Optimizing Data-Driven Models for Summarization as Parallel Tasks

Aleš Zamuda

Faculty of Electrical Engineering and Computer Science, University of Maribor
Koroška cesta 46, 2000 Maribor, Slovenia

Elena Lloret

Department of Software and Computing Systems, University of Alicante
Apdo. de correos 99, E-03080 Alicante, Spain

Abstract

This paper presents tackling of a hard optimization problem of computational linguistics, specifically automatic multi-document text summarization, using grid computing. The main challenge of multi-document summarization is to extract the most relevant and unique information effectively and efficiently from a set of topic-related documents, constrained to a specified length. In the Big Data/Text era, where the information increases exponentially, optimization becomes essential in selection of the most representative sentences for generating the best summaries. Therefore, a data-driven summarization model is proposed and optimized during a run of Differential Evolution (DE).

Different DE runs are distributed to a grid in parallel as optimization tasks, seeking high processing throughput despite the demanding complexity of the linguistic model, especially on longer multi-documents where DE improves results given more iterations. Namely, parallelization and the grid enable, running several independent DE runs at same time within fixed real-time budget. Such approach results in improving a Document Understanding Conference (DUC) benchmark recall metric over a previous setting.

Keywords: Text Summarization, Discrete Optimization, Distributed Computing, Data-Driven Model, Differential Evolution

Email addresses: ales.zamuda@um.si (Aleš Zamuda), elloret@dlsi.ua.es (Elena Lloret)

1. Introduction

In the era in which the information increases at an exponential rate, and it is impossible to digest it in an efficient and effective manner, the task of automatic text summarization can be of great help, both for humans and other computer-based tasks. On the one hand, providing summaries that condense all the relevant information for a specific topic instead of all the source documents to users, avoids them the costly task of having to read and determine which information should be kept [1]. On the other hand, using summaries can benefit other processes when it comes to managing large amounts of information at intermediate stages, thus decreasing the execution time, and consequently, providing the output faster, since, in this case, the text summarization process would act as a filtering step, keeping only the relevant information to process, and considering the remaining information as noisy information [2].

Formally, the task of text summarization can be defined as "the process of distilling the most important information from a source (or sources) to produce an abridged version for a particular user (or users) and task (or tasks)" [3]. This task poses great challenges to the research community, due partly to the inherent subjectivity associated with the process of determining "the most important" information. When and how can information be considered as relevant? This will very much depend on the person who will read the summary, or even the final goal of the summary itself, since the same piece of information could be considered relevant or irrelevant, depending on many factors. Different taxonomies have been proposed to classify the summaries according to different factors, such as its input, output, purpose, or language [4]. For instance, multi-document summarization specifically takes as input a number of documents — that could, potentially, be related to the same topic, or not — and has to produce an effective summary based on them. This type of input increases the complexity of the task, because it has to deal with redundancy, as well as the type of input documents —formal and informal text documents could also be input.

The usefulness of text summarization, either for users or for applications, also depends on the extent to which the summary can be obtained in real time, regardless of the number of input documents it has to cope with. The real time limitation to be considered may vary from one task specification

to another, and might even be different when planning the operation's scenario [5, 6]. Also, for real case scenarios, the optimum values might not be known beforehand, and running an optimization algorithm for a longer time setting would be useful [7, 5].

The aim of this paper is to elaborate on the possibility that the text summarization model can be run on a parallel computer grid, and the feedback obtained is applied for design of the text summarization approach. We motivate the fact that Natural Language Processing (NLP) tasks do not normally take care of the efficiency in terms of time. They just focus on solving the problem with high accurary, which is good, but not enough to transfer the High-Performance Computing (HPC) technology to it by improving the NLP approaches with HPC support. In this context, we emphasize the need to investigate methods/approaches that take into account accuracy, as well as efficiency. To investigate this issue would be the main contribution of our research work, since, to the best of our knowledge, existing publications only focus on either accuracy or efficiency at a time, thus, not analyzing how both aspects could be integrated appropriately into a summarization task to produce effective parallelized automatic summaries. But just to note, we have limited our approach to extractive summarization, i.e. our approach cannot be applied for abstractive summarization, and it is also not applicable for summarization with word length less than the shortest sentence, because it selects whole sentences to generate a summary.

Obtaining an appropriate balance between accuracy and efficiency would be possible by including parallel computing and using an HPC grid to run these summarization tasks. We investigate that, during working on a design of a summarization algorithm, one first conducts a preliminary research for showing that this is feasible. The knowledge obtained through the proposed research would allow the design and development of summarization approaches that can be both effective and accurate, so they can be used further in real contexts, being directly transferable to society. The main novelty in the newly proposed approach in this paper is, therefore, the parallelization of tasks for optimization that are applied in the perspective of benchmarking, i.e. the individual benchmark tests are parallelized in such a manner that summarization parallel tasks are submitted and distributed to a computer grid individually and merged after tasks are complete. Also, the information identification and quantitative evaluation is obtained using Freeling [8], coreference resolution, semantic analysis, and concept matrix that take place in the pre-processing stage for summarization (in the corpus

preprocessing phase for task construction, before the parallell task is submitted — much before the loop for summarization optimization commences). The computational perspective defining static (precomputed) and changing (computed during evaluation call) parts of fitness function are also newly discussed, as this perspective is important in high-performance computing scenarios; namely, the parts of fitness function that do not need to be recomputed, are precomputed. Regarding the optimization part novelty, a new algorithm combination is presented, a constrained version of a self-adaptive Differential Evolution (DE) with binarization. Additionally, an important computationally-intensive correlation analysis is, finally, presented empirically, providing insight into the correlation among ROUGE metric values on the dedicated benchmark (DUC 2002) along with the number of fitness functions, testing the trend and payoff of prolonged optimization execution runs (quality improvement after extended optimization run time).

In the following, Section 2 provides more related work, covering optimization and Differential Evolution, then text summarization and task parallelization on HPC. Section 3 presents our proposed contributions and defines novelty of the paper, defining the multi-document extractive summarization approach via DE that is run in parallel on HPC. Section 4 reports and discusses the results of the proposed approach. Section 5 provides conclusions and outlines future work.

2. Related Work

This section presents related work, as covered in the following subsections, in which, first, text summarization is explained, then optimization is addressed, followed by covering task parallelization and HPC.

2.1. Text Summarization

The task of text summarization has been researched for more than 50 years. Despite this, it is still a very challenging task in NLP [9]. As was outlined in the previous Section, its difficulty is due partly to the fact that there is a lot of subjectivity in the process that is influenced by many cognitive aspects [10, 11]. Whereas the main objective of text summarization is to produce a summary automatically, i.e., with no human intervention, such a summary could be output in many different forms, also being influenced by a wide range of factors that should be considered during the process of summary generation. In this sense, there are different classical taxonomies proposed in the literature that lead to different types of summaries [12, 13].

A special type of summarization is sentence extraction from multiple documents, i.e. multi-document extractive summarization. Different approaches can be found in the literature for addressing multi-document summarization, which range from simple approaches using statistical techniques, such as tf-idf [14], to more recent ones that use neural computing [15, 16, 17]. However, efficiency is not normally taken into account, and although summaries can obtain good results in terms of their content, the associated drawback concerns the impossibility to apply those approaches in realtime scenarios, especially those approaches based on Neural Networks (NNs) that require a lot of training time. One approach in-between is to take into account optimization issues and integrate them within the summarization approach. For instance, Alguliev et al. [18] applied Differential Evolution to multi-document summarization using sentence extraction, being formalized as a discrete optimization problem. Further on, Alguliev et al. also extended their work, modeling the multi-document summarization tasks as different algorithmic problems, such as a quadratic boolean programing problem; a non-linear programing problem; or as a modified p-median problem [19, 20, 21, 22, 23].

The summarization approach presented in this paper is based on the multi-document sentence extraction summarization approach of Alguliev et al. [18]. Therefore, in the next subsection, the basic text summarization approach [18] is covered over three subsubsections, followed by a subsubsection on the DUC 2002 multi-document summarization set, and an explanation of the assessment through ROUGE peer evaluation.

2.1.1. Multi-Document Summary: Text Indexing

This subsubsection presents how a summarization task is defined mathematically, based on the approach from [18]. Addressing it from an optimization perspective, summarization is optimization of a summarization text model (fitness function), constrained to word length. From the perspective of optimization problems, this model does not have a more general problem description, as it is a special case of a very complex feature selection with constraints, like, e.g. feature preselection for stability selection in Data Mining of Big Data from healthcare inpatients records [24].

To define the features that can be selected during Feature Selection (FS) for text summarization, the approach in [18] extracts a definition of the text summarization model from a given dataset. This dataset is a collection of documents, which are converted to a text representation and concatenated, composed of e.g. UTF-8 encoded character bytes. The input text is split into

sentences, so that if the sentences are extracted in the resulting summary, the summary can be printed by recomposing and concatenating the text written in these sentences. The sentences in the input text are indexed, as well as the words that these sentences contain, as explained in the following.

Within one text D of a multi-document set from the whole benchmark set, the sentences in all contained documents are indexed. A statement consists of words (terms), words consist of letters. Within the text, beginnings of statements are marked as text character indices $D = \{s_1, s_2, s_3, ..., n\}$, and a dictionary of different words (terms) and their appearences in the text is composed of $W = \{w_1, w_2, w_3, ...\}$. A symbol table (e.g. implemented using hash table) is then used to compare terms and build the dictionary.

2.1.2. Multi-Document Summary: Term Weights' Computation

Term weights' computation contains three steps. In the first step, for each i-th term (w_i), during indexing, the number of occurrences in the text is gathered, and the number of occurrences (n_k) of a term in some k-th statement. Then, in the second step, for each term w_i in the document, an inverse frequency of each term is calculated:

$$isf_{w_i} = \log(\frac{n}{n_k}), \tag{1}$$

where n denotes the number of statements in the document, and n_k the number of statements including a term w_i. In the third step, for each term in the document, a weight is calculated:

$$w_{i,k} = tf_{i,k} isf_k, \tag{2}$$

where $tf_{i,k}$ is the number of occurrences (term frequency) of a term w_k in a statement s_i.

2.1.3. Multi-Document Summary: Summary Model Fitness Evaluation

Based on the weights, the similarity between two selected statements $s_i = [w_{i,1}, w_{i,2}, ..., w_{i,m}]$ and $s_j = [w_{j,1}, w_{j,2}, ..., w_{j,m}]$ is computed as in [18]:

$$sim(s_i, s_j) = \sum_{k=1}^{m} \frac{w_{i,k} w_{j,k}}{\sqrt{\sum_{k=1}^{m} w_{i,k} w_{i,k} \sum_{k=1}^{m} w_{j,k} w_{j,k}}}, \tag{3}$$

where $w_{i,k}$ is a term weight defined in the previous subsubsection, and m is the number of all terms in the summarized text.

A sentence set $\mathbf{x}$ is evaluated as a 0/1 knapsack-like problem, the content of which is to be maximized. To include a selection of statements in the extractive summary, an i-th sentence s_i is selected ($x_i = 1$), or unselected ($x_i = 0$). The selected statements from the sentence combination $\mathbf{x}$ are printed in order as they appear in the text, to recompose an extractive summary text. To define a sentence selection $\mathbf{x}$, needed to provide the selected text summaries to be extracted from a multi-document set $\mathbf{D}$ with N sentences, a set of sentences $\mathbf{x}$ needs to be selected:

$$\mathbf{x} = \{x_j\}, \forall j \in \{1, 2, ..., 1, N\}, \forall x_j \in \{0, 1\}. \tag{4}$$

For a sentence selection $\mathbf{x}$, content coverage $V(\mathbf{x})$ is computed as a cross-sentence multiplied double sum of similarities [18]:

$$V(\mathbf{x}) = \sum_{i=1}^{N-1} \sum_{j=i+1}^{N} (sim(s_i, O) + sim(s_j, O))x_{i,j}, \tag{5}$$

where $x_{i,j}$ denotes the inclusion of both statements, s_i, and s_j, and $x_{i,j}$ is only 1 if $x_i = x_j = 1$, otherwise 0. The $O = (o_1, o_2, ..., o_M)$ represents an average term vector, which is an auxiliary weight used for all different indexed M terms $i = \{1, 2, ..., M\}$ based on the corresponding term weights $w_{i,k}$:

$$o_i = \frac{\sum_{j=1}^{N} w_{i,j}}{N}. \tag{6}$$

For a sentence combination $\mathbf{x}$, redundance $R(\mathbf{x})$ is then calculated as a similarity sum among all statements, similar to content coverage $V(\mathbf{x})$ by reusing the $sim(s_i, s_j)$ calculations [18]:

$$R(\mathbf{x}) = \sum_{i=1}^{N-1} \sum_{j=i+1}^{N} sim(s_i, s_j)x_{i,j}, \tag{7}$$

where $x_{i,j}$ denotes the inclusion of both statements, s_i, and s_j, and again, $x_{i,j}$ is only 1 if $x_i = x_j = 1$, otherwise 0.

Finally, the fitness of a generated text model represents the ratio between content coverage ($V(\mathbf{x})$) and redundancy ($R(\mathbf{x})$), defined using negation as a minimization function [18]:

$$f(\mathbf{x}) = -\frac{V(\mathbf{x})}{R(\mathbf{x})}. \tag{8}$$

As the constraint of the summary length is $L \pm \epsilon_w$ words, therefore, a constraint handling mechanism also needs to be used to address this during optimization.

The meaning and evaluation of a recomposed summary, hence, does not have the same features as a 0/1 knapsack problem with a fixed knapsack capacity and a linear relationship between the weights and profit values of unsorted items, as addressed for standard strongly correlated sets of unsorted data and efficiently solved approximately using NNs, like recently in [25]. These features are more complex and the weights (of sentences) are dynamic, according to which sentences are chosen (coverage and redundancy change without preservation of linear relationships among selected sentences due to distance non-preserving multiplications with term weights, see Eqs. (5) and (7)). Also, a summarization fitness function as modeled here does not merely change the dynamic capacity of a knapsack [26], but changes the values of knapsack items added up together at higher powers than linearly. The problem selected as the case study in this paper is not only an important challenge within NLP due to the real and big data processing, but also, moreover, technically, as a large scale, non-linear, constrained, and non-separable problem in the benchmarking domain [27].

2.1.4. Multi-document Summarization Sets and Peer Evaluation

The summarization task addressed in this research focus on the scenario of news digest. Everyday, a high number of news are published, and it is impossible to read them all and keep up-to-date. The use of summaries to help digest them fits perfectly in this context.

The existing collection of English newswire documents from the Document Understanding Conferences[1] (DUC) fits very well for experimenting within the aforementioned scenario. On the one hand, it provides pairs of documents-summary or cluster-summary for diverse summarization types (extractive, single-document, multi-document, etc.), and, on the other hand, there is a wide number of previous summarization systems to be compared with, and determine to what extent our approach is effective. In particular, we used the DUC 2002 dataset, which contains 567 documents grouped in 59 clusters (denoted as `d061` to `d120`), where each cluster represents a set of topic-related documents (the average number of documents per cluster is

[1] https://duc.nist.gov/

10). Besides this dataset, a recent new dataset, i.e., CNN/DailyMail[2] has also been made available for the research community [28]. This dataset contains more than 300,000 documents, and it provides summaries of about 50 words. However, this dataset can only be used for single-document summarization, which is not the type of summaries we are addressing in this research.

Concerning the evaluation of summaries, ROUGE [29] is one of the common standard, and most used tools. The idea behind ROUGE is that, if two texts have a similar meaning, they must also share similar words or phrases. As a consequence, it relies on n-gram co-occurrence, and the idea behind it is to compare the content of a peer summary with one or more model summaries, and compute the number of n-gram of words they all have in common. Different types of n-grams can be obtained, such as unigrams (ROUGE-1), bigrams (ROUGE-2), the longest common subsequence (ROUGE-L), or bigrams with a maximum distance of four words in-between (ROUGE-SU4), and, based on them, values for recall, precision and F-measure can be obtained, thus determining the summary accuracy in terms of content (the higher recall, precision and F-measure values, better). Among all the metrices, recall values are then usually reported for peer evaluation of generated model summaries. Other metrices also exist, like AutoSummENG [30] which is an automatic character n-gram based evaluation method with high correlation with human judgments, or SumTriver [31], which does not need to have human summaries for the evaluation; however, they are not often used by the research community, thus it is difficult to find results for comparison purposes. Other ways to evaluate the summaries would be to consider standard similarity metrices, such as Simmetrics[3]. A comprehensive literature review on summarization evaluation methods and metrices can be found in [32].

2.2. Optimization and Differential Evolution

Differential Evolution (DE) [33] is a floating-point encoding Evolutionary Algorithm [34, 35] for global optimization over continuous spaces. The DE algorithm was introduced in 1995 by Storn and Price [33] and, since then, has formed the basis for a set of successful algorithms for optimization domains, such as continuous, discrete, mixed-integer, or other search spaces and features [36]. The whole encompassing research field around

[2]https://github.com/abisee/cnn-dailymail

[3]https://github.com/Simmetrics/simmetrics

DE was surveyed most recently in [37], and even since then, several other domain- and feature-specific surveys, studies, and comparisons have also followed [38, 39, 40, 41]. Theoretical insight and insights to inner workings and behaviors of DE during consecutive generations has been studied in works like [42, 43, 44, 45, 46, 47, 48].

The main performance advantages of DE over other Evolutionary Algorithms [49, 50, 51, 52, 53, 54] lie in floating-point encoding, and a good combination of evolutionary operators, mutation step size adaptation, and elitist selection. The DE algorithm has a main evolution loop in which a population of vectors is computed for each generation of the evolution loop. During one generation g, for each vector $\mathbf{x}_i$, $\forall i \in \{1, 2, ..., NP\}$ in the current population, DE employs evolutionary operators, namely mutation, crossover, and selection, to produce a trial vector (offspring) and to select one of the vectors with the best fitness value. NP denotes population size and $g \in \{1, 2, ..., G\}$, the current generation number.

Mutation creates a mutant vector $\mathbf{v}_{i,g+1}$ for each corresponding population vector. Among many proposed, one of the most popular DE mutation strategies [55, 33] are the *'rand/1'*:

$$\mathbf{v}_{i,g+1} = \mathbf{x}_{r_1,g} + F(\mathbf{x}_{r_2,g} - \mathbf{x}_{r_3,g}) \tag{9}$$

and the *'best/1'*:

$$\mathbf{v}_{i,g+1} = \mathbf{x}_{best,g} + F(\mathbf{x}_{r_1,g} - \mathbf{x}_{r_2,g}), \tag{10}$$

where the indexes r_1, r_2, and r_3 represent the random and mutually different integers generated within the range $\{1, 2, ..., NP\}$ and are also different from index i. The $\mathbf{x}_{best,g}$ denotes the currently best vector. F is an amplification factor of the difference vector within the range $[0, 2]$, but usually less than 1. The vector at index r_1 is a base vector. The term $\mathbf{x}_{r_2,g} - \mathbf{x}_{r_3,G}$ denotes a difference vector which after multiplication with F, is named an amplified difference vector. The simple DE mutation 'rand/1' is by far most the widely used [56], however, a form of 'best/1' mutation has also been signified beneficial, especially in more restrictive evaluation scenarios [57, 58, 45, 59, 6].

After mutation, the mutant vector $\mathbf{v}_{i,g+1}$ is taken into a recombination process with the target vector $\mathbf{x}_{i,g}$ to create a trial vector $\mathbf{u}_{i,g+1} = \{u_{i,1,g+1}, u_{i,2,g+1}, ..., u_{i,D,g+1}\}$. The binary crossover operates as follows:

$$u_{i,j,g+1} = \begin{cases} v_{i,j,g+1} & \text{if } rand(0,1) \leq CR \text{ or } j = j_{rand} \\ x_{i,j,g} & \text{otherwise} \end{cases}, \tag{11}$$

where $j \in \{1, 2, ..., D\}$ denotes the j-th search parameter of D-dimensional search space, $rand(0, 1) \in [0, 1]$ denotes a uniformly distributed random number, and j_{rand} denotes a uniform randomly chosen index of the search parameter, which is always exchanged to prevent cloning of target vectors. CR denotes the crossover rate [60]. Finally, the selection operator propagates the fittest individual in the new generation (for a minimization problem):

$$\mathbf{x}_{i,g+1} = \begin{cases} \mathbf{u}_{i,g+1} & \text{if } f(\mathbf{u}_{i,g+1}) < f(\mathbf{x}_{i,g}) \\ \mathbf{x}_{i,g} & \text{otherwise} \end{cases}. \tag{12}$$

2.2.1. Self-adaptation of Control Parameters

As studied in [49, 45], the original DE algorithm [33] keeps all three control parameters fixed during the optimization process. As observed from the initial experiments in [49], the necessity was confirmed for changing control parameters during the optimization process. In the jDE algorithm [49], which extends the original DE algorithm, a self-adaptive control mechanism is introduced to change the control parameters F and CR during the evolutionary process after their initial setting at $F = 0.5$ and $CR = 0.9$. New control parameters $F_{i,G+1}$ and $CR_{i,G+1}$ are calculated for jDE as in [49]:

$$F_{i,G+1} = \begin{cases} F_{\mathrm{l}} + rand_1 \times F_{\mathrm{u}} & \text{if } rand_2 < \tau_{\mathrm{F}} \\ F_{i,G} & \text{otherwise} \end{cases} \tag{13}$$

$$CR_{i,G+1} = \begin{cases} rand_3 & \text{if } rand_4 < \tau_{\mathrm{CR}} \\ CR_{i,G} & \text{otherwise.} \end{cases} \tag{14}$$

This produces control parameters F and CR within a new vector. The $rand_j \in [0, 1]$, $j \in \{1, 2, 3, 4\}$ are uniform random values. The τ_{F} and τ_{CR} (generally denoted in literature as τ_1 and τ_2, respectively) represent the probabilities of adjusting control parameters F and CR, respectively. The τ_{F}, τ_{CR}, F_{l}, F_{u} are taken as fixed values, as proposed in [49], but in [45], a thorough study was conducted on a full benchmark using HPC, showing that adjusting these values can have significant improvement effects.

2.2.2. Constraints' Definition and Handling

As explained and compared experimentally in [59], several methods have been proposed for defining constraints, and addressing them in Genetic Algorithms for parameter optimization problems [61]. A summary of constraint

handling techniques can be found in [62, 63, 64], which also contain information on many stochastic techniques. An application of the constraint definition is to use it to guide the search towards feasible areas of the search space. Given constraints, there are different methods for consideration when defining feasibility and possible violation estimation of a solution, varying with the ease of defining such methods on the one hand, and guiding feedback amount suitability for constraint violation handling on the other [59].

A solution $\mathbf{x}$ is regarded as *feasible* with regard to inequality constraints $g_i(\mathbf{x}) \leq \mathbf{0}$ and equality constraints $h_j(\mathbf{x}) = \mathbf{0}$ if

$$g_i(\mathbf{x}) \leq 0, \quad i = \quad 1, 2, ..., q, \tag{15}$$

$$|h_j(\mathbf{x})| - \epsilon \leq 0, \quad j = \quad q+1, ..., m, \tag{16}$$

where equality constraints are transformed into inequalities. The mean value of all constraints' violations $\overline{\nu}$ is defined as:

$$\overline{\nu} = \frac{(\sum_{i=1}^{q} G_i(\mathbf{x})) + (\sum_{j=q+1}^{m} H_j(\mathbf{x}))}{m}, \tag{17}$$

where the sum of all constraint violations is zero for feasible solutions, and positive when at least one constraint is violated:

$$G_i(\mathbf{x}) = \begin{cases} g_i(\mathbf{x}), & g_i(\mathbf{x}) > 0, \\ 0, & g_i(\mathbf{x}) \leq 0, \end{cases} \tag{18}$$

$$H_j(\mathbf{x}) = \begin{cases} |h_j(\mathbf{x})|, & |h_j(\mathbf{x})| - \epsilon > 0, \\ 0, & |h_j(\mathbf{x})| - \epsilon \leq 0. \end{cases} \tag{19}$$

The ϵ-jDE algorithm [65] follows the jDE-2 algorithm [66], and emphasizes constraints as follows. It compares two solutions, say i and j, during the selection operation:

$$\mathbf{x}_{i,g+1} = \begin{cases} \mathbf{x}_{j,g} & \text{if } (\overline{\nu}_{i,g} > \overline{\nu}_{j,g}), \\ \mathbf{x}_{j,g} & \text{else if } (\overline{\nu}_{j,g} = 0) \wedge (f(\mathbf{x}_{i,g}) > f(\mathbf{x}_{j,g})), \\ \mathbf{x}_{i,g} & \text{otherwise.} \end{cases} \tag{20}$$

The algorithm distinguishes between feasible ($\overline{\nu} = 0$) and infeasible individuals: Any feasible solution being better than any infeasible one. Namely, the jDE-2 algorithm had difficulties when solving constrained optimization problems with equality constraints. Since Takahama and Sakai in [67] pointed out

that, for problems with equality constraints, the ϵ level should be controlled properly in order to obtain high quality solutions, the ϵ-jDE algorithm, therefore, uses ϵ level controlling, where the ϵ level constraint violation precedes the Objective Function when the aggregated constraints' value is small. The ϵ-jDE also uses an adaptive and self-adaptive control of ϵ level, but thereby introduces some additional control parameters. The ϵ level is updated until the number of generations g reaches the control generation g_c. After the number of generations exceeds g_c, the ϵ level is set at 0 to obtain solutions with minimum constraint violations [65].

2.3. Task Parallelization and High-Performance Computing

High-Performance Computing (HPC) architectures involve the use of many interconnected processing elements to reduce the time to solution of a given problem, such as processing of huge amounts of Big Data, as defined in [68]. As presented in [68], many powerful HPC systems are heterogeneous, in the sense that they combine general-purpose CPUs with *accelerators* such as, Graphics Processing Units (GPUs), or Field Programable Gate Arrays (FPGAs) [69]. Several HPC approaches exist [70, 71, 72, 73] developed to improve the performance of advanced and data intensive modeling and simulation applications. The parallel computing paradigm may be used on multi-core CPUs, many-core processing units (such as, GPUs [74]), re-configurable hardware platforms (such as FPGAs), or over distributed infrastructure (such as, cluster, Grid, or Cloud). To access the HPC, tools like ARC from ATLAS [75] can be utilized, to submit workload tasks as, e.g. Bash scripts and other executable and data files. Regarding widely used parallel programing frameworks [76] for heterogeneous systems, these include OpenACC [77], OpenCL [78], OpenMP [79], and NVIDIA CUDA [80]. OpenMP is a set of compiler directives, library routines, and environment variables for programing shared-memory parallel computing systems. Furthermore, OpenMP has been extended to support programing of heterogeneous systems that contain CPUs and accelerators. OpenCL supports portable programing of hardware provided by various vendors, while CUDA runs only on NVIDIA hardware. CUDA C/C++ compiler, libraries, and run-time software enable programers to develop and accelerate data-intensive applications on a GPU. With regard to distributed parallel computing, the available frameworks include the Message Passing Interface (MPI) [81], MapReduce/Hadoop [82] or Apache Spark [83]. For Big Data processing, an important platform is also Data Flow [84], specifically, Maxeller platform [85].

