UNIVERSIDAD AUTÓNOMA DE MADRID

ESCUELA POLITÉCNICA SUPERIOR

TRABAJO FIN DE MÁSTER

Desarrollo de un simulador de redes de procesadores que evolucionan (NEPS) en la nube (SPARK)

Máster Universitario en Investigación e Innovación en Tecnologías de la Información y las Comunicaciones (i2-TIC)

Autor: Subhi Mahdi AL-Rubaie, Ali

Tutora: María Elena Gómez Martínez

Departamento de Escuela Politécnica Superior

FECHA: SEPTIEMBRE 2021

DESARROLLO DE UN SIMULADOR DE REDES DE PROCESADORES QUE EVOLUCIONAN (NEPS) EN LA NUBE (SPARK)

ACKNOWLEDGEMENTS

Throughout the writing of this dissertation I have received a great deal of support and assistance.

I would first like to thank my supervisor, Professor María Elena Gómez Martínez, whose expertise was invaluable in formulating the research questions and methodology. Your insightful feedback pushed me to sharpen my thinking and brought my work to a higher level.

I would also like to thank my tutors, Dr. Alfonso Ortega, for their valuable guidance throughout my studies. You provided me with the tools that I needed to choose the right direction and successfully complete my dissertation.

In addition, I would like to thank my parents for their wise counsel and sympathetic ear. You are always there for me. Finally, I could not have completed this dissertation without the support of my brothers, sisters, and wife, who provided stimulating discussions as well as happy distractions to rest my mind outside of my research.

Thanks a lot,

Ali Subhi Mahdi Al-Rubaye

Abstract

The natural-inspired computing has become one of the most frequently used techniques to handle complex problems such as the NP-Hard optimization problems. This kind of computing has several advantages over traditional computing, including resiliency, parallel data processing, and low consumption of power. One of the active research areas of the natural-inspired algorithms is Network of Evolutionary Processors (NEPs). A NEP consists of several cells that are attached together; at the same time the edges of the graph are to transfer data between the nodes in system, while cells are representing the nodes.

In this thesis we construct a NEPs system which is implemented over the Hadoop spark environment. The use of the spark platform is essential in this work due to the capabilities supplied by this platform. It is a suitable environment used solving some complicated problems. Using the environment is a possible choice in order to design the NEPs system. For this reason, in this thesis, we detailed on how to install, design and operate this system on the Apache the spark environment is used because it has the capability to implement the NEPs system in a distributed manner. The NEPs simulation is delivered in this work. An analysis of system's parameters was also provided in this work for the system performance evaluation via the examination of each single factor affecting the performance of the NEPs individually.

After testing the system, it become clear that using NEPs on the decentralize cloud eco-system can be thought as an effective method to handle data of different formats and also to execute optimization problems such as Adelman, 3-colorabilty and Massive-NEP problems. Moreover, this scheme is also robust that can be adaptable to handle data which might be scaled up to be big data which is characterized by its volume and heterogeneity. In this context heterogeneity might be referring to collecting data from different sources. Moreover, the utilization of the spark environment as a platform to operate the NEPs system has it is prospects. This environment is characterized by its fast task handing chunks of data to Hadoop architecture that is used to implement the spark system which is mainly based on the map and reduce functions. Thus, the task is distributed on NEPs system using the cloud based environment system made it possible to have logical result in all of the three examples investigated and examined in this method.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1. INTRODUCTION

1.1 Background

Modern computing systems have been developed with a fast pace through the years. The mechanisms of computers to deal with the problems have been shifted to be more realistic and efficient to suit the requirements of users and customers. The use of these mechanisms was successful in different real-world and complex problems that require the intervention of modern tools and techniques. The computing system is composed of the input, output, processing, and storage components in their simplest forms. The processing unit is the heart of the computing system. In the beginning, the processor was simple chips used to conduct simple mathematical or logical operations. With the increase of complexity of the modern world and its huge data generation, it becomes necessary to improve traditional processing using the recent advances in this domain.

