**This is the accepted version of the article:**

Jesús M. Sánchez-Gómez, Miguel A. Vega-Rodríguez, Carlos J. Pérez (2020). A decomposition-based multi-objective optimization approach for extractive multi-document text summarization, Applied Soft Computing, Volume 91, 2020, 106231, ISSN: 1568-4946, DOI: 10.1016/j.asoc.2020.106231

# A Decomposition-based Multi-Objective Optimization Approach for Extractive Multi-Document Text Summarization

Jesus M. Sanchez-Gomez[a,*], Miguel A. Vega-Rodríguez[a], Carlos J. Pérez[b]

[a]*Department of Computer and Communications Technologies, University of Extremadura, Campus Universitario s/n, 10003 Caceres, Spain.*
[b]*Department of Mathematics, University of Extremadura, Campus Universitario s/n, 10003 Caceres, Spain.*

**Abstract**

Currently, due to the overflow of textual information on the Internet, automatic text summarization methods are becoming increasingly important in many fields of knowledge. Extractive multi-document text summarization approaches are intended to automatically generate summaries from a document collection, covering the main content and avoiding redundant information. These approaches can be addressed through optimization techniques. In the scientific literature, most of them are single-objective optimization approaches, but recently multi-objective approaches have been developed and they have improved the single-objective existing results. In addition, in the field of multi-objective optimization, decomposition-based approaches are being successfully applied increasingly. For this reason, a Multi-Objective Artificial Bee Colony algorithm based on Decomposition (MOABC/D) is proposed to solve the extractive multi-document text summarization problem. An asynchronous parallel design of MOABC/D algorithm has been implemented in order to take advantage of multi-core architectures. Experiments have been carried out with Document Understanding Conferences (DUC) datasets, and the results have been evaluated with Recall-Oriented Understudy for Gisting Evaluation (ROUGE) metrics. The obtained results have improved the existing ones in the scientific literature for ROUGE-1,

*Corresponding author

*Email addresses:* jmsanchezgomez@unex.es (Jesus M. Sanchez-Gomez), mavega@unex.es (Miguel A. Vega-Rodríguez), carper@unex.es (Carlos J. Pérez)

ROUGE-2, and ROUGE-L scores, also reporting a very good speedup.

*Keywords:* Multi-Document Summarization, Multi-Objective Optimization, Artificial Bee Colony, Decomposition-based.

## 1. Introduction

Nowadays, there is a large amount of information contained on the Internet, and it follows growing day-to-day. This makes that the number of digital documents is also increasing. This data volume makes difficult to obtain the most important information of a specific topic, whereas the Internet users demand to obtain such information as quickly as possible. According to Fan and Bifet [1], text mining tools are capable of extracting relevant information from a large set of documents. Besides, from the textual information contained in the documents, these tools should be able to automatically create a summary (Hashimi et al. [2]), satisfying the needs of the users by covering the main content and reducing the number of redundant sentences.

The summaries generated by text mining tools can be of various kinds. Firstly, a summary can be abstractive or extractive: abstractive summarization generates summaries made by words and phrases which may not be contained in the document collection, and extractive summarization selects sentences that exist in the original text (Wan [3]). Secondly, summaries can be single-document or multi-document depending on where the information comes from: single-document methods are limited to reducing the information of a single document to the required length, and multi-document methods gather textual information of the entire set of documents (Zajic et al. [4]). Extractive multi-document text summarization is one of the most used methods to generate summaries in an automatic way. Its main goal is to represent the most relevant information contained in a document collection, covering the main content and avoiding redundant information. On the other hand, abstractive multi-document text summarization is becoming a current research trend, since it requires new and challenging natural language processing techniques. These techniques are based on linguistic and semantic approaches, and they can also be addressed with evolutionary algorithms (see e.g. Khan et al. [5], Khan et al. [6], and Mendoza et al. [7]).

The extractive multi-document text summarization has been addressed from single-objective and multi-objective approaches. In particular, single-objective approaches are focused on the optimization of only one objective

function (Alguliev et al. [8]). This optimization technique requires weighting all the criteria included in the single objective function, which means that the weight allocation is performed subjectively, therefore, influencing the final result. In contrast, in multi-objective approaches, every criterion corresponds to a different objective function and all the objective functions are optimized simultaneously (Sanchez-Gomez et al. [9]). In the scientific literature, multi-objective optimization for extractive multi-document text summarization has provided better results than the single-objective optimization.

In recent years, the multi-objective optimization approaches have evolved in different ways, and new techniques have emerged in this field. This is the case of approaches based on decomposition (Zhang and Li [10]). Unlike traditional multi-objective optimization approach, which is based on Pareto dominance, this technique consists of the decomposition of a multi-objective optimization problem into a number of scalar optimization subproblems. In addition, every subproblem is optimized taking into account the information from its neighboring subproblems.

This decomposition-based technique has been applied in several cases, improving the results that already existed. A decomposition-based multi-objective approach was compared in Li and Zhang [11] with the Non-dominated Sorting Genetic Algorithm II (NSGA-II), and the obtained results indicated a significant improvement. Moreover, Zhang et al. [12] proposed another decomposition-based algorithm for dealing with expensive multi-objective optimization problems in real-world applications. In Ke et al. [13], an ant colony optimization algorithm was adopted to derive a decomposition-based approach, improving the existing results in two well-known problems: multi-objective 0-1 knapsack problem and bi-objective traveling salesman problem. In addition, in decomposition-based approaches, there are different methods to transform the approximation of the Pareto front to scalar optimization subproblems. Some of the most popular approaches are boundary intersection (Das and Dennis [14]), non-normalized Tchebycheff (Jaszkiewicz [15]), and normalized Tchebycheff (Tang et al. [16]).

In this paper, a Multi-Objective Artificial Bee Colony based on Decomposition (MOABC/D) is proposed to address the extractive multi-document text summarization problem. MOABC/D algorithm generates summaries automatically by maximizing two objective functions simultaneously: the content coverage and the redundancy reduction. Experiments have been carried out with Document Understanding Conferences (DUC) datasets, and the Recall-Oriented Understudy for Gisting Evaluation (ROUGE) metrics has

been used. The main contributions of this paper are:

- MOABC/D has been designed and applied to the extractive multi-document text summarization problem.
- A parallel version of MOABC/D has been designed and implemented, which is based on an asynchronous parallel model.
- Quality of the summaries provided by the proposed MOABC/D approach has been statistically analyzed and compared to other approaches. For this purpose, DUC datasets and ROUGE metrics (the standard in the field) have been used.
- Execution times have been analyzed and compared. For this goal, speedup and efficiency (the standard in the field) have been used, obtaining very good results for both metrics with 64 threads.

The remainder of this paper is organized as follows. Section 2 includes the state-of-the-art. In Section 3, the extractive multi-document text summarization problem is defined as a multi-objective optimization problem. Then, Section 4 describes the proposed MOABC/D algorithm. In Section 5, the datasets used, the experimental settings, the evaluation metrics, and the obtained results are reported. And, finally, Section 6 presents the conclusions and the future research.

## 2. Related Work

Firstly, single-objective approaches are presented. An unsupervised text summarization model based on integer linear programming that used the Branch and Bound (B&B) algorithm and a Particle Swarm Optimization (PSO) algorithm was proposed by Alguliev et al. [17]. Alguliev et al. [8] addressed document summarization as a discrete optimization problem, implementing an adaptive Differential Evolution (DE) algorithm in order to solve it. A multi-document text summarization based on modified $p$-median problem was proposed by Alguliev et al. [18] (Self-Adaptive DE algorithm) and Alguliev et al. [19] (DE algorithm based on Self-Adaptive Mutation and Crossover parameters, DESAMC). Alguliev et al. [20] and Alguliev et al. [21] tackled the document text summarization problem, in which the objective function was defined as the Heronian mean of all criteria in a 0-1 non-linear

programming problem. In both works the PSO algorithm was used. A quadratic boolean programming approach was considered by Alguliev et al. [22] and Alguliev et al. [23] with a binary DE algorithm. Alguliev et al. [24] used PSO algorithm in a quadratic integer programming framework. An improved DE algorithm was developed by Alguliev et al. [25] with the aim of solving the optimization-based approach for generic document summarization. Alguliev et al. [26] modeled document summarization as a linear and nonlinear optimization problem, and used a PSO algorithm to solve it. Multi-document summarization was described as a binary optimization problem in Mendoza et al. [27], where a Cross-generational elitist selection, Heterogeneous recombination and Cataclysmic mutation (CHC) algorithm was proposed to solve it. Benjumea and León [28] presented a novel approach for automatic extractive text summarization based on sentence clustering, which used a genetic clustering algorithm. Multi-document summarization was also considered as a topical closeness approach by Umam et al. [29], which proposed a Self-Adaptive DE algorithm. Alguliev et al. [30] addressed the text summarization problem as boolean programming using a DE algorithm. A pattern-based model for generic multi-document summarization by using a closed pattern mining algorithm was presented in Qiang et al. [31]. A two-stage sentences selection model based on clustering and optimization techniques was presented by Alguliyev et al. [32], using an adaptive DE algorithm with a novel mutation strategy. Finally, Verma and Om [33] proposed a novel extraction-based method for multi-document summarization, using a meta-heuristic approach by means of Shark Smell Optimization (SSO) algorithm.

The scientific literature on multi-objective optimization for extractive multi-document text summarization is much more limited. Saleh et al. [34] proposed a model based on discrete optimization using a Non-dominated Sorting Genetic Algorithm-II (NSGA-II), whereas Sanchez-Gomez et al. [9] used a Multi-Objective Artificial Bee Colony (MOABC) optimization approach. These two approaches were based on the Pareto dominance, that is, they are not decomposition-based multi-objective approaches.