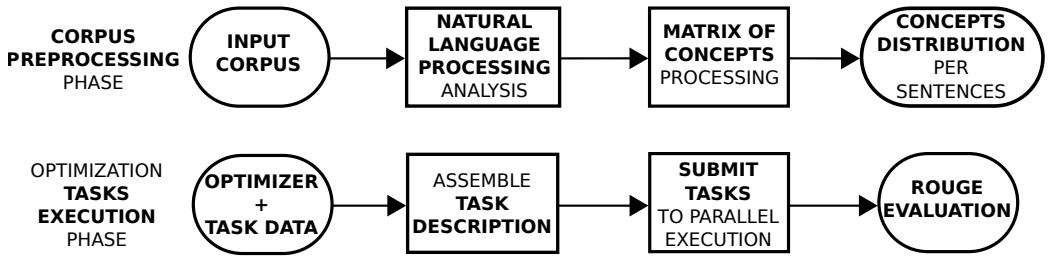

Figure 1: Architecture of the Multi-document Extractive Summarization Approach. Tasks are prepared in the first phase, then executed as the second phase in parallel.

3. Multi-document Extractive Summarization Approach via Differential Evolution and Sentence Optimization

The architecture, with phases, is provided in Figure 1. The two distinct phases are preprocessing of the corpus and execution of optimization tasks. The first phase needs to be completed before submitting a task in the next phase.

The proposed multi-document extractive summarization approach has been designed and developed following the standard stages for the automatic summarization process described in [86]: i) Information identification; ii) Information interpretation; and iii) Summary generation. The first stage consists of determining the particular subject the document is about. It is usually approached by assigning each unit (words, sentences, phrases, etc.) a score which is indicative of its importance. In the end, the top score units are extracted up to a desired length. This is far from trivial, since NLP techniques are necessary to understand the meaning of the documents' content. In the information interpretation stage, information in the form of topics identified as important are fused, represented in new terms, and expressed using a new formulation, which includes concepts or words not found in the original text. This stage is what distinguishes extractive from abstractive summarization. Finally, the last stage only makes sense if abstractive summaries are generated. In these cases, natural language generation techniques (text planning, sentence planning, and sentence realization) are needed to produce the final text of the summary.

Since, in this research work, we are focusing on an extractive summarization approach, i.e., the most important sentences would be extracted without any modification in vocabulary or format, the last two stages will not be necessary. Despite this, the extractive task is far from trivial, and it

still requires exploring novel methods and techniques to address it properly, as far as the results obtained by the generated summaries is concerned, as well as the processing time required to generate it, especially when dealing with a high number of input documents. The main novelty in the newly proposed approach in this paper is, therefore, the parallelization of tasks for optimization that are applied in the perspective of benchmarking, i.e. the individual benchmark tests are parallelized in such manner that summarization parallel tasks are submitted individually to a computer grid and merged after tasks are complete. Also, the information identification and quantitative evaluation using Freeling [8], coreference resolution, semantic analysis, and concept matrix in the pre-processing stage for summarization before the task execution for summary optimization (that loops DE generations, etc.) commences. The computational perspective defining static (precomputed) and changing (computed during evaluation call) parts of fitness function are also discussed, as this perspective is important in high-performance computing scenarios; namely, the parts of fitness function that do not need to be recomputed are precomputed. Regarding the optimization part novelty, a new algorithm combination is presented, a constrained version of a self-adaptive DE with binarization. In the following subsections, we explain in detail how the process has been designed, by first elaborating on the information identification and information interpretation and then presenting the distribution of parallel tasks to a grid.

3.1. Information identification and Quantitative evaluation

In this stage, a linguistic analysis process is applied to the input. Using NLP tools, such as Freeling linguistic analyzer [8], the linguistic process consists of performing sentence splitting, tokenization, semantic analysis and coreference resolution. Sentence splitting aims at delimiting when a sentence starts and ends. Tokenization identifies the beginning and end of an element (token) in a specific language. A token could be a word, but also a punctuation mark. Semantic analysis will allow us to obtain the specific meaning of a word within its context, together with its semantic relations, such as synonymy, hypernymy, meronymy, etc. Coreference resolution, and, specifically, pronominal anaphora resolution will allow us to determine whether a pronoun and a word are referring to the same item within the text. Once this process is finished, the text will be represented, associating each of its words to its corresponding linguistic meaning.

Although the linguistic process may resemble the compilation process

of a programing language [87], specially the tokenization part included in the lexical analysis of a compiler, understanding and interpreting natural language automatically gets even harder, since contrary to programing languages, natural languages include ambiguity, so a deeper linguistic knowledge and appropriate linguistic resources have to be used, such a Freeling [8].

3.1.1. Sentence Splitting and Tokenization

Sentence and word text processing is different between ROUGE and Freeling. Therefore, we did a couple of experiments with different word whitespace configurations. We got the same results with texts containing punctuation marks as with those texts from the same source that did not contain any punctuation marks. On the contrary, we observed that spaces do influence the ROUGE results. Therefore, since the difference in some punctuation marks with the Freeling recomposed texts compared to the SPL (i.e., the format the source documents have), the following processing was used to create SPL from Freeling recomposed text (deleting spaces infront "."", "","", ")", "(", and splitting multi-word phrases "_") using command `sed -e 's/ \././g' -e 's/ ,/,/g'-e 's/_/ /g' -e 's/( /(/g' -e 's/ )/)/g'`.

3.1.2. Semantic Analysis and Coreference Resolution

Before applying an evolutionary algorithm, we carried out a linguistic process over the DUC datasets in order that semantic understanding of texts could also be obtained. In addition, input documents were cleansed by removing XML tags, and a sentence segmentation step was then applied to detect sentence boundaries. Further on, documents were passed through a coreference resolution system (i.e. JavaRap[4]), so that pronoun references were replaced by their correct antecedent in the document. Once the references were removed from documents, a semantic analysis process was carried out. This is the most relevant step in the proposed linguistic process, since it involved the detection of concepts semantically-related in the documents. WordNet 3.0[5] was employed for this semantic analysis. WordNet is a lexico-semantic English resource that groups content words (i.e., nouns, verbs, adjectives and adverbs) into sets of synonyms called synsets, providing information about the semantic relationships between them. In our process,

[4]http://ilo.nus.edu.sg/for-industry/online-software-licences/g-lonline-software-licences/java-rap-resolution-of-anaphora-program/

[5]https://wordnet.princeton.edu/

we used such information to detect sets of synonyms in the documents, so in this manner, we could work with concepts instead of literal terms. For example, *detonation* and *explosion* are different words but their WordNet's synsets are the same (*07323181*), so they belong to the same concept. Specifically, WordNet was used through the Freeling analyzer[6], which, among other linguistic information, also provides the WordNet synset corresponding to a word. The advantage of using WordNet through Freeling tool, is that we can configure the word disambiguation method used for determining the meaning of a word in a particular context (this is especially useful in dealing with ambiguity). We set this option to the most frequent sense, that corresponds to the first WordNet synset of a term, but also taking also into consideration its part-of-speech (i.e., the grammatical class of a term). This guarantees that the meaning of that word in the document is its most probable meaning. In this manner, if two words have the same first synset, it will be considered that they are synonyms and they will be represented under the same concept.

It is worth mentioning that more sophisticated disambiguation methods could have been employed; however, they are not matured enough yet [88], and, therefore, they could introduce errors that would be propagated to further stages of the approach, and, therefore, be detrimental for the whole approach.

3.1.3. Concept Matrix

The information from the previous stages is used to build a concept Matrix (M), where the rows represent the concepts[7] of the document —extracted through the previous linguistic analysis— and the columns represent the sentences of the document/s. Each cell $M_{i,j}$ contains the frequency of appearance of each concept. For those cases where the concept is not included in the sentence, a 0 is assigned to the cell. Once the Matrix has been filled in, the next stage of the process is to determine which concepts are the most relevant ones in order to extract the sentences with the highest relevance.

Specifically, we used concept matrix and stopword omitting for information identification in multi-document summarization on DUC2002 (using a max. 200 words per summary). The sentences were split using Freeling, and, therefore differed from the SPL sentences provided in the source documents;

[6]http://nlp.lsi.upc.edu/freeling/

[7]Concepts provide a higher level of abstraction compared to words, since they can group together sets of synonyms

however, they could be used in the ROUGE, since the human models are free text and different from the provided SPL input. The wordcount and number of sentences for the optimizer were, therefore, reconstructed from the Freeling sentence analysis.

The words were classified into concepts (concept matrix) using the Freeling tool, and statistics were gathered on how many times some concepts appeared in a certain sentence. The Freeling was also used for sentence splitting, not only for word splitting and word classification. Also, no punctuation signs were treated as concepts, and, therefore, were omitted in the concept-matrix. The punctuation marks were not considered as stopwords (i.e. to be omitted), but they do not affect the concept matrix computations, because WordNet was used, which only identifies as concepts those words that are nouns, verbs, adjectives, or adverbs.

3.1.4. Obtained Identified Multi-Document Abstractions to be Optimized

Coherent with the definitions of the above subsubsections in this subsection, we then obtained the following number of sentences (i.e. features to select from), total number of words in the multi-documents for each DUC benchmark. The number of sencences and terms for DUC 2002 is listed in Table 1. In SPL-based input from DUC (using the original sentence splitting tool), 22,114 sentences are given. In Freeling format, 16,878 sentences were outputted for the DUC 2002. This gives a 23,68% difference in number of reduced number of sentences when using the Freeling. Namely, using Freeling sentence splitting approach, we used it for input, as well as for the reconstruction of terms into sentences, to match the correct number of words in sentences that were used in the input to the Concepts matrix that used WordNet term classification.

3.2. Information interpretation: Sentence Selection Optimization

The phasing of a summarization task using the multi-document sets is as follows (see algorithm pseudocode in Fig. 2):

1. All single multi-document sentences are packed together in one set to be summarized,
2. Each of these sets is sent as input to the summarization algorithm 20 times independently (i.e. for 20 independent runs),
3. In a summarization optimization run, each sentence in the set is received as indexed, and also each term in a sentence, then, for each term, a term dictionary and weight are built, and for sentences $i =$

Figure 2: Algorithm Constraint-adjusting Binary Self-adaptive Differential Evolution for Data-Driven Models Extractive Text Summarization Optimization (CaBiSDETS)

algorithm CaBiSDETS(RNi, MDOC, L, $GMAX$, NP, F_l, F_u, $\tau_\mathrm{F}, \tau_\mathrm{CR}$)

Require: variable parameters — RNi (current random seed number, run), MDOC (multi-document input to be summarized, already prepared by Freeling decomposition of text to sentences and terms), L (summary target length); DE constants — $GMAX$ (number of generations to run DE), NP (DE population size), F_l, F_u, (bounds of F and CR control-parameters' self-adaptation); $\tau_\mathrm{F}, \tau_\mathrm{CR}$ (frequencies of F and CR control-parameters' self-adaptation).

Ensure: $\mathbf{x}$ – list of sentence indexes (D binary variables setting on/off switch for each of N sentences of a MDOC) for an extractive summary.

Read the input files $MDOC$ previously processed by Freeling, precompute the ISF and weights for all M terms and pan-sentence similarities between all N sentences:

$\forall n \in \{1,2,...,N\}$, $\forall m \in \{1,2,...,M\}$: $isf_m = \log(\frac{N}{|\{\forall n : m \in s_n\}|})$;

$\forall n \in \{1,2,...,N\}$, $\forall m \in \{1,2,...,M\}$: $w_{n,m} = tf_{n,m} isf_m$;

$\forall n \in \{1,2,...,N\}$: $o_n = \frac{\sum_{j=1}^{n} w_{i,j}}{n}$;

$\forall k, \forall l \in \{1,2,...,N\}$: $sim(s_k, s_l) = \sum_{m=1}^{M} \frac{w_{k,m} w_{l,m}}{\sqrt{\sum_{m=1}^{M} w_{k,m} w_{k,m} \sum_{m=1}^{M} w_{l,m} w_{l,m}}}$;

initialize random number generator function $\mathcal{U}$ seed to RNi; set dimension D to N;

uniform randomly generate DE initial population $\mathbf{x}_{i,0} := \mathcal{U}[-5,5]$, $\forall i \in \{1,2,...,NP\}$;

evaluate initial population and set constraint level ϵ at $\overline{\nu}$ of 30% least fit DE vector;

for DE generation loop $g = 1$ to $g = 10000$ **do**

  for DE iteration loop i (all individuals $\mathbf{x}_{i,g}$ in a population) **do**

    DE new trial vector $\mathbf{x}_{i,g}$ computation:

      choose mutually and from i distinct indices $r_1, r_2, r_3 \in \mathcal{U}\{1,2,...,NP\}$;

      $F_{i,g+1} = \begin{cases} F_\mathrm{l} + \mathcal{U}[0,1] \times F_\mathrm{u} & \text{if } \mathcal{U}[0,1] < \tau_\mathrm{F} \\ F_{i,g} & \text{otherwise} \end{cases}$;

      $CR_{i,g+1} = \begin{cases} \mathcal{U}[0,1] & \text{if } \mathcal{U}[0,1] < \tau_\mathrm{CR} \\ CR_{i,g} & \text{otherwise} \end{cases}$;

      $\mathbf{v}_{i,g+1} = \mathbf{x}_{r_1,g} + F_{i,g+1}(\mathbf{x}_{r_2,g} - \mathbf{x}_{r_3,g})$

      $\forall j \in \{1,..,D\}$: $u_{i,j,g+1} = \begin{cases} v_{i,j,g+1} & \text{if } \mathcal{U}[0,1] \le CR_{i,g+1} \text{ or } j = j_{rand} \\ x_{i,j,g} & \text{otherwise} \end{cases}$;

    DE fitness evaluation (summary model calculation) for i-th vector $\mathbf{u}_{i,g+1}$:

      $\mathbf{u}_{i,g+1}$ as $\mathbf{x} := \{x_1, x_2, ..., x_D\}$, $\forall j : x_j := \mathcal{U}[0,1] < |(\frac{2}{\pi} \arctan\left(\frac{\pi}{2} u_{i,j,g+1}\right)|$;

      $V(\mathbf{x}) = \sum_{k=1}^{N-1} \sum_{l=k+1}^{N} (sim(s_k, O) + sim(s_l, O) x_k x_l)$;

      $R(\mathbf{x}) = \sum_{k=1}^{N-1} \sum_{l=k+1}^{N} (sim(s_k, s_l) x_k x_l)$;

      $f(\mathbf{u}_{i,g+1}) = f(\mathbf{x}) = -\frac{V(\mathbf{x})}{R(\mathbf{x})}$;

      $\mathbf{x}_{i,g+1} = \begin{cases} \mathbf{u}_{i,g+1} & \text{if } \overline{\nu}_{u,g} < \epsilon \text{ and } \overline{\nu}_{i,g} < \epsilon \text{ and } f(\mathbf{u}_{i,g+1}) < f(\mathbf{x}_{i,g}) \\ \mathbf{x}_{i,g} & \text{if } \overline{\nu}_{u,g} < \epsilon \text{ and } \overline{\nu}_{i,g} < \epsilon \text{ and } f(\mathbf{u}_{i,g+1}) \ge f(\mathbf{x}_{i,g}) \\ \mathbf{u}_{i,g+1} & \text{if } (\overline{\nu}_{u,g} \ge \epsilon \text{ or } \overline{\nu}_{i,g} \ge \epsilon) \text{ and } \overline{\nu}_{u,g} < \overline{\nu}_{i,g} \\ \mathbf{x}_{i,g} & \text{if } (\overline{\nu}_{u,g} \ge \epsilon \text{ or } \overline{\nu}_{i,g} \ge \epsilon) \text{ and } \overline{\nu}_{u,g} > \overline{\nu}_{i,g} \\ \mathbf{u}_{i,g+1} & \text{if } \overline{\nu}_{u,g} \ge \epsilon \text{ and } \overline{\nu}_{i,g} = \overline{\nu}_{u,g} \text{ and } f(\mathbf{u}_{i,g+1}) \le f(\mathbf{x}_{i,g}) \\ \mathbf{x}_{i,g} & \text{if } \overline{\nu}_{u,g} \ge \epsilon \text{ and } \overline{\nu}_{i,g} = \overline{\nu}_{u,g} \text{ and } f(\mathbf{u}_{i,g+1}) > f(\mathbf{x}_{i,g}) \end{cases}$;

  end for

  update constraint level ϵ to 0 if $g > 0.2GMAX$, else to population's 30%-worst $\overline{\nu}$;

end for

return the best individual obtained among $\mathbf{x}_{i,g}$;

Table 1: DUC 2002 multi-document (MDOC), number of terms per one MDOC (Freeling wordcount). SntD stands for number of sentences in an MDOC, as used for input to the summarization, as given from the DUC repository for DUC2002 (`docs.with.sentence.breaks`, counting number of lines containing `"<s docid="`). SntF stands for the number of sentences in an MDOC, as used for input to the summarization, as given from the Freeling decomposition of each MDOC. SntR stands for the number of sentences in an MDOC, as used for input to the ROUGE evaluation, based on the recomposition of the Freeling-decomposed sentences. After that column, for the latter case, the number of terms in total follow, for a certaint MDOC.

MDOC	#SntD	#SntF	#SntR	#Terms	MDOC	#SntD	#SntF	#SntR	#Terms
d061j	238	202	202	4039	d092c	375	227	227	5026
d062j	158	118	118	2998	d093c	343	211	211	4350
d063j	319	261	261	5449	d094c	253	201	201	4740
d064j	254	192	192	4759	d095c	320	244	244	5278
d065j	372	328	328	6557	d096c	353	212	212	4478
d066j	250	200	200	4650	d097e	316	229	229	5323
d067f	168	121	121	3170	d098e	579	498	498	10245
d068f	182	134	134	2749	d099e	540	420	420	8416
d069f	450	360	360	8519	d100e	712	691	691	11693
d070f	250	163	163	3506	d101e	620	354	354	8798
d071f	204	126	126	2354	d102e	802	770	770	13831
d072f	483	480	480	9141	d103g	299	194	194	4414
d073b	306	168	168	3816	d104g	421	332	332	6265
d074b	318	176	176	4007	d105g	355	246	246	6888
d075b	362	274	274	6512	d106g	313	213	213	4285
d076b	421	321	321	7146	d107g	336	228	228	5355
d077b	432	323	323	6883	d108g	296	185	185	4092
d078b	359	337	337	5475	d109h	275	139	139	4189
d079a	398	343	343	7146	d110h	598	433	433	9341
d080a	653	548	548	12249	d111h	292	210	210	4781
d081a	403	291	291	6763	d112h	306	178	178	4868
d082a	284	216	216	5450	d113h	183	112	112	2903
d083a	275	209	209	4754	d114h	485	380	380	7629
d084a	460	395	395	8211	d115i	338	264	264	5611
d085d	307	232	232	4905	d116i	430	363	363	7727
d086d	514	395	395	8673	d117i	344	322	322	6214
d087d	363	293	293	6514	d118i	467	389	389	7904
d089d	480	376	376	8022	d119i	319	218	218	5035
d090d	344	247	247	5327	d120i	477	358	358	8755
d091c	360	286	286	5307	TOTAL:	22114	16878	16878	480197

$1, 2, ..., D$, the isf_{w_i}, $sim(i, O)$, and $sim(i, j)$ are calculated for all $i, j \in \{1, 2, ..., N\}$, then

4. The DE generation loop is started, repeating $GMAX$ times over the population of NP vectors representing candidate model summaries, yielding the obtained optimized summary model sentence selection,
5. Printing of the statistics and ROUGE evaluations.

After information identification (pre-processing), a system summary is computed, i.e. evolved by optimization. We chose BinaryDE [89] as the Evolutionary Algorithm, like in [18], but adding to the BinaryDE the ϵ-jDE self-adaptation of control parameters [65, 53, 45], which evolves *sentence in-*

clusion vectors (i.e. a 0/1 set of flags if a sentence is to be included in the final summary). The genotype-phenotype mapping from floating points to binary values is done the same as in BinaryDE [89]. Also, we applied inside DE, that the epsilon constraint definition was such that for the upper limit, the sentence length constraint bound value was strictly violated, but for the lower bound, a difference was allowed for an epsilon between the maximum and minimum sentence. During evolution, the DE evaluates the trial vectors using the evaluation as defined in the previous subsection (3.1). The best vector sentences obtained by BinaryDE (each sentence marked "1") is output as a reconstructed sentence text (in Sentence-Per-Line, SPL format), prepared by Freeling decomposition using each sentence's corresponding sentence number. The obtained system summary (evolved by DE) sentences are mapped using its Freeling printed SPL format to unstemmed original SPL sentences, and compared to human model sentences using jackknifing on ROUGE 1.5.5 with Porter stemmer enabled. The algorithm listed using pseudocode in Fig. 2 is executed as a call to CaBiSDETS(RNi=`$RNi`, MDOC=`$MDOC`, $L = 200$, $GMAX = 10000$, $NP = 500$, $F_{\mathrm{l}} = 0$, $F_{\mathrm{u}} = 0.9$, $\tau_{\mathrm{F}} = 0.1$, $\tau_{\mathrm{CR}} = 0.1$) using different RNi settings, where the `$RNi` seed and `$MDOC` filename are chosen in lines 4 and 5 of the script in Fig. 4. The pseudocode in Fig. 2 shows the use of input files and summary model preparation in lines 2–6, DE initializations in lines 7–9, then the DE loop follows in lines 10–26, which return the result in line 27. The inner DE loop of the algorithm in Fig. 2 runs trial vector computation in lines 12–17 and summary fitness evaluation in lines 18–22 (binarization happens in line 19), then it calculates the propagated DE vector in line 23, and updates the constraint handling level in line 25. After this, the algorithm is executed, and it determines the list of sentences that are in the optimized summary set of extracted sentences.

3.3. Distributing Parallel Tasks to a Grid

This subsection provides the definition of executing parallel tasks for summarization, where the benchmark for summarization is first split to different parallel tasks, and these tasks are then executed independently and, finally, merged.

The following two subsubsections provide two approaches to run the tasks in parallel, i.e. to run each task in the dataset in parallel in the same batch on one machine, or, to send these tasks as independent batches to be scheduled on different machines. Then, gathering of data from the parallel runs is explained.

```
1  #!/bin/bash
2  mkdir -p volatile
3
4  for RNi in {1..20}; do
5   for MDOC in `ls -1 input/duc2002_FreelingSentences_recomposed_TEXTO/\
6    | sed -e 's/\.txt//g'`; do
7    mkdir -p output/unstemmedmodels/$MDOC
8
9    ./summarizer & --constraintHandlingEpsilonAdaptivePopulations --useBinaryDE\
10    --RNi $RNi --GMAX 10000 --NP 500 --summarylength 200\
11    --inputfile input/duc2002docs_CONCEPT_COUNTS_CONCEPT_ID/${MDOC}.txt\
12    --withStatementMarkers --MDOCmatrix\
13    --useConceptMatrix input/duc2002docs_matrixConcepts_NoStopW_OK/${MDOC}.txt.
          matriz\
14    input/DUC-2002_FreelingSentences_WORDCOUNT_PER_SENTENCE/${MDOC}.txt.wordcount\
15    --printSummaryStatementIDsONLY --printOptimizationNewBests\
16    --printParameterVectors --printOptimizationNewBests\
17    --printOptimizationNewBestsOutputROUGE\
18    | tee volatile/output-smdoc-$MDOC-$RNi;
19   done # MDOC
20  done # RNi
21  wait # all tasks to finish
```

Figure 3: Script listing for running in parallel for summarization tasks in DUC2002 using a single multi-processor machine.

3.3.1. Running each task in parallel on one machine

Running all tasks in parallel, the optimization execution file `summarizer` is called on a multi-processor Bash-operated machine using following script listed in Figure 3. The script generates output directories and uses input directories from a preprocessed DUC 2002 datasets. Lines 4 and 5 represent the loops over different multi-documents for all independent runs. The lines 9–17 run the specific code of the summarization algorithm, using the CaBiS-DETS algorithm with adaptive constraint handling (line 9), settings for DE (line 10), input text file (line 11), concepts matrix format settings and files (lines 11–14). Lines 2 and 7 prepare the output structure, while line 18 saves the results, and the lines 15–17 set the output print format for the optimizer executable.

3.3.2. Sending tasks as independent batches to be scheduled on different machines

Given the algorithm code from `code` directory put in package together with the corresponding input files as one task input file, the tasks are sent as independent batches for scheduling on different machines as shown in Figure 4. While here the optimization executable inputs and outputs (on

lines 12–21) and setup of output structure are same as in Figure 3, the difference is how the loops over different multi-documents for all independent runs are provided. Namely, these loops generate task shell Bash script (line 23), packaged together with input files (lines 26–32) and prepare the task package (lines 34–40) to be submitted and retrieved as seen in Figure 5 using ARC [75] commands, `arcsub ${task}.xrsl` (line 7) for every parallel task submitted and `arcget -a` command (line 13) after all tasks complete to retrieve the summaries (line 12 as proof-of-concept that could be changed to e.g. some loop checking using `[ $(arcstat -s Finished | wc -l) == 1180 ]`) for these 1180 tasks.

3.3.3. Merging of distributed tasks results

For sake of clarity, merging of ROUGE results on DUC 2002 dataset is listed in script in Figure 6. This shell script uses any of the outputs provided (lines 3-11) from the files structure generated from previous two parallelization approaches in Figures 4 and 3 and then prints the average ROUGE values (lines 13-16) or plots the results (lines 18-23).

Table 2: Average ROUGE 1, 2, L, and SU recall values through generations (logscale), for the best obtained solution during optimization with CaBiSDETS algorithm, using parameters $NP = 500$ and $GMAX$=10000. Final generation yielded results are in bold.