In recent decades, natural-inspired computing has piqued people's interest. There were many algorithms and models presented. Natural-inspired computing has several benefits over traditional computing, including resiliency, parallel data processing, and low power consumption. One of the potential natural-inspired algorithms is the Network of Evolutionary Processors (NEPs) [1,2]. A NEP comprises many linked cells, while the links transfer data between the processing units (i.e. nodes) in the system, while cells represent the processing units (nodes). In natural analogy, each one of the nodes has chain of chromosomes representing its DNA. Simple actions such as deletion and insertion of symbols from words and receiving and transmitting words to/from other nodes are used for processing words. NEPs are distinguished through their simple structure, intrinsic parallel data processing, and ability to adapt to a variety of problems. NEPs were suggested in numerous forms [3] and were successfully used to a variety of issues [4]. NEPs, for example, are more capable of processing NP-problems than traditional computing [5]. NEPs were implemented in a variety of ways [6,8].

In this thesis, we design a system based on NEPs implemented in the spark environment. The use of the spark platform is important because it has the facilities and capabilities provided by this platform. It seems to be a convenient environment to solve some complicated problems, such as solving NP-hard optimization problems. Using the environment might be an ideal choice to design the NEPs system. For this reason, in this thesis, we focus on details on how to install, design and operate this system on the Apache, Apache is an open-source software. can be highly customized to meet the needs of many different environments by using extensions and modules. The spark environment is used because it has the capability to implement the NEPs system in a distributed manner.

Moreover, the system design is one of the most vital objectives considered in this thesis. This might be including three sub objectives. Firstly, it is the scalability of the NEPs nodes and studying the effect of scaling the system to increase the number of nodes. The second aspect of this characteristic is the system's scalability in terms of increasing the number of edges among the different processing units, which is basically increased as a natural result of the increase of the processing units. The last aspect

considered in this thesis is that the increment of words compromises the number of words transmitted in the processing units. One of the other objectives followed in this thesis is to discuss Spark's parameter tuning. The main idea behind this is to make this tuning to obtain the best performance of parallelization that might be gained from the Spark system, which is problem dependent. which means the parameters related to any problem are distinct from the parameters associated with other issues. In this context, some problems rely on the problem itself, where each problem depends on a specific value of these parameters that must be set carefully.

1.2 Aim of thesis

In this thesis, we design a method to operate NEPs in parallel to get the best result using the Apache Spark Platform. This involves the scaling of the NEPs system and conducting the parameter tuning, which is considered one of the most important objectives in our proposed system presented in this thesis. We can conclude our objectives of this thesis as follows:

1- Simulate a NEP system to solve the problem optimization type using the Spark Apache Platform.
2- Analyze parameters tuning for the system. Parameter tuning is the process that examines every single factor affecting the performance of the NEPs individually.
3- Scale the system to be capable of handling more complex problems, such as increasing number of processors, number of connections, size and number of words transmitted.

1.3 Thesis Layout

This thesis is designed to be arranged as follows:

- In chapter two we describe the theoretical background about the Networks of Evolutionary Processors.
- In chapter three, we describe the practical implementation of the proposed system.
- In chapter four some test results are explained.
- A set of conclusions and future works were explained in the last chapter.

CHAPTER 2. THEORETICAL BACKGROUND

2.1 Background

This chapter explains the main theoretical issues related to our proposed system. This chapter is designed to give clear descriptions and state of the art about the NEPs technology and other technologies related to it. We start explaining the algorithms inspired by nature such as Darwin's theory of evolution.