Table 1 shows a summary of the previous approaches. In addition to the algorithm described and whether it is a multi-objective optimization algorithm, the table includes the used criteria, which have been mainly content coverage and redundancy reduction. Besides, all these previous works have used DUC datasets and ROUGE metrics.

| Work (year) | Algorithm | Criteria | Multi-objective |
|---|---|---|---|
| Alguliev et al. [8] (2011) | Adaptive DE | Coverage and redundancy reduction | No |
| Alguliev et al. [17] (2011) | B&B and PSO | Coverage and redundancy reduction | No |
| Alguliev et al. [18] (2011) | Self-Adaptive DE | Coverage, redundancy reduction and relevance | No |
| Alguliev et al. [20] (2011) | PSO | Coverage and redundancy reduction | No |
| Alguliev et al. [19] (2012) | DESAMC | Coverage, redundancy reduction and relevance | No |
| Alguliev et al. [22] (2012) | DE | Coverage and redundancy reduction | No |
| Alguliev et al. [23] (2012) | DE | Coverage and redundancy reduction | No |
| Alguliev et al. [21] (2013) | PSO | Coverage and redundancy reduction | No |
| Alguliev et al. [24] (2013) | PSO | Coverage and redundancy reduction | No |
| Alguliev et al. [25] (2013) | DE | Coverage and redundancy reduction | No |
| Alguliev et al. [26] (2013) | PSO | Coverage and redundancy reduction | No |
| Mendoza et al. [27] (2014) | CHC | Coverage and redundancy reduction | No |
| Benjumea and León [28] (2015) | Genetic clustering | Coverage and redundancy reduction | No |
| Umam et al. [29] (2015) | Self-Adaptive DE | Coverage, redundancy reduction and coherence | No |
| Alguliev et al. [30] (2015) | DE | Coverage and redundancy reduction | No |
| Qiang et al. [31] (2016) | Closed pattern mining | Coverage and redundancy reduction | No |
| Alguliyev et al. [32] (2019) | Adaptive DE | Coverage and redundancy reduction | No |
| Verma and Om [33] (2019) | SSO | Coverage, redundancy reduction and relevance | No |
| Saleh et al. [34] (2015) | NSGA-II | Coverage and redundancy reduction | Yes |
| Sanchez-Gomez et al. [9] (2018) | MOABC | Coverage and redundancy reduction | Yes |

Table 1: Works related with extractive multi-document text summarization.

## 3. Problem Definition

In this section, extractive multi-document text summarization is formalized as an optimization problem. In this field, the most common methods use the term-based vector space model (VSM). For this kind of methods, every sentence is represented as a term vector, and the similarity measure used for pairwise comparison is usually cosine similarity (Alguliev et al. [8], Saleh et al. [34], Sanchez-Gomez et al. [9]).

Table 2 contains the notation used through the paper.

| **Symbol** | **Definition** |
|---|---|
| $D$ | Document collection |
| $t_k$ | Distinct term $k$ of the document collection $D$ |
| $T$ | Set of different terms from the document collection $D$ |
| $m$ | Number of distinct terms in the document collection $D$ |
| $s_i$ | Sentence $i$ of the document collection $D$ |
| $w_{ik}$ | Weight associated to the term $t_k$ in the sentence $s_i$ |
| $tf_{ik}$ | Frequency of the term $t_k$ in the sentence $s_i$ |
| $n$ | Number of sentences in the document collection $D$ |
| $n_k$ | Number of sentences containing the term $t_k$ |
| $O$ | Mean vector or centre of the document collection $D$ |
| $o_k$ | Component $k$ of mean vector $O$ |
| $sim(s_i, s_j)$ | Cosine similarity measure between sentences $s_i$ and $s_j$ |
| $N$ | Number of documents in the document collection $D$ |
| $S$ | Generated summary |
| $L$ | Summary length constraint |
| $x_i$ | Binary decision variable equal to 1 if sentence $s_i$ is presented in the generated summary $S$ |
| $y_{ij}$ | Binary decision variable equal to 1 if sentences $s_i$ and $s_j$ are presented in the generated summary $S$ |
| $X$ | Solution representation or decision vector |
| $\Phi(X)$ | Objective function |
| $l_i$ | Length of the sentence $s_i$ |
| $\varepsilon$ | Summary length tolerance |

Table 2: Notations and symbols used in this work.

*3.1. Similarity Measure*

Firstly, the cosine similarity measure is described. Let $T = \{t_1, t_2, \ldots, t_m\}$ be the set that contains all the distinct terms from the document collection $D$, where $m$ is the number of terms. Every sentence $s_i$ from $D$ can be represented as a vector of terms $s_i = (w_{i1}, w_{i2}, \ldots, w_{im}), i = 1, 2, \ldots, n$, where every component $w_{ik}$ of the vector is associated to the weight of the term $t_k$ in the sentence $s_i$. The most common way to calculate this weight is using the *term-frequency inverse-sentence-frequency* scheme, called $tf\_isf$ (Salton and Buckley [35]). According to this scheme, the weight $w_{ik}$ is calculated as follows:

$$w_{ik} = tf_{ik} \cdot log(n/n_k). \tag{1}$$

The first component $tf_{ik}$ counts the times that the term $t_k$ occurs in the sentence $s_i$, and the second component $log(n/n_k)$ ($isf$ expression) takes into account the number of sentences $n_k$ in $D$ which contain the term $t_k$. The total number of sentences in $D$ is $n$.

The main content of the document collection $D$ can be represented through the average weights of the $m$ terms in the set of sentences. The vector $O = (o_1, o_2, \ldots, o_m)$, also called mean vector, represents the center of the document collection $D$. Its components are calculated as follows:

$$o_k = \frac{1}{n} \sum_{i=1}^{n} w_{ik}, \quad k = 1, 2, \ldots, m. \tag{2}$$

Finally, the cosine similarity is based on the weights presented previously. It is defined as a resemblance measure between two sentences $s_i = (w_{i1}, w_{i2}, \ldots, w_{im})$ and $s_j = (w_{j1}, w_{j2}, \ldots, w_{jm})$ as follows:

$$sim(s_i, s_j) = \frac{\sum_{k=1}^{m} w_{ik} w_{jk}}{\sqrt{\sum_{k=1}^{m} w_{ik}^2 \cdot \sum_{k=1}^{m} w_{jk}^2}}, \quad i, j = 1, 2, \ldots, n. \tag{3}$$

*3.2. Formulation of the Optimization Problem*

After explaining the cosine similarity measure, the formulation of the optimization problem is described. Let $D = \{d_1, d_2, \ldots, d_N\}$ be the document collection, which is a set of $N$ documents. In addition, $D$ can be represented as a set of the $n$ sentences contained in the document collection: $D = \{s_1, s_2, \ldots, s_n\}$. The aim is to generate a summary $S \subset D$ satisfying three aspects:

- *Content coverage*: the generated summary should cover the main content of the document collection by including the most relevant sentences.
- *Redundancy reduction*: the generated summary should not contain similar sentences from the document collection.
- *Length*: the generated summary must have a restricted length $L$. This constraint is explained later.

The presented extractive multi-document text summarization problem implies the simultaneous optimization of the content coverage and the redundancy reduction. Nevertheless, these two criteria are in conflict. Thus, the natural way to address this problem is by a multi-objective optimization approach.

Let $x_i \in \{0, 1\}$ be a binary decision variable which takes into account the presence or absence ($x_i = 1$ or $x_i = 0$) of the sentence $s_i$ in the generated summary $S$. So, the representation of the solution is denoted as $X = (x_1, x_2, \ldots, x_n)$, which is also called the decision vector.

Firstly, the objective function $\Phi_{Cov}(X)$ concerns the content coverage criterion. Given the sentence $s_i \in S$, the content coverage is formulated by using the cosine similarity between the sentence $s_i$ and the set of sentences in $D$ represented by the mean vector $O$. Thus, the following objective function should be maximized:

$$\Phi_{Cov}(X) = \sum_{i=1}^{n} sim(s_i, O) \cdot x_i. \tag{4}$$

Secondly, the objective function $\Phi_{ReR}(X)$ concerns the redundancy reduction criterion. A new binary decision variable $y_{ij}$ must be defined. This variable is related to the pair of sentences $s_i$ and $s_j$, i.e., $y_{ij} = 1$ if $s_i$ and $s_j$ are presented simultaneously in the generated summary $S$, and $y_{ij} = 0$ otherwise. For each pair of sentences $s_i, s_j \in S$, the cosine similarity between them $sim(s_i, s_j)$ should be minimized. This is the same as maximizing the following objective function:

$$\Phi_{ReR}(X) = \frac{1}{\left(\sum_{i=1}^{n-1} \sum_{j=i+1}^{n} sim(s_i, s_j) \cdot y_{ij}\right) \cdot \sum_{i=1}^{n} x_i}. \tag{5}$$

Finally, after defining the objective functions to be maximized, the multi-objective extractive multi-document text summarization problem can be formulated as follows:

$$\max \ \Phi(X) = \{\Phi_{Cov}(X), \Phi_{ReR}(X)\}, \tag{6}$$

$$\text{subject to } L - \varepsilon \leq \sum_{i=1}^{n} l_i \cdot x_i \leq L + \varepsilon. \tag{7}$$

The variable $l_i$ is the length of the sentence $s_i$ and $\varepsilon$ is the tolerance for the summary length constraint, defined as $\max_{i=1,2,\ldots,n} l_i - \min_{i=1,2,\ldots,n} l_i$.

## 4. Multi-Objective Artificial Bee Colony based on Decomposition

This section presents the proposed Multi-Objective Artificial Bee Colony based on Decomposition (MOABC/D). The fundamentals of the standard ABC (Artificial Bee Colony) algorithm can be found in Karaboga and Basturk [36]. ABC algorithm has been successfully applied to different real-world problems (Karaboga et al. [37]), motivating its selection as part of the approach proposed here. In the following subsections, firstly, the decomposition-based multi-objective optimization is described. Secondly, indications about the preprocessing of the input documents are presented, and then, the main steps of the MOABC/D algorithm and its main operators are detailed. Finally, the asynchronous parallel design of the algorithm is explained.

### *4.1. Decomposition-based Multi-Objective Optimization*

The principles of multi-objective optimization can be consulted in Deb [38]. This subsection describes the decomposition-based multi-objective optimization technique applied in this work. Without loss of generality, we assume that all the objective functions have to be maximized. In traditional multi-objective optimization problems, the objectives contained in the set of objective functions are conflicting among them. That is, there is no point in the objective space maximizing all objective functions simultaneously, so the best tradeoffs found are called Pareto optimal points. In these points, which make up the Pareto front, any improvement in one objective leads to a deterioration in, at least, other. This is the case of the problem presented here (Equation 6).

The basic idea of decomposition-based multi-objective optimization consists of the decomposition of the Pareto front into a number of scalar subproblems. In addition, every subproblem is optimized by using the information from its neighboring subproblems. Mathematically, the optimization problem is defined as follows.

Supposing two objective functions (like in Equation 6), let $\lambda^i = (\lambda_1^i, \lambda_2^i)$ be a weight vector with two components (one for each objective), and $\sum_{j=1}^{2} \lambda_j^i = 1$. In addition, let $\Lambda = \{\lambda^1, \lambda^2, \ldots, \lambda^{pop_{size}}\}$ be a set with $pop_{size}$ (the population size) weight vectors, that is, the Pareto front is divided into $pop_{size}$ scalar subproblems. The neighborhood of the weight vector $\lambda^i$ is defined as a set of its closest weight vectors in $\{\lambda^{i,1}, \lambda^{i,2}, \ldots, \lambda^{i,niche}\}$, being $niche$ the predetermined size of the neighborhood. Let $z^* = (z_1^*, z_2^*)$ be the reference point, where $z_j^*$ is the best value found for the objective $j$. Figure 1 shows how the Pareto front is divided into several scalar subproblems. In this example, ten weight vectors ($\lambda^i$) divide the objective space into ten evenly-distributed subspaces. In every subspace, each point is optimized as a single-objective subproblem according to its corresponding weight vector, also taking into account the information from its neighboring subproblems.