Metric / Generation	ROUGE 1	ROUGE 2	ROUGE L	ROUGE SU4	Fitness
13	0.22588	0.0380907	0.184536	0.0573557	-80.1165
147	0.272849	0.054693	0.240132	0.0882645	-164.708
1008	0.287191	0.0580012	0.258406	0.0967963	-251.043
10000	**0.318715**	**0.0655228**	**0.290371**	**0.107963**	-511.336

4. Results and Discussion

This section presents results by reporting the quantitative evaluation of the obtained summary models using comparison to human peer summaries. The optimization algorithm parameter settings used were: Population size was set at NP=500 and the maximum number of generations was set at $GMAX$=10000.

Figure 7 shows the text summarization model and ROUGE metric recall values' convergence during generations, averaged over 20 runs. As observed, all the values improved with adding more generations and, therefore, it is

```
#!/bin/bash
mkdir -p xrsl

for RNi in {1..20}; do
 for MDOC in `ls -1 input/duc2002_FreelingSentences_recomposed_TEXTO/\
 | sed -e 's/\.txt//g'`; do

  task="$MDOC-seed-${RNi}"

  echo -e "#!/bin/bash\ntar xvf summarizer-code-and-inputdata-${task}.tgz;
  mkdir -p volatile;
  ./summarizer --constraintHandlingEpsilonAdaptivePopulations --useBinaryDE\
   --RNi $RNi --GMAX 10000 --NP 500 --summarylength 200\
   --inputfile input/duc2002docs_CONCEPT_COUNTS_CONCEPT_ID/${MDOC}.txt\
   --withStatementMarkers --MDOCmatrix\
   --useConceptMatrix input/duc2002docs_matrixConcepts_NoStopW_OK/${MDOC}.txt.
       matriz\
   input/DUC-2002_FreelingSentences_WORDCOUNT_PER_SENTENCE/${MDOC}.txt.wordcount\
   --printSummaryStatementIDsONLY --printOptimizationNewBests\
   --printParameterVectors --printOptimizationNewBests\
   --printOptimizationNewBestsOutputROUGE\
  | tee volatile/output-smdoc-$MDOC-$RNi;
  tar cvzf ${task}.tgz volatile/output-smdoc-$MDOC-$RNi;
  echo Experiment successfuly finished: ${task}" > ${task}.sh
  chmod 755 ${task}.sh

  cd code;
  tar cvzf ../summarizer-code-and-inputdata-${task}.tgz .\
   ../input/duc2002_FreelingSentences_recomposed_TEXTO/$MDOC\
   ../input/duc2002docs_CONCEPT_COUNTS_CONCEPT_ID/$MDOC\
   ../input/duc2002docs_matrixConcepts_NoStopW_OK/$MDOC\
   ../input/DUC-2002_FreelingSentences_WORDCOUNT_PER_SENTENCE/$MDOC
  cd ..

  cat experiment.xrsl | sed -e "s/experiment.tgz/${task}.tgz/g"\
   | sed -e "s/jobname=experiment/jobname=${task}/g"\
   | sed -e "s/executable=task.sh/executable=${task}.sh/g"\
   | sed -e "s/workload.tgz/workload-${task}.tgz/g"\
  > ${task}.xrsl

  mv ${task}.xrsl workload-${task}.tgz ${task}.sh xrsl
 done # MDOC
done # RNi
```

Figure 4: Script listing for preparing parallel summarization tasks in DUC2002 using a grid to call the algorithm from Fig. 2.

reasonable to run the summarization processing for more generations. The ROUGE values, as well as the fitness of the model, were improving throughout the evolution. As an example, values over generation numbers at natural values 13, 147, 1008, and the last generation, the ROUGE values and fitness are seen in Table 2. The results in Figure 7 show an increasing trend as far as

```
#!/bin/bash
for RNi in {1..20}; do
 for MDOC in `ls -1 input/duc2002_FreelingSentences_recomposed_TEXTO/\
 | sed -e 's/\.txt//g'`; do

  task="$MDOC-seed-${RNi}"
  cd xrsl; arcsub ${task}.xrsl -j jobs.dat; cd ..

 done # MDOC
done # RNi

read -p "Finished?" # wait for parallel tasks completion
arcget -a -j jobs.dat
for F in *.tgz; do tar xf $F; done
```

Figure 5: Submitting, retrieving, and merging results of parallel tasks execution on a grid (runs 1180 CaBiSDETS algorithm executions from Fig. 2 as Parallel Tasks, possibly merely simultaneously if allocated 1180 nodes at once by Slurm).

```
#!/bin/bash

for RUN in {1..20}; do
 for DOC in $(ls -1 *-$RUN|sed -e 's/output-smdoc-//g' -e 's/-'$RUN'$//g'); do
  export DOC; cat output-smdoc-$DOC-$RUN |\
  awk '/ROUGE/{print ENVIRON["DOC"], $5, $(NF-3), $(NF-2), $(NF-1), $NF}' ;
done | tee ../plots/rougeCross4-$RUN; done

for RUN in {1..20}; do for i in {061..120}; do
  cat rougeCross4-$RUN|grep d${i} > plot4-$RUN-$i; tail -n 1 plot4-$RUN-$i
done; done

for RUN in {1..20}; do
  for i in {061..120}; do tail -n 1 plot4-$RUN-$i; done;
done > COMBINED
for i in {2..6}; do cat COMBINED | avg $i; done # prints the average ROUGE values

echo -n "plot " # generate plotting commands for Gnuplot
for RUN in {1..20}; do for i in {61..120}; do
  cat rougeCross4-$RUN|grep ${i}..txt > plot-$RUN-$i; echo -n "'plot-"$RUN"-"$i;
  echo -n "' u 3 w l lc "$i" t 'Run #"$RUN" for model ";
  cat plot-$RUN-$i | awk 'NR==1 {printf("%s", $1);}'; echo -n "', ";
done; done
```

Figure 6: Merging of ROUGE results on DUC 2002 dataset from optimization tasks run in parallel.

the ROUGE values obtained, which indicates that the more generations, the better the model is for selecting relevant information. As the language model for multi-documents are usually not presented and might vary for different computations, we, hence, also included the obtained count of words for each

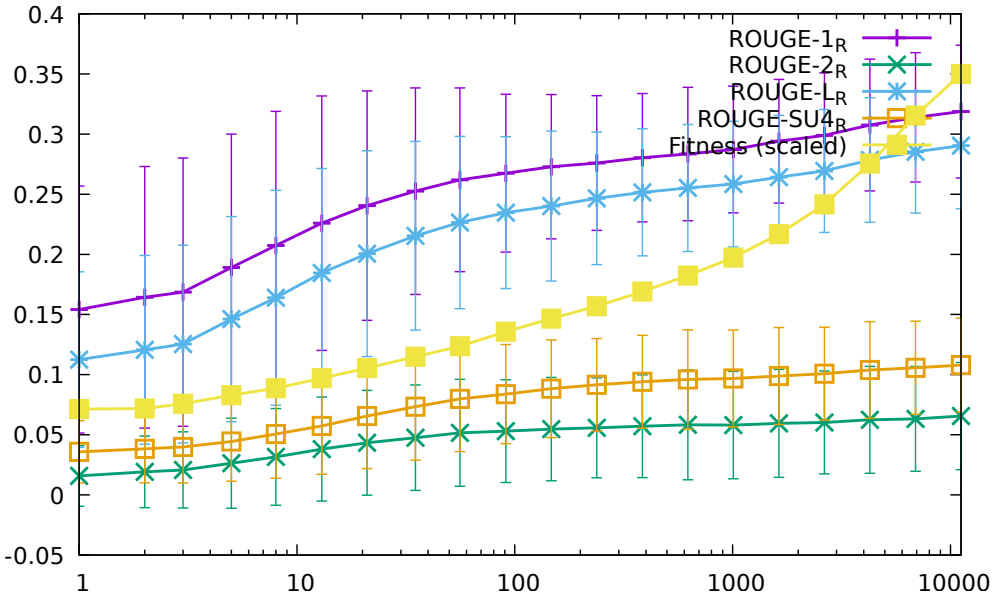

Figure 7: Average ROUGE 1, 2, L, and SU recall values through generations (logscale) for the best obtained solution during optimization with CaBiSDETS, using parameters $NP = 500$ and $GMAX$=10000. Also, fitness obtained on average, is plotted along this evolution, as a scaled graph starting at 0.05 and divided by the best fitness value (-511.336). A vertical line on each data point draws the standard deviation at this point.

sentence in each multi-document set, in detail Figures 8 and 9. Furthermore, for the demonstrative purpose of describing the research, three outputs with different approaches for a sample summarization task (document d061j from DUC 2002) are demonstrated in Figure 10.

In terms of accuracy performance, the resulting summaries obtained acceptable results for the recall value compared to other summarization approaches. In this manner, with 10,000 generations, results overperformed the COMPENDIUM system, which is a very powerful summarization system in NLP [90]. Moreover, the results also improved with respect to a very competitive baseline (i.e. *Lead*) when the model was run with at least 1,000 generations. This baseline extracts the first sentences of the documents, and, taking into account the structure of a news document, in which its most important information is stated at the very beginning, the first paragraph contains the answer to the five Ws questions (i.e., *Who, What, When, Where, and Why), so it can be appropriate to be considered as a summary.*

	ROUGE-1	ROUGE-2	ROUGE-L	ROUGE-SU4
Our model (10000 generations)	0.31872 (0.05518)	0.065523 (0.04456)	0.29037 (0.05241)	0.10796 (0.03898)
COMPENDIUM	0.30340	0.05357	0.26554	0.09282
Lead baseline	0.28684	0.05283	-	-

Table 3: Comparison in terms of summary information appropriateness (recall values of ROUGE-1, ROUGE-2, ROUGE-L and ROUGE-SU4). Results in paranthesis for our approach report the standard deviation of the value considering different independent runs of DE over the multi-document sets.

Enabling factors of the anaylsis conducted were also the setup and definition of the approach, which included that the information identification and quantitative evaluation use Freeling [8], coreference resolution, semantic analysis, and concept matrix in the pre-processing stage before summary optimization (task) commenced; and, in the active part, a constrained version was used of a self-adaptive DE with binarization as an optimizer. As the parts of fitness function that do not need to be recomputed during optimization, are precomputed, the computational perspective defining static (pre-computed) and changing (computed during evaluation call) parts of fitness function yielded important and efficient high-performance computing scenarios. Under these scenarios, the parallelization of summarization tasks submitted, distributed to a computer grid and merged after tasks were complete, an important computationally-intensive correlation analysis was able to be reported, empirically providing insight into the correlation and trend among ROUGE metric values on the dedicated benchmark (DUC 2002) along with the number of fitness functions. Thereby, testing the trend and payoff of prolonged optimization execution runs showed summarization quality improvement after extended optimization run time, that was possible due to task parallelization on grid.

Therefore, the experiments conducted and the results obtained confirm the feasibility of using parallelization and high-performance computing in summarization tasks, which renders it beneficial to be applied in such costly and very time-consuming tasks.

5. Conclusions and Future Work

This paper presented tackling of a hard optimization problem of computational linguistics, specifically automatic multi-document text summarization. Feasibility and analysis were demonstrated for the problem, and parallelization was explained for the case of High-Performance Computing. A Differential Evolution algorithm was applied to optimize the text summarization model.

Different DE runs were distributed to a grid in parallel as optimization tasks, seeking high processing throughput despite the demanding complexity of the linguistic model, especially on longer multi-documents, where DE improves results given more iterations. The impact of a larger number of function evaluations was demonstrated to be beneficial to improve summarization performance along with evolution steps in terms of ROUGE metrices.

As prospects for future research, we see further improvement of the optimization approach, including multi-modal and multi-objective approaches. A multi-objective approach, using multiple criteria functions, could be an interesting challenge for DE, and, although it is difficult to define more objective functions for the current problem (coverage vs. redundancy), multi-objective optimization seems likely for deeper knowledge insight, but criteria still need to be defined. Also, more comparisons with other techniques could be done based on different aspects of evaluation, like qualitative human-based evaluations.

Acknowledgments

This paper is based upon work from COST Action IC1406 High-Performance Modelling and Simulation for Big Data Applications (cHiPSet), supported by COST (European Cooperation in Science and Technology). This paper is also based upon work from COST Actions CA15140 "Improving Applicability of Nature-Inspired Optimisation by Joining Theory and Practice (ImAppNIO)", and CA18231 "Multi3Generation: Multi-task, Multilingual, Multi-modal Language Generation", both supported by COST. The author AZ acknowledges the financial support from the Slovenian Research Agency (Research Core Funding No. P2-0041). AZ also acknowledges EU support under Project No. 5442-24/2017/6 (HPC – RIVR). AZ also acknowledges the EU Interreg Alpine Space project SmartVillages and Erasmus TSM grant. The author EL acknowledges the financial support by the Generalitat Valenciana through the Research Project PROMETEU/2018/089, and by the Spanish Government through the INTEGER project (RTI2018-094649-B-I00), and network RED iGLN (TIN2017-90773-REDT).

Figure 8: DUC 2002 multi-document (MDOC) statistics Table, together with word count in each sentence for sets `d061` to `d091`.

