tagger	Prob = 0.7	Prob = 0.7.	Prob = 0.75.

In Table 5, the results of the experiments carried out on the three complete corpuses are presented. It can be seen that GBHS4Tagger surpassed the other algorithms in precision value, in the IULA and Brown corpus, with the Nasa Yuwe corpus being the second best. It should be noted that the adapted algorithms obtained very good results for this problem, but there are differences between the precision values obtained in each algorithm, which allow us to appreciate that some algorithms perform better than others, as established in the second theorem of No Free Lunch Theorems for Optimization (NFLT) [28].

Table 5. Results of the experiments.

Results	IULA corpus			BROWN corpus			Nasa Yuwe corpus		
Algorithm	**Phrases**	**Precision**	**Std dev**	**Phrases**	**Precision**	**Std dev**	**Phrases**	**Precision**	**Std dev**
GBHS4Tagger	**42079**	**97.5403**	**5.4795**	**52998**	**94.8803**	**6.0289**	175	61.4185	15.5199
RRHCTagger	42079	97.2678	5.5006	52998	94.5342	6.0106	**175**	**63.7096**	**16.2249**
GBHS Tagger2	42079	97.4306	5.7844	52998	94.8236	6.2051	175	60.0749	15.5199
JayaTagger	42079	96.8592	5.5963	52998	93.0519	6.3054	175	57.084	15.977
PSOTagger	42079	96.7738	5.7844	52998	89.6933	7.0810	175	55.936	16.052

Table 6 shows the ranking of each algorithm in the experiments carried out in each corpus once the Friedman NxN non-parametric statistical test has been applied, obtaining a p value smaller than 0.05, therefore, it makes the ranking statistically significant, complementing the evaluation of the algorithms.

Table 6. Friedman test results - IULA corpus. Made with KEEL software [29]

IULA corpus	
Algorithm	**Ranking**
GBHS4Tagger	1
GBHS Tagger2	2
RRHCTagger	3
JayaTagger	4
PSOTagger	5

Brown corpus	
Algorithm	**Ranking**
GBHS4Tagger	1
GBHS Tagger2	2
RRHCTagger	3
JayaTagger	4
PSOTagger	5

Nasa Yuwe	
Algorithm	**Ranking**
RRHCTagger	1
GBHS4Tagger 2	2
GBHS Tagger2	3
JayaTagger	4
PSOTagger	5

Additionally, the Wilcoxon test showed, with a significance level of 90%, that the results obtained for the winning algorithms, GBHS4Tagger for Spanish and English, and RRHCTagger for Nasa Yuwe, are better in contrast with the other proposed taggers.

V. Discussion and conclusions

This work achieved the adaptation of the metaheuristic algorithms PSO, Jaya, and RRHC to the problem of part of speech tagging (POST), taking into account the characteristics of each algorithm, and performing the parameter adjustment required for each algorithm on each corpus, obtaining competitive results with respect to one of the state-of-the-art algorithms. It was also possible to propose an improvement to the state-of-the-art GBHS Tagger 2 memetic algorithm, which continued to demonstrate that the performance of the tagger improves by including knowledge of the problem, as seen in the IULA (Spanish) and Brown (English) corpus. Consequently, the presented research reinforced the idea that metaheuristic approaches are capable of performing tagging with good results, with acceptable resources and times. Metaheuristic algorithms should continue to be used for tagging on other traditional and non-traditional languages, and seek new improvements for the proposed taggers in combination with other optimization techniques that improve the results of the tagging.

AUTHOR'S CONTRIBUTION

Miguel-Alexis Solano-Jiménez: Formal Analysis, Data curation, Investigation, Methodology, Software, Validation, Visualization, Writing – original draft, Writing – review & editing.

Jose-Julio Tobar-Cifuentes: Formal Analysis, Data curation, Investigation, Methodology, Software, Validation, Writing – original draft, Writing – review & editing.