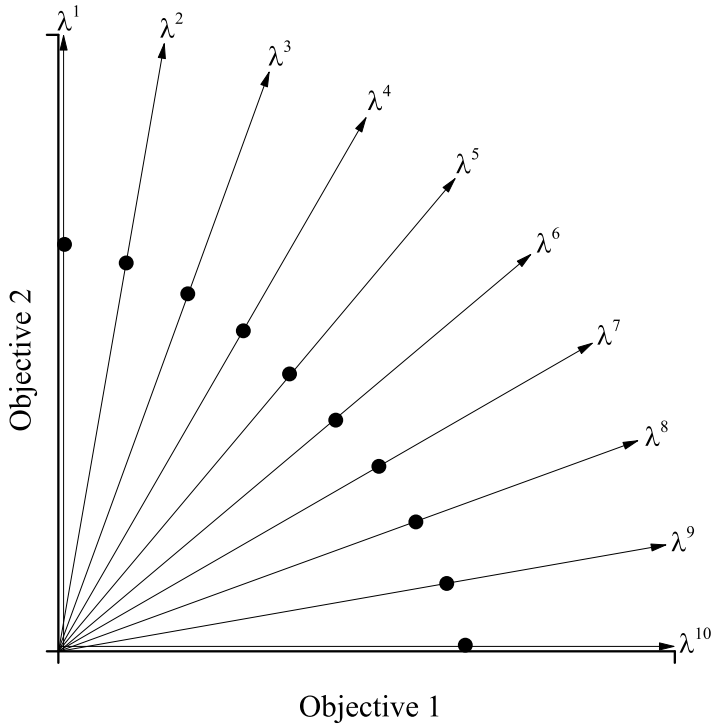


Figure 1: Decomposition-based approximation for dividing the Pareto front into several scalar subproblems.

In the scientific literature, there are several mathematical methods to transform the approximation of the Pareto front to scalar optimization subproblems. Three of the most common approaches, and the ones experimented in this work, are: boundary intersection (Das and Dennis [14]), non-normalized Tchebycheff (Jaszkiewicz [15]), and normalized Tchebycheff (Tang et al. [16]). First, the boundary intersection method aims to find intersection points of the most top boundary and a set of lines evenly distributed. Secondly, in the non-normalized Tchebycheff method, it is possible to obtain different Pareto optimal solutions by altering the weight vector of every Pareto optimal point. Finally, the normalized Tchebycheff method is similar to the previous one, only taking into account the distance between the reference point and the Pareto optimal point for each objective function.

*4.2. Preprocessing*

The input documents from the document collection need to be preprocessed before the execution of the algorithm. In the Natural Language Processing

(NLP) field, there are various techniques for preprocessing the text to a normalized form, such as lemmatization and stemming algorithms (Toman et al. [39]). Lemmatization replaces or removes the suffix of a word to get the basic form, which is called lemma, whereas word stemming produces an approximation of the basic form, called stem. According to Toman et al. [39], the Porter stemming algorithm is the most appropriate algorithm for word normalization in English language. In fact, it is the most used one in the extractive multi-document text summarization field (see the related works in Section 2). Therefore, word stemming is applied in this paper with the following preprocessing steps:

1. Sentence segmentation. All the sentences in the document collection are separated with the aim of defining their beginning and ending.

2. Word tokenization. All the words from the sentences are extracted separately. Exclamation, interrogation, punctuation, and other marks are removed.

3. Stop word removal. Stop words are common words that have no relevant meaning, such as prepositions, conjunctions, articles, possessives, pronouns and others. Thus, these words are deleted from the sentences. The ROUGE package (ROUGE [40]) provides a list of stop words in English language, which contains a total of 598 words.

4. Word stemming. Finally, by means of the Porter stemming algorithm (Porter [41]), the roots of the remaining words are extracted, allowing that the words with the same lexical root will be processed as a same term. Porter stemming algorithm is one of the most adopted and extended algorithms, becoming a standard to word conflation for information retrieval in a wide range of languages (Willett [42]).

Figure 2 represents a flow chart with the steps followed by the proposed method.

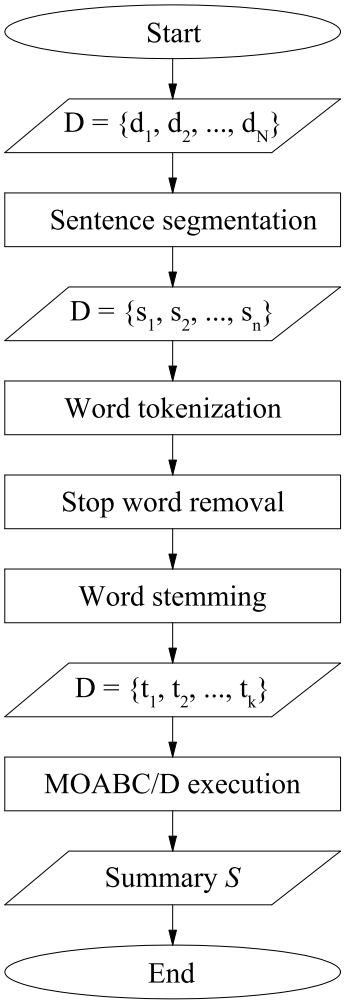


Figure 2: Flow chart of the steps followed by the proposed method.

### *4.3. Main Steps of the Algorithm*

In this subsection, the main steps of the MOABC/D algorithm are presented by the pseudocode in Algorithm 1 and later fully described.

**Algorithm 1** MOABC/D pseudocode.

```
1: NDS_file ← ∅
2: initialize_reference(refPoint)
3: for i = 1 to pop_size do
4:    initialize_colony(i, Colony)
5: end for
6: for i = 1 to pop_size do
7:    update_reference(i, Colony, refPoint)
8: end for
9: for cycle = 1 to cycles_max do
10:   for i = 1 to pop_size do
11:      send_employed_bee(i, Colony)
12:   end for
13:   calculate_probabilities(Colony)
14:   for i = 1 to pop_size do
15:      send_onlooker_bee(i, Colony)
16:   end for
17:   for i = 1 to pop_size do
18:      send_scout_bee(i, Colony, cycle)
19:   end for
20:   for i = 1 to pop_size do
21:      update_reference(i, Colony, refPoint)
22:   end for
23:   for i = 1 to pop_size do
24:      update_problem(i)
25:   end for
26:   for i = 1 to pop_size do
27:      export_bee(i, NDS_file)
28:   end for
29: end for
```

In the first place, the storing file $NDS_{file}$, that will contain the non-dominated solutions, is initialized to an empty set in line 1. Then, the reference point ($refPoint$) is initialized in line 2. After that, in line 4, the $Colony$ (a population of $pop_{size}$ solutions) is initialized. In this step, firstly, the set of evenly-distributed weight vectors is initialized. Secondly, the neighborhood of the subproblems is initialized based on the distances of the weight vectors.

And finally, the initial solution population is randomly generated. The last step of the initialization is the update of the reference point in line 7.

Once the initialization steps are performed, the following steps of the algorithm (lines 9 to 29) are repeated during a maximum number of cycles $cycles_{max}$. At the beginning of a cycle, the employed bees step is performed (line 11). Every employed bee has a solution associated. The mutation operator, that is explained in Subsection 4.4, is applied in this case with the aim of generating a neighbor solution to each current solution. If the mutated solution improves the original one, the mutated solution will replace it. Otherwise, the mutated solution is discarded. Then, the selection probabilities are calculated in line 13. They are calculated by means of a scalar function, which will depend on the considered decomposition approach. After that, the onlooker bees step is carried out (line 15). Each one of the onlooker bees selects its corresponding employed bee solution taking into account the selection probabilities previously calculated. In this way, the onlooker bee tries to improve its employed bee solution. The mutation operator is applied in this step too. The colony size is duplicated at this point of the algorithm, due to the onlooker bee solutions.

The last bee step is the scout bees step (line 18). It verifies if the current solutions are exhausted. A solution is exhausted if it has not been improved after a determined number of tries. The exhausted bees/solutions are replaced by scout bee solutions, which are randomly generated and mutated with a number of mutations proportional to the current number of cycles.

At the end of a cycle, the colony size is reduced to the half, that is, its original size ($pop_{size}$). Two steps related to decomposition are carried out before ending the cycle in lines 21 and 24 respectively: updating the reference point and updating the best solutions of neighboring subproblems. Finally, the solutions of the current cycle are repaired (as explained in Subsection 4.5), and they are exported to $NDS_{file}$ (line 27).

### *4.4. Mutation Operator*

The mutation operation carried out here consists of adding or removing sentences in the summary. Let $p_m \in (0, 1)$ be the mutation probability. A uniform pseudo-random number $r_i \sim U(0, 1)$ is generated for every sentence $s_i$ contained in the summary $S$. The sentence $s_i$ is candidate for mutation if

$r_i \leq p_m$, and finally, if the following condition is fulfilled:

$$sim(s_i, O) \geq \frac{1}{n} \sum_{j=1}^{n} sim(s_j, O). \tag{8}$$

Then, the sentence $s_i$ will be included in the summary $S$. Otherwise, the sentence will be removed. The purpose of the previous condition is checking that the cosine similarity between the candidate sentence $s_i$ and the mean vector $O$ is greater than or equal to the average cosine similarity of all sentences from the document collection. In this way, the best quality sentences will remain in the solution.

*4.5. Repair Operator*

In order to satisfy the length constraint of the generated summary defined in Equation 7, the repair operation is performed. This operator is the same as the one used in NSGA-II and MOABC algorithms (Saleh et al. [34] and Sanchez-Gomez et al. [9], respectively). At the end of a cycle, the generated summary may violate the length constraint. The length constraint must be checked in two directions: on the one hand, if the generated summary has a length smaller than $L - \varepsilon$, the summary must be discarded; and on the other hand, if the generated summary has a length greater than $L + \varepsilon$, the summary must be repaired. This decision has been taken due to the fact that the number of generated summaries having a length smaller than the length constraint is negligible. Now, the repair operator is explained.