MDOC	#Words
d061j	31, 17, 7, 12, 15, 23, 14, 16, 28, 32, 20, 2, 24, 7, 23, 20, 18, 18, 18, 26, 8, 5, 20, 14, 33, 3, 29, 29, 27, 36, 12, 7, 22, 19, 46, 14, 13, 28, 19, 35, 17, 28, 18, 25, 10, 25, 37, 30, 24, 30, 32, 37, 25, 33, 29, 26, 26, 17, 16, 22, 8, 29, 28, 19, 21, 19, 32, 13, 29, 24, 20, 43, 34, 37, 20, 14, 18, 10, 12, 24, 16, 31, 21, 21, 20, 23, 37, 16, 11, 19, 18, 22, 17, 35, 12, 20, 18, 24, 2, 12, 25, 22, 15, 7, 23, 29, 19, 11, 17, 16, 11, 17, 17, 5, 9, 44, 18, 14, 18, 19, 16, 17, 21, 39, 20, 30, 33, 12, 34, 7, 19, 28, 22, 2, 26, 38, 15, 31, 17, 24, 15, 17, 10, 15, 12, 34, 35, 4, 7, 9, 22, 12, 9, 10, 13, 11, 15, 11, 22, 32, 11, 14, 22, 34, 3, 22, 29, 7, 13, 15, 3, 8, 21, 20, 15, 21, 13, 3, 23, 19, 8, 24, 25, 15, 28, 14, 18, 42, 7, 32, 6, 22, 22, 19, 19, 26, 19, 14, 14, 32, 28, 21
d062j	39, 39, 22, 26, 46, 20, 38, 16, 20, 29, 6, 50, 19, 32, 25, 31, 12, 40, 14, 25, 12, 28, 26, 30, 17, 25, 12, 18, 19, 13, 7, 9, 19, 7, 16, 9, 10, 22, 14, 13, 26, 45, 32, 47, 20, 11, 34, 10, 38, 22, 25, 10, 26, 28, 31, 21, 23, 50, 27, 20, 27, 22, 10, 13, 38, 26, 15, 23, 28, 44, 27, 42, 18, 41, 15, 18, 20, 32, 35, 36, 10, 19, 22, 37, 18, 30, 25, 49, 21, 29, 17, 39, 30, 32, 21, 18, 21, 42, 37, 26, 18, 41, 19, 22, 37, 11, 40, 17, 61, 16, 23, 37, 20, 27, 23, 27, 29, 36
d063j	25, 17, 29, 18, 14, 17, 18, 22, 10, 7, 23, 42, 23, 16, 45, 16, 29, 21, 34, 34, 15, 37, 12, 33, 18, 5, 21, 14, 7, 8, 9, 17, 8, 9, 24, 25, 12, 28, 31, 20, 20, 21, 9, 10, 14, 11, 10, 10, 26, 21, 7, 16, 14, 15, 11, 23, 8, 22, 11, 30, 11, 12, 16, 9, 30, 47, 17, 26, 15, 8, 16, 13, 19, 25, 31, 31, 24, 16, 36, 22, 32, 15, 20, 37, 23, 12, 6, 38, 44, 20, 21, 9, 10, 14, 3, 13, 14, 8, 14, 8, 23, 27, 29, 24, 19, 16, 31, 17, 33, 29, 26, 15, 16, 20, 26, 18, 6, 17, 24, 16, 17, 20, 30, 5, 28, 19, 9, 9, 68, 38, 29, 39, 37, 38, 12, 21, 12, 26, 31, 36, 9, 11, 28, 29, 21, 21, 2, 19, 16, 13, 39, 33, 21, 12, 2, 15, 2, 19, 61, 15, 33, 32, 17, 13, 18, 13, 14, 11, 38, 15, 17, 31, 30, 16, 14, 4, 16, 26, 13, 30, 18, 7, 17, 3, 12, 23, 19, 8, 19, 6, 11, 30, 19, 54, 21, 13, 26, 14, 26, 46, 29, 10, 84, 20, 19, 3, 20, 16, 17, 28, 27, 18, 21, 28, 34, 18, 25, 20, 23, 27, 30, 6, 8, 12, 31, 8, 31, 32, 28, 18, 14, 15, 22, 21, 15, 12, 47, 9, 3, 25, 13, 58, 33, 7, 40, 28, 28, 35, 31, 31, 26, 10, 17, 4, 17, 27, 11, 35, 26, 24, 30
d064j	37, 47, 33, 21, 14, 31, 36, 28, 23, 39, 27, 31, 18, 36, 25, 21, 25, 23, 45, 36, 23, 16, 22, 12, 25, 39, 34, 20, 15, 36, 13, 21, 17, 24, 16, 24, 18, 16, 32, 31, 20, 26, 23, 24, 43, 33, 35, 34, 14, 29, 17, 30, 26, 44, 18, 25, 5, 11, 6, 9, 11, 38, 57, 6, 15, 32, 15, 46, 36, 15, 34, 34, 28, 10, 18, 16, 26, 12, 35, 30, 24, 29, 24, 30, 11, 32, 21, 13, 23, 18, 14, 20, 41, 31, 33, 9, 12, 76, 35, 36, 44, 33, 46, 46, 64, 31, 13, 10, 37, 32, 14, 37, 45, 23, 12, 9, 7, 20, 9, 23, 12, 19, 7, 8, 19, 22, 38, 45, 45, 16, 39, 44, 12, 29, 8, 3, 11, 26, 35, 12, 35, 21, 27, 11, 3, 40, 22, 36, 28, 8, 61, 30, 24, 27, 13, 16, 35, 35, 22, 31, 19, 13, 20, 16, 28, 19, 21, 12, 36, 17, 19, 23, 28, 24, 40, 23, 23, 8, 14, 12, 31, 31, 32, 11, 17, 25, 30, 21, 4, 22, 19, 3
d065j	32, 27, 26, 9, 26, 38, 34, 39, 24, 40, 36, 41, 39, 27, 31, 7, 8, 28, 32, 11, 9, 31, 18, 24, 10, 22, 21, 18, 20, 2, 18, 17, 11, 22, 12, 24, 22, 10, 16, 18, 11, 5, 5, 17, 61, 26, 16, 16, 24, 22, 19, 10, 19, 4, 35, 4, 20, 11, 2, 6, 10, 5, 10, 3, 3, 24, 34, 8, 6, 14, 31, 9, 7, 12, 18, 28, 12, 44, 32, 11, 24, 6, 16, 42, 18, 9, 8, 34, 18, 16, 12, 30, 2, 24, 65, 37, 7, 10, 36, 33, 24, 22, 3, 20, 43, 35, 42, 20, 22, 30, 22, 8, 12, 12, 23, 2, 31, 12, 18, 3, 2, 32, 17, 16, 17, 18, 31, 13, 20, 2, 34, 27, 8, 20, 8, 6, 21, 11, 10, 9, 19, 31, 15, 27, 14, 40, 23, 6, 3, 31, 33, 15, 22, 27, 27, 19, 27, 19, 30, 36, 50, 15, 18, 39, 12, 40, 23, 17, 29, 11, 3, 10, 35, 14, 12, 10, 3, 49, 34, 4, 13, 3, 31, 15, 36, 8, 23, 26, 11, 10, 3, 36, 3, 12, 13, 11, 18, 28, 22, 3, 45, 9, 25, 27, 9, 40, 27, 13, 41, 14, 3, 13, 20, 19, 5, 6, 20, 11, 16, 17, 9, 13, 5, 60, 25, 17, 12, 19, 28, 19, 20, 9, 11, 10, 11, 12, 6, 14, 15, 5, 5, 16, 15, 8, 22, 15, 3, 19, 6, 8, 3, 8, 8, 17, 18, 3, 24, 16, 31, 20, 9, 29, 21, 15, 11, 3, 15, 9, 13, 11, 10, 15, 2, 10, 16, 29, 26, 13, 25, 3, 16, 12, 25, 31, 8, 48, 37, 12, 23, 104, 17, 32, 22, 41, 24, 30, 44, 40, 14, 23, 10, 10, 7, 26, 13, 27, 44, 15, 27, 44, 29, 48, 19, 17, 30, 42, 44, 13, 33, 30, 38, 21, 17, 54, 13, 42, 28, 19
d066j	31, 20, 29, 18, 22, 14, 18, 13, 38, 9, 10, 16, 25, 26, 14, 13, 17, 29, 22, 25, 23, 23, 7, 19, 23, 27, 26, 15, 12, 22, 33, 15, 8, 12, 12, 5, 12, 8, 10, 13, 29, 41, 16, 12, 20, 33, 22, 9, 19, 26, 3, 11, 35, 18, 6, 12, 54, 37, 44, 19, 20, 8, 29, 3, 10, 14, 10, 53, 6, 15, 14, 12, 11, 31, 17, 13, 14, 9, 30, 19, 18, 7, 18, 4, 29, 23, 19, 20, 25, 14, 38, 9, 22, 15, 22, 33, 20, 24, 19, 13, 8, 29, 39, 16, 19, 3, 9, 46, 12, 35, 35, 68, 22, 18, 12, 37, 34, 17, 18, 52, 24, 27, 44, 31, 38, 10, 25, 45, 15, 24, 19, 14, 25, 38, 33, 5, 37, 17, 13, 18, 10, 22, 24, 39, 11, 31, 15, 50, 38, 20, 40, 9, 40, 16, 18, 18, 31, 16, 6, 37, 19, 12, 38, 41, 15, 20, 23, 25, 50, 50, 14, 37, 41, 24, 27, 11, 22, 40, 17, 7, 47, 54, 9, 39, 50, 16, 26, 60, 38, 34, 33, 12, 43, 17, 29, 55, 24, 28, 30, 21
d067f	63, 40, 50, 10, 15, 33, 26, 32, 43, 31, 15, 20, 42, 48, 7, 48, 26, 18, 62, 37, 32, 27, 29, 31, 21, 29, 37, 53, 31, 27, 22, 14, 38, 25, 23, 16, 16, 14, 21, 31, 62, 26, 13, 20, 24, 14, 15, 23, 25, 18, 16, 39, 37, 49, 28, 10, 14, 22, 33, 24, 24, 27, 8, 11, 31, 20, 21, 15, 26, 21, 13, 36, 21, 38, 12, 38, 17, 25, 22, 26, 40, 23, 7, 18, 23, 10, 18, 33, 19, 22, 7, 26, 25, 19, 40, 40, 34, 25, 18, 24, 36, 19, 29, 35, 32, 14, 20, 14, 30, 38, 26, 16, 18, 36, 24, 33, 25, 28, 24, 14, 21
d068f	19, 24, 42, 33, 8, 30, 25, 24, 21, 33, 24, 25, 24, 50, 27, 8, 12, 25, 28, 34, 32, 14, 21, 27, 24, 27, 29, 13, 17, 26, 29, 26, 14, 13, 16, 18, 14, 8, 12, 7, 30, 23, 10, 28, 11, 20, 31, 29, 26, 4, 18, 25, 12, 21, 17, 33, 12, 14, 31, 43, 28, 33, 36, 32, 14, 30, 11, 7, 11, 13, 52, 38, 46, 6, 2, 19, 25, 13, 18, 16, 13, 16, 9, 17, 12, 20, 21, 4, 11, 7, 8, 10, 26, 25, 14, 15, 15, 15, 15, 30, 17, 6, 7, 22, 10, 23, 23, 20, 14, 36, 21, 4, 29, 3, 13, 14, 8, 13, 7, 18, 45, 31, 21, 7, 19, 16, 26, 29, 33, 10, 22, 41, 30, 17
d069f	24, 24, 27, 19, 27, 29, 8, 42, 41, 17, 22, 30, 26, 20, 36, 25, 47, 28, 26, 23, 45, 12, 14, 28, 22, 4, 26, 35, 15, 25, 46, 40, 27, 25, 22, 21, 14, 8, 23, 32, 36, 29, 11, 17, 5, 22, 19, 21, 35, 33, 38, 31, 39, 29, 17, 47, 30, 16, 3, 34, 13, 14, 6, 32, 23, 16, 20, 3, 18, 15, 33, 38, 3, 18, 17, 3, 27, 33, 26, 3, 27, 26, 31, 13, 2, 16, 4, 28, 4, 22, 25, 18, 28, 15, 32, 17, 27, 13, 12, 34, 7, 24, 28, 31, 15, 14, 3, 21, 37, 16, 35, 16, 29, 16, 28, 30, 23, 9, 28, 15, 22, 23, 24, 21, 11, 11, 39, 42, 19, 34, 16, 3, 47, 3, 11, 6, 12, 27, 16, 14, 34, 31, 17, 26, 23, 23, 21, 19, 13, 38, 17, 35, 22, 39, 12, 3, 15, 20, 32, 18, 15, 31, 30, 15, 35, 4, 19, 11, 11, 24, 3, 16, 23, 44, 30, 27, 18, 17, 7, 31, 27, 35, 32, 16, 31, 12, 21, 36, 21, 11, 24, 21, 12, 6, 9, 18, 21, 31, 24, 15, 37, 39, 7, 32, 13, 15, 16, 15, 21, 24, 28, 10, 35, 27, 34, 29, 21, 4, 9, 8, 6, 17, 33, 32, 7, 7, 6, 39, 32, 15, 33, 20, 7, 21, 31, 26, 9, 27, 34, 43, 9, 203, 39, 25, 31, 11, 3, 42, 80, 67, 9, 8, 18, 20, 10, 23, 7, 17, 24, 32, 14, 27, 14, 18, 16, 7, 33, 22, 15, 5, 22, 41, 38, 26, 34, 28, 4, 71, 28, 20, 19, 39, 43, 20, 24, 3, 33, 25, 17, 15, 25, 19, 30, 49, 26, 53, 18, 29, 14, 37, 11, 24, 23, 11, 39, 16, 9, 20, 6, 24, 37, 40, 33, 33, 7, 3, 36, 45, 33, 19, 40, 31, 30, 22, 27, 59, 54, 30, 10, 27, 18, 49, 44, 3, 42, 8, 3, 22, 27, 17, 28, 14, 39, 37, 24, 5, 45, 17, 24, 12, 13, 20, 23, 32, 30, 41, 25, 28, 23, 29
d070f	26, 10, 7, 34, 27, 24, 13, 6, 20, 16, 7, 25, 23, 3, 22, 3, 21, 22, 3, 38, 37, 30, 8, 8, 25, 9, 3, 24, 19, 24, 28, 3, 13, 22, 25, 15, 40, 32, 40, 14, 16, 37, 29, 11, 17, 12, 11, 23, 35, 29, 13, 15, 38, 31, 38, 29, 14, 25, 26, 35, 27, 9, 20, 14, 39, 8, 31, 25, 17, 24, 11, 23, 19, 24, 36, 28, 14, 25, 31, 28, 24, 44, 35, 30, 12, 6, 13, 23, 12, 28, 29, 26, 25, 26, 28, 22, 10, 24, 5, 39, 24, 37, 30, 30, 14, 28, 18, 35, 24, 6, 6, 14, 31, 16, 5, 23, 12, 29, 9, 28, 5, 9, 12, 34, 18, 25, 40, 25, 21, 18, 30, 16, 12, 18, 17, 29, 15, 26, 31, 29, 39, 29, 19, 13, 34, 20, 8, 27, 26, 20, 25, 20, 12, 10, 15, 23, 16, 14, 30, 23, 37, 11, 19
d071f	25, 21, 28, 15, 15, 9, 17, 28, 16, 27, 16, 21, 19, 28, 32, 41, 11, 37, 28, 22, 24, 42, 17, 22, 23, 33, 18, 18, 23, 13, 62, 21, 14, 14, 7, 40, 16, 15, 25, 11, 40, 13, 14, 18, 19, 25, 24, 13, 31, 11, 29, 9, 16, 33, 11, 9, 18, 8, 12, 26, 11, 15, 15, 21, 33, 7, 22, 27, 14, 10, 35, 27, 6, 29, 22, 16, 15, 10, 22, 3, 19, 21, 9, 4, 5, 4, 14, 35, 20, 14, 15, 10, 11, 12, 10, 31, 2, 2, 2, 3, 4, 18, 3, 16, 1, 10, 17, 14, 23, 14, 51, 15, 22, 13, 36, 19, 12, 11, 15, 14, 13, 32, 29, 15, 7, 34
d072f	29, 32, 5, 28, 12, 2, 12, 2, 13, 3, 6, 13, 2, 14, 29, 31, 34, 34, 35, 25, 27, 12, 2, 12, 2, 13, 12, 19, 4, 19, 26, 15, 29, 34, 22, 43, 8, 11, 29, 27, 12, 12, 24, 22, 25, 47, 16, 20, 64, 18, 16, 28, 8, 29, 30, 21, 38, 8, 12, 49, 12, 7, 7, 21, 26, 15, 18, 28, 9, 7, 23, 29, 29, 33, 34, 27, 13, 22, 33, 35, 7, 23, 22, 33, 4, 26, 38, 18, 12, 27, 15, 10, 9, 15, 39, 23, 24, 3, 29, 12, 14, 7, 5, 18, 46, 24, 6, 5, 9, 8, 6, 16, 4, 20, 6, 9, 5, 4, 5, 23, 13, 33, 24, 21, 22, 16, 15, 11, 19, 25, 4, 2, 2, 10, 30, 34, 36, 3, 37, 16, 9, 13, 36, 10, 41, 13, 5, 15, 6, 11, 10, 9, 5, 13, 9, 12, 10, 8, 4, 16, 2, 15, 15, 24, 8, 26, 11, 27, 25, 13, 4, 3, 5, 8, 8, 9, 15, 7, 9, 2, 10, 10, 12, 6, 16, 11, 10, 6, 21, 4, 6, 4, 4, 21, 7, 3, 31, 2, 2, 22, 24, 26, 24, 8, 26, 2, 10, 8, 21, 7, 21, 41, 7, 17, 27, 70, 34, 56, 16, 24, 11, 15, 29, 10, 14, 2, 42, 29, 35, 22, 46, 21, 48, 52, 35, 33, 27, 22, 29, 18, 5, 7, 6, 22, 20, 12, 39, 26, 45, 30, 11, 16, 18, 23, 18, 22, 18, 3, 19, 5, 22, 24, 35, 21, 22, 24, 42, 53, 21, 6, 17, 20, 29, 9, 13, 29, 17, 11, 16, 25, 24, 38, 9, 4, 47, 27, 15, 2, 7, 29, 5, 42, 24, 15, 16, 8, 15, 18, 7, 15, 39, 15, 5, 15, 14, 13, 14, 9, 14, 8, 9, 13, 4, 5, 10, 6, 5, 20, 2, 6, 10, 13, 4, 7, 7, 35, 5, 14, 8, 20, 14, 31, 25, 34, 13, 3, 8, 46, 24, 36, 26, 10, 24, 25, 26, 26, 32, 29, 24, 3, 3, 6, 7, 17, 7, 30, 25, 29, 41, 47, 25, 9, 43, 25, 2, 15, 22, 12, 12, 9, 13, 40, 34, 25, 25, 34, 33, 43, 15, 26, 26, 12, 22, 27, 1, 15, 15, 6, 25, 51, 9, 21, 37, 22, 27, 46, 44, 45, 15, 73, 12, 8, 31, 11, 31, 13, 8, 24, 30, 35, 64, 22, 32, 24, 15, 11, 15, 22, 26, 22, 5, 3, 27, 48, 24, 38, 14, 26, 32, 29, 10, 19, 29, 80, 7, 17, 28, 1, 28, 17, 49, 3, 14, 15, 15, 10, 37, 6, 11, 7, 13, 2, 1, 58, 20, 5, 2, 4, 44, 3, 8, 21, 2, 5, 16, 27, 3, 7, 5, 30, 42, 55, 5, 6, 32, 7, 1, 6, 24, 9
d073b	22, 18, 30, 26, 37, 29, 31, 22, 35, 10, 18, 17, 17, 20, 14, 28, 18, 17, 42, 31, 31, 25, 7, 27, 21, 17, 17, 30, 23, 39, 30, 21, 22, 13, 21, 6, 28, 34, 23, 25, 27, 25, 41, 12, 13, 76, 30, 24, 65, 23, 15, 11, 19, 34, 13, 23, 35, 21, 13, 20, 28, 24, 17, 13, 6, 22, 7, 18, 8, 4, 8, 28, 17, 22, 17, 11, 11, 22, 32, 20, 34, 29, 12, 20, 24, 36, 22, 24, 28, 8, 9, 20, 15, 9, 17, 16, 12, 13, 30, 13, 17, 36, 10, 27, 29, 10, 25, 69, 20, 12, 24, 12, 13, 18, 7, 32, 27, 26, 47, 11, 42, 37, 36, 21, 32, 22, 14, 27, 26, 34, 20, 5, 31, 19, 9, 18, 46, 19, 14, 38, 10, 6, 26, 37, 32, 34, 41, 7, 13, 38, 33, 11, 12, 32, 14, 29, 10, 29, 18, 33, 32, 37, 12, 16, 26, 14, 18, 6
d074b	56, 73, 155, 21, 45, 19, 33, 24, 30, 31, 31, 17, 30, 20, 7, 6, 46, 16, 21, 8, 41, 43, 29, 25, 15, 13, 40, 39, 19, 33, 25, 24, 31, 31, 17, 32, 41, 34, 20, 7, 46, 16, 21, 12, 41, 44, 29, 25, 24, 20, 13, 40, 8, 21, 3, 8, 7, 16, 16, 27, 20, 21, 4, 22, 16, 12, 11, 27, 20, 14, 36, 30, 27, 42, 31, 25, 29, 9, 34, 40, 2, 8, 27, 8, 8, 3, 20, 9, 19, 7, 32, 13, 24, 15, 26, 5, 25, 5, 26, 32, 43, 27, 6, 20, 10, 24, 10, 11, 11, 12, 19, 23, 36, 12, 19, 9, 11, 17, 27, 4, 32, 7, 12, 3, 4, 28, 18, 3, 23, 8, 21, 4, 16, 8, 29, 26, 5, 5, 8, 19, 9, 10, 27, 20, 18, 7, 32, 14, 31, 30, 58, 43, 27, 30, 26, 15, 30, 28, 26, 2, 7, 7, 38, 11, 9, 12, 16, 33, 98, 23, 10, 3, 39, 23, 60, 36
d075b	50, 18, 25, 15, 9, 24, 21, 15, 24, 19, 22, 19, 13, 13, 19, 14, 27, 14, 10, 18, 22, 42, 27, 33, 32, 34, 31, 28, 41, 26, 29, 13, 3, 9, 22, 13, 21, 15, 21, 37, 30, 41, 19, 3, 20, 38, 16, 48, 19, 20, 26, 32, 35, 24, 11, 26, 14, 13, 25, 40, 23, 14, 10, 19, 36, 7, 36, 41, 29, 32, 31, 16, 21, 14, 19, 44, 29, 30, 38, 26, 24, 29, 25, 23, 12, 39, 33, 31, 9, 28, 19, 22, 34, 35, 34, 23, 9, 22, 31, 13, 10, 12, 3, 3, 17, 27, 35, 13, 36, 15, 17, 21, 20, 12, 19, 20, 14, 38, 34, 17, 26, 15, 27, 23, 6, 17, 41, 42, 14, 15, 37, 11, 18, 21, 33, 16, 22, 27, 25, 13, 46, 22, 77, 22, 35, 31, 16, 27, 29, 28, 25, 9, 21, 13, 21, 27, 30, 16, 66, 15, 6, 3, 39, 22, 17, 11, 11, 10, 44, 21, 22, 20, 14, 32, 43, 26, 22, 12, 14, 21, 18, 23, 24, 20, 32, 19, 35, 29, 32, 15, 14, 29, 26, 25, 13, 17, 22, 26, 14, 22, 25, 3, 26, 22, 22, 24, 26, 18, 3, 23, 28, 21, 9, 27, 18, 28, 33, 23, 38, 17, 22, 39, 26, 32, 23, 17, 36, 19, 33, 41, 32, 41, 32, 30, 31, 21, 16, 24, 19, 14, 12, 38, 34, 12, 36, 14, 30, 17, 28, 43, 22, 18, 18, 10, 29, 51, 51, 27, 31, 19, 6, 49, 24, 26, 17, 9, 35, 22, 11, 25, 14, 13, 43, 25
d076b	34, 28, 31, 22, 8, 7, 30, 27, 32, 55, 27, 49, 24, 37, 21, 3, 39, 11, 36, 32, 14, 12, 24, 22, 19, 26, 27, 19, 13, 3, 5, 18, 13, 14, 19, 10, 15, 21, 21, 13, 37, 15, 37, 39, 3, 21, 28, 45, 9, 39, 24, 20, 21, 35, 36, 41, 3, 37, 63, 3, 40, 12, 24, 18, 8, 6, 24, 29, 15, 13, 17, 18, 46, 31, 21, 17, 11, 11, 19, 5, 28, 17, 11, 17, 8, 19, 16, 14, 11, 40, 26, 11, 44, 17, 28, 23, 5, 14, 32, 15, 11, 16, 23, 21, 11, 33, 15, 8, 29, 7, 23, 23, 10, 20, 28, 29, 14, 13, 21, 3, 21, 29, 19, 41, 50, 29, 33, 15, 5, 29, 23, 22, 17, 12, 31, 20, 31, 29, 15, 41, 27, 14, 6, 44, 29, 22, 26, 34, 20, 10, 25, 25, 5, 24, 27, 17, 15, 25, 19, 24, 13, 41, 20, 26, 47, 6, 12, 52, 37, 3, 27, 29, 35, 31, 16, 23, 26, 29, 8, 35, 27, 7, 17, 11, 16, 10, 20, 55, 15, 10, 16, 12, 13, 25, 43, 12, 52, 6, 15, 20, 50, 3, 28, 11, 5, 7, 3, 18, 32, 10, 33, 29, 15, 7, 24, 23, 3, 26, 22, 11, 25, 26, 13, 3, 24, 23, 30, 25, 26, 23, 24, 8, 7, 4, 6, 4, 26, 13, 24, 25, 27, 23, 45, 27, 30, 15, 21, 32, 28, 43, 10, 21, 21, 12, 55, 11, 40, 19, 35, 38, 7, 16, 28, 12, 24, 25, 13, 4, 4, 39, 39, 5, 69, 17, 24, 34, 15, 12, 14, 22, 37, 7, 16, 30, 38, 13, 21, 17, 18, 18, 25, 56, 9, 20, 49, 26, 26, 22, 14, 18, 19, 23, 24, 7, 12, 24, 30, 30, 44, 23, 32, 46, 13, 34, 12, 40, 5, 36, 22, 23, 28
d077b	23, 9, 29, 25, 13, 23, 12, 23, 7, 28, 17, 14, 12, 19, 31, 25, 27, 33, 11, 35, 22, 11, 14, 21, 15, 22, 17, 15, 17, 11, 15, 6, 30, 13, 32, 10, 19, 16, 21, 26, 16, 9, 20, 17, 16, 18, 20, 18, 29, 14, 7, 9, 25, 19, 17, 21, 15, 13, 11, 23, 14, 12, 19, 12, 14, 27, 27, 32, 38, 10, 15, 3, 10, 19, 13, 16, 17, 14, 9, 17, 18, 24, 17, 3, 36, 19, 13, 25, 22, 17, 32, 26, 41, 36, 56, 17, 30, 36, 26, 22, 16, 3, 22, 20, 25, 14, 13, 3, 17, 23, 29, 38, 17, 6, 11, 5, 3, 37, 27, 42, 18, 3, 30, 13, 28, 16, 10, 17, 27, 14, 21, 19, 13, 20, 26, 27, 30, 21, 22, 32, 24, 35, 18, 27, 22, 32, 27, 20, 81, 40, 11, 36, 59, 44, 37, 35, 10, 31, 19, 45, 24, 25, 19, 21, 30, 84, 32, 30, 15, 30, 27, 37, 35, 70, 27, 27, 20, 31, 19, 20, 58, 42, 37, 117, 29, 15, 8, 11, 7, 7, 20, 17, 20, 12, 21, 24, 11, 55, 35, 28, 9, 8, 19, 18, 19, 22, 23, 9, 6, 40, 16, 15, 20, 16, 13, 34, 30, 36, 44, 30, 25, 24, 3, 13, 12, 14, 11, 25, 18, 21, 32, 20, 13, 13, 29, 26, 22, 20, 32, 29, 30, 37, 28, 14, 16, 19, 27, 8, 4, 7, 11, 29, 18, 23, 7, 23, 19, 13, 7, 9, 19, 7, 16, 9, 10, 22, 14, 13, 35, 31, 53, 27, 8, 24, 4, 5, 32, 25, 38, 22, 6, 16, 17, 6, 10, 12, 15, 11, 10, 31, 36, 23, 10, 23, 26, 17, 11, 7, 11, 18, 19, 11, 3, 6, 4, 10, 31, 6, 8, 11, 8, 28, 9, 31, 24, 16, 24, 35, 36, 38, 12, 7, 10