Luz-Marina Sierra-Martínez: Conceptualization, Methodology, Supervision, Project administration, Writing -original draft, Writing – review & editing.

Carlos-Alberto Cobos-Lozada: Conceptualization, Methodology, Supervision, Project administration, Writing -original draft, Writing – review & editing.

FUNDING

This work was partially financed by Universidad del Cauca.

ACKNOWLEDGMENTS

The authors express especially gratitude to Colin McLachlan for suggestions relating to the English text.

REFERENCES

[1] T. Güngör, *Handbook of Natural Language Processing (2 Edition)*, 2011.

[2] D. Jurafsky, and J. H. Martin, "Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition," *Computational Linguistics*, vol. 26(4), pp. 638-641, 2009. https://doi.org/10.1162/089120100750105975

[3] A. Alhasan, and A. T. Al-taani, "POS Tagging for Arabic Text Using Bee Colony Algorithm," *Procedia Computer Science*, pp. 158-165, 2018. https://doi.org/10.1016/j.procs.2018.10.471

[4] L. M. Sierra Martínez, C. A. Cobos, and J. C. Corrales, "Memetic algorithm based on global-best harmony search and hill climbing for part of speech tagging," in *International Conference on Mining Intelligence and Knowledge Exploration*, 2017, pp. 198-211. https://doi.org/10.1007/978-3-319-71928-3_20

[5] R. Forsati, and M. Shamsfard, "Novel harmony search-based algorithms for part-of-speech tagging," *Knowledge and Information Systems*, vol. 42, pp. 709-736, 2014. https://doi.org/10.1007/s10115-013-0719-6

[6] A. Ekbal, and S. Saha, "Simulated annealing based classifier ensemble techniques: Application to part of speech tagging," *Information Fusion*, vol. 14 (3), pp. 288-300, 2013. https://doi.org/10.1016/j.inffus.2012.06.002

[7] S. Bandyopadhyay, S. Saha, U. Maulik, and K. Deb, "A Simulated Annealing-Based Multiobjective

Optimization Algorithm: AMOSA," *IEEE Transactions on Evolutionary Computation*, vol. 12 (3), pp. 269-283, Jun. 2008. https://doi.org/10.1109/TEVC.2007.900837

[8] W. N. Francis, and H. Kucera, *Brown Corpus Manual*, 1979. http://clu.uni.no/icame/manuals/BROWN/INDEX.HTM#bc8

[9] M. P. Marcus, M. A. Marcinkiewicz, and B. Santorini, "Building a large annotated corpus of English: the penn treebank," *Computational Linguistics*, vol. 19 (2), pp. 313-330, 1993. https://doi.org/10.1162/coli.2010.36.1.36100

[10] M. El-Haj, and R. Koulali, "KALIMAT a multipurpose Arabic Corpus," in *Second Workshop on Arabic Corpus Linguistics*, 2013.

[11] T. Chakraborty, "Identification of Reduplication in Bengali Corpus and their Semantic Analysis : A Rule-Based Approach," in *Proceedings of the Workshop on Multiword Expressions: from Theory to Applications*, 2010, pp. 73-76.

[12] O. Bojar, V. Diatka, P. Rychly, P. Straňak, V. Suchomel, Al. Tamchyna, and D. Zeman, "HindEnCorp - Hindi-English and Hindi-only corpus for machine translation," in *Proceedings of the Ninth International Conference on Language Resources and Evaluation*, 2014, pp. 3550-3555.

[13] S. S. Mukku, and R. Mamidi, "ACTSA: Annotated Corpus for Telugu Sentiment Analysis," in *Proceedings of the First Workshop on Building Linguistically Generalizable NLP Systems*, 2018, pp. 54-58. https://doi.org/10.18653/v1/w17-5408

[14] L. M. Sierra Martínez, C. A. Cobos, C. J. Muñoz Corrales, T. Curieux Rojas, E. Herrera-viedma, and D. H. Peluffo-ordóñez, "Building a Nasa Yuwe Language Corpus and Tagging with a Metaheuristic Approach," *Computación y Sistemas*, vol. 22 (3), pp. 881-894, 2018. https://doi.org/10.13053/CyS-22-3-3018

[15] S. Petrov, D. Das, and R. McDonald, "A Universal Part-of-Speech Tagset," in *Proceedings of the Eighth International Conference on Language Resources and Evaluation*, 2012, pp. 2089-2096.