Let $S^*$ be a generated summary that has more sentences than what is allowed. The repair operation removes from $S^*$ one or more sentences that have a high cosine similarity degree among them in order to obtain a valid generated summary. For this purpose, it has been considered a similarity threshold $\delta = 0.9$ (Saleh et al. [34]), since 90% of concordance seems reasonable to check if sentences match. Then, every pair of sentences $s_i$ and $s_j$ from the generated summary $S^*$ is checked, and the repair operation is performed if the following conditions are fulfilled:

$$\{s_i, s_j \in S^*\} \wedge \{i \neq j\} \wedge sim(s_i, s_j) \geq \delta \quad i, j = 1, 2, \ldots, n. \tag{9}$$

Finally, the repair operation removes the worst sentence from this pair. That is, the sentence with lower score is removed. The score is calculated as

follows:

$$score_{s_i} = sim(s_i, O) + \big((sim(O^{sum}, O) - sim(o^{sum-s_i}, O)\big) \cdot 10. \qquad (10)$$

The expression $sim(O^{sum}, O)$ is the cosine similarity between the center of the generated summary (including the sentence $s_i$) and the center of document collection $O$, and $sim(O^{sum-s_i}, O)$ is the cosine similarity between the center of the generated summary (excluding the sentence $s_i$) and the center of document collection $O$. The second term is multiplied by 10, that is, it has an order of magnitude more, because it measures the quality of the generated summary when sentence $s_i$ is removed. The repair operator will be carried out until the length constraint is satisfied.

### *4.6. Asynchronous Parallel MOABC/D Algorithm*

This subsection presents an asynchronous parallel design for the MOABC/D algorithm. In contrast to the traditional design, which follows a generational synchronous scheme, the proposed design is based on the real behavior of the honey bees in the nature, i.e., when a bee finishes its task, it is not necessary to wait for other bees to continue with the following task. Therefore, its behavior is asynchronous.

The scheme followed by the asynchronous parallel MOABC/D algorithm is based on a master-worker parallel scheme (Zăvoianu et al. [43]). It is the most suitable way to parallelize evolutionary algorithms as MOABC (Talbi et al. [44]). Two roles for execution threads are considered in this scheme: master thread and worker thread. Firstly, the master thread is the one that manages the entire colony. It carries out the calculation of the selection probabilities (*calculate_probabilities* step) and the update of the reference point and subproblems (*update_reference* and *update_problem* steps), and manages the shared data structures needed for communication with the worker threads. And secondly, the worker threads perform the tasks related to the bees steps until the master thread signalizes them the end of the execution. During these tasks, the worker threads use the shared data structures to communicate with the master thread in every iteration without waiting.

Once the master-worker scheme has been described, the asynchronous parallel MOABC/D algorithm is presented. Algorithm 2 shows its global design, including the OpenMP directives (Chapman et al. [45]) that mark the parallel regions.

**Algorithm 2** Asynchronous parallel MOABC/D pseudocode - global design.

```
1: NDS_file ← ∅
2: initialize_reference(refPoint)
3: initialize_beeFIFOs(BeeFIFOs)
4: #pragma omp parallel num_threads(threads_num)
5: {
6:    #pragma omp for
7:    for i = 1 to pop_size do
8:       initialize_colony(i, Colony)
9:    end for
10:   #pragma omp for
11:   for i = 1 to pop_size do
12:      update_reference(i, Colony, refPoint)
13:   end for
14: }
15: Colony_aux ← Colony
16: refPoint_aux ← refPoint
17: calculate_probabilities(Colony)
18: #pragma omp parallel num_threads(threads_num)
19: {
20:   thread_id ← omp_get_thread_num()
21:   if thread_id == threads_num then
22:      master_tasks(Colony, Colony_aux, refPoint, refPoint_aux, BeeFIFOs)
23:   else
24:      worker_tasks(Colony, BeeFIFOs[thread_id], NDS_file)
25:   end if
26: }
```

The initializations before the asynchronous parallel region (the second parallel region) are performed in lines 1 to 17. There are some variations regarding Algorithm 1. Firstly, in line 3, the shared data structures for communication between the master thread and the worker threads are initialized. They are queue-type data structures (FIFO, First In First Out), in which each worker thread has its own queue where it stores its generated solutions, and finally the master thread will gather them. Secondly, since the other two initialization steps (lines 8 and 12) can be parallelized, a first parallel region is defined from lines 4 to 14 in order to parallelize the two loops. Thirdly, in

lines 15 and 16, the initialized colony and the reference point are stored in auxiliary copies ($Colony_{aux}$ and $refPoint_{aux}$ respectively). The reason why it is necessary to have a copy of the $Colony$ is because its integrity must be ensured: $Colony$ will be read by the worker threads and will be written by the master thread, and $Colony_{aux}$ will be exclusively handled by the master thread. As for the reference point, the purpose is the same. Finally, in line 17, the *calculate_probabilities* step has to be performed for the initial colony. After that, the asynchronous parallel region (the second parallel region) begins (lines 18 to 26). As can be seen, the last thread will perform the master tasks, while the rest of threads will carry out worker tasks.

**Algorithm 3** Asynchronous parallel MOABC/D pseudocode - master tasks.

```
1:  eval_cur ← 0
2:  while eval_cur < eval_max do
3:    updated ← false
4:    for i = 1 to threads_num − 1 do
5:      while not is_empty(BeeFIFOs[i]) do
6:        bee ← pop(BeeFIFOs[i])
7:        if update_problem(bee) then
8:          Colony_aux[bee.position] ← bee
9:          eval_cur ← eval_cur + bee.eval_num
10:         updated ← true
11:       end if
12:     end while
13:   end for
14:   if updated then
15:     for i = 1 to pop_size do
16:       update_reference(i, Colony_aux, refPoint_aux)
17:     end for
18:     interchange_reference_points(refPoint, refPoint_aux)
19:     calculate_probabilities(Colony_aux)
20:     interchange_colonies(Colony, Colony_aux)
21:   end if
22: end while
23: send_finalization_signal()
```

As shown in Algorithm 3, the master tasks are carried out until reaching

a maximum number of evaluations $eval_{max}$. In the first place, in lines 4 to 13, the master thread empties all shared queues of the worker threads, and for each solution generated, it checks if the solution updates the best solutions of its neighboring subproblems (line 7). If so, the generated solution is stored in the auxiliary colony $Colony_{aux}$, the evaluation counter $eval_{cur}$ is increased, and the boolean variable *updated* is changed to true (lines 8, 9, and 10). In the second place, if the auxiliary colony $Colony_{aux}$ has been updated (*updated* is true), the master thread performs the tasks related to the management of the colony (lines 14 to 21). Once the auxiliary colony $Colony_{aux}$ has been modified, its auxiliary reference point $refPoint_{aux}$ has to be updated (line 16). Then, in line 18 the main reference point $refPoint$ is interchanged in order to take the updated information from the auxiliary reference point $refPoint_{aux}$. After that, the *calculate_probabilities* step is carried out (line 19), and the main colony $Colony$ is also interchanged with the auxiliary colony $Colony_{aux}$ (line 20), taking the updated information of the colony. Finally, in line 23 the master thread broadcasts the finalization signal to each worker thread when the maximum number of evaluations $eval_{max}$ has been reached.

**Algorithm 4** Asynchronous parallel MOABC/D pseudocode - worker tasks.

```
1: i ← 0
2: while not finalization_signal() do
3:    index ← get_index(pop_size, threads_num, thread_id, i)
4:    if thread_id < threads_num/2 then
5:       bee ← send_employed_bee(index, Colony)
6:    else
7:       bee ← send_onlooker_bee(index, Colony)
8:    end if
9:    if bee.trials > limit then
10:      bee ← send_scout_bee(index, Colony, eval_cur)
11:   end if
12:   push(BeeFIFOs[thread_id], bee)
13:   i ← i + 1
14: end while
15: while not is_empty(BeeFIFOs[thread_id]) do
16:   export_bee(pop(BeeFIFOs[thread_id]), NDS_file)
17: end while
```

In the second place, the worker tasks are presented in Algorithm 4. The worker tasks can be performed by the employed workers or the onlooker workers. They are executed until the finalization signal is received from the master thread (lines 2 to 14). At the beginning of the task (line 3), the worker thread must know its corresponding chunk of the colony by means of its specific index. Then, the employed tasks or the onlooker tasks are carried out depending on the thread identifier ($thread_{id}$): the first half of the execution threads sends the employed bees (line 5), and the second one sends the onlooker bees (line 7) except the last one (the master thread). Afterward, in line 9, if the trial counter of *bee* surpasses the maximum number of trials ($limit$), a scout bee is sent in order to replace it (line 10). At the end, in line 12, the *bee* is stored in its corresponding shared queue $BeeFIFOs[thread_{id}]$. To finish the worker tasks, once the finalization signal is sent by the master thread, the remaining solutions in $BeeFIFOs[thread_{id}]$ are exported to $NDS_{file}$ (lines 15 to 17).

## 5. Experimental Results

This section includes the datasets used, the experimental settings, the evaluation metrics and performance measures, and the results obtained.

### *5.1. Datasets*

The performance has been evaluated using the multi-document summarization datasets provided by *Document Understanding Conferences* (DUC). It is an open benchmark from the *National Institute of Standards and Technology* (NIST) for the evaluation of generic automatic summarization. The used datasets have been obtained from DUC2002 (NIST [46]), which have been widely used in the field. They consist on a set of topics, and each topic contains several newspaper articles that inform about a single specific subject. Table 3 shows some features.

| **Description** | **Features** |
|---|---|
| Topics used | 10 (d061j through d070f) |
| Total of documents | 77 |
| Average documents in each topic | $\sim$ 8 (from 5 to 14) |
| Summary length constraint | 200 words |

Table 3: Some features of the used topics from DUC2002.

The documents contained in DUC2002 have been preprocessed following the steps explained in Subsection 4.2. Table 4 presents the number of words before and after preprocessing and the number of sentences of each topic.

| **Topic** | **No. of words before preprocessing** | **No. of words after preprocessing** | **No. of sentences** |
|---|---|---|---|
| d061j | 3679 | 693 | 184 |
| d062j | 2691 | 630 | 118 |
| d063j | 4793 | 846 | 249 |
| d064j | 4080 | 924 | 183 |
| d065j | 5500 | 1080 | 284 |
| d066j | 3894 | 923 | 190 |
| d067f | 2805 | 644 | 122 |
| d068f | 2565 | 531 | 131 |
| d069f | 7767 | 1306 | 327 |
| d070f | 3116 | 620 | 148 |

Table 4: Some counts of each used topic.