2.2 Algorithms Inspired by Nature

This study presents a range of algorithms that nature has inspired in the next sections. Various research lines have looked at how nature solves problems and develop new approaches for developing creative algorithmic combinations. Evolutionary computing can be defined as a collection of algorithms that are inspired by natural selection and described by Darwin's evolution theory first. The evolution process is popular for resulting in the survival of the strongest people and the death of the weakest. In addition, the strongest individuals have a higher chance of surviving and, as a result, reproducing more frequently. The new offspring is not an identical copy of the progenitors, yet it does contain small random differences that may allow for greater adaption to meet future environmental requirements. In this approach, a species' population evolves, producing individuals who have a better adaptation to the environment. A general evolutionary system, based on such natural process, contains the next basic characteristics:

- Population: A group of people who differ significantly in their features and behavior.
- Environment: The entity in which persons are placed. Also, individuals interact with their environment and must meet its demands in order to increase their chances of survival.
- Reproduction process: Individuals all die sooner or later. To keep the population, a reproduction process must occur on a regular basis. Reproduction creates new individuals who are comparable yet not identical to their parents.
- This source of diversity stems from a wide range of operations with varying characteristics. By making such minor changes, the system attempts to locate better people.

  Considering such 4 components, the system's dynamic is one of constant population renovation, in which people improve their quality of behavior as the system gradually changes the descendants. Environmental fitness is used to determine its quality.

  From a computational standpoint, one can deduce that the system is looking for the highest-quality person. Put differently, given a problem (fitting into the environment). The system uses an iterative stochastic process to find a suitable individual. Evolutionary algorithms were motivated by such metaphor. Any problem-solving task is viewed as one of the search problems which could be addressed using evolutionary methods in this field. In addition, the search occurs in a solution space, in which each one of the points represents an individual/solution. The system explores the space on a path that should

eventually lead to the best solution. [43] provide a precise formal characterization regarding the evolutionary search; this study attempts to show how evolutionary algorithms work in such a brief introduction. Because of the prior viewpoint, evolutionary algorithms are a versatile tool that might be applied to practically any situation. This assertion will be studied thoroughly once going over the components that an evolutionary algorithm must be having:

- Representation of persons: Each one of the persons in the evolutionary algorithm can be considered one of the potential solutions to the issue. Individuals must be specified in a formal manner in such away. Because the solution to the issue is usually a complicated expression informal language, the solutions are encoded using a simplified encoding termed genotype. The genotype is the individual's representation, whereas the phenotype is the solution's formal expression. Mapping refers to the process of translating a genotype into a phenotype.

  This study attempts to construct a function for evaluating them to assess the quality of the persons, which can be represented as the quality of the found solutions. It denotes the requirements that the solutions must meet.

  Population: This is a list of possible solutions. Every reproduction cycle, in which a few individuals are born, and others die, alters.

Parent selection mechanism: prior to reproduction, the new offspring's parents must be selected. Even though other factors could be utilized, the better-adapted persons (those with high fitness values) are usually picked to reproduce.

Variation operators: the new individuals are comparable to their progenitors, as anticipated, yet there are certain distinctions as well. There are a variety of factors that cause the offspring to differ fairly. Mainly, they are an analogy of natural evolution, as mutation or chromosomes crossover. Put another way, they all change small aspects of the progenitor's representation to form a new individual.

In the pseudo-code shown in Figure 1, the evolutionary process of NEPs is clearly structured. The reason behind this pseudo-code is to show the logical arrangement of the NEPs evolutionary steps.

```
BEGIN
        INITIALIZE 'Population' with random candidate
solutions;
        EVOLUATE each candidate;
        REPEAT UNTIL ('termination condition' is satisfied)
            (1) SELECT parents;
            (2) REPRODUCE;
            (3) EVALUATE new candidates;
            (4) SELECT survivors;
        END
END
```

Figure 1 pseudo-code of evolutionary processing

In Figure 1, the pseudo-code of evolutionary processing shown several steps, the first step is to initialize the population with random candidate solution, each solution according to the genetic algorithm represent the chromosome, after initializing the first population second step is to apply evolutionary processes each single solution until the termination condition (stopping condition) is reached, the selection of the parents will be conducted first and then the reproduction constative are applied on the selective chromosome, the third step is evaluate the new candidate or the new solution generated from the above to it, afterword, the forth step is principle for electing the best solution from the newly generated solutions.