[16] X. S. Yang, and S. Deb, "Cuckoo search: Recent advances and applications," *Neural Computing and Applications*, vol. 24 (1), pp. 169-174, 2014. https://doi.org/10.1007/s00521-013-1367-1

[17] J. Brownlee, *Clever Algorithms*, 2011.

[18] F. Neri, and C. Cotta, "A Primer on Memetic Algorithms," in *Handbook of Memetic Algorithm*, pp. 43-52, 2012.

[19] C. Cotta, *Una Visión General de los Algoritmos Meméticos*. http://www.lcc.uma.es/~ccottap/papers/memeticos.pdf

[20] E. R. R. Kato, G. D. de A. Aranha, and R. H. Tsunaki, "A new approach to solve the flexible job shop problem based on a hybrid particle swarm optimization and Random-Restart Hill Climbing," *Computers & Industrial Engineering*, vol. 125, pp. 178-189, Nov. 2018. https://doi.org/10.1016/j.cie.2018.08.022

[21] J. Kennedy, and R. Eberhart, "Particle Swarm Optimization," in *Proceedings of ICNN'95 - International Conference on Neural Networks*, 1995, pp. 1942-1948. https://doi.org/10.1109/ICNN.1995.488968

[22] A. Nickabadi, M. M. Ebadzadeh, and R. Safabakhsh, "A novel particle swarm optimization algorithm with adaptive inertia weight," *Applied Soft Computing*, vol. 11 (4), pp. 3658-3670, 2011. https://doi.org/10.1016/j.asoc.2011.01.037

[23] R. Venkata Rao, "Jaya: A simple and new optimization algorithm for solving constrained and unconstrained optimization problems," *International Journal of Industrial Engineering Computations*, vol. 7 (1), pp. 19-34, Dec. 2016. https://doi.org/10.5267/j.ijiec.2015.8.004

[24] Institut Universitari de Lingüística Aplicada, *IULA Spanish LSP Treebank*, 2012.

[25] K. S. Pratt, “Design Patterns for Research Methods: Iterative Field Research,” in *AAAI Spring Symposium: Experimental Design for Real*, 2009, pp. 1-7.

[26] Q. Pan, M. F. Tasgetiren, and Y. Liang, “A discrete particle swarm optimization algorithm for the no-wait flowshop scheduling problem,” *Computers & Operations Research*, vol. 35, pp. 2807-2839, 2008. https://doi.org/10.1016/j.cor.2006.12.030

[27] K. Gao, F. Yang, M. Zhou, Q. Pan, and P. N. Suganthan, “Flexible job-shop rescheduling for new job insertion by using discrete Jaya algorithm,” *IEEE Transactions on Cybernetics*, vol. 49 (5), pp. 1944-1955, 2019. https://doi.org/10.1109/TCYB.2018.2817240

[28] D. H. Wolpert, and W. G. Macready, “No free lunch theorems for optimization,” *IEEE Transactions on Evolutionary Computation*, vol. 1 (1), pp. 67-82, Apr. 1997. https://doi.org/10.1109/4235.585893

[29] J. Alcalá-Fdez, L. Sánchez, S. García, M. J. del Jesus, S. Ventura, J. M. Garrell, J. Otero, C. Romero, J. Bacardit, V. M. Rivas, J. C. Fernández, and F. Herrera, “KEEL: A software tool to assess evolutionary algorithms for data mining problems,” *Soft Computing*, vol. 13, pp. 307-318, 2009. https://doi.org/10.1007/s00500-008-0323-y