*5.2. Experimental Settings*

The parameter setting for the MOABC/D algorithm has been the same as used in Sanchez-Gomez et al. [9] in order to provide fair comparisons: population size, $pop_{size} = 64$; mutation probability, $p_m = 0.1$; number of cycles, $cycles_{max} = 1000$; and number of independent runs or repetitions, $reps_{max} = 20$. In addition to these settings, two more related to decomposition have been used: the neighborhood or *niche* size and the decomposition approach. After carrying out an experimental study, it has been determined that the best configuration for the decomposition settings are: $niche = 3$ and the Normalized Tchebycheff approach.

The experiments have been executed in a compute node with 4 processors (16 cores per processor) AMD Opteron Abu Dhabi 6376 2.3GHz and 96 GB RAM. The approach has been implemented in C/C++ language, and developed in Eclipse platform on Ubuntu 16.04 LTS. The OpenMP version was 4.0, and the C/C++ compiler version was GCC 5.3.0.

*5.3. Evaluation Metrics*

The approach performance has been evaluated by using *Recall-Oriented Understudy for Gisting Evaluation* (ROUGE) metric (Lin [47]). The *Document*

*Understanding Conferences* consider ROUGE as the official evaluation metric for text summarization. This metric calculates the similarity between a system-generated summary and a human-generated summary by counting the number of overlapping units. Two variants of ROUGE scores have been used in this work: ROUGE-N and ROUGE-L. ROUGE-N compares the $N - gram$ recall of the system-generated summary and a set of human-generated summaries. In this case, ROUGE-1 and ROUGE-2 have been used. ROUGE-L measures the ratio between the length of the summaries' longest common subsequence and the length of the reference summary. These two metrics have been selected to provide fair comparisons with results from other approaches in the scientific literature.

Two dispersion statistics have also been considered: range and a coefficient of variation-type statistics. The range is calculated as follows:

$$Range = ROUGE_{best} - ROUGE_{worst}. \tag{11}$$

It provides information about the data dispersion. Nevertheless, the range is not an adimensional measure. For this reason, it is complemented with a modification of the Pearson's coefficient of variation (CV). This coefficient involves the relation between the range and the arithmetic mean, leading to an adimensional measure of the data relative dispersion. The CV is calculated as follows:

$$CV = \frac{Range}{ROUGE_{average}} \cdot 100. \tag{12}$$

In addition to the previous metrics, other measures have been used to assess the performance of the asynchronous parallel design of the MOABC/D algorithm. One of them is the execution time (ET) that represents the total computing time of one execution, whereas the other two measures, speedup (S) and efficiency (E), are the most common ones in this field to measure the performance improvement. On the one hand, speedup measures the improvement factor for a number $c$ of threads in execution, and it is calculated as:

$$S_c = \frac{ET_1}{ET_c}. \tag{13}$$

$ET_1$ is the execution time of the serial version and $ET_c$ is the execution time with a number of threads $c$ (being $c > 1$). On the other hand, efficiency

is expressed in percentage terms, taking into account the number of threads $c$:

$$E_c = \frac{S_c}{c} \cdot 100 = \frac{ET_1}{c \cdot ET_c} \cdot 100. \quad (14)$$

Finally, hypothesis tests have been considered. Specifically, Mann-Whitney $U$ test has been applied on ROUGE scores to report statistically significant differences between two approaches. Two-side $p$-values smaller than 0.05 have been considered as statistically significant. R software (R Development Core Team [48]) has been used for the statistical analysis.

### *5.4. Comparing MOABC/D with MOABC*

In this subsection, the results obtained by the proposed MOABC/D algorithm are compared with those of the MOABC algorithm presented in Sanchez-Gomez et al. [9]. In that work, the results obtained by the MOABC algorithm improved to those existing in the scientific literature. Firstly, Table 5 presents average, range, and CV for ROUGE-1 scores for each one of the used topics and the average values obtained. The $p$-values obtained from pairwise comparison between both approaches are also shown.

| **Topic** | **MOABC (Sanchez-Gomez et al. [9])** | | | **MOABC/D** | | | $p$**-values** |
|---|---|---|---|---|---|---|---|
| | **Average** | **Range** | **CV** | **Average** | **Range** | **CV** | |
| d061j | 0.539 | 0.067 | 12.38 | 0.670 | 0.040 | 5.94 | <0.001 |
| d062j | 0.536 | 0.033 | 6.10 | 0.661 | 0.061 | 9.18 | <0.001 |
| d063j | 0.456 | 0.040 | 8.70 | 0.533 | 0.057 | 10.69 | <0.001 |
| d064j | 0.495 | 0.015 | 3.06 | 0.549 | 0.061 | 11.06 | <0.001 |
| d065j | 0.373 | 0.046 | 12.33 | 0.513 | 0.034 | 6.71 | <0.001 |
| d066j | 0.471 | 0.015 | 3.25 | 0.541 | 0.055 | 10.23 | <0.001 |
| d067f | 0.567 | 0.020 | 3.53 | 0.571 | 0.072 | 12.55 | 0.081 |
| d068f | 0.655 | 0.093 | 14.25 | 0.622 | 0.064 | 10.22 | <0.001 |
| d069f | 0.417 | 0.020 | 4.81 | 0.415 | 0.033 | 8.07 | 0.253 |
| d070f | 0.482 | 0.012 | 2.42 | 0.454 | 0.019 | 4.08 | <0.001 |
| **Average** | **0.499** | **0.036** | **7.08** | **0.553** | **0.050** | **8.87** | |

Table 5: Average, range, and CV of ROUGE-1 scores. The best values appear in grey background. In the case of the results are not statistically significant, both approaches are highlighted in grey background.

The results shown in Table 5 report that the MOABC/D algorithm outperforms MOABC algorithm. The proposed MOABC/D algorithm obtains the best ROUGE-1 scores in 6 out of 10 topics, while MOABC algorithm is better in only 2 topics. 8 out of 10 topics present statistically significant differences (topics d067f and d069f did not obtain statistically significant differences). Besides, both algorithms provide similar average values regarding the data dispersion. In average terms, the MOABC/D algorithm has obtained an improvement percentage of 10.82% for ROUGE-1 score. Figure 3 shows graphically the results for ROUGE-1 scores. The results of the 10 topics are represented as columns, and the average ROUGE-1 scores are symbolized as discontinuous lines.

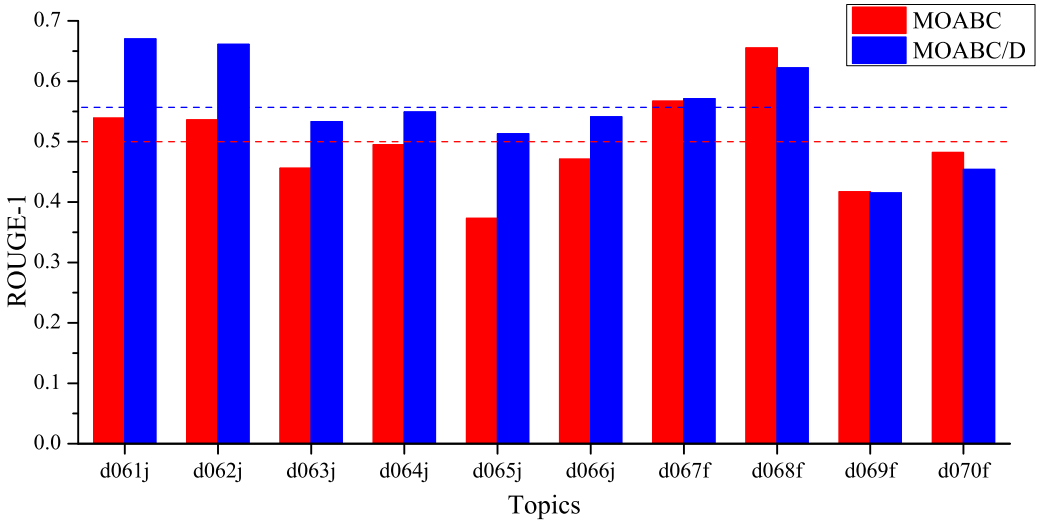


Figure 3: Comparison of MOABC/D with MOABC for ROUGE-1 score for the 10 topics. The discontinuous lines represent the average ROUGE-1 scores.

Secondly, the results obtained for ROUGE-2 score are presented in Table 6. It includes the same information as ROUGE-1 score.

| Topic | MOABC (Sanchez-Gomez et al. [9]) | | | MOABC/D | | | $p$-values |
|---|---|---|---|---|---|---|---|
| | Average | Range | CV | Average | Range | CV | |
| d061j | 0.365 | 0.093 | 25.43 | 0.472 | 0.022 | 4.70 | <0.001 |
| d062j | 0.342 | 0.023 | 6.60 | 0.466 | 0.027 | 5.74 | <0.001 |
| d063j | 0.272 | 0.005 | 1.84 | 0.329 | 0.052 | 15.93 | <0.001 |
| d064j | 0.308 | 0.009 | 2.83 | 0.352 | 0.049 | 13.97 | <0.001 |
| d065j | 0.198 | 0.026 | 13.32 | 0.231 | 0.031 | 13.30 | <0.001 |
| d066j | 0.290 | 0.019 | 6.54 | 0.329 | 0.023 | 7.05 | <0.001 |
| d067f | 0.356 | 0.005 | 1.39 | 0.353 | 0.029 | 8.34 | <0.001 |
| d068f | 0.444 | 0.084 | 18.83 | 0.396 | 0.037 | 9.30 | <0.001 |
| d069f | 0.240 | 0.008 | 3.15 | 0.238 | 0.037 | 15.54 | 0.068 |
| d070f | 0.305 | 0.002 | 0.76 | 0.254 | 0.018 | 7.26 | <0.001 |
| **Average** | **0.312** | **0.027** | **8.07** | **0.342** | **0.033** | **10.11** | |

Table 6: Average, range, and CV of ROUGE-2 scores. The best values appear in grey background. In the case of the results are not statistically significant, both approaches are highlighted in grey background.

In the same way, the results reported in Table 6 show that the MOABC/D algorithm outperforms MOABC algorithm. The proposed MOABC/D algorithm obtains the best ROUGE-2 scores in 6 out of 10 topics, while MOABC algorithm is better in 3 topics. All the topics present statistically significant differences, except topic d069f, where a tie is obtained. Regarding the data dispersion, it can be observed that both algorithms provide similar average values. In average, an improvement percentage of 9.62% has been obtained for ROUGE-2 score by using MOABC/D algorithm. A visual representation of the results for ROUGE-2 scores is shown in Figure 4. Both the 10 topics' results and the average ROUGE-2 scores are represented.