d078b	27, 27, 24, 22, 22, 5, 5, 32, 4, 5, 5, 19, 15, 10, 19, 23, 9, 21, 5, 1, 26, 24, 19, 8, 17, 2, 19, 4, 12, 10, 6, 7, 10, 18, 5, 36, 2, 2, 6, 22, 18, 17, 14, 14, 35, 39, 12, 27, 15, 9, 20, 2, 59, 4, 7, 7, 6, 15, 3, 2, 5, 2, 4, 8, 3, 4, 40, 17, 8, 6, 8, 6, 8, 6, 7, 6, 7, 6, 7, 6, 18, 8, 8, 7, 3, 2, 10, 8, 8, 10, 9, 7, 11, 10, 15, 7, 7, 3, 6, 10, 10, 7, 12, 2, 5, 12, 13, 2, 16, 8, 6, 9, 16, 15, 13, 13, 5, 23, 37, 29, 30, 10, 19, 11, 11, 34, 19, 13, 27, 11, 21, 32, 38, 24, 23, 28, 18, 22, 21, 17, 46, 35, 27, 14, 9, 15, 18, 35, 22, 3, 3, 3, 9, 30, 16, 3, 17, 10, 23, 6, 14, 22, 18, 18, 18, 11, 12, 11, 18, 41, 50, 50, 36, 15, 19, 30, 36, 5, 6, 23, 20, 15, 10, 29, 17, 18, 7, 23, 30, 27, 30, 32, 3, 10, 16, 19, 8, 9, 22, 27, 20, 9, 11, 13, 37, 20, 5, 22, 12, 16, 4, 6, 7, 11, 25, 26, 28, 15, 24, 8, 17, 21, 7, 7, 6, 7, 15, 11, 4, 5, 8, 18, 7, 4, 2, 9, 12, 6, 11, 11, 12, 11, 26, 19, 26, 12, 8, 28, 24, 24, 52, 25, 41, 32, 42, 6, 29, 22, 12, 27, 15, 32, 20, 15, 6, 18, 33, 5, 10, 8, 40, 20, 22, 13, 15, 28, 34, 71, 4, 6, 3, 6, 11, 43, 1, 33, 20, 18, 19, 25, 3, 6, 3, 7, 2, 19, 11, 46, 3, 14, 3, 35, 17, 28, 22, 20, 10, 23, 25, 24, 17, 14, 18, 29, 35, 12, 11, 18, 33, 21, 10, 15, 26, 27, 27, 17, 44, 9, 28, 3, 22, 6, 3, 36, 15, 14, 3
d079a	31, 17, 7, 12, 15, 23, 14, 16, 28, 32, 20, 2, 24, 7, 23, 20, 18, 18, 18, 34, 16, 17, 21, 18, 6, 21, 24, 25, 16, 37, 35, 24, 3, 18, 34, 24, 14, 27, 18, 23, 20, 17, 9, 24, 15, 27, 10, 24, 38, 17, 44, 36, 18, 13, 21, 36, 21, 3, 12, 36, 31, 41, 24, 23, 15, 20, 20, 3, 24, 32, 4, 6, 11, 3, 41, 25, 35, 29, 20, 36, 16, 16, 26, 34, 25, 33, 9, 3, 16, 28, 28, 29, 17, 10, 24, 22, 36, 18, 23, 17, 21, 20, 26, 7, 8, 25, 11, 11, 3, 12, 19, 7, 18, 5, 8, 11, 16, 18, 27, 11, 16, 16, 14, 31, 20, 9, 9, 20, 38, 11, 25, 31, 20, 18, 25, 15, 15, 16, 33, 33, 45, 12, 15, 35, 40, 32, 31, 21, 35, 7, 7, 37, 24, 15, 30, 21, 19, 25, 22, 30, 14, 5, 11, 27, 14, 5, 3, 17, 15, 41, 43, 19, 18, 24, 17, 30, 15, 20, 16, 20, 34, 30, 14, 9, 26, 8, 12, 6, 55, 26, 17, 32, 26, 20, 28, 35, 20, 12, 13, 15, 21, 15, 22, 19, 16, 23, 23, 28, 16, 10, 4, 10, 23, 29, 54, 15, 24, 35, 26, 30, 21, 31, 31, 6, 16, 17, 4, 29, 37, 24, 15, 29, 43, 17, 12, 7, 47, 20, 16, 19, 30, 38, 13, 35, 19, 13, 28, 17, 52, 11, 18, 14, 11, 34, 5, 6, 40, 34, 11, 3, 15, 27, 16, 3, 38, 33, 17, 4, 27, 35, 15, 32, 25, 9, 23, 21, 5, 9, 12, 25, 8, 39, 27, 20, 10, 25, 38, 25, 21, 16, 32, 29, 26, 14, 6, 14, 20, 20, 23, 18, 34, 10, 29, 7, 9, 11, 6, 39, 24, 18, 8, 30, 16, 19, 22, 35, 9, 15, 16, 29, 31, 9, 23, 26, 22, 28, 14, 18, 42, 7, 32, 6, 22, 22, 19, 19, 26, 19, 14, 14, 32, 28, 21
d080a	47, 34, 42, 14, 6, 16, 22, 57, 17, 7, 21, 10, 21, 19, 33, 33, 24, 24, 23, 19, 21, 26, 13, 21, 24, 32, 18, 19, 13, 10, 4, 11, 7, 13, 16, 20, 18, 19, 15, 43, 19, 9, 3, 31, 16, 7, 39, 17, 15, 20, 23, 4, 38, 33, 14, 23, 16, 26, 15, 3, 20, 13, 31, 31, 26, 7, 9, 15, 20, 4, 12, 9, 20, 25, 14, 5, 31, 24, 19, 27, 69, 48, 26, 22, 20, 11, 3, 13, 38, 29, 4, 8, 20, 18, 6, 29, 18, 16, 20, 18, 19, 11, 5, 22, 20, 3, 26, 8, 29, 15, 32, 22, 27, 21, 10, 13, 10, 11, 10, 5, 3, 8, 8, 12, 29, 16, 4, 15, 15, 13, 13, 14, 10, 13, 63, 160, 10, 12, 12, 62, 97, 50, 84, 30, 131, 138, 34, 24, 31, 28, 34, 21, 25, 29, 31, 25, 17, 15, 22, 6, 33, 21, 13, 12, 22, 20, 19, 49, 42, 26, 8, 15, 15, 14, 31, 15, 16, 24, 54, 18, 27, 15, 28, 48, 7, 3, 27, 29, 28, 21, 8, 27, 12, 51, 24, 3, 7, 42, 26, 22, 18, 15, 7, 15, 10, 40, 15, 7, 3, 13, 15, 3, 21, 18, 16, 9, 40, 44, 21, 40, 12, 18, 23, 34, 24, 24, 27, 13, 51, 20, 29, 14, 14, 19, 7, 17, 4, 9, 24, 29, 7, 44, 45, 30, 11, 17, 25, 29, 11, 27, 22, 3, 13, 4, 6, 38, 18, 8, 5, 12, 6, 14, 18, 16, 4, 24, 5, 13, 8, 9, 24, 29, 28, 4, 9, 11, 10, 8, 33, 39, 56, 10, 9, 2, 26, 22, 10, 9, 10, 17, 4, 18, 1, 2, 22, 13, 4, 23, 14, 8, 12, 4, 35, 35, 20, 59, 23, 13, 50, 28, 33, 34, 30, 15, 2, 2, 22, 60, 3, 24, 23, 44, 26, 21, 40, 19, 13, 19, 22, 57, 10, 5, 22, 28, 36, 9, 14, 22, 29, 44, 30, 21, 28, 66, 4, 4, 4, 215, 119, 96, 12, 33, 57, 19, 28, 37, 23, 7, 19, 28, 34, 63, 8, 23, 24, 2, 33, 17, 6, 32, 40, 11, 27, 20, 7, 9, 23, 47, 11, 12, 11, 11, 27, 16, 8, 13, 19, 15, 8, 10, 18, 6, 22, 13, 9, 11, 13, 9, 12, 8, 39, 9, 46, 23, 28, 9, 16, 31, 48, 17, 16, 21, 24, 16, 14, 35, 9, 2, 13, 17, 35, 23, 30, 37, 19, 3, 15, 25, 12, 3, 8, 15, 32, 21, 37, 26, 35, 36, 24, 47, 19, 6, 19, 22, 8, 20, 3, 17, 23, 16, 25, 14, 18, 12, 36, 32, 13, 24, 23, 19, 40, 31, 21, 23, 24, 21, 15, 55, 20, 4, 10, 7, 7, 13, 22, 34, 36, 24, 32, 46, 31, 24, 17, 17, 19, 8, 37, 19, 30, 11, 20, 34, 9, 16, 20, 16, 10, 19, 9, 12, 20, 17, 7, 12, 8, 10, 5, 23, 16, 19, 30, 55, 21, 20, 22, 24, 9, 32, 25, 11, 7, 8, 9, 8, 13, 70, 75, 41, 8, 18, 27, 46, 27, 31, 32, 33, 16, 39, 38, 38, 29, 27, 4, 36, 43, 19, 10, 23
d081a	31, 25, 31, 31, 10, 20, 29, 16, 30, 10, 23, 35, 24, 46, 33, 13, 12, 13, 34, 27, 28, 21, 13, 23, 20, 26, 12, 14, 13, 23, 25, 10, 13, 22, 20, 26, 33, 18, 20, 13, 31, 7, 21, 16, 24, 21, 24, 21, 37, 19, 16, 34, 19, 14, 40, 29, 16, 49, 20, 20, 40, 3, 20, 30, 58, 25, 15, 27, 24, 17, 19, 29, 14, 21, 26, 9, 26, 28, 3, 8, 30, 8, 33, 23, 22, 25, 18, 33, 18, 15, 41, 11, 10, 15, 21, 30, 25, 46, 17, 11, 19, 11, 24, 28, 20, 28, 26, 32, 16, 24, 21, 37, 34, 29, 18, 20, 20, 27, 33, 15, 24, 51, 36, 18, 24, 19, 36, 12, 14, 8, 12, 18, 38, 3, 36, 20, 25, 23, 8, 30, 6, 18, 21, 16, 20, 41, 28, 24, 20, 19, 16, 18, 28, 38, 38, 32, 30, 37, 23, 12, 31, 30, 21, 42, 52, 20, 17, 18, 32, 25, 17, 26, 26, 9, 20, 21, 46, 7, 17, 34, 16, 29, 8, 9, 8, 13, 34, 15, 17, 30, 49, 21, 11, 36, 13, 55, 32, 18, 14, 25, 7, 6, 2, 16, 9, 8, 2, 3, 6, 14, 30, 4, 50, 11, 4, 22, 26, 11, 2, 20, 44, 18, 60, 36, 42, 26, 48, 36, 30, 22, 31, 27, 30, 39, 7, 27, 8, 23, 27, 37, 21, 26, 15, 10, 31, 37, 30, 24, 17, 18, 7, 17, 38, 9, 33, 14, 31, 47, 13, 35, 18, 28, 16, 15, 16, 34, 20, 34, 38, 46, 32, 31, 14, 37, 16, 27, 7, 32, 22, 26, 34, 39, 12, 17, 18, 23, 40, 16, 21, 30, 16
d082a	30, 34, 48, 16, 30, 49, 31, 49, 37, 31, 12, 16, 33, 25, 18, 30, 22, 34, 20, 49, 31, 3, 4, 6, 43, 40, 30, 3, 12, 34, 16, 3, 18, 44, 10, 5, 10, 10, 22, 3, 13, 36, 22, 24, 40, 29, 14, 11, 3, 35, 35, 21, 40, 35, 11, 38, 40, 30, 31, 35, 23, 36, 20, 22, 8, 24, 41, 36, 30, 41, 14, 33, 27, 11, 30, 30, 21, 34, 23, 18, 16, 18, 18, 20, 41, 13, 15, 9, 33, 36, 47, 51, 21, 18, 22, 12, 19, 21, 31, 41, 40, 15, 17, 15, 15, 13, 21, 36, 5, 17, 45, 17, 46, 14, 9, 39, 38, 22, 55, 3, 34, 17, 38, 7, 40, 39, 21, 30, 34, 41, 8, 20, 36, 40, 9, 40, 18, 16, 44, 14, 49, 25, 18, 25, 16, 8, 33, 51, 6, 20, 14, 30, 14, 34, 40, 22, 41, 48, 38, 20, 31, 56, 16, 31, 24, 23, 34, 33, 30, 19, 4, 21, 17, 47, 47, 20, 12, 35, 30, 35, 41, 14, 16, 21, 13, 15, 15, 39, 15, 11, 32, 24, 30, 20, 45, 22, 6, 3, 22, 17, 16, 24, 16, 22, 36, 33, 36, 14, 25, 16, 32, 25, 26, 18, 3, 14
d083a	30, 12, 28, 28, 22, 34, 33, 18, 19, 33, 24, 27, 8, 32, 28, 18, 14, 9, 24, 41, 19, 11, 19, 30, 22, 29, 17, 36, 10, 28, 22, 28, 17, 38, 16, 11, 12, 26, 10, 11, 16, 21, 30, 12, 33, 12, 20, 10, 25, 12, 3, 22, 47, 29, 19, 3, 80, 22, 8, 15, 24, 11, 21, 31, 24, 40, 25, 14, 23, 35, 29, 18, 17, 32, 14, 11, 27, 23, 20, 8, 28, 23, 18, 9, 11, 21, 14, 9, 16, 19, 20, 14, 13, 18, 12, 15, 10, 36, 25, 18, 54, 14, 38, 12, 64, 27, 15, 29, 22, 52, 14, 11, 33, 29, 20, 7, 27, 44, 25, 32, 39, 44, 32, 44, 10, 29, 11, 26, 32, 15, 9, 25, 19, 11, 7, 8, 26, 34, 33, 51, 17, 30, 5, 39, 34, 23, 20, 20, 31, 21, 16, 20, 27, 22, 18, 18, 31, 21, 17, 28, 7, 3, 14, 22, 18, 16, 54, 13, 24, 8, 23, 14, 32, 28, 16, 27, 17, 27, 24, 39, 37, 31, 30, 16, 18, 21, 30, 12, 33, 12, 20, 10, 12, 3, 25, 22, 47, 28, 19, 3, 80, 22, 8, 15, 24, 11, 21, 31, 26
d084a	36, 24, 35, 37, 55, 22, 35, 22, 35, 3, 24, 39, 23, 59, 4, 12, 37, 9, 17, 19, 23, 20, 13, 21, 19, 20, 8, 9, 10, 14, 7, 29, 17, 15, 13, 36, 57, 2, 17, 13, 4, 10, 22, 22, 21, 10, 35, 46, 16, 16, 7, 33, 43, 26, 43, 43, 31, 18, 9, 16, 36, 2, 35, 41, 14, 31, 21, 16, 9, 21, 37, 37, 11, 14, 16, 12, 2, 23, 11, 34, 16, 18, 19, 34, 33, 43, 26, 35, 25, 28, 4, 9, 16, 37, 12, 28, 26, 31, 30, 7, 29, 31, 11, 16, 35, 16, 17, 5, 4, 3, 14, 22, 35, 24, 15, 23, 14, 3, 15, 23, 40, 7, 22, 19, 24, 22, 17, 21, 29, 2, 36, 2, 44, 20, 33, 31, 37, 31, 39, 6, 17, 7, 2, 13, 23, 13, 18, 2, 6, 30, 12, 18, 17, 13, 28, 17, 20, 12, 29, 17, 18, 19, 25, 2, 29, 14, 42, 2, 13, 19, 33, 25, 24, 17, 19, 12, 36, 9, 7, 10, 11, 5, 46, 5, 25, 14, 8, 13, 22, 32, 11, 12, 3, 22, 40, 36, 19, 17, 20, 25, 39, 25, 12, 28, 21, 2, 18, 3, 25, 26, 21, 24, 40, 17, 16, 15, 12, 17, 29, 25, 19, 21, 13, 8, 18, 27, 22, 40, 25, 25, 40, 26, 11, 20, 37, 5, 4, 24, 35, 20, 4, 23, 38, 8, 9, 6, 11, 14, 25, 36, 40, 28, 25, 40, 34, 8, 22, 51, 31, 19, 19, 25, 17, 41, 20, 14, 39, 25, 24, 12, 38, 18, 12, 3, 45, 19, 43, 33, 14, 8, 16, 32, 10, 12, 26, 14, 11, 30, 11, 16, 11, 22, 14, 28, 6, 3, 12, 12, 11, 3, 31, 20, 15, 11, 13, 2, 33, 18, 41, 18, 27, 25, 31, 34, 34, 11, 14, 5, 35, 10, 14, 22, 8, 9, 6, 14, 13, 3, 17, 7, 16, 10, 10, 36, 12, 19, 44, 9, 2, 12, 23, 27, 9, 22, 45, 55, 25, 57, 2, 24, 18, 4, 30, 20, 14, 35, 2, 19, 2, 56, 9, 40, 20, 19, 20, 34, 39, 27, 2, 16, 16, 25, 21, 24, 42, 16, 14, 15, 12, 17, 29, 19, 21, 8, 18, 27, 26, 7, 28, 3, 4, 26, 28, 35, 15
d085d	34, 11, 26, 19, 12, 16, 11, 27, 12, 12, 33, 14, 20, 26, 35, 14, 11, 23, 18, 12, 9, 27, 36, 28, 16, 27, 17, 22, 8, 6, 18, 26, 15, 32, 20, 22, 48, 13, 18, 38, 8, 19, 34, 16, 29, 20, 14, 32, 22, 29, 35, 14, 36, 28, 15, 15, 29, 20, 38, 34, 6, 36, 30, 27, 27, 14, 9, 8, 32, 22, 21, 32, 17, 3, 31, 36, 36, 28, 8, 47, 25, 33, 35, 21, 3, 12, 24, 31, 18, 3, 18, 21, 41, 30, 11, 9, 27, 3, 28, 33, 3, 19, 13, 20, 32, 16, 13, 11, 16, 7, 13, 11, 15, 9, 10, 20, 12, 10, 11, 14, 18, 8, 22, 15, 19, 16, 9, 18, 17, 5, 14, 21, 14, 20, 6, 13, 20, 17, 20, 18, 10, 4, 15, 25, 13, 9, 8, 23, 18, 16, 2, 35, 27, 9, 32, 21, 19, 21, 23, 9, 26, 42, 31, 14, 17, 12, 48, 15, 14, 28, 39, 49, 20, 11, 9, 3, 14, 36, 35, 10, 27, 21, 21, 24, 17, 13, 6, 22, 20, 18, 16, 12, 22, 17, 36, 18, 26, 32, 40, 25, 26, 68, 12, 47, 16, 32, 29, 54, 42, 29, 20, 24, 19, 16, 37, 26, 10, 21, 37, 13, 23, 19, 31, 17, 20, 21, 16, 35, 21, 30, 20, 29
d086d	15, 11, 15, 20, 23, 36, 38, 23, 21, 53, 30, 18, 35, 35, 39, 17, 34, 22, 32, 12, 19, 18, 22, 12, 16, 15, 36, 17, 10, 24, 13, 29, 34, 12, 29, 12, 11, 22, 15, 17, 5, 31, 25, 25, 24, 16, 22, 29, 24, 22, 10, 18, 15, 28, 38, 15, 7, 12, 32, 27, 23, 23, 12, 18, 22, 24, 22, 8, 6, 5, 14, 28, 17, 13, 10, 25, 23, 10, 24, 25, 22, 23, 27, 12, 30, 24, 26, 9, 17, 10, 29, 22, 20, 30, 28, 14, 31, 23, 17, 15, 38, 19, 25, 12, 23, 23, 8, 24, 18, 16, 24, 3, 11, 26, 24, 25, 27, 24, 95, 32, 11, 10, 7, 17, 9, 16, 10, 6, 11, 23, 18, 27, 22, 18, 14, 15, 22, 16, 12, 11, 20, 18, 21, 33, 21, 16, 22, 14, 24, 27, 21, 26, 8, 11, 25, 18, 15, 11, 19, 22, 28, 31, 17, 24, 31, 45, 16, 25, 31, 24, 8, 20, 19, 8, 11, 17, 22, 17, 26, 29, 37, 20, 33, 34, 28, 10, 23, 17, 25, 23, 21, 20, 25, 24, 16, 25, 18, 19, 8, 3, 11, 18, 2, 2, 6, 8, 23, 32, 31, 40, 42, 30, 23, 41, 22, 22, 11, 11, 21, 20, 16, 38, 25, 40, 37, 36, 64, 24, 37, 86, 72, 20, 48, 23, 12, 17, 16, 20, 36, 10, 29, 14, 20, 29, 25, 14, 17, 20, 14, 37, 23, 33, 36, 20, 13, 25, 16, 20, 22, 8, 24, 8, 15, 31, 25, 7, 6, 35, 33, 16, 30, 33, 14, 19, 21, 22, 29, 38, 11, 30, 22, 23, 23, 8, 18, 16, 32, 11, 22, 26, 24, 17, 9, 24, 18, 22, 18, 16, 15, 24, 16, 13, 11, 21, 18, 21, 33, 14, 114, 25, 26, 14, 18, 17, 7, 7, 40, 40, 28, 14, 20, 14, 30, 24, 33, 26, 9, 44, 19, 23, 38, 38, 13, 29, 37, 18, 18, 18, 10, 22, 20, 22, 32, 20, 19, 31, 19, 12, 9, 3, 21, 6, 39, 24, 3, 10, 15, 17, 22, 18, 5, 21, 13, 16, 33, 17, 3, 19, 25, 15, 6, 24, 20, 22, 39, 22, 10, 28, 29, 26, 15, 39, 22, 42, 33, 42, 12, 37, 35, 31, 16, 20, 23, 24, 4
d087d	32, 5, 17, 9, 8, 12, 4, 16, 2, 15, 28, 26, 19, 34, 13, 29, 7, 20, 14, 18, 24, 25, 22, 33, 43, 22, 20, 16, 16, 40, 11, 14, 38, 23, 8, 8, 19, 18, 16, 9, 27, 31, 22, 19, 23, 29, 27, 17, 40, 18, 23, 5, 5, 23, 43, 30, 7, 38, 25, 22, 18, 35, 38, 34, 15, 14, 28, 25, 25, 19, 69, 20, 14, 35, 72, 6, 61, 8, 38, 30, 17, 41, 25, 16, 8, 34, 28, 32, 28, 12, 42, 53, 18, 27, 18, 13, 14, 41, 38, 19, 10, 6, 26, 3, 33, 34, 18, 45, 29, 27, 26, 8, 35, 34, 20, 14, 27, 36, 30, 14, 28, 29, 3, 9, 21, 35, 8, 3, 7, 27, 28, 7, 14, 29, 19, 32, 24, 29, 4, 11, 16, 5, 17, 17, 7, 39, 40, 29, 26, 25, 33, 20, 13, 17, 26, 43, 70, 18, 28, 10, 17, 8, 12, 31, 25, 20, 26, 11, 15, 16, 34, 21, 16, 17, 30, 16, 9, 40, 30, 12, 25, 15, 22, 24, 8, 14, 13, 8, 12, 29, 26, 18, 19, 8, 15, 19, 19, 6, 15, 13, 17, 27, 12, 25, 9, 16, 7, 34, 31, 40, 20, 17, 25, 15, 3, 16, 32, 12, 3, 20, 45, 24, 21, 20, 28, 13, 14, 16, 11, 16, 22, 25, 39, 25, 17, 14, 20, 23, 19, 38, 20, 27, 20, 8, 25, 29, 27, 21, 17, 39, 28, 40, 12, 30, 22, 51, 8, 21, 27, 30, 33, 18, 14, 23, 35, 44, 25, 12, 39, 32, 36, 20, 54, 15, 11, 16, 32, 13, 29, 13, 15, 37, 5, 20, 30, 29, 9, 23, 21, 15, 5, 12, 16
d089d	30, 12, 28, 28, 22, 34, 33, 18, 19, 33, 24, 27, 8, 32, 28, 18, 14, 9, 24, 41, 19, 11, 19, 30, 22, 29, 17, 36, 10, 28, 22, 28, 17, 38, 16, 11, 12, 26, 10, 11, 16, 21, 30, 12, 33, 12, 20, 10, 25, 12, 3, 22, 47, 29, 19, 3, 80, 22, 8, 15, 24, 11, 21, 31, 24, 40, 25, 14, 34, 36, 20, 25, 12, 31, 14, 27, 11, 35, 31, 24, 68, 35, 23, 28, 14, 13, 27, 13, 11, 20, 2, 30, 35, 15, 27, 12, 21, 33, 45, 26, 22, 18, 14, 19, 58, 19, 3, 20, 15, 33, 14, 3, 15, 16, 16, 25, 19, 30, 24, 13, 7, 8, 12, 11, 17, 7, 31, 21, 24, 51, 22, 7, 5, 43, 19, 26, 20, 23, 11, 27, 16, 12, 8, 19, 11, 14, 14, 10, 19, 8, 11, 14, 22, 20, 14, 13, 12, 3, 12, 23, 35, 29, 18, 17, 32, 14, 11, 27, 23, 20, 8, 28, 23, 18, 9, 11, 21, 14, 9, 16, 19, 20, 14, 13, 18, 12, 15, 10, 36, 25, 18, 54, 14, 38, 12, 64, 27, 15, 29, 22, 52, 14, 11, 33, 29, 20, 7, 27, 44, 25, 32, 39, 44, 32, 44, 10, 29, 11, 26, 32, 15, 9, 25, 19, 11, 7, 8, 26, 34, 33, 51, 17, 30, 5, 39, 34, 23, 20, 20, 31, 21, 16, 20, 27, 22, 18, 18, 31, 21, 17, 28, 7, 3, 14, 23, 18, 16, 54, 13, 24, 8, 23, 14, 32, 28, 16, 27, 17, 27, 24, 37, 28, 13, 33, 21, 32, 15, 23, 9, 36, 19, 23, 15, 35, 22, 14, 23, 15, 26, 31, 11, 18, 3, 12, 11, 24, 24, 10, 17, 22, 27, 15, 17, 37, 4, 33, 27, 9, 12, 7, 28, 12, 21, 3, 40, 15, 15, 20, 22, 18, 4, 16, 33, 15, 17, 9, 15, 34, 25, 9, 11, 14, 12, 20, 22, 32, 11, 9, 23, 7, 3, 11, 15, 16, 13, 9, 39, 37, 31, 30, 16, 18, 21, 30, 12, 33, 12, 20, 10, 12, 3, 25, 22, 62, 33, 19, 3, 80, 22, 8, 15, 24, 11, 21, 31, 26
d090d	30, 31, 28, 14, 7, 27, 18, 8, 24, 22, 43, 28, 6, 15, 36, 28, 17, 31, 24, 23, 17, 18, 32, 16, 9, 47, 14, 3, 16, 29, 50, 3, 36, 20, 21, 15, 27, 19, 20, 31, 27, 37, 17, 16, 43, 31, 22, 30, 3, 16, 34, 19, 28, 19, 35, 39, 3, 9, 5, 50, 9, 15, 6, 19, 14, 15, 16, 33, 22, 10, 20, 25, 32, 38, 31, 36, 45, 21, 3, 23, 34, 20, 15, 28, 21, 35, 44, 16, 14, 28, 30, 22, 34, 21, 12, 26, 26, 15, 26, 2, 15, 14, 17, 22, 11, 1, 1, 9, 20, 7, 18, 26, 26, 19, 29, 8, 20, 24, 14, 14, 25, 27, 19, 2, 9, 28, 15, 11, 2, 29, 38, 27, 25, 26, 20, 34, 6, 23, 21, 25, 16, 18, 37, 28, 11, 28, 37, 25, 19, 34, 8, 33, 31, 28, 12, 20, 31, 32, 31, 32, 22, 25, 17, 8, 13, 31, 66, 36, 19, 12, 15, 16, 17, 68, 28, 34, 14, 5, 59, 15, 18, 25, 17, 21, 13, 6, 25, 19, 38, 33, 28, 47, 21, 21, 42, 29, 19, 15, 42, 24, 22, 11, 14, 8, 3, 6, 14, 2, 2, 13, 23, 19, 14, 14, 23, 13, 11, 30, 19, 24, 22, 17, 21, 17, 15, 16, 11, 27, 19, 16, 35, 24, 24, 16, 25, 32, 34, 23, 9, 10, 24, 23, 11, 4, 4, 6, 8
d091c	33, 23, 10, 17, 12, 12, 11, 27, 14, 17, 13, 17, 15, 13, 9, 20, 6, 24, 17, 16, 21, 14, 18, 20, 20, 3, 17, 31, 31, 19, 9, 12, 3, 25, 32, 41, 25, 28, 18, 17, 33, 13, 10, 7, 14, 11, 17, 24, 16, 16, 11, 16, 13, 10, 39, 20, 23, 24, 16, 27, 31, 22, 20, 30, 29, 22, 15, 16, 34, 37, 24, 28, 14, 7, 9, 5, 3, 21, 9, 11, 3, 27, 24, 20, 6, 25, 29, 14, 15, 35, 10, 18, 10, 39, 28, 37, 13, 34, 40, 38, 14, 10, 24, 9, 25, 23, 15, 13, 23, 8, 8, 3, 17, 22, 18, 35, 20, 6, 18, 27, 13, 21, 12, 10, 12, 20, 16, 3, 7, 46, 24, 22, 15, 40, 21, 29, 22, 25, 13, 25, 8, 15, 36, 25, 11, 16, 3, 34, 17, 15, 34, 22, 22, 3, 31, 5, 37, 11, 6, 24, 49, 23, 23, 17, 14, 11, 31, 3, 9, 14, 7, 15, 12, 21, 15, 26, 16, 21, 23, 18, 8, 9, 20, 17, 18, 41, 8, 13, 16, 16, 8, 38, 16, 11, 20, 18, 23, 23, 12, 28, 44, 8, 18, 6, 38, 16, 15, 44, 30, 11, 3, 5, 3, 9, 21, 9, 17, 3, 22, 55, 15, 20, 3, 26, 20, 34, 22, 18, 9, 3, 30, 33, 24, 10, 17, 12, 12, 11, 27, 15, 13, 9, 20, 6, 24, 17, 16, 21, 14, 18, 20, 20, 3, 17, 31, 31, 5, 37, 11, 27, 23, 30, 17, 11, 11, 31, 3, 9, 13, 7, 15, 12, 23, 15, 26, 16, 21, 9, 21, 24, 18, 8, 9, 20, 17, 18