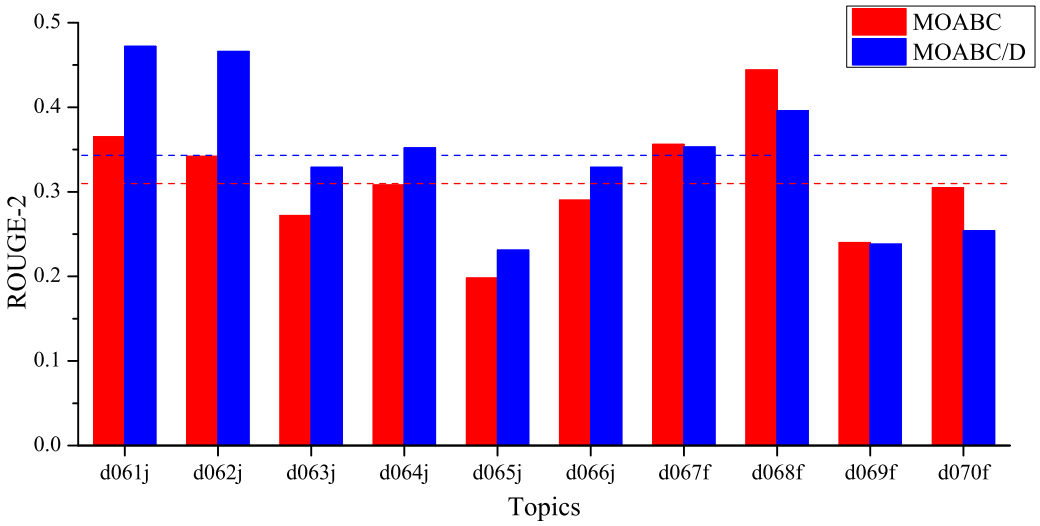


Figure 4: Comparison of MOABC/D with MOABC for ROUGE-2 score for the 10 topics. The discontinuous lines represent the average ROUGE-2 scores.

Finally, the results obtained for ROUGE-L scores are presented in Table 7.

| **Topic** | **MOABC (Sanchez-Gomez et al. [9])** | | | **MOABC/D** | | | ***p*-values** |
|---|---|---|---|---|---|---|---|
| | **Average** | **Range** | **CV** | **Average** | **Range** | **CV** | |
| d061j | 0.590 | 0.056 | 9.46 | 0.616 | 0.032 | 5.27 | <0.001 |
| d062j | 0.536 | 0.017 | 3.26 | 0.588 | 0.045 | 7.62 | <0.001 |
| d063j | 0.509 | 0.050 | 9.74 | 0.524 | 0.017 | 3.33 | <0.001 |
| d064j | 0.495 | 0.011 | 2.19 | 0.539 | 0.025 | 4.56 | <0.001 |
| d065j | 0.464 | 0.057 | 12.29 | 0.517 | 0.029 | 5.52 | <0.001 |
| d066j | 0.519 | 0.007 | 1.36 | 0.573 | 0.068 | 11.92 | <0.001 |
| d067f | 0.580 | 0.012 | 2.15 | 0.551 | 0.039 | 7.10 | <0.001 |
| d068f | 0.639 | 0.071 | 11.09 | 0.653 | 0.032 | 4.90 | 0.011 |
| d069f | 0.554 | 0.010 | 1.81 | 0.544 | 0.048 | 8.82 | 0.565 |
| d070f | 0.515 | 0.005 | 0.90 | 0.513 | 0.030 | 5.86 | 0.289 |
| **Average** | **0.540** | **0.030** | **5.43** | **0.562** | **0.037** | **6.49** | |

Table 7: Average, range, and CV of ROUGE-L scores. The best values appear in grey background. In the case of the results are not statistically significant, both approaches are highlighted in grey background.

From the results reported in Table 7, it can also be concluded that the

MOABC/D algorithm outperforms MOABC algorithm. The MOABC/D algorithm performs better than MOABC in 7 out of 10 topics, while MOABC algorithm does it in only 1 topic. All topics, except d069f and d070f, present statistically significant differences. Again both algorithms provide similar average values in range and CV. In this case, in average, an improvement percentage of 4.07% has been obtained for ROUGE-L score by using MOABC/D algorithm. Figure 5 shows the results of the ROUGE-L score in a visual way, including the 10 topics used and the average ROUGE-L scores.

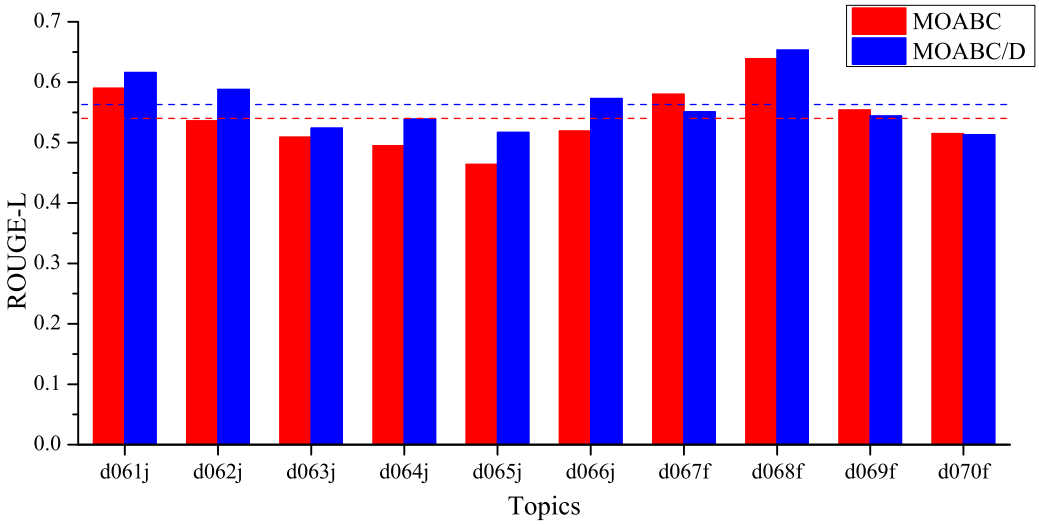


Figure 5: Comparison of MOABC/D with MOABC for ROUGE-L score for the 10 topics. The discontinuous lines represent the average ROUGE-L scores.

### *5.5. Comparing MOABC/D with NSGA-II and Adaptive DE*

This subsection presents the comparison of the results obtained among the proposed MOABC/D algorithm and the algorithms presented in works Alguliev et al. [8] and Saleh et al. [34]. On the one hand, Alguliev et al. [8] is a single-objective approach with an Adaptive DE algorithm which obtained very good results in this problem. On the other hand, Saleh et al. [34] is based on NSGA-II and it is the multi-objective approach from other authors that obtained the best results in the scientific literature. Both of them experimented with the same datasets (DUC2002), and used the same ROUGE scores that are used in this work.

In the first place, Table 8 shows average, range, and CV of ROUGE-2 scores for each used topic and their average values for these three approaches.

| Topic | Adaptive DE (Alguliev et al. [8]) | | | NSGA-II (Saleh et al. [34]) | | | MOABC/D | | |
|---|---|---|---|---|---|---|---|---|---|
| | **Average** | **Range** | **CV** | **Average** | **Range** | **CV** | **Average** | **Range** | **CV** |
| d061j | 0.266 | 0.290 | 109.02 | 0.306 | 0.263 | 85.95 | 0.472 | 0.022 | 4.70 |
| d062j | 0.188 | 0.275 | 146.28 | 0.200 | 0.422 | 211.00 | 0.466 | 0.027 | 5.74 |
| d063j | 0.245 | 0.208 | 84.90 | 0.275 | 0.279 | 101.45 | 0.329 | 0.052 | 15.93 |
| d064j | 0.194 | 0.280 | 144.33 | 0.233 | 0.356 | 152.79 | 0.352 | 0.049 | 13.97 |
| d065j | 0.144 | 0.209 | 145.14 | 0.182 | 0.208 | 114.29 | 0.231 | 0.031 | 13.30 |
| d066j | 0.201 | 0.257 | 127.86 | 0.181 | 0.245 | 136.36 | 0.329 | 0.023 | 7.05 |
| d067f | 0.239 | 0.235 | 98.33 | 0.260 | 0.298 | 114.62 | 0.353 | 0.029 | 8.34 |
| d068f | 0.491 | 0.384 | 78.21 | 0.496 | 0.281 | 56.65 | 0.396 | 0.037 | 9.30 |
| d069f | 0.184 | 0.166 | 90.22 | 0.232 | 0.239 | 103.02 | 0.238 | 0.037 | 15.54 |
| d070f | 0.224 | 0.260 | 116.07 | 0.262 | 0.215 | 82.03 | 0.254 | 0.018 | 7.26 |
| **Average** | **0.238** | **0.256** | **114.03** | **0.263** | **0.281** | **115.72** | **0.342** | **0.033** | **10.11** |

Table 8: Average, range, and CV of ROUGE-2 scores. Comparison among MOABC/D, NSGA-II, and Adaptive DE. The best values appear in grey background.

From the results reported in Table 8, it can be noted that the proposed MOABC/D algorithm outperforms the other two approaches. MOABC/D algorithm outperforms the Adaptive DE algorithm in Alguliev et al. [8] in 9 out of 10 topics, and to NSGA-II in Saleh et al. [34] in 8 out of 10 topics. In average terms, for the ROUGE-2 score, the proposed approach improves by 43.70% and 30.04% to the other two approaches, respectively. Besides, ranges and CVs show that the results of the proposed algorithm are very robust, being much smaller than those of the other two approaches. This means that the results from all the repetitions are more stable. Considering the average values of the ten topics for CV, the MOABC/D algorithm obtains only 10.11% versus 114.03% and 115.72% obtained by the other two algorithms (Adaptive DE and NSGA-II). Figure 6 shows graphically the results of the ROUGE-2 score for the three approaches compared. In the same way as in previous figures, the results of the 10 topics are represented as columns, and the average ROUGE-2 scores are represented as discontinuous lines.

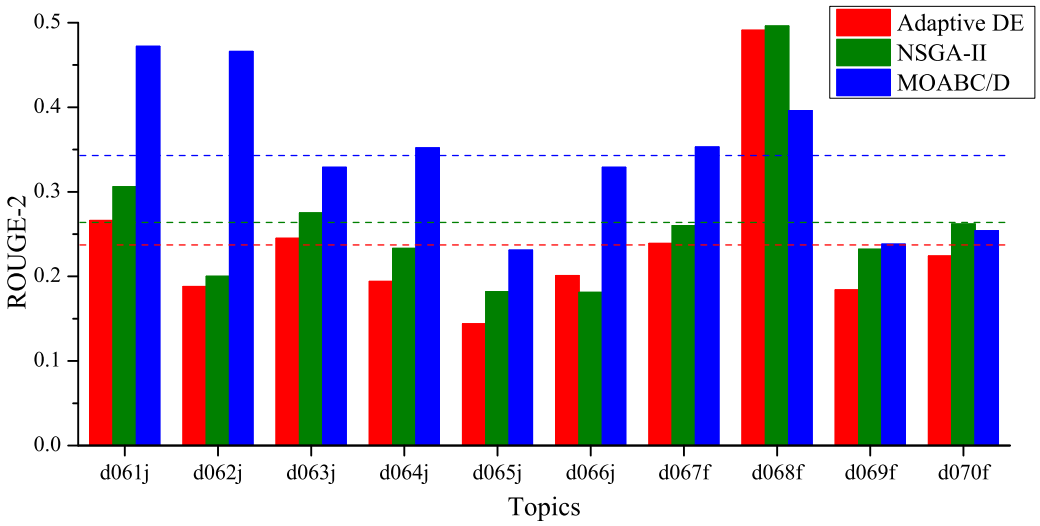


Figure 6: Comparison of MOABC/D with Adaptive DE and NSGA-II for ROUGE-2 score for the 10 topics. The discontinuous lines represent the average ROUGE-2 scores.

Now, average, range, and CV for ROUGE-L scores for each used topic and their average values are presented in Table 9.

| **Topic** | **Adaptive DE (Alguliev et al. [8])** | | | **NSGA-II (Saleh et al. [34])** | | | **MOABC/D** | | |
|---|---|---|---|---|---|---|---|---|---|
| | **Average** | **Range** | **CV** | **Average** | **Range** | **CV** | **Average** | **Range** | **CV** |
| d061j | 0.542 | 0.208 | 38.38 | 0.554 | 0.205 | 37.00 | 0.616 | 0.032 | 5.27 |
| d062j | 0.473 | 0.239 | 50.53 | 0.481 | 0.306 | 63.62 | 0.588 | 0.045 | 7.62 |
| d063j | 0.493 | 0.156 | 31.64 | 0.528 | 0.171 | 32.39 | 0.524 | 0.017 | 3.33 |
| d064j | 0.462 | 0.235 | 50.87 | 0.488 | 0.287 | 58.81 | 0.539 | 0.025 | 4.56 |
| d065j | 0.431 | 0.141 | 32.71 | 0.457 | 0.174 | 38.07 | 0.517 | 0.029 | 5.52 |
| d066j | 0.455 | 0.196 | 43.08 | 0.441 | 0.149 | 33.79 | 0.573 | 0.068 | 11.92 |
| d067f | 0.509 | 0.232 | 45.58 | 0.529 | 0.244 | 46.12 | 0.551 | 0.039 | 7.10 |
| d068f | 0.666 | 0.226 | 33.93 | 0.626 | 0.226 | 36.10 | 0.653 | 0.032 | 4.90 |
| d069f | 0.454 | 0.135 | 29.74 | 0.476 | 0.191 | 40.13 | 0.544 | 0.048 | 8.82 |
| d070f | 0.496 | 0.173 | 34.88 | 0.513 | 0.158 | 30.80 | 0.513 | 0.030 | 5.86 |
| **Average** | **0.498** | **0.194** | **39.13** | **0.509** | **0.211** | **41.68** | **0.562** | **0.037** | **6.49** |

Table 9: Average, range, and CV of ROUGE-L scores. Comparison among MOABC/D, NSGA-II, and Adaptive DE. The best values appear in grey background.

The results presented in Table 9 show that the MOABC/D algorithm also outperforms the other two approaches. The proposed MOABC/D algorithm obtains better values in 9 out of 10 topics when comparing with Adaptive

DE algorithm in Alguliev et al. [8], and in 8 out of 10 topics when comparing with NSGA-II in Saleh et al. [34]. Regarding the average ROUGE-L scores, the proposed approach improves by 12.85% and 10.41% to the other two approaches, respectively. Again ranges and CVs report that the results obtained by MOABC/D algorithm are much more stable. In this case, the proposed algorithm provides a CV of only 6.49%, in comparison with 39.13% and 41.68% for the other two algorithms (Adaptive DE and NSGA-II, respectively). Figure 7 presents the results of the ROUGE-L score in a visual way. Both the 10 topics' results and the average ROUGE-L scores are represented.

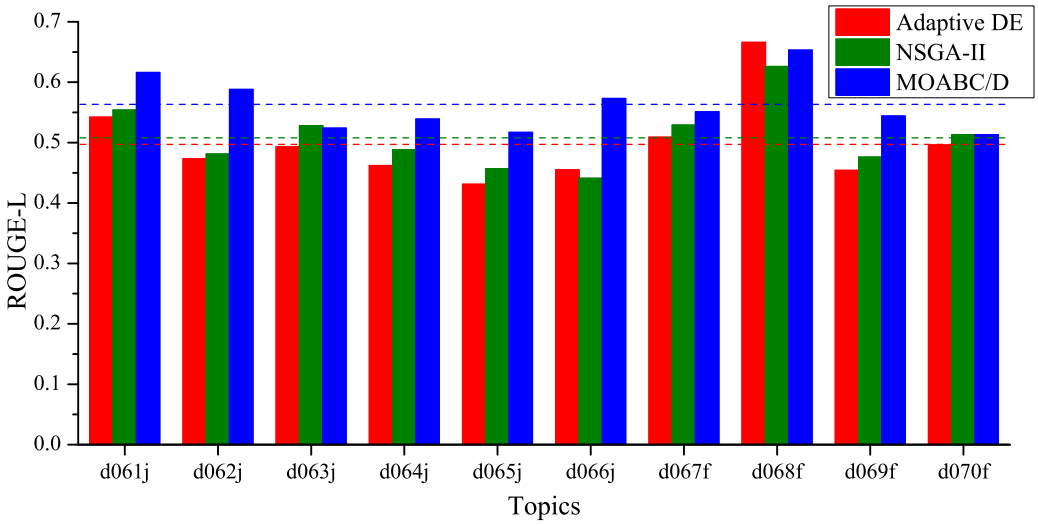


Figure 7: Comparison of MOABC/D with Adaptive DE and NSGA-II for ROUGE-L score for the 10 topics. The discontinuous lines represent the average ROUGE-L scores.

### *5.6. Analyzing the Efficiency*

Finally, this subsection presents a study of the efficiency of the MOABC/D algorithm in comparison with MOABC algorithm in Sanchez-Gomez et al. [9], which provided the best results in the scientific literature. In order to make fair comparisons, the total number of evaluations executed $eval_{max}$ is set to $pop_{size} \cdot cycles_{max}$. The experiments have been performed with 64 threads. Table 10 displays, for every topic, the execution time for MOABC algorithm, the execution time with 64 threads for the proposed MOABC/D algorithm, and the speedup and efficiency obtained.

| Topic | ET for MOABC | ET for MOABC/D | S for MOABC/D | E for MOABC/D |
|---|---|---|---|---|
| d061j | 197058.27 | 3536.18 | 55.73 | 87.07% |
| d062j | 73496.95 | 1408.37 | 52.19 | 81.54% |
| d063j | 475206.18 | 8982.26 | 52.90 | 82.66% |
| d064j | 270498.76 | 4832.18 | 55.98 | 87.47% |
| d065j | 775763.26 | 14220.60 | 54.55 | 85.24% |
| d066j | 295616.36 | 5589.73 | 52.89 | 82.63% |
| d067f | 78333.87 | 1479.17 | 52.96 | 82.75% |
| d068f | 78924.33 | 1476.08 | 53.47 | 83.55% |
| d069f | 1286430.37 | 23688.97 | 54.31 | 84.85% |
| d070f | 114875.14 | 2068.70 | 55.53 | 86.77% |
| **Average** | **364620.35** | **6728.22** | **54.05** | **84.45%** |

Table 10: Comparison between MOABC/D and MOABC: execution times (ET) in seconds, speedups (S), and efficiencies (E).

The results reported in Table 10 show that the proposed MOABC/D algorithm provides excellent speedups and efficiencies. The asynchronous parallel design of MOABC/D achieves an average speedup of 54.05 with 64 threads of execution, that is, an average efficiency of 84.45%, what makes it very efficient in exploiting the potential of multi-core architectures.

## 6. Conclusions

Extractive multi-document text summarization is a problem that demands the simultaneous optimization of more than one objective function, so it requires the application of multi-objective optimization techniques. In the field of multi-objective optimization there are different ways to address this kind of problems. In recent years, decomposition-based approaches have been increasing their popularity. In this paper, a decomposition-based multi-objective optimization approach (MOABC/D) has been proposed to solve this problem. The MOABC/D algorithm has been designed and implemented following an asynchronous parallel design in order to take advantage of multi-core architectures.

The results obtained by using MOABC/D have improved those obtained by other works in the scientific literature. It has been compared with a very good single-objective approach based on an Adaptive DE algorithm (Alguliev et al. [8]) and with the multi-objective approach based on NSGA-II (Saleh et al. [34]) from other authors that obtained the best results for this problem. When MOABC/D is used, the average improvement percentages obtained range from 30.04% to 43.70% for ROUGE-2 scores, and from 10.41% to

12.85% for ROUGE-L scores. Furthermore, the average dispersion measures show that MOABC/D is much more robust (approximately 11 times more for ROUGE-2 and 6 times more for ROUGE-L). Even more, when comparing with the MOABC algorithm presented in Sanchez-Gomez et al. [9], the average improvement percentage obtained is 9.62% and 4.07% for ROUGE-2 and ROUGE-L scores, respectively. In addition, the asynchronous parallel design of MOABC/D algorithm makes it very efficient compared with MOABC. More specifically, the proposed MOABC/D algorithm has reported an average speedup of 54.05 for 64 threads, that is, an average efficiency of 84.45%.

As a future work, the approach will be implemented in NeuroK software [1]. NeuroK is an e-learning platform that is based on social networks principles and neurodidactics (Calle-Alonso et al. [49]). The purpose of the MOABC/D algorithm will be to generate summaries automatically of the texts written by the students in the platform. These texts can belong to learning units or learning activities from an specific course. This would provide teachers a tool that would allow them to easily supervise the students' learning process. Moreover, this tool may also include an automatic qualification of the summary generated with the texts written by each student.

## Acknowledgments

This research has been supported by Agencia Estatal de Investigación - Spain (Projects TIN2016-76259-P and MTM2017-86875-C3-2-R), Junta de Extremadura - Spain (Projects GR18090 and GR18108), and European Union (European Regional Development Fund). Jesus M. Sanchez-Gomez is supported by a doctoral fellowship granted by Junta de Extremadura (Contract PD18057) and European Union (European Social Fund).