Figure 9: DUC 2002 multi-document (MDOC) statistics Table, together with word count in each sentence for sets d092 to d120.

MDOC	#Words
d092c	30, 30, 37, 34, 19, 27, 16, 29, 43, 28, 21, 19, 21, 17, 21, 11, 21, 32, 28, 19, 21, 24, 36, 32, 21, 19, 21, 35, 18, 10, 28, 21, 26, 29, 15, 27, 25, 22, 11, 29, 17, 25, 10, 20, 21, 17, 2, 14, 30, 31, 21, 27, 30, 45, 50, 39, 21, 32, 28, 34, 8, 42, 16, 22, 18, 14, 10, 39, 30, 23, 24, 11, 9, 19, 6, 11, 9, 28, 22, 39, 24, 23, 28, 22, 11, 6, 14, 29, 11, 25, 27, 19, 29, 12, 24, 25, 6, 19, 21, 8, 16, 25, 33, 8, 16, 14, 31, 19, 13, 24, 20, 23, 11, 19, 18, 10, 12, 14, 16, 20, 34, 30, 11, 16, 20, 35, 15, 21, 12, 22, 34, 12, 30, 39, 19, 17, 22, 19, 28, 15, 26, 14, 15, 20, 35, 21, 26, 12, 24, 17, 9, 30, 19, 11, 29, 12, 21, 8, 4, 8, 32, 34, 43, 34, 23, 34, 18, 27, 15, 22, 16, 42, 44, 22, 24, 30, 30, 12, 3, 16, 38, 13, 29, 18, 34, 14, 37, 17, 9, 88, 13, 9, 68, 27, 12, 14, 8, 22, 13, 10, 34, 28, 35, 18, 22, 18, 14, 15, 18, 27, 26, 13, 15, 31, 11, 23, 12, 28, 17, 17, 24, 21, 20, 18, 18, 17, 28
d093c	26, 12, 15, 7, 31, 12, 14, 22, 8, 14, 24, 15, 21, 34, 21, 3, 6, 16, 18, 10, 12, 8, 12, 3, 13, 24, 20, 19, 20, 15, 14, 6, 15, 28, 14, 8, 13, 14, 8, 15, 8, 14, 17, 16, 16, 13, 12, 17, 24, 11, 18, 24, 33, 15, 21, 28, 23, 31, 28, 26, 27, 17, 26, 33, 11, 17, 33, 29, 14, 23, 11, 35, 22, 44, 40, 30, 19, 18, 34, 15, 21, 23, 19, 13, 13, 22, 11, 15, 25, 8, 20, 33, 39, 48, 17, 19, 23, 28, 18, 9, 9, 16, 17, 16, 16, 20, 8, 33, 21, 10, 12, 13, 21, 31, 21, 16, 13, 27, 9, 3, 20, 12, 21, 27, 14, 14, 39, 49, 29, 23, 30, 29, 24, 19, 32, 18, 27, 7, 12, 22, 33, 9, 15, 36, 12, 20, 33, 12, 24, 29, 17, 24, 26, 27, 34, 19, 20, 15, 22, 12, 11, 46, 31, 55, 7, 41, 39, 31, 35, 26, 29, 31, 15, 26, 25, 14, 34, 33, 28, 8, 18, 30, 11, 18, 22, 17, 17, 12, 3, 20, 38, 29, 25, 15, 10, 18, 24, 19, 19, 17, 3, 17, 36, 16, 7, 21, 30, 27, 19, 32, 29
d094c	28, 26, 47, 32, 3, 29, 37, 13, 17, 35, 17, 35, 21, 18, 34, 16, 29, 31, 16, 14, 4, 13, 25, 17, 11, 10, 12, 25, 28, 43, 39, 14, 9, 13, 57, 5, 26, 25, 53, 16, 40, 41, 16, 27, 38, 11, 14, 29, 17, 31, 37, 18, 37, 30, 18, 19, 57, 27, 12, 36, 10, 15, 47, 17, 18, 6, 35, 12, 51, 9, 30, 19, 7, 22, 23, 24, 9, 26, 33, 37, 32, 12, 15, 9, 23, 26, 24, 36, 30, 36, 9, 32, 6, 19, 10, 71, 13, 22, 12, 15, 26, 3, 38, 24, 3, 19, 12, 20, 22, 30, 36, 25, 20, 17, 18, 3, 32, 28, 12, 22, 12, 15, 21, 14, 20, 15, 22, 13, 15, 15, 7, 35, 31, 11, 27, 46, 36, 26, 4, 36, 18, 17, 11, 32, 30, 15, 18, 43, 31, 38, 78, 4, 7, 6, 14, 3, 39, 24, 33, 23, 28, 9, 27, 24, 10, 37, 28, 27, 38, 39, 18, 20, 43, 31, 25, 15, 23, 19, 7, 30, 15, 20, 26, 13, 11, 11, 10, 30, 19, 34, 10, 45, 34, 19, 30, 29, 70, 23, 27, 50, 1
d095c	50, 8, 39, 51, 39, 18, 26, 20, 14, 35, 26, 10, 37, 16, 4, 17, 28, 17, 9, 17, 25, 13, 57, 54, 29, 63, 4, 5, 23, 16, 9, 11, 4, 8, 23, 21, 37, 18, 14, 32, 15, 19, 16, 12, 20, 33, 21, 41, 30, 16, 10, 6, 28, 7, 73, 24, 10, 18, 12, 26, 25, 33, 16, 12, 25, 14, 25, 17, 9, 11, 36, 20, 13, 31, 20, 11, 9, 3, 26, 22, 39, 12, 3, 10, 33, 29, 16, 6, 6, 29, 10, 46, 11, 32, 3, 40, 6, 5, 13, 38, 9, 23, 19, 9, 4, 12, 15, 22, 20, 11, 24, 6, 15, 15, 18, 15, 18, 24, 4, 15, 15, 46, 10, 34, 28, 24, 14, 29, 17, 28, 25, 33, 28, 8, 15, 23, 17, 36, 26, 66, 45, 27, 18, 50, 17, 10, 9, 9, 23, 26, 10, 16, 23, 46, 12, 17, 15, 3, 29, 8, 27, 22, 21, 14, 15, 17, 47, 13, 6, 24, 37, 40, 27, 32, 28, 29, 24, 27, 25, 21, 13, 10, 18, 21, 14, 7, 9, 24, 17, 17, 22, 22, 13, 13, 15, 18, 34, 10, 3, 10, 41, 51, 21, 34, 9, 3, 37, 9, 23, 26, 17, 28, 70, 39, 21, 5, 3, 33, 12, 25, 35, 15, 3, 20, 16, 31, 23, 3, 47, 69, 17, 22, 18, 40, 27, 11, 27, 32, 49, 35, 11, 12, 9, 11
d096c	23, 16, 26, 10, 30, 32, 13, 18, 21, 24, 19, 20, 13, 18, 16, 13, 10, 29, 26, 11, 12, 21, 9, 10, 22, 10, 5, 14, 33, 14, 19, 9, 10, 17, 9, 31, 23, 24, 35, 11, 37, 46, 20, 23, 7, 27, 9, 16, 8, 26, 13, 12, 16, 35, 14, 15, 38, 29, 11, 13, 23, 17, 7, 18, 31, 21, 47, 17, 13, 40, 18, 28, 11, 11, 24, 7, 23, 30, 13, 48, 28, 2, 17, 11, 9, 42, 31, 23, 22, 25, 23, 16, 25, 31, 27, 15, 13, 9, 7, 17, 26, 11, 11, 11, 10, 10, 11, 11, 11, 11, 2, 36, 33, 11, 17, 40, 25, 3, 9, 21, 40, 23, 31, 25, 33, 18, 33, 7, 58, 17, 34, 17, 29, 15, 32, 54, 38, 20, 2, 2, 26, 15, 35, 6, 5, 8, 9, 33, 14, 16, 36, 22, 23, 7, 15, 33, 36, 24, 13, 21, 12, 25, 22, 7, 35, 45, 32, 10, 3, 46, 15, 34, 25, 11, 21, 15, 13, 27, 20, 32, 23, 32, 49, 32, 35, 35, 19, 29, 22, 48, 14, 14, 20, 46, 23, 12, 25, 23, 23, 36, 28, 12, 18, 44, 27, 6, 9, 14, 18, 17, 33, 16
d097e	34, 11, 26, 19, 12, 16, 11, 27, 12, 12, 33, 14, 20, 26, 35, 14, 11, 23, 18, 12, 9, 27, 36, 28, 16, 27, 17, 22, 8, 6, 18, 26, 15, 32, 20, 22, 48, 13, 18, 38, 8, 19, 34, 16, 29, 20, 14, 32, 22, 29, 35, 14, 36, 28, 15, 15, 23, 34, 13, 31, 28, 23, 17, 29, 13, 21, 29, 32, 9, 8, 16, 3, 8, 29, 8, 28, 8, 47, 25, 33, 35, 21, 3, 12, 24, 31, 18, 3, 18, 21, 41, 30, 11, 9, 27, 3, 28, 33, 3, 19, 13, 20, 36, 35, 11, 27, 21, 21, 24, 17, 13, 6, 22, 20, 18, 16, 12, 22, 17, 27, 23, 23, 18, 37, 5, 12, 13, 3, 16, 22, 24, 20, 23, 31, 41, 52, 36, 16, 38, 32, 14, 18, 19, 8, 16, 12, 27, 16, 17, 12, 8, 32, 35, 10, 32, 12, 26, 36, 18, 26, 32, 40, 25, 26, 68, 12, 47, 16, 32, 29, 54, 42, 29, 20, 24, 19, 16, 37, 26, 10, 21, 37, 13, 37, 38, 34, 30, 26, 26, 23, 38, 8, 15, 40, 21, 49, 29, 10, 9, 70, 15, 45, 24, 44, 21, 44, 38, 26, 17, 14, 24, 24, 17, 21, 59, 15, 35, 23, 19, 31, 17, 20, 21, 16, 35, 21, 30, 20, 29
d098e	36, 14, 3, 23, 9, 23, 36, 6, 9, 10, 8, 6, 3, 17, 14, 27, 22, 30, 31, 18, 26, 21, 22, 33, 19, 44, 14, 3, 27, 25, 24, 17, 36, 24, 21, 18, 23, 20, 3, 14, 33, 22, 30, 27, 29, 2, 23, 23, 39, 20, 25, 36, 8, 45, 23, 26, 28, 23, 12, 30, 19, 26, 30, 13, 17, 24, 27, 28, 16, 21, 20, 21, 10, 18, 25, 51, 12, 24, 32, 23, 15, 3, 16, 24, 21, 20, 9, 25, 3, 13, 37, 13, 35, 36, 15, 18, 19, 33, 23, 6, 17, 30, 12, 23, 27, 20, 23, 37, 25, 33, 24, 19, 31, 34, 23, 20, 5, 17, 16, 17, 52, 14, 32, 37, 12, 28, 15, 7, 11, 6, 22, 30, 25, 6, 22, 15, 6, 5, 16, 20, 9, 4, 11, 37, 35, 25, 14, 26, 17, 13, 2, 46, 3, 7, 4, 2, 24, 8, 14, 38, 32, 23, 12, 40, 21, 32, 17, 22, 26, 14, 13, 26, 34, 7, 11, 2, 15, 35, 19, 17, 11, 24, 15, 30, 41, 24, 4, 5, 5, 5, 31, 43, 27, 41, 15, 18, 40, 3, 20, 19, 31, 19, 6, 38, 24, 22, 40, 20, 29, 42, 39, 29, 19, 4, 22, 6, 14, 23, 6, 43, 6, 18, 3, 16, 34, 19, 12, 32, 10, 6, 4, 15, 22, 27, 22, 3, 5, 5, 5, 21, 31, 17, 18, 10, 29, 5, 15, 22, 34, 21, 23, 12, 18, 27, 30, 18, 15, 25, 9, 33, 22, 22, 15, 32, 27, 22, 38, 35, 27, 13, 15, 24, 29, 22, 12, 28, 15, 16, 35, 14, 7, 5, 15, 8, 17, 11, 32, 7, 22, 4, 23, 7, 34, 30, 29, 24, 9, 26, 25, 30, 11, 29, 39, 9, 29, 12, 37, 10, 18, 17, 36, 21, 20, 18, 20, 8, 32, 32, 18, 19, 3, 15, 18, 16, 20, 13, 20, 21, 11, 12, 35, 30, 29, 42, 3, 21, 26, 35, 27, 31, 12, 14, 25, 10, 24, 33, 12, 36, 15, 12, 13, 33, 14, 20, 23, 7, 6, 9, 32, 35, 19, 32, 16, 12, 21, 21, 13, 26, 25, 11, 20, 19, 31, 40, 16, 10, 24, 19, 27, 10, 11, 2, 16, 29, 27, 13, 14, 34, 19, 21, 24, 25, 10, 24, 33, 35, 25, 8, 6, 8, 10, 10, 19, 5, 9, 3, 13, 13, 3, 13, 10, 7, 32, 28, 33, 21, 10, 17, 30, 39, 26, 32, 16, 29, 30, 16, 16, 17, 21, 42, 3, 29, 12, 25, 14, 9, 3, 21, 21, 32, 18, 26, 39, 17, 22, 29, 30, 45, 14, 38, 23, 27, 35, 35, 20, 9, 16, 35, 14, 49, 15, 36, 20, 7, 42, 45, 17, 4, 10, 39, 9, 19, 14, 13, 19, 13, 7, 2, 16, 21, 20, 17, 29, 26, 23, 28, 29, 31, 24, 10, 33, 26, 12, 10, 42, 17, 11, 28
d099e	32, 35, 15, 15, 34, 18, 9, 16, 10, 24, 20, 8, 3, 22, 11, 16, 15, 26, 16, 16, 16, 19, 23, 13, 8, 8, 11, 6, 6, 3, 9, 43, 34, 26, 10, 33, 9, 27, 28, 17, 17, 60, 18, 33, 23, 33, 43, 30, 39, 29, 27, 17, 22, 19, 23, 38, 63, 28, 36, 18, 12, 23, 27, 24, 24, 28, 6, 15, 15, 20, 13, 14, 18, 26, 41, 13, 16, 10, 8, 21, 32, 23, 40, 22, 16, 36, 28, 44, 8, 48, 11, 12, 24, 21, 32, 14, 26, 9, 13, 12, 12, 19, 34, 36, 21, 7, 22, 16, 13, 4, 34, 9, 30, 20, 25, 10, 21, 25, 41, 29, 19, 15, 23, 32, 16, 33, 11, 15, 21, 43, 11, 27, 24, 12, 3, 5, 5, 10, 28, 2, 34, 22, 18, 25, 47, 6, 16, 23, 20, 46, 17, 9, 13, 21, 15, 39, 36, 12, 31, 20, 12, 3, 19, 8, 24, 7, 10, 3, 12, 8, 20, 22, 40, 21, 16, 14, 22, 22, 17, 13, 12, 25, 30, 34, 25, 3, 21, 54, 9, 11, 11, 65, 68, 12, 54, 14, 13, 20, 18, 22, 9, 20, 12, 20, 3, 16, 10, 6, 8, 13, 13, 29, 32, 22, 22, 7, 3, 10, 34, 12, 3, 12, 18, 14, 3, 12, 29, 14, 31, 20, 6, 23, 18, 37, 22, 17, 38, 13, 10, 13, 41, 23, 25, 23, 10, 11, 12, 21, 23, 11, 3, 21, 14, 9, 18, 32, 19, 97, 22, 18, 42, 25, 11, 20, 13, 30, 7, 15, 52, 18, 26, 68, 26, 20, 9, 31, 18, 34, 22, 30, 7, 36, 30, 7, 27, 30, 11, 43, 17, 22, 9, 19, 16, 12, 21, 13, 17, 27, 19, 25, 12, 15, 10, 23, 19, 14, 12, 6, 10, 5, 11, 9, 33, 30, 48, 8, 23, 10, 15, 16, 9, 18, 11, 24, 11, 15, 4, 9, 13, 33, 19, 33, 9, 5, 21, 24, 11, 12, 17, 37, 24, 15, 36, 27, 8, 17, 14, 7, 22, 35, 13, 25, 46, 18, 11, 9, 13, 34, 26, 20, 13, 28, 12, 8, 11, 8, 10, 21, 12, 10, 11, 29, 19, 21, 6, 20, 24, 12, 18, 11, 7, 19, 11, 16, 16, 19, 40, 22, 10, 13, 22, 14, 15, 16, 14, 8, 14, 48, 60, 16, 6, 12, 26, 31, 18, 25, 7, 13, 36, 9, 14, 26, 20, 5, 21, 14, 7, 7, 20, 21
d100e	46, 6, 50, 13, 15, 20, 13, 30, 29, 19, 22, 14, 8, 16, 14, 11, 29, 24, 8, 11, 8, 15, 15, 12, 21, 22, 32, 30, 5, 14, 12, 9, 7, 3, 13, 24, 29, 14, 28, 11, 29, 19, 23, 19, 37, 24, 36, 29, 8, 22, 7, 20, 16, 25, 13, 4, 4, 24, 39, 45, 13, 13, 3, 9, 14, 33, 7, 11, 23, 7, 11, 15, 7, 11, 8, 9, 32, 9, 6, 11, 12, 15, 12, 9, 11, 20, 3, 3, 9, 17, 19, 13, 31, 32, 5, 6, 41, 9, 9, 17, 10, 7, 29, 30, 13, 14, 5, 24, 7, 28, 15, 37, 19, 3, 4, 12, 13, 6, 6, 10, 7, 13, 24, 11, 4, 9, 7, 8, 40, 45, 15, 3, 18, 10, 11, 18, 10, 5, 16, 49, 16, 23, 2, 16, 13, 6, 17, 5, 10, 5, 23, 12, 26, 5, 6, 4, 4, 4, 8, 42, 9, 31, 4, 3, 8, 6, 23, 6, 8, 14, 7, 16, 9, 6, 7, 16, 23, 27, 5, 6, 6, 4, 4, 5, 7, 21, 8, 14, 9, 13, 46, 15, 44, 13, 4, 3, 26, 15, 22, 5, 18, 6, 3, 14, 4, 27, 17, 22, 12, 19, 18, 9, 7, 13, 6, 10, 7, 29, 11, 4, 9, 3, 6, 16, 12, 28, 12, 26, 18, 24, 25, 45, 6, 23, 11, 9, 19, 33, 40, 12, 18, 12, 36, 56, 28, 59, 21, 25, 14, 3, 11, 13, 21, 6, 29, 12, 17, 19, 34, 20, 35, 34, 28, 10, 3, 24, 10, 11, 14, 41, 21, 3, 18, 4, 10, 3, 13, 7, 12, 11, 8, 3, 4, 38, 47, 18, 9, 3, 18, 13, 16, 22, 6, 10, 10, 9, 41, 12, 7, 20, 29, 8, 16, 6, 8, 21, 2, 11, 17, 5, 4, 16, 23, 30, 17, 19, 16, 7, 14, 9, 13, 34, 14, 13, 25, 12, 27, 17, 11, 15, 3, 27, 31, 22, 29, 20, 21, 11, 15, 22, 9, 12, 27, 26, 66, 25, 32, 21, 15, 3, 10, 11, 15, 24, 15, 21, 19, 15, 38, 12, 24, 6, 5, 33, 7, 21, 24, 12, 8, 12, 30, 33, 43, 54, 13, 24, 5, 83, 38, 12, 14, 5, 24, 2, 14, 38, 5, 18, 21, 19, 5, 12, 12, 22, 12, 5, 12, 12, 14, 33, 2, 6, 17, 35, 11, 23, 11, 16, 30, 34, 3, 9, 15, 8, 8, 17, 8, 38, 14, 18, 8, 10, 15, 13, 3, 15, 34, 6, 6, 11, 7, 10, 15, 19, 19, 20, 10, 9, 20, 11, 17, 4, 5, 3, 16, 2, 17, 7, 16, 27, 16, 17, 31, 6, 12, 25, 27, 3, 32, 38, 20, 15, 15, 32, 33, 15, 7, 13, 10, 32, 11, 5, 14, 16, 8, 3, 8, 16, 15, 15, 32, 27, 27, 14, 16, 20, 3, 26, 11, 12, 13, 17, 3, 3, 8, 31, 17, 13, 24, 20, 4, 5, 17, 6, 5, 22, 14, 7, 6, 7, 23, 23, 3, 9, 10, 23, 10, 12, 7, 10, 7, 10, 10, 8, 30, 20, 3, 19, 6, 8, 28, 10, 23, 12, 4, 56, 10, 5, 36, 21, 31, 3, 13, 6, 15, 29, 29, 30, 23, 21, 2, 28, 37, 21, 58, 4, 26, 23, 61, 30, 24, 16, 31, 35, 20, 25, 3, 2, 4, 7, 8, 24, 7, 24, 22, 4, 6, 5, 42, 15, 18, 3, 15, 17, 19, 28, 17, 15, 24, 24, 16, 15, 22, 20, 18, 4, 12, 2, 14, 16, 15, 13, 3, 16, 13, 11, 11, 15, 24, 9, 10, 6, 12, 11, 8, 11, 8, 12, 6, 7, 3, 13, 31, 22, 3, 7, 5, 16, 15, 19, 30, 12, 16, 4, 18, 4, 8, 23, 12, 23, 20, 19, 6, 16, 35, 77, 24, 16, 12, 31, 22, 14, 2, 32, 64, 54, 6, 10, 22, 6, 36, 20, 45, 20, 19, 36, 19, 30, 38, 17, 23, 49, 15, 18, 37, 23, 16, 31, 37, 1, 25, 12, 24, 42, 27, 51, 5, 55, 24, 9
d101e	20, 19, 6, 15, 19, 35, 23, 40, 12, 3, 7, 22, 42, 18, 20, 13, 30, 12, 11, 42, 10, 22, 64, 9, 20, 24, 51, 27, 19, 87, 13, 43, 15, 18, 35, 22, 20, 10, 14, 18, 38, 17, 24, 10, 28, 20, 40, 20, 22, 38, 12, 31, 16, 45, 39, 21, 7, 24, 25, 22, 28, 29, 26, 36, 15, 38, 34, 52, 27, 19, 87, 13, 43, 15, 18, 31, 36, 21, 14, 17, 24, 10, 28, 18, 32, 40, 20, 28, 12, 31, 16, 39, 21, 24, 25, 22, 28, 29, 26, 32, 44, 20, 10, 13, 30, 19, 8, 7, 16, 9, 32, 16, 33, 13, 36, 29, 35, 40, 25, 29, 45, 14, 15, 8, 3, 4, 35, 15, 17, 34, 15, 10, 40, 14, 30, 16, 4, 15, 54, 52, 12, 39, 28, 20, 9, 18, 38, 55, 43, 46, 18, 32, 38, 27, 12, 22, 23, 12, 45, 37, 8, 46, 29, 11, 31, 36, 11, 18, 59, 19, 13, 16, 24, 66, 41, 25, 23, 42, 30, 22, 44, 25, 15, 24, 33, 6, 5, 44, 30, 26, 41, 15, 9, 23, 12, 14, 16, 2, 34, 24, 19, 18, 27, 30, 29, 26, 19, 27, 17, 38, 60, 8, 18, 29, 36, 31, 26, 20, 19, 27, 11, 22, 45, 12, 2, 2, 18, 19, 13, 31, 12, 14, 16, 2, 35, 16, 25, 29, 27, 25, 19, 19, 12, 38, 21, 38, 26, 36, 27, 26, 39, 44, 21, 19, 36, 24, 11, 20, 28, 31, 6, 11, 31, 43, 19, 26, 40, 43, 34, 48, 13, 17, 7, 13, 54, 43, 54, 7, 8, 12, 20, 5, 1, 13, 19, 16, 48, 22, 30, 27, 8, 72, 24, 26, 38, 21, 8, 25, 35, 30, 14, 22, 8, 24, 42, 18, 3, 3, 4, 32, 8, 12, 16, 18, 25, 28, 26, 51, 18, 33, 3, 20, 9, 13, 12, 29, 19, 29, 15, 15, 21, 15, 18, 15, 50, 28, 16, 5, 29, 28, 48, 33, 20, 60, 31, 19, 65, 46, 22, 40, 50, 41, 10, 24
d102e	80, 13, 11, 13, 89, 13, 6, 4, 43, 3, 23, 3, 7, 7, 30, 17, 5, 51, 14, 19, 60, 26, 43, 19, 7, 5, 29, 29, 24, 25, 12, 20, 17, 12, 9, 20, 27, 12, 20, 10, 33, 25, 10, 32, 22, 18, 31, 14, 21, 9, 22, 29, 26, 19, 31, 20, 5, 7, 22, 10, 20, 28, 13, 26, 29, 6, 8, 20, 42, 24, 9, 3, 13, 15, 31, 32, 28, 12, 23, 7, 14, 7, 12, 10, 9, 37, 11, 10, 36, 30, 28, 18, 12, 11, 35, 41, 14, 23, 19, 27, 3, 3, 3, 47, 5, 5, 24, 27, 4, 3, 24, 27, 17, 15, 20, 25, 10, 20, 7, 13, 15, 16, 28, 12, 4, 6, 12, 23, 34, 7, 21, 35, 13, 22, 11, 16, 21, 22, 3, 10, 21, 5, 34, 88, 12, 51, 11, 39, 20, 11, 17, 3, 10, 6, 4, 17, 8, 24, 23, 33, 17, 15, 3, 23, 17, 19, 10, 17, 4, 8, 16, 22, 13, 44, 5, 6, 16, 2, 14, 10, 22, 41, 10, 12, 32, 17, 10, 17, 18, 15, 10, 4, 8, 39, 41, 13, 5, 6, 19, 11, 7, 29, 4, 7, 10, 30, 25, 28, 13, 13, 3, 8, 29, 30, 21, 18, 23, 16, 29, 6, 7, 26, 10, 34, 22, 15, 19, 46, 20, 7, 10, 13, 26, 19, 40, 30, 30, 18, 26, 20, 24, 14, 20, 18, 20, 18, 3, 13, 38, 19, 29, 20, 4, 17, 29, 12, 7, 23, 4, 9, 15, 12, 16, 16, 4, 19, 14, 52, 6, 10, 11, 10, 11, 33, 9, 24, 11, 14, 15, 17, 2, 12, 18, 41, 11, 10, 7, 3, 14, 14, 12, 14, 14, 11, 10, 20, 4, 19, 17, 29, 33, 18, 35, 8, 9, 16, 30, 17, 11, 14, 14, 23, 25, 16, 31, 25, 28, 34, 15, 3, 10, 16, 12, 3, 8, 10, 17, 17, 16, 13, 4, 19, 1, 2, 38, 32, 11, 36, 2, 27, 3, 3, 9, 29, 7, 3, 11, 12, 10, 8, 12, 7, 29, 34, 8, 5, 4, 3, 3, 16, 17, 38, 8, 3, 9, 13, 13, 11, 16, 5, 1, 14, 28, 14, 19, 14, 24, 11, 3, 16, 14, 13, 18, 6, 16, 5, 11, 15, 15, 7, 9, 10, 21, 21, 28, 35, 20, 22, 11, 34, 13, 13, 2, 10, 15, 19, 7, 29, 27, 10, 9, 7, 40, 31, 43, 62, 42, 12, 25, 20, 14, 10, 36, 11, 9, 6, 10, 13, 14, 10, 10, 17, 21, 17, 25, 18, 9, 19, 13, 27, 3, 3, 3, 49, 7, 19, 30, 5, 20, 36, 11, 16, 10, 11, 40, 20, 41, 23, 18, 29, 27, 21, 25, 10, 19, 6, 13, 15, 17, 32, 62, 13, 23, 24, 27, 13, 12, 37, 36, 55, 27, 27, 9, 3, 13, 25, 14, 39, 15, 45, 35, 41, 5, 10, 5, 5, 5, 6, 6, 18, 2, 2, 21, 21, 25, 31, 23, 29, 22, 17, 22, 23, 32, 10, 18, 42, 19, 36, 33, 44, 7, 37, 17, 21, 18, 12, 13, 29, 11, 4, 12, 36, 6, 10, 10, 15, 42, 21, 36, 8, 30, 28, 9, 35, 12, 29, 17, 22, 7, 30, 32, 19, 9, 23, 3, 44, 21, 12, 21, 3, 12, 7, 3, 14, 7, 14, 19, 17, 30, 16, 5, 10, 9, 10, 8, 2, 43, 16, 5, 8, 11, 8, 8, 9, 16, 26, 2, 23, 17, 5, 6, 12, 17, 33, 12, 9, 29, 13, 16, 19, 18, 22, 10, 15, 2, 26, 5, 9, 33, 17, 3, 9, 39, 4, 5, 4, 19, 12, 45, 11, 20, 21, 14, 3, 18, 11, 21, 11, 9, 7, 9, 17, 4, 6, 34, 19, 12, 9, 20, 5, 23, 38, 30, 14, 13, 3, 53, 6, 4, 26, 3, 23, 20, 13, 11, 29, 23, 32, 12, 20, 29, 5, 11, 9, 46, 13, 21, 17, 26, 9, 6, 35, 9, 14, 10, 17, 6, 14, 27, 23, 5, 5, 7, 13, 17, 16, 30, 23, 30, 30, 25, 6, 13, 37, 11, 17, 7, 5, 9, 31, 12, 4, 4, 19, 7, 23, 11, 13, 14, 17, 5, 9, 12, 35, 2, 12, 14, 12, 11, 23, 8, 7, 10, 17, 21, 11, 19, 27, 8, 11, 14, 35, 60, 7, 8, 5, 16, 8, 33, 62, 18, 20, 17, 45, 4, 15, 45, 39, 4, 22, 20, 44, 40, 36, 16, 26, 20, 18, 21, 25, 13, 25, 3, 32, 43, 9, 8, 26, 21, 55
d103g	30, 30, 37, 34, 19, 27, 16, 29, 43, 28, 21, 19, 21, 17, 21, 11, 21, 32, 28, 19, 21, 24, 36, 32, 21, 19, 21, 35, 18, 10, 28, 21, 26, 29, 15, 27, 25, 22, 11, 29, 17, 25, 10, 20, 21, 17, 2, 14, 30, 31, 21, 31, 23, 20, 59, 26, 22, 48, 18, 30, 32, 13, 27, 23, 16, 39, 19, 82, 23, 20, 19, 30, 32, 21, 19, 30, 11, 24, 27, 30, 45, 50, 39, 21, 32, 28, 34, 8, 28, 14, 15, 22, 14, 13, 14, 13, 13, 13, 13, 13, 17, 39, 19, 17, 22, 19, 28, 15, 26, 14, 15, 20, 35, 21, 26, 12, 24, 17, 9, 30, 19, 11, 29, 12, 25, 8, 4, 8, 32, 8, 37, 25, 3, 26, 18, 30, 5, 3, 23, 70, 12, 7, 18, 26, 23, 11, 25, 12, 29, 15, 9, 24, 3, 58, 27, 9, 33, 6, 14, 20, 36, 44, 23, 31, 35, 28, 34, 8, 18, 16, 17, 17, 21, 15, 34, 24, 24, 31, 32, 17, 20, 17, 28, 10, 29, 38, 12, 30, 26, 11, 29, 5, 7, 8
d104g	31, 4, 29, 33, 9, 6, 18, 10, 3, 6, 33, 20, 12, 11, 14, 27, 26, 25, 17, 17, 25, 19, 20, 15, 19, 17, 35, 37, 11, 34, 26, 5, 8, 20, 28, 27, 20, 37, 16, 29, 14, 26, 13, 28, 33, 22, 13, 4, 4, 32, 25, 13, 23, 11, 9, 11, 17, 11, 13, 13, 7, 14, 16, 10, 7, 10, 15, 33, 16, 11, 7, 10, 12, 14, 25, 20, 27, 40, 9, 14, 14, 23, 23, 15, 8, 26, 16, 14, 11, 18, 19, 12, 20, 7, 17, 32, 10, 17, 18, 7, 17, 28, 120, 39, 6, 100, 62, 2, 20, 9, 74, 52, 32, 23, 25, 22, 36, 31, 14, 29, 18, 28, 30, 14, 22, 16, 14, 29, 29, 15, 7, 16, 28, 28, 29, 40, 9, 9, 32, 22, 19, 4, 4, 23, 10, 44, 25, 23, 11, 9, 13, 13, 7, 14, 16, 10, 7, 10, 15, 33, 16, 11, 7, 10, 12, 14, 25, 19, 27, 40, 9, 14, 14, 21, 23, 15, 8, 26, 16, 14, 11, 18, 19, 12, 20, 7, 17, 32, 10, 17, 18, 7, 11, 7, 15, 16, 22, 10, 8, 15, 11, 24, 10, 31, 25, 14, 24, 12, 35, 18, 18, 12, 19, 10, 28, 6, 22, 27, 14, 18, 32, 15, 11, 27, 6, 16, 34, 22, 19, 4, 4, 23, 10, 44, 25, 23, 11, 9, 13, 13, 7, 14, 16, 10, 7, 10, 18, 33, 16, 11, 7, 10, 12, 14, 25, 19, 27, 40, 9, 14, 14, 21, 23, 15, 8, 26, 16, 14, 11, 18, 19, 12, 20, 7, 17, 32, 10, 17, 18, 7, 12, 7, 15, 17, 8, 22, 13, 8, 15, 11, 22, 7, 9, 23, 13, 29, 42, 34, 20, 24, 30, 8, 17, 19, 29, 27, 22, 17, 16, 27, 22, 28, 20, 32, 14, 17, 23, 10, 13, 5, 22, 11, 12, 19, 10, 31, 31, 24, 30, 23, 13, 12