## References

[1] W. Fan, A. Bifet, Mining big data: current status, and forecast to the future, ACM SIGKDD Explorations Newsletter 14 (2013) 1–5.

[2] H. Hashimi, A. Hafez, H. Mathkour, Selection criteria for text mining approaches, Computers in Human Behavior 51 (2015) 729–733.

[1] `https://neurok.es/en/`

[3] X. Wan, An exploration of document impact on graph-based multi-document summarization, in: Proceedings of the Conference on Empirical Methods in Natural Language Processing, Association for Computational Linguistics, 2008, pp. 755–762.

[4] D. M. Zajic, B. J. Dorr, J. Lin, Single-document and multi-document summarization techniques for email threads using sentence compression, Information Processing & Management 44 (2008) 1600–1610.

[5] A. Khan, N. Salim, Y. J. Kumar, A framework for multi-document abstractive summarization based on semantic role labelling, Applied Soft Computing 30 (2015) 737–747.

[6] A. Khan, N. Salim, H. Farman, Clustered genetic semantic graph approach for multi-document abstractive summarization, in: 2016 International Conference on Intelligent Systems Engineering (ICISE), IEEE, 2016, pp. 63–70. doi:10.1109/INTELSE.2016.7475163.

[7] V. N. Mendoza, Y. Ledeneva, R. A. García-Hernández, Abstractive multi-document text summarization using a genetic algorithm, in: Mexican Conference on Pattern Recognition, Springer, 2019, pp. 422–432. doi:10.1007/978-3-030-21077-9_39.

[8] R. M. Alguliev, R. M. Aliguliyev, C. A. Mehdiyev, Sentence selection for generic document summarization using an adaptive differential evolution algorithm, Swarm and Evolutionary Computation 1 (2011) 213–222.

[9] J. M. Sanchez-Gomez, M. A. Vega-Rodríguez, C. J. Pérez, Extractive multi-document text summarization using a multi-objective artificial bee colony optimization approach, Knowledge-Based Systems 159 (2018) 1–8.

[10] Q. Zhang, H. Li, MOEA/D: A multiobjective evolutionary algorithm based on decomposition, IEEE Transactions on Evolutionary Computation 11 (2007) 712–731.

[11] H. Li, Q. Zhang, Multiobjective optimization problems with complicated Pareto sets, MOEA/D and NSGA-II, IEEE Transactions on Evolutionary Computation 13 (2009) 284–302.

[12] Q. Zhang, W. Liu, E. Tsang, B. Virginas, Expensive multiobjective optimization by MOEA/D with gaussian process model, IEEE Transactions on Evolutionary Computation 14 (2010) 456–474.

[13] L. Ke, Q. Zhang, R. Battiti, MOEA/D-ACO: A multiobjective evolutionary algorithm using decomposition and ant colony, IEEE Transactions on Cybernetics 43 (2013) 1845–1859.

[14] I. Das, J. E. Dennis, Normal-boundary intersection: A new method for generating the Pareto surface in nonlinear multicriteria optimization problems, SIAM Journal on Optimization 8 (1998) 631–657.

[15] A. Jaszkiewicz, On the performance of multiple-objective genetic local search on the 0/1 knapsack problem - A comparative experiment, IEEE Transactions on Evolutionary Computation 6 (2002) 402–412.

[16] L. Tang, X. Zuo, C. Wang, X. Zhao, A MOEA/D based approach for solving robust double row layout problem, in: Evolutionary Computation (CEC), 2015 IEEE Congress on, IEEE, 2015, pp. 1966–1973. doi:10.1109/CEC.2015.7257126.

[17] R. M. Alguliev, R. M. Aliguliyev, M. S. Hajirahimova, C. A. Mehdiyev, MCMR: Maximum coverage and minimum redundant text summarization model, Expert Systems with Applications 38 (2011) 14514–14522.

[18] R. M. Alguliev, R. M. Aliguliyev, C. A. Mehdiyev, pSum-Sade: a modified p-median problem and self-adaptive differential evolution algorithm for text summarization, Applied Computational Intelligence and Soft Computing 2011 (2011) 1–13.

[19] R. M. Alguliev, R. M. Aliguliyev, N. R. Isazade, DESAMC+DocSum: Differential evolution with self-adaptive mutation and crossover parameters for multi-document summarization, Knowledge-Based Systems 36 (2012) 21–38.

[20] R. M. Alguliev, R. M. Aliguliyev, C. A. Mehdiyev, An optimization model and DPSO-EDA for document summarization, International Journal of Information Technology and Computer Science (IJITCS) 3 (2011) 59–68.

[21] R. M. Alguliev, R. M. Aliguliyev, N. R. Isazade, Formulation of document summarization as a 0–1 nonlinear programming problem, Computers & Industrial Engineering 64 (2013) 94–102.

[22] R. M. Alguliev, R. M. Aliguliyev, M. S. Hajirahimova, Quadratic boolean programming model and binary differential evolution algorithm for text summarization, Problems of Information Technology 3 (2012) 20–29.

[23] R. M. Alguliev, R. M. Aliguliyev, M. S. Hajirahimova, GenDocSum+MCLR: Generic document summarization based on maximum coverage and less redundancy, Expert Systems with Applications 39 (2012) 12460–12473.

[24] R. M. Alguliev, R. M. Aliguliyev, N. R. Isazade, CDDS: Constraint-driven document summarization models, Expert Systems with Applications 40 (2013) 458–465.

[25] R. M. Alguliev, R. M. Aliguliyev, N. R. Isazade, Multiple documents summarization based on evolutionary optimization algorithm, Expert Systems with Applications 40 (2013) 1675–1689.

[26] R. M. Alguliev, R. M. Aliguliyev, C. A. Mehdiyev, An optimization approach to automatic generic document summarization, Computational Intelligence 29 (2013) 129–155.

[27] M. Mendoza, C. Cobos, E. Leon, M. Lozano, F. Rodriguez, E. Herrera-Viedma, A new memetic algorithm for multi-document summarization based on CHC algorithm and greedy search, in: Mexican International Conference on Artificial Intelligence, Springer, 2014, pp. 125–138. doi:10.1007/978-3-319-13647-9_14.

[28] S. S. Benjumea, E. León, Genetic clustering algorithm for extractive text summarization, in: 2015 IEEE Symposium Series on Computational Intelligence, IEEE, 2015, pp. 949–956. doi:10.1109/SSCI.2015.139.

[29] K. Umam, F. W. Putro, G. Q. O. Pratamasunu, A. Z. Arifin, D. Purwitasari, Coverage, Diversity, and Coherence Optimization for Multi-Document Summarization, Jurnal Ilmu Komputer dan Informasi 8 (2015) 1–10.

[30] R. M. Alguliev, R. M. Aliguliyev, N. R. Isazade, An unsupervised approach to generating generic summaries of documents, Applied Soft Computing 34 (2015) 236–250.

[31] J.-P. Qiang, P. Chen, W. Ding, F. Xie, X. Wu, Multi-document summarization using closed patterns, Knowledge-Based Systems 99 (2016) 28–38.

[32] R. M. Alguliyev, R. M. Aliguliyev, N. R. Isazade, A. Abdi, N. Idris, COSUM: Text summarization based on clustering and optimization, Expert Systems 36 (2019) e12340: 1–17.

[33] P. Verma, H. Om, MCRMR: Maximum coverage and relevancy with minimal redundancy based multi-document summarization, Expert Systems with Applications 120 (2019) 43–56.

[34] H. H. Saleh, N. J. Kadhim, B. A. Attea, A genetic based optimization model for extractive multi-document text summarization, Iraqi Journal of Science 56 (2015) 1489–1498.

[35] G. Salton, C. Buckley, Term-weighting approaches in automatic text retrieval, Information Processing & Management 24 (1988) 513–523.

[36] D. Karaboga, B. Basturk, A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm, Journal of Global Optimization 39 (2007) 459–471.

[37] D. Karaboga, B. Gorkemli, C. Ozturk, N. Karaboga, A comprehensive survey: artificial bee colony (ABC) algorithm and applications, Artificial Intelligence Review 42 (2014) 21–57.

[38] K. Deb, Multi-objective evolutionary algorithms, in: Springer Handbook of Computational Intelligence, Springer, 2015, pp. 995–1015. doi:10.1007/978-3-662-43505-2_49.

[39] M. Toman, R. Tesar, K. Jezek, Influence of word normalization on text classification, Proceedings of InSciT 4 (2006) 354–358.

[40] ROUGE, Summary Evaluation Package, http://www.berouge.com/, 2017. Last accessed: 14-July-2017.

[41] M. Porter, The Porter Stemming Algorithm, http://www.tartarus.org/martin/PorterStemmer/, 2019. Last accessed: 18-February-2020.

[42] P. Willett, The Porter stemming algorithm: then and now, Program: electronic library and information systems 40 (2006) 219–223.

[43] A.-C. Zăvoianu, E. Lughofer, W. Koppelstätter, G. Weidenholzer, W. Amrhein, E. P. Klement, On the performance of master-slave parallelization methods for multi-objective evolutionary algorithms, in: International Conference on Artificial Intelligence and Soft Computing, Springer, 2013, pp. 122–134. doi:10.1007/978-3-642-38610-7_12.

[44] E.-G. Talbi, S. Mostaghim, T. Okabe, H. Ishibuchi, G. Rudolph, C. A. C. Coello, Parallel approaches for multiobjective optimization, in: Multiobjective Optimization, Springer, 2008, pp. 349–372. doi:10.1007/978-3-540-88908-3_13.

[45] B. Chapman, G. Jost, R. van der Pas, Using OpenMP: Portable Shared Memory Parallel Programming, volume 10, MIT press, 2008.

[46] NIST, Document Understanding Conferences, http://duc.nist.gov, 2014. Last accessed: 18-February-2020.

[47] C.-Y. Lin, ROUGE: A package for automatic evaluation of summaries, in: Text summarization branches out: Proceedings of the ACL-04 Workshop, volume 8, Association for Computational Linguistics, 2004, pp. 74–81.

[48] R Development Core Team, R: A Language and Environment for Statistical Computing. R Foundation for Statistical Computing. Vienna, Austria, 2008.

[49] F. Calle-Alonso, A. Cuenca-Guevara, D. de la Mata Lara, J. M. Sanchez-Gomez, M. A. Vega-Rodríguez, C. J. Perez Sanchez, NeuroK: A Collaborative e-Learning Platform based on Pedagogical Principles from Neuroscience, in: Proceedings of the 9th International Conference on Computer Supported Education (CSEDU 2017), volume 1, Science and Technology Publications, 2017, pp. 550–555.