d105g	26, 36, 16, 32, 26, 14, 30, 26, 31, 16, 15, 23, 46, 26, 10, 28, 35, 54, 50, 46, 30, 25, 27, 46, 6, 6, 29, 23, 8, 15, 48, 49, 32, 35, 30, 17, 28, 17, 36, 21, 42, 27, 28, 9, 22, 25, 24, 37, 16, 18, 2, 31, 2, 32, 18, 10, 34, 29, 36, 24, 31, 24, 26, 18, 27, 27, 18, 25, 8, 34, 13, 15, 11, 24, 32, 14, 40, 28, 12, 32, 38, 18, 14, 60, 21, 35, 59, 83, 17, 55, 34, 58, 42, 38, 20, 6, 18, 25, 37, 28, 28, 18, 3, 18, 12, 31, 36, 13, 23, 26, 26, 15, 36, 30, 20, 19, 52, 66, 39, 10, 24, 35, 32, 28, 12, 32, 15, 18, 37, 9, 12, 26, 31, 35, 18, 37, 33, 29, 25, 33, 25, 32, 34, 22, 13, 19, 29, 22, 34, 21, 37, 25, 28, 19, 19, 40, 30, 14, 22, 33, 22, 26, 21, 23, 35, 36, 8, 23, 50, 43, 18, 17, 20, 38, 34, 26, 44, 10, 23, 23, 32, 9, 25, 39, 32, 12, 40, 18, 17, 13, 84, 35, 9, 57, 22, 11, 12, 27, 32, 24, 17, 11, 23, 16, 9, 27, 52, 45, 36, 28, 31, 34, 70, 32, 38, 53, 10, 35, 67, 19, 18, 63, 65, 12, 22, 43, 34, 32, 7, 24, 26, 42, 15, 56, 62, 3, 49, 45, 56, 22, 16, 30, 43, 10, 3, 42
d106g	29, 39, 21, 14, 16, 3, 14, 13, 10, 35, 38, 25, 18, 15, 21, 11, 10, 13, 16, 40, 16, 20, 35, 26, 16, 27, 20, 8, 31, 26, 9, 43, 39, 24, 33, 15, 16, 25, 33, 3, 24, 10, 38, 33, 28, 12, 20, 16, 17, 8, 11, 18, 43, 33, 22, 29, 12, 24, 9, 37, 25, 11, 31, 31, 37, 23, 7, 20, 20, 5, 39, 3, 68, 9, 16, 30, 24, 17, 27, 52, 8, 14, 16, 25, 24, 38, 19, 10, 10, 27, 18, 3, 17, 25, 29, 10, 7, 21, 5, 16, 9, 14, 14, 3, 13, 25, 4, 14, 32, 17, 17, 13, 6, 20, 19, 17, 11, 17, 25, 14, 12, 5, 20, 32, 15, 23, 16, 25, 8, 17, 23, 30, 8, 31, 27, 28, 45, 25, 3, 27, 17, 35, 19, 3, 46, 14, 22, 10, 26, 14, 17, 42, 25, 10, 33, 48, 35, 25, 15, 32, 14, 26, 18, 11, 15, 22, 12, 23, 13, 18, 16, 15, 14, 17, 30, 5, 16, 7, 11, 23, 12, 21, 15, 11, 12, 16, 33, 19, 31, 16, 8, 11, 6, 26, 30, 27, 21, 30, 13, 18, 13, 11, 19, 11, 38, 9, 23, 7, 37, 17, 7, 25, 7
d107g	31, 31, 41, 34, 19, 24, 23, 16, 20, 12, 26, 13, 23, 13, 28, 40, 23, 35, 30, 23, 32, 3, 40, 21, 35, 45, 25, 37, 27, 41, 30, 36, 22, 21, 8, 40, 27, 35, 28, 40, 20, 16, 9, 16, 11, 5, 14, 14, 39, 42, 18, 31, 25, 18, 26, 14, 5, 19, 16, 32, 18, 7, 16, 9, 36, 18, 17, 33, 24, 31, 16, 11, 29, 16, 45, 16, 17, 33, 32, 18, 21, 25, 14, 11, 19, 31, 19, 3, 20, 31, 43, 40, 34, 26, 39, 29, 3, 26, 14, 21, 21, 22, 50, 12, 19, 21, 36, 17, 21, 28, 11, 13, 42, 30, 13, 12, 21, 10, 10, 17, 17, 33, 14, 38, 11, 12, 3, 23, 31, 15, 13, 42, 27, 26, 32, 28, 14, 31, 30, 21, 19, 21, 28, 35, 26, 31, 31, 30, 15, 12, 37, 25, 43, 23, 27, 31, 28, 12, 16, 35, 22, 33, 30, 43, 53, 39, 25, 30, 22, 25, 18, 25, 32, 6, 30, 12, 3, 21, 35, 34, 27, 36, 21, 43, 41, 13, 32, 5, 24, 12, 30, 17, 13, 19, 9, 2, 1, 21, 31, 14, 13, 23, 18, 37, 16, 21, 22, 22, 21, 28, 26, 33, 3, 24, 51, 18, 13, 25, 16, 10, 27, 22, 21, 3, 22, 24, 18, 18
d108g	11, 23, 41, 16, 33, 20, 40, 30, 12, 21, 25, 9, 32, 23, 24, 18, 11, 8, 19, 21, 13, 12, 14, 17, 23, 32, 31, 40, 42, 30, 23, 41, 22, 22, 11, 11, 21, 20, 16, 38, 25, 26, 4, 11, 21, 8, 16, 20, 20, 16, 13, 46, 29, 12, 8, 31, 20, 8, 24, 9, 13, 9, 39, 16, 12, 12, 37, 39, 12, 15, 20, 17, 29, 12, 17, 22, 18, 28, 20, 32, 20, 33, 7, 15, 22, 39, 34, 20, 18, 32, 22, 16, 9, 30, 15, 15, 33, 9, 26, 21, 30, 31, 38, 12, 32, 25, 40, 23, 20, 15, 37, 42, 31, 7, 32, 13, 22, 11, 20, 18, 16, 5, 16, 22, 23, 23, 8, 18, 16, 32, 11, 22, 26, 24, 17, 9, 24, 18, 22, 18, 16, 15, 24, 16, 13, 11, 21, 18, 21, 33, 14, 82, 22, 10, 28, 29, 26, 15, 39, 22, 42, 33, 42, 12, 37, 35, 31, 16, 20, 23, 24, 42, 44, 24, 30, 22, 21, 12, 25, 5, 10, 10, 22, 24, 4
d109h	43, 21, 35, 23, 22, 17, 29, 31, 22, 41, 20, 23, 19, 42, 24, 46, 33, 18, 41, 27, 20, 30, 47, 16, 33, 24, 22, 13, 14, 23, 16, 34, 19, 39, 23, 19, 25, 24, 43, 21, 56, 17, 66, 75, 18, 29, 25, 29, 12, 54, 14, 34, 12, 30, 23, 9, 10, 28, 23, 32, 59, 27, 11, 42, 14, 15, 12, 14, 27, 37, 28, 34, 66, 22, 58, 22, 29, 41, 24, 23, 16, 86, 34, 45, 28, 20, 29, 34, 37, 15, 40, 11, 49, 46, 32, 20, 12, 41, 21, 32, 8, 11, 11, 14, 33, 61, 37, 21, 21, 9, 54, 126, 54, 52, 35, 37, 17, 27, 46, 7, 38, 27, 33, 21, 48, 19, 30, 36, 40, 29, 44, 30, 20, 53, 37, 24, 23, 13, 12
d110h	28, 13, 30, 29, 21, 14, 26, 17, 31, 31, 12, 31, 20, 9, 22, 25, 18, 23, 22, 33, 18, 13, 28, 14, 36, 27, 29, 23, 10, 29, 16, 7, 21, 5, 12, 23, 25, 27, 33, 10, 51, 12, 16, 20, 16, 18, 26, 29, 14, 15, 21, 17, 26, 26, 25, 40, 12, 23, 23, 8, 24, 18, 16, 24, 3, 11, 26, 24, 25, 27, 24, 95, 32, 11, 10, 7, 17, 9, 16, 10, 6, 11, 23, 18, 27, 22, 18, 14, 15, 22, 16, 12, 11, 20, 18, 21, 33, 21, 16, 22, 14, 24, 27, 21, 26, 8, 11, 25, 18, 15, 11, 19, 37, 19, 12, 20, 6, 29, 8, 25, 7, 26, 18, 30, 30, 43, 4, 16, 16, 15, 14, 40, 3, 20, 11, 6, 7, 8, 15, 21, 27, 44, 14, 10, 24, 22, 24, 25, 21, 9, 34, 17, 15, 3, 18, 6, 35, 11, 31, 29, 31, 18, 23, 17, 21, 12, 30, 12, 21, 25, 9, 32, 23, 24, 18, 11, 8, 19, 21, 13, 12, 14, 17, 22, 28, 31, 17, 24, 31, 45, 16, 25, 31, 24, 8, 20, 19, 8, 11, 17, 22, 17, 26, 29, 37, 20, 33, 34, 28, 10, 23, 17, 25, 23, 21, 20, 25, 24, 16, 25, 18, 19, 8, 3, 11, 18, 2, 2, 6, 8, 30, 26, 15, 42, 32, 9, 25, 10, 13, 31, 53, 33, 41, 22, 28, 39, 8, 20, 9, 28, 15, 25, 25, 27, 33, 44, 26, 36, 8, 37, 14, 28, 12, 31, 53, 16, 34, 8, 20, 9, 21, 25, 25, 27, 28, 20, 13, 25, 16, 20, 22, 8, 24, 8, 15, 31, 25, 7, 6, 35, 33, 16, 30, 33, 14, 19, 21, 22, 29, 38, 11, 30, 33, 31, 21, 20, 17, 21, 34, 27, 51, 30, 20, 14, 16, 28, 25, 31, 3, 7, 24, 20, 27, 16, 29, 11, 11, 13, 8, 26, 43, 26, 17, 16, 51, 21, 26, 25, 24, 14, 25, 15, 26, 35, 7, 23, 26, 26, 26, 21, 15, 34, 20, 14, 10, 16, 32, 22, 27, 27, 21, 46, 22, 8, 14, 22, 21, 10, 21, 23, 23, 15, 34, 16, 23, 34, 12, 23, 26, 22, 23, 23, 8, 18, 16, 32, 11, 22, 26, 24, 17, 9, 24, 18, 22, 18, 16, 15, 24, 16, 13, 11, 21, 18, 21, 33, 14, 82, 22, 10, 28, 29, 26, 15, 39, 22, 42, 33, 42, 12, 37, 35, 31, 16, 20, 23, 24, 18, 39, 14, 26, 15, 40
d111h	25, 41, 23, 27, 6, 9, 8, 22, 9, 50, 11, 43, 10, 14, 13, 13, 30, 18, 20, 16, 9, 29, 29, 9, 23, 24, 20, 33, 25, 29, 8, 42, 13, 15, 30, 14, 32, 29, 32, 24, 15, 16, 24, 4, 21, 27, 29, 26, 30, 13, 5, 58, 32, 18, 13, 38, 32, 8, 24, 10, 25, 20, 15, 37, 22, 29, 38, 18, 24, 22, 16, 11, 8, 24, 29, 33, 11, 25, 22, 6, 25, 18, 3, 36, 19, 13, 36, 12, 23, 7, 33, 12, 18, 11, 13, 23, 22, 13, 22, 14, 22, 33, 30, 11, 30, 35, 17, 28, 20, 32, 20, 3, 16, 14, 21, 25, 15, 8, 24, 15, 15, 47, 24, 45, 34, 6, 7, 11, 47, 49, 24, 3, 16, 10, 17, 40, 19, 48, 17, 41, 23, 20, 15, 24, 4, 12, 27, 9, 13, 21, 36, 43, 33, 18, 39, 35, 36, 27, 12, 27, 9, 3, 8, 24, 31, 14, 11, 38, 37, 21, 26, 23, 9, 37, 57, 24, 32, 42, 23, 31, 22, 44, 8, 47, 23, 51, 27, 16, 27, 26, 12, 10, 26, 24, 19, 36, 30, 22, 19, 27, 25, 24, 20, 21, 23, 16, 13, 21, 21, 40
d112h	52, 24, 14, 38, 51, 40, 36, 14, 11, 15, 25, 12, 23, 32, 39, 48, 125, 74, 18, 35, 21, 33, 31, 20, 38, 42, 32, 24, 25, 25, 19, 16, 30, 47, 28, 41, 20, 34, 22, 20, 41, 54, 59, 26, 39, 28, 26, 30, 38, 14, 10, 26, 15, 15, 20, 34, 24, 32, 52, 37, 34, 30, 26, 24, 44, 36, 25, 23, 44, 49, 21, 18, 18, 31, 21, 36, 34, 14, 21, 20, 32, 38, 19, 17, 21, 71, 31, 23, 16, 27, 37, 30, 13, 45, 19, 33, 39, 13, 55, 23, 21, 36, 43, 51, 42, 29, 26, 19, 30, 17, 32, 14, 34, 9, 17, 31, 27, 34, 37, 38, 44, 20, 20, 37, 33, 31, 30, 17, 4, 33, 38, 17, 8, 23, 7, 24, 9, 8, 11, 14, 33, 21, 6, 24, 15, 23, 7, 7, 41, 18, 29, 6, 40, 14, 18, 29, 9, 15, 2, 21, 13, 18, 26, 39, 25, 16, 19, 38, 19, 11, 23, 26, 18, 23, 2, 34, 18, 22
d113h	24, 13, 34, 21, 33, 50, 19, 27, 37, 23, 31, 21, 11, 42, 33, 43, 21, 24, 26, 31, 24, 35, 34, 11, 18, 9, 13, 40, 28, 36, 35, 42, 43, 53, 48, 15, 24, 23, 30, 31, 36, 21, 16, 45, 17, 5, 7, 13, 21, 26, 14, 29, 22, 25, 11, 17, 27, 57, 29, 6, 16, 29, 18, 17, 23, 14, 21, 17, 10, 28, 13, 32, 24, 16, 28, 29, 31, 19, 9, 23, 37, 31, 32, 31, 19, 17, 30, 68, 23, 23, 16, 32, 38, 10, 11, 38, 30, 18, 24, 58, 7, 31, 56, 27, 17, 14, 26, 21, 14, 23, 25, 39
d114h	34, 25, 18, 24, 29, 17, 18, 21, 28, 17, 29, 22, 16, 23, 13, 6, 8, 17, 8, 25, 14, 26, 24, 15, 35, 36, 13, 12, 21, 19, 7, 17, 13, 12, 18, 31, 35, 9, 28, 4, 5, 16, 8, 8, 8, 7, 22, 15, 24, 24, 13, 32, 29, 27, 36, 36, 21, 12, 8, 14, 16, 32, 17, 17, 9, 5, 22, 29, 41, 39, 15, 20, 11, 5, 18, 15, 8, 5, 6, 6, 27, 10, 29, 27, 16, 31, 20, 34, 5, 17, 8, 9, 19, 17, 29, 49, 19, 13, 36, 13, 20, 40, 55, 34, 21, 21, 26, 34, 27, 14, 14, 27, 27, 18, 4, 9, 22, 33, 11, 18, 29, 28, 29, 19, 9, 23, 17, 27, 23, 16, 27, 11, 27, 20, 14, 11, 15, 24, 24, 32, 13, 5, 32, 28, 31, 19, 18, 25, 17, 19, 24, 21, 30, 19, 15, 31, 18, 28, 12, 6, 21, 23, 19, 6, 2, 26, 24, 26, 34, 39, 36, 22, 21, 4, 33, 23, 25, 20, 25, 10, 24, 26, 26, 9, 13, 18, 23, 15, 39, 23, 33, 38, 25, 8, 32, 29, 18, 16, 21, 19, 16, 18, 37, 25, 9, 19, 22, 11, 13, 24, 3, 24, 26, 26, 13, 13, 12, 18, 14, 19, 26, 30, 15, 36, 17, 14, 38, 15, 26, 10, 14, 27, 31, 9, 21, 11, 11, 25, 29, 11, 17, 16, 13, 26, 12, 12, 26, 26, 23, 3, 7, 12, 7, 20, 23, 10, 39, 18, 10, 23, 13, 9, 24, 23, 17, 18, 12, 19, 40, 34, 34, 23, 5, 48, 18, 29, 5, 19, 4, 21, 25, 21, 47, 6, 23, 12, 9, 16, 15, 40, 24, 13, 22, 32, 5, 16, 12, 12, 25, 8, 21, 12, 18, 7, 30, 34, 24, 21, 27, 11, 24, 19, 11, 18, 19, 25, 16, 29, 29, 11, 17, 29, 23, 30, 30, 23, 24, 25, 20, 16, 6, 27, 13, 9, 29, 29, 34, 27, 16, 28, 25, 21, 16, 16, 15, 26, 27, 13, 9, 28, 31, 18, 17, 16, 42, 22, 27, 15, 11, 18, 23, 10, 13, 5, 5, 7, 7, 8, 10, 9, 16, 15, 8, 22, 28, 15, 22, 25, 37, 30
d115i	23, 9, 30, 26, 8, 17, 26, 31, 21, 19, 21, 16, 22, 23, 24, 14, 18, 23, 13, 21, 33, 17, 20, 11, 25, 20, 25, 30, 22, 12, 34, 22, 27, 39, 25, 11, 9, 19, 16, 12, 26, 15, 15, 22, 44, 14, 16, 8, 24, 24, 21, 12, 8, 22, 5, 18, 31, 33, 40, 25, 19, 20, 19, 35, 23, 21, 19, 11, 20, 21, 16, 22, 9, 23, 12, 30, 3, 17, 14, 19, 16, 21, 14, 28, 15, 21, 30, 17, 22, 16, 23, 13, 16, 20, 23, 13, 13, 32, 24, 28, 28, 16, 49, 14, 25, 17, 21, 27, 15, 28, 12, 16, 11, 22, 17, 8, 20, 25, 17, 30, 24, 19, 17, 22, 15, 27, 20, 28, 13, 20, 23, 27, 26, 32, 17, 24, 9, 8, 33, 22, 36, 21, 25, 47, 20, 12, 31, 16, 24, 17, 25, 23, 20, 16, 15, 29, 20, 26, 38, 21, 14, 11, 10, 20, 30, 33, 14, 16, 27, 25, 10, 12, 15, 11, 5, 11, 47, 14, 11, 16, 46, 10, 5, 7, 19, 9, 8, 20, 31, 16, 24, 23, 20, 53, 33, 14, 16, 34, 39, 9, 39, 18, 15, 35, 17, 22, 27, 9, 16, 22, 30, 15, 25, 38, 29, 19, 26, 19, 15, 20, 11, 3, 18, 17, 22, 25, 20, 15, 41, 43, 18, 40, 20, 27, 20, 11, 27, 11, 3, 15, 27, 36, 35, 25, 15, 50, 29, 10, 19, 27, 18, 18, 14, 7, 3, 13, 16, 37, 42, 41, 20, 25, 46, 26
d116i	29, 12, 3, 21, 22, 11, 3, 22, 7, 19, 16, 13, 3, 29, 17, 13, 28, 19, 17, 25, 28, 19, 10, 11, 20, 30, 26, 36, 30, 17, 31, 19, 30, 27, 29, 2, 23, 23, 39, 20, 25, 36, 8, 45, 23, 26, 28, 23, 12, 30, 19, 26, 30, 13, 17, 24, 27, 28, 16, 21, 20, 21, 10, 18, 35, 36, 15, 18, 19, 33, 23, 6, 17, 30, 12, 23, 27, 20, 23, 37, 25, 33, 24, 19, 31, 34, 23, 20, 5, 17, 16, 17, 52, 14, 32, 37, 12, 28, 15, 7, 11, 6, 22, 30, 25, 6, 39, 29, 19, 4, 22, 6, 14, 23, 6, 43, 6, 18, 3, 16, 34, 19, 12, 32, 10, 6, 4, 15, 22, 27, 22, 3, 5, 5, 5, 21, 31, 17, 18, 10, 29, 5, 15, 22, 34, 21, 23, 12, 18, 27, 30, 18, 15, 25, 9, 33, 22, 22, 15, 32, 27, 22, 38, 35, 27, 13, 15, 24, 29, 22, 12, 28, 15, 16, 35, 14, 7, 5, 15, 8, 17, 11, 32, 7, 22, 4, 23, 7, 34, 30, 29, 24, 9, 26, 25, 30, 11, 29, 39, 9, 29, 12, 37, 10, 18, 17, 36, 21, 20, 18, 33, 47, 15, 27, 26, 14, 24, 3, 10, 32, 26, 26, 40, 22, 26, 21, 11, 38, 23, 27, 35, 35, 20, 9, 16, 35, 14, 49, 15, 36, 20, 7, 42, 45, 17, 4, 10, 39, 9, 19, 14, 13, 19, 13, 7, 2, 16, 21, 20, 17, 29, 26, 23, 28, 29, 31, 24, 10, 33, 26, 12, 10, 42, 17, 11, 28, 24, 21, 14, 16, 21, 6, 5, 9, 20, 33, 22, 25, 30, 26, 32, 47, 33, 27, 13, 21, 19, 28, 20, 17, 18, 32, 42, 45, 21, 27, 13, 24, 22, 13, 33, 18, 60, 32, 22, 27, 27, 8, 7, 19, 12, 25, 10, 22, 24, 22, 38, 16, 54, 19, 30, 28, 15, 5, 33, 30, 7, 11, 2, 35, 35, 19, 17, 12, 24, 11, 30, 42, 33, 25, 19, 12, 27, 10, 6, 4, 15, 24, 15, 18, 10, 33, 5
d117i	18, 4, 27, 27, 46, 48, 24, 8, 31, 18, 13, 30, 17, 37, 22, 3, 10, 39, 13, 8, 27, 22, 7, 20, 25, 43, 7, 31, 22, 21, 3, 29, 2, 20, 27, 4, 19, 4, 34, 39, 3, 20, 7, 7, 24, 7, 4, 4, 4, 2, 7, 4, 2, 37, 7, 38, 3, 5, 17, 8, 9, 26, 8, 13, 3, 14, 34, 27, 50, 14, 27, 31, 13, 7, 38, 23, 42, 7, 3, 1, 2, 1, 3, 1, 3, 1, 5, 3, 41, 15, 11, 14, 3, 27, 32, 32, 27, 41, 15, 18, 10, 22, 4, 28, 20, 10, 3, 4, 3, 2, 3, 2, 7, 3, 17, 19, 9, 29, 71, 48, 8, 39, 13, 15, 11, 31, 17, 12, 29, 22, 12, 13, 9, 24, 11, 42, 13, 34, 12, 13, 11, 19, 9, 4, 14, 11, 19, 14, 23, 27, 14, 16, 71, 27, 23, 45, 21, 29, 13, 22, 5, 8, 41, 15, 24, 10, 6, 23, 6, 41, 35, 14, 32, 32, 21, 25, 24, 15, 10, 12, 21, 21, 28, 31, 13, 10, 42, 14, 30, 18, 27, 23, 23, 13, 36, 43, 20, 33, 23, 29, 17, 31, 8, 20, 56, 6, 42, 35, 30, 16, 67, 52, 47, 21, 28, 49, 44, 34, 24, 16, 45, 46, 20, 26, 30, 30, 8, 7, 11, 13, 20, 37, 40, 17, 13, 13, 33, 18, 24, 22, 11, 17, 16, 20, 24, 15, 22, 18, 43, 20, 18, 27, 28, 5, 23, 15, 3, 6, 28, 16, 16, 34, 19, 23, 12, 18, 6, 41, 21, 9, 2, 2, 13, 1, 15, 3, 2, 17, 5, 16, 1, 13, 2, 13, 41, 17, 17, 6, 7, 3, 23, 14, 10, 8, 3, 21, 18, 16, 8, 19, 22, 27, 30, 11, 14, 4, 25, 31, 23, 38, 19, 21, 5, 15, 37, 17, 19, 14, 9, 3, 27, 10
d118i	3, 20, 18, 10, 34, 36, 11, 16, 15, 3, 27, 16, 17, 13, 8, 28, 20, 24, 48, 38, 32, 17, 39, 32, 49, 27, 14, 35, 28, 24, 30, 44, 25, 42, 4, 10, 7, 14, 6, 25, 20, 25, 6, 10, 2, 24, 22, 23, 22, 35, 14, 15, 28, 17, 19, 11, 18, 30, 33, 21, 21, 22, 24, 18, 23, 17, 11, 24, 10, 20, 34, 40, 19, 20, 13, 6, 2, 14, 9, 4, 2, 4, 21, 2, 19, 29, 13, 10, 13, 3, 22, 22, 16, 23, 36, 26, 14, 2, 8, 10, 2, 8, 10, 20, 10, 22, 4, 14, 16, 20, 15, 11, 2, 16, 66, 3, 11, 12, 6, 23, 20, 20, 21, 10, 12, 3, 14, 19, 46, 2, 25, 12, 48, 10, 23, 15, 26, 18, 15, 22, 13, 33, 15, 4, 14, 27, 19, 32, 31, 13, 4, 9, 32, 22, 26, 26, 17, 35, 35, 26, 13, 17, 32, 3, 20, 31, 6, 21, 17, 43, 19, 2, 17, 2, 18, 29, 13, 12, 36, 49, 20, 23, 35, 31, 31, 38, 27, 3, 6, 28, 49, 26, 2, 5, 16, 33, 22, 40, 8, 32, 2, 11, 2, 45, 34, 26, 6, 18, 48, 51, 21, 6, 13, 3, 4, 32, 48, 18, 52, 42, 24, 45, 23, 26, 19, 42, 20, 21, 40, 26, 17, 20, 15, 20, 29, 39, 27, 14, 11, 22, 25, 3, 25, 25, 28, 1, 36, 24, 27, 30, 40, 30, 6, 3, 8, 14, 47, 23, 28, 24, 12, 19, 33, 25, 19, 58, 27, 27, 27, 12, 38, 34, 40, 20, 42, 28, 44, 29, 30, 38, 21, 37, 38, 27, 23, 25, 38, 29, 12, 20, 11, 29, 9, 21, 62, 14, 13, 25, 7, 27, 40, 32, 8, 22, 12, 27, 14, 13, 6, 40, 12, 32, 13, 7, 13, 33, 17, 34, 28, 24, 10, 25, 20, 33, 21, 25, 11, 18, 6, 16, 21, 7, 17, 18, 36, 12, 16, 28, 8, 30, 17, 38, 15, 19, 23, 21, 20, 39, 18, 3, 18, 2, 30, 8, 12, 2, 18, 11, 2, 6, 2, 6, 22, 2, 26, 3, 21, 11, 4, 20, 2, 20, 15, 2, 9, 4, 12, 33, 2, 13, 12, 2, 28, 12, 10, 6, 10, 9, 10
d119i	40, 23, 5, 4, 5, 14, 9, 17, 42, 7, 10, 3, 17, 31, 24, 3, 25, 36, 39, 30, 13, 63, 15, 26, 11, 21, 30, 27, 38, 8, 56, 23, 15, 35, 11, 9, 4, 9, 4, 4, 12, 4, 23, 13, 15, 10, 55, 14, 3, 29, 11, 27, 37, 45, 50, 17, 3, 4, 42, 7, 32, 3, 20, 11, 3, 6, 30, 18, 26, 33, 23, 30, 15, 17, 20, 29, 23, 17, 33, 23, 14, 14, 21, 39, 27, 9, 21, 54, 21, 28, 25, 27, 30, 30, 22, 23, 16, 11, 30, 15, 16, 26, 16, 6, 20, 20, 27, 17, 24, 25, 32, 8, 2, 4, 44, 22, 27, 41, 38, 21, 11, 30, 15, 32, 43, 37, 31, 36, 27, 6, 4, 23, 26, 23, 21, 25, 25, 55, 36, 24, 34, 21, 56, 33, 38, 35, 19, 24, 17, 15, 18, 15, 44, 26, 28, 39, 10, 4, 24, 48, 18, 32, 18, 14, 34, 35, 26, 23, 11, 34, 27, 18, 21, 20, 18, 33, 10, 4, 24, 30, 37, 42, 34, 38, 14, 16, 9, 39, 32, 16, 12, 10, 25, 2, 39, 43, 48, 2, 24, 16, 5, 16, 25, 30, 19, 50, 22, 24, 22, 10, 33, 10, 38, 23, 31, 30, 40, 29
d120i	27, 42, 37, 36, 20, 13, 10, 35, 3, 9, 35, 11, 9, 14, 32, 26, 14, 26, 31, 45, 39, 33, 11, 29, 17, 18, 9, 9, 13, 15, 24, 22, 42, 62, 20, 11, 15, 28, 12, 21, 9, 29, 29, 14, 39, 20, 34, 29, 5, 29, 13, 24, 35, 13, 3, 25, 21, 22, 25, 32, 18, 39, 13, 11, 43, 18, 7, 13, 16, 11, 22, 7, 9, 14, 22, 30, 37, 4, 13, 7, 9, 14, 11, 13, 23, 20, 31, 43, 18, 20, 12, 10, 33, 28, 25, 25, 10, 25, 37, 14, 28, 35, 30, 28, 14, 14, 25, 47, 26, 17, 26, 35, 26, 24, 36, 31, 9, 40, 35, 47, 14, 30, 41, 29, 16, 12, 9, 23, 28, 20, 19, 36, 27, 22, 40, 31, 23, 18, 38, 48, 18, 33, 17, 28, 19, 27, 31, 49, 33, 27, 29, 41, 20, 16, 84, 34, 107, 34, 96, 45, 52, 31, 215, 142, 18, 3, 40, 20, 26, 22, 6, 5, 6, 7, 6, 15, 18, 27, 10, 5, 8, 24, 18, 18, 6, 35, 23, 28, 6, 10, 15, 5, 20, 12, 38, 8, 11, 7, 5, 46, 14, 49, 16, 7, 14, 21, 36, 78, 8, 27, 24, 51, 30, 14, 6, 8, 26, 14, 14, 25, 20, 12, 10, 6, 11, 29, 30, 14, 7, 38, 22, 17, 20, 18, 17, 24, 26, 40, 14, 28, 36, 31, 20, 21, 5, 32, 23, 6, 27, 34, 3, 10, 13, 22, 9, 60, 30, 9, 5, 28, 27, 28, 21, 31, 14, 11, 32, 21, 23, 25, 33, 32, 21, 40, 26, 34, 48, 23, 14, 28, 12, 22, 19, 34, 3, 15, 51, 24, 23, 22, 22, 13, 26, 11, 10, 3, 16, 40, 18, 16, 31, 32, 11, 23, 48, 42, 28, 14, 17, 3, 23, 7, 31, 37, 25, 15, 48, 41, 38, 29, 51, 23, 30, 19, 32, 31, 17, 34, 4, 4, 23, 7, 19, 43, 8, 38, 29, 34, 27, 15, 33, 24, 12, 46, 37, 13, 14, 38, 35, 70, 3, 22, 34, 13, 10, 7, 12, 8

1. Automatic summary (our method, for peer 1, run 1, cluster d061j):
An estimated 100,000 people live in the province, including 70,000 in the city of Barahona, about 125 miles west of Santo Domingo.
On Saturday, Hurricane Florence was downgraded to a tropical storm and storm 's remnants pushed inland from the U. S. Gulf Coast.
The National Weather Service reported heavy damage to Kingston 's airport and aircraft parked on The National Weather Service 's fields.
Public buildings in Cancun were used as shelters, said Cecila Lavalle, a spokesman for Quintana Roo state government in Chetumal, 155 miles southeast of Cozumel.
Thursday does n't happen very often.
If clouds form, the heat of condensation in the clouds occasionally provides ' positive feedback ' to the convergence pattern.
The sun puts energy into the water, the top of the oceans and lowest part of the atmosphere.
Hurricane Gilbert swept toward Jamaica yesterday with 100-mile-an-hour winds, and officials issued warnings to residents on the southern coasts of the Dominican Republic, Haiti and Cuba.

2. Human (d061jb):
Gilbert reached Jamaica after skirting southern Puerto Rico, Haiti and the Dominican Republic.
Hurricane Gilbert, one of the strongest storms ever, slammed into the Yucatan Peninsula Wednesday and leveled thatched homes, tore off roofs, uprooted trees and cut off the Caribbean resorts of Cancun and Cozumel.
The storm killed 19 people in Jamaica and five in the Dominican Republic before moving west to Mexico.
Prime Minister Edward Seaga of Jamaica said Wednesday the storm destroyed an estimated 100,000 of Jamaica's 500,000 homes when it throttled the island Monday.
More than 120,000 people on the northeast Yucatan coast were evacuated, the Yucatan state government said. Shelters had little or no food, water or blankets and power was out.
The Mexican National Weather Service reported winds gusting as high as 218 mph earlier Wednesday with sustained winds of 179 mph.
The National Hurricane Center said a hurricane watch was in effect on the Texas coast from Brownsville to Port Arthur and along the coast of northeast Mexico from Tampico north.
The National Hurricane Center said Gilbert was the most intense storm on record in terms of barometric pressure.
Tropical Storm Gilbert formed in the eastern Caribbean and strengthened into a hurricane Saturday night.

3. Lead baseline (d061j):
AP880912-0137 AP-NR-09-12-88 1555EDT u i AM-HurricaneGilbert Bjt 09-12 0681 AM-Hurricane Gilbert, Bjt,0702 Hurricane Hits Jamaica With 115 mph Winds; Communications Disrupted By LLOYD WILLIAMS Associated Press Writer KINGSTON, Jamaica AP Hurricane Gilbert slammed into Kingston on Monday with torrential rains and 115 mph winds that ripped roofs off homes and buildings, uprooted trees and downed power lines.
WSJ880912-0064 Hurricane Gilbert Heading for Jamaica With 100 MPH Winds LATAM SANTO DOMINGO, Dominican Republic AP Hurricane Gilbert swept toward Jamaica yesterday with 100-mile-an-hour winds, and officials issued warnings to residents on the southern coasts of the Dominican Republic, Haiti and Cuba.

4. COMPENDIUM (d061j):
Hurricane Gilbert swept toward the Dominican Republic Sunday, and the Civil Defense alerted its heavily populated south coast to prepare for high winds, heavy rains and high seas.
The storm was approaching from the southeast with sustained winds of 75 mph gusting to 92 mph. There is no need for alarm," Civil Defense Director Eugenio Cabral said in a television alert shortly before midnight Saturday.
Cabral said residents of the province of Barahona should closely follow Gilbert's movement.
An estimated 100,000 people live in the province, including 70,000 in the city of Barahona, about 125 miles west of Santo Domingo.

Figure 10: Sample summaries generated for document cluster d061j, using different approaches.

References

[1] K. McKeown, R. J. Passonneau, D. K. Elson, A. Nenkova, J. Hirschberg, Do Summaries Help? A Task-Based Evaluation of Multi-Document Summarization, in: In Proceedings of the ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 1–8.

[2] K. Kaczmarek-Majer, O. Hryniewicz, Application of linguistic summarization methods in time series forecasting, Information Sciences 478 (2019) 580 – 594.

[3] I. Mani, D. House, G. Klein, L. Hirschman, T. Firmin, B. Sundheim, The tipster summac text summarization evaluation, in: Proceedings of the Ninth Conference on European Chapter of the Association for Computational Linguistics, EACL '99, Association for Computational Linguistics, Stroudsburg, PA, USA, 1999, pp. 77–85.

[4] E. Lloret, M. Palomar, Text summarisation in progress: a literature review, Artif. Intell. Rev. 37 (2012) 1–41.

[5] A. Zamuda, J. Brest, On Tenfold Execution Time in Real World Optimization Problems with Differential Evolution in Perspective of Algorithm Design, in: 2018 25th International Conference on Systems, Signals and Image Processing (IWSSIP), IEEE, pp. 1–5.

[6] A. Zamuda, J. D. H. Sosa, Success history applied to expert system for underwater glider path planning using differential evolution, Expert Systems with Applications 119 (2019) 155–170.

[7] A. Zamuda, J. Brest, Population Reduction Differential Evolution with Multiple Mutation Strategies in Real World Industry Challenges, in: L. Rutkowski, M. Korytkowski, R. Scherer, R. Tadeusiewicz, L. Zadeh, J. Zurada (Eds.), Swarm and Evolutionary Computation, Lecture Notes in Computer Science, Springer, 2012, pp. 154–161.

[8] L. Padro, E. Stanilovsky, Freeling 3.0: Towards wider multilinguality, in: Proceedings of the Language Resources and Evaluation Conference (LREC 2012), ELRA, Istanbul, Turkey, pp. 2473–2479.

[9] M. Allahyari, S. A. Pouriyeh, M. Assefi, S. Safaei, E. D. Trippe, J. B. Gutierrez, K. Kochut, Text summarization techniques: A brief survey, CoRR abs/1707.02268 (2017).

[10] W. Dressler, Current trends in textlinguistics, Research in text theory, W. de Gruyter, 1978.

[11] C. Leopold, A. Brückner, S. Dutke, Summarizing as a strategy for science text comprehension: Text-based versus content-based processing, Discourse Processes 0 (2019) 1–20.

[12] K. S. Jones, Automatic summarising: factors and directions, CoRR cmp-lg/9805011 (1998).

[13] I. Mani, M. T. Maybury (Eds.), Advances in Automatic Text Summarization, MIT Press, 1999.

[14] R. Ferreira, L. de Souza Cabral, F. Freitas, R. D. Lins, G. de França Silva, S. J. Simske, L. Favaro, A multi-document summarization system based on statistics and linguistic treatment, Expert Systems with Applications 41 (2014) 5780 – 5787.

[15] W. Yin, Y. Pei, Optimizing sentence modeling and selection for document summarization, in: Proceedings of the 24th International Conference on Artificial Intelligence, IJCAI'15, AAAI Press, 2015, pp. 1383–1389.

[16] X. Zhang, M. Lapata, F. Wei, M. Zhou, Neural latent extractive document summarization, in: Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Association for Computational Linguistics, Brussels, Belgium, 2018, pp. 779–784.

[17] M. Zhong, P. Liu, D. Wang, X. Qiu, X. Huang, Searching for effective neural extractive summarization: What works and what's next, in: Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, Association for Computational Linguistics, Florence, Italy, 2019, pp. 1049–1058.

[18] 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.

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

[20] 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.

[21] 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.

[22] R. M. Alguliev, R. M. Aliguliyev, N. R. Isazade, Multiple documents summarization based on evolutionary optimization algorithm, Expert Systems with Applications (2012). DOI 10.1016/j.eswa.2012.09.014.

[23] R. M. Alguliev, R. M. Aliguliyev, N. R. Isazade, Formulation of document summarization as a 0-1 nonlinear programming problem, Computers & Industrial Engineering (2012). DOI 10.1016/j.cie.2012.09.005.

[24] A. Zamuda, C. Zarges, G. Stiglic, G. Hrovat, Stability selection using a genetic algorithm and logistic linear regression on healthcare records, in: Proceedings of the Genetic and Evolutionary Computation Conference Companion (GECCO 2017), pp. 143–144.

[25] G. Zhang, H. Rong, F. Neri, M. J. Pérez-Jiménez, An optimization spiking neural p system for approximately solving combinatorial optimization problems, International Journal of Neural Systems 24 (2014) 1440006.

[26] V. Roostapour, A. Neumann, F. Neumann, On the performance of baseline evolutionary algorithms on the dynamic knapsack problem, in: International Conference on Parallel Problem Solving from Nature, Springer, pp. 158–169.

[27] A. Zamuda, G. Hrovat, E. Lloret, M. Nicolau, C. Zarges, Examples implementing black-box discrete optimization benchmarking survey for BB-DOB@GECCO and BB-DOB@PPSN, in: Black Box Discrete Optimization Benchmarking (BB-DOB) Workshop at 15th International Conference on Parallel Problem Solving from Nature (PPSN 2018), September 8-12, 2018, Coimbra, Portugal, p. 1.

[28] K. M. Hermann, T. Kočiský, E. Grefenstette, L. Espeholt, W. Kay, M. Suleyman, P. Blunsom, Teaching machines to read and comprehend, in: Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 1, NIPS'15, MIT Press, Cambridge, MA, USA, 2015, pp. 1693–1701.

[29] C.-Y. Lin, ROUGE: A package for automatic evaluation of summaries, in: Proceedings of the ACL-04 Workshop on Text Summarization Branches Out, Barcelona, Spain, pp. 74–81.

[30] G. Giannakopoulos, V. Karkaletsis, G. Vouros, P. Stamatopoulos, Summarization system evaluation revisited: N-gram graphs, ACM Trans. Speech Lang. Process. 5 (2008) 5:1–5:39.

[31] L. A. Cabrera-Diego, J. Torres-Moreno, Summtriver: A new trivergent model to evaluate summaries automatically without human references, Data Knowl. Eng. 113 (2018) 184–197.

[32] E. Lloret, L. Plaza, A. Aker, The challenging task of summary evaluation: an overview, Language Resources and Evaluation 52 (2018) 101–148.

[33] R. Storn, K. Price, Differential Evolution – A Simple and Efficient Heuristic for Global Optimization over Continuous Spaces, Journal of Global Optimization 11 (1997) 341–359.

[34] J. Holland, Adaptation In Natural and Artificial Systems, The University of Michigan Press, Ann Arbor, 1975.

[35] A. E. Eiben, J. E. Smith, Introduction to Evolutionary Computing (Natural Computing Series), Springer, 2003.

[36] A. Zamuda, M. Nicolau, C. Zarges, A black-box discrete optimization benchmarking (BB-DOB) pipeline survey: taxonomy, evaluation, and ranking, in: Proceedings of the Genetic and Evolutionary Computation Conference Companion (GECCO 2018), pp. 1777–1782.

[37] S. Das, S. S. Mullick, P. Suganthan, Recent advances in differential evolution – An updated survey, Swarm and Evolutionary Computation 27 (2016) 1–30.

[38] A. P. Piotrowski, Review of differential evolution population size, Swarm and Evolutionary Computation 32 (2017) 1–24.

[39] A. P. Piotrowski, J. J. Napiorkowski, Some metaheuristics should be simplified, Information Sciences 427 (2018) 32–62.

[40] R. D. Al-Dabbagh, F. Neri, N. Idris, M. S. Baba, Algorithmic design issues in adaptive differential evolution schemes: Review and taxonomy, Swarm and Evolutionary Computation 43 (2018) 284–311.

[41] A. P. Piotrowski, J. J. Napiorkowski, Step-by-step improvement of JADE and SHADE-based algorithms: Success or failure?, Swarm and Evolutionary Computation 43 (2018) 88–108.

[42] M. Weber, F. Neri, V. Tirronen, A Study on Scale Factor in Distributed Differential Evolution, Information Sciences 181 (2011).

[43] F. Neri, G. Iacca, E. Mininno, Disturbed exploitation compact differential evolution for limited memory optimization problems, Information Sciences 181 (2011) 2469–2487.

[44] M. Weber, F. Neri, V. Tirronen, A study on scale factor/crossover interaction in distributed differential evolution, Artificial Intelligence Review 39 (2013) 195–224.

[45] A. Zamuda, J. Brest, Self-adaptive control parameters' randomization frequency and propagations in differential evolution, Swarm and Evolutionary Computation 25 (2015) 72–99.

[46] A. Viktorin, R. Senkerik, M. Pluhacek, A. Zamuda, Steady success clusters in Differential Evolution, in: 2016 IEEE Symposium Series on Computational Intelligence (SSCI), IEEE, pp. 1–8.

[47] R. Tanabe, A. S. Fukunaga, How Far Are We From an Optimal, Adaptive DE?, in: 14th International Conference on Parallel Problem Solving from Nature (PPSN XIV), IEEE, p. accepted.

[48] K. R. Opara, J. Arabas, Differential Evolution: A survey of theoretical analyses, Swarm and Evolutionary Computation 44 (2019) 546–558.

[49] J. Brest, S. Greiner, B. Bošković, M. Mernik, V. Žumer, Self-Adapting Control Parameters in Differential Evolution: A Comparative Study on Numerical Benchmark Problems, IEEE Transactions on Evolutionary Computation 10 (2006) 646–657.

[50] E. Mezura-Montes, B. C. Lopez-Ramirez, Comparing bio-inspired algorithms in constrained optimization problems, The 2007 IEEE Congress on Evolutionary Computation (25-28 Sept. 2007) 662–669.

[51] X. Yao, Y. Liu, G. Lin, Evolutionary Programming Made Faster, IEEE Transactions on Evolutionary Computation 3 (1999) 82–102.

[52] Y. Liu, X. Yao, Q. Zhao, T. Higuchi, Scaling Up Fast Evolutionary Programming with Cooperative Coevolution, in: Proceedings of the 2001 Congress on Evolutionary Computation CEC 2001, IEEE Press, 2001, pp. 1101–1108.

[53] J. Brest, P. Korošec, J. Šilc, A. Zamuda, B. Bošković, M. S. Maučec, Differential evolution and differential ant-stigmergy on dynamic optimisation problems, International Journal of Systems Science 44 (2013) 663–679.

[54] A. Viktorin, R. Senkerik, M. Pluhacek, T. Kadavy, A. Zamuda, Distance Based Parameter Adaptation for Success-History based Differential Evolution, Swarm and Evolutionary Computation 50 (2019) 100462.

[55] K. V. Price, R. M. Storn, J. A. Lampinen, Differential Evolution: A Practical Approach to Global Optimization, Natural Computing Series, Springer-Verlag, Berlin, Germany, 2005.

[56] S. Das, P. N. Suganthan, Differential Evolution: A Survey of the State-of-the-art, IEEE Transactions on Evolutionary Computation 15 (2011) 4–31.

[57] R. Joshi, A. Sanderson, Minimal representation multisensor fusion using differential evolution, IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans 29 (1999) 1083–4427.

[58] B. Bošković, J. Brest, A. Zamuda, S. Greiner, V. Žumer, History Mechanism Supported Differential Evolution for Chess Evaluation Function Tuning, Soft Computing – A Fusion of Foundations, Methodologies and Applications 15 (2011) 667–682.

[59] A. Zamuda, J. D. H. Sosa, L. Adler, Constrained Differential Evolution Optimization for Underwater Glider Path Planning in Sub-mesoscale Eddy Sampling, Applied Soft Computing 42 (2016) 93–118.

[60] D. Zaharie, Influence of crossover on the behavior of Differential Evolution Algorithms, Applied Soft Computing 9 (2009) 1126–1138.

[61] Z. Michalewicz, M. Schoenauer, Evolutionary Algorithms for Constrained Parameter Optimization Problems, Evolutionary Computation 4 (1996) 1–32.

[62] Z. Michalewicz, D. B. Fogel, How to Solve It: Modern Heuristics, Springer, Berlin, 2000.

[63] C. A. Coello Coello, Theoretical and numerical constraint-handling techniques used with evolutionary algorithms: a survey of the state of the art, Computer Methods in Applied Mechanics and Engineering 191 (2002) 1245–1287.

[64] E. Mezura-Montes, C. A. C. Coello, Constraint-handling in nature-inspired numerical optimization: past, present and future, Swarm and Evolutionary Computation 1 (2011) 173–194.

[65] J. Brest, Constrained Real-Parameter Optimization with ϵ-Self-Adaptive Differential Evolution, in: Constraint-Handling in Evolutionary Optimization, Springer, 2009, pp. 73–93.

[66] J. Brest, V. Žumer, M. S. Maučec, Self-adaptive Differential Evolution Algorithm in Constrained Real-Parameter Optimization, in: The 2006 IEEE Congress on Evolutionary Computation CEC 2006, IEEE Press, 2006, pp. 919–926.

[67] T. Takahama, S. Sakai, N. Iwane, Solving Nonlinear Constrained Optimization Problems by the ϵ Constrained Differential Evolution, IEEE International Conference on Systems, Man and Cybernetics 2006 (SMC 2006) 3 (2006) 2322–2327.

[68] S. Spolaor, M. Gribaudo, M. Iacono, T. Kadavy, Z. K. Oplatkova, G. Mauri, S. Pllana, R. Senkerik, N. Stojanovic, E. Turunen, A. Viktorin, S. Vitabile, A. Zamuda, M. S. Nobile, Towards human cell simulation, in: Lecture Notes In Computer Science, volume 11400 of *High-*

Performance Modelling and Simulation for Big Data Applications: Selected Results of the COST Action IC1406 cHiPSet (Kolodziej, Joanna, Gonzalez-Velez, Horacio (Eds.)), pp. 221–249.

[69] A. Biasizzo, F. Novak, P. Korošec, A multi–alphabet arithmetic coding hardware implementation for small fpga devices, Journal of Electrical Engineering 64 (2013) 44–49.

[70] D. Padua, Encyclopedia of Parallel Computing, Springer Publishing Company, Incorporated, 2011.

[71] C. Kessler, U. Dastgeer, S. Thibault, R. Namyst, A. Richards, U. Dolinsky, S. Benkner, J. L. Träff, S. Pllana, Programmability and performance portability aspects of heterogeneous multi-/manycore systems, in: Design, Automation & Test in Europe Conference & Exhibition (DATE), 2012, IEEE, pp. 1403–1408.

[72] S. Benkner, S. Pllana, J. L. Traff, P. Tsigas, U. Dolinsky, C. Augonnet, B. Bachmayer, C. Kessler, D. Moloney, V. Osipov, Peppher: Efficient and productive usage of hybrid computing systems, IEEE Micro 31 (2011) 28–41.

[73] S. Memeti, S. Pllana, Hstream: A directive-based language extension for heterogeneous stream computing, in: 2018 IEEE International Conference on Computational Science and Engineering (CSE), pp. 138–145.

[74] M. S. Nobile, P. Cazzaniga, A. Tangherloni, D. Besozzi, Graphics processing units in bioinformatics, computational biology and systems biology, Brief. Bioinform. 18 (2017) 870–885.

[75] S. Haug, M. Hostettler, F. Sciacca, M. Weber, The atlas arc backend to hpc, Journal of Physics: Conference Series 664 (2015) 062057.

[76] S. Memeti, L. Li, S. Pllana, J. Kolodziej, C. Kessler, Benchmarking OpenCL, OpenACC, OpenMP, and CUDA: Programming Productivity, Performance, and Energy Consumption, in: Proceedings of the 2017 Workshop on Adaptive Resource Management and Scheduling for Cloud Computing, ARMS-CC '17, ACM, New York, NY, USA, 2017, pp. 1–6.

[77] S. Wienke, P. Springer, C. Terboven, D. an Mey, OpenACC: First Experiences with Real-world Applications, in: Proceedings of the 18th

International Conference on Parallel Processing, Euro-Par'12, Springer-Verlag, Berlin, Heidelberg, 2012, pp. 859–870.

[78] J. E. Stone, D. Gohara, G. Shi, OpenCL: A parallel programming standard for heterogeneous computing systems, Computing in science & engineering 12 (2010) 66–73.

[79] OpenMP, OpenMP 4.0 Specifications, `http://www.openmp.org/specifications/`, 2013. Accessed: 2019-02-28.

[80] NVIDIA, CUDA C Programming Guide, `http://docs.nvidia.com/cuda/cuda-c-programming-guide/`, 2016. Accessed: 2019-02-28.

[81] W. Gropp, E. Lusk, A. Skjellum, Using MPI: portable parallel programming with the message-passing interface, volume 1, MIT press, 1999.

[82] Apache, Apache Hadoop project, `https://hadoop.apache.org/`, 2019. Accessed: 2019-02-28.

[83] Apache, Apache Spark project, `https://spark.apache.org/`, 2019. Accessed: 2019-02-28.

[84] A. Kos, S. Tomažič, J. Salom, N. Trifunovic, M. Valero, V. Milutinovic, New benchmarking methodology and programming model for big data processing, International Journal of Distributed Sensor Networks 11 (2015) 271752.

[85] V. Milutinovic, A. Hurson, Dataflow Processing, volume 1st edition, Academic Press, 2015.

[86] E. Hovy, Text summarization, in: The Oxford Handbook of Computational Linguistics 2nd edition, Oxford University Press, 2003, pp. 583–598.

[87] T. A. Mogensen, Introduction to Compiler Design, Springer Publishing Company, Incorporated, 1st edition, 2011.

[88] D. McCarthy, R. Navigli, Word sense disambiguation: An overview, Proceedings of the 4th International Workshop on Semantic Evaluations (2007) 7–12.

[89] G. Pampara, A. P. Engelbrecht, N. Franken, Binary differential evolution, in: 2006 IEEE International Conference on Evolutionary Computation, IEEE, pp. 1873–1879.

[90] E. Lloret, M. Palomar, COMPENDIUM: a text summarisation tool for generating summaries of multiple purposes, domains, and genres, Natural Language Engineering 19 (2013) 147–186.