Survivor selection mechanism: Similar to parent selection, a few evolutionary algorithms choose the individuals who will survive and so be able to reproduce. High fitness values, once again, indicate a greater chance of survival. A broad scheme regarding an evolutionary algorithm could be given using previous elements. It is made up of a loop in which reproduction and selection repeatedly occur until a stopping condition is met. Finding a fitness value greater than a pre-determined value of the threshold is commonly the termination condition. Figure 1 shows the algorithm in pseudo-code in a more formal approach. The most well-known evolutionary algorithms type for this purpose will be exhibited: genetic algorithms. Even though there are different types, most genetic algorithms employ a string of binary digits or integers as representation. As one of the analogies to natural genetics, those strings are commonly referred to as chromosomes.

In the example used in this work, a genetic algorithm is created to tackle the Eight Queens problem, in which a standard chessboard is given and must arrange eight queens so that none of them can check each other. Various potential representations for a candidate solution to such an issue might be imagined. A string of 8 integers ranging as (1,64) are selected. Also, every one of the integers corresponds to the corresponding queen's location on the chessboard (the chessboard has 8x8 = 64 locations). The number of feasible checks in the board may be defined as the fitness function. In terms of reproduction, this algorithm will create two new children from 2 individuals by performing the next steps: first, a random splitting point is determined, and then the 2 parents are cut into 2 segments. Ultimately, parent 1's segment one is combined with parent 2's segment 2 and the other way around. Such a method of child formation is known as chromosome recombination or crossover, a common variation operator in natural genetics. Following crossover, a new variation operation is performed that involves randomly modifying one of the integers. This type of variation operation is known as mutation because an integer could be thought of as a gene on a chromosome. Only the selection processes must be decided at this point.

There are several options; in this example used in this work, the best two out of five random individuals are selected each time to have 2 children, and the 2 worst individuals of the population will die afterwards.

As a result, Table 1 summarizes the parameters of the method. They are only suggestions; every evolutionary algorithm requires parameter tuning to function effectively. In this scenario, finding the most complicated parameters does not require much effort because the problem can be addressed quickly using a genetic algorithm like the one provided or something similar.

Table 1 Algorithm's parameters

Representation	String of eight integers
Fitness function	Number of possible checks
Variation operators	Mutation and one point crossover
Mutation probability	90%
Parent selection	Best 2 out of 5 random individuals
Survival selection	Replace worst
Population size	500
Offspring size	2
Initialization	Random
Stopping condition	Fitness = 0 or 10000 cycles

The NEPs algorithm parameters are illustrated in Table 1. The representation of the solutions is constructed as a string of eight integers which the fitness function represented as the number of possible checks. The variation operators are the one-point crossover and the mutation. The mutation probability equals 90%, while the parent selection scheme is based on choosing the best 2 out of 5 random individuals. The survival selection is conducted according to the replace worst individual by the best. The population size equals 500, and the offspring size is 2. The initialization size is random. Finally, the stopping condition is when the fitness function equals 0 or when the number of cycles (iterations) reaches 10000.

2.3 General System Description

The Logic Flow paradigm [5] and Connection Machine [7] both include a basic architecture with regard to parallel and distributed symbolic processing that consists of multiple processors, each one of them is put in a node of a complete virtual graph and can handle data that is related to that node. Each node processor processes the local data according to certain pre-defined rules. After that, local data is transformed into a mobile agent that could navigate the network using a specific protocol. Only data that passes a filtering process may be sent.

This process of filtering might be required to meet conditions set forth through the receiving and/or sending processors. All nodes send their data at the same time, and all receiving nodes handle all arriving messages at the same time. See for example, [6,7]. Based on the premise that data could be delivered in strings, [3] proposes a notion referred to as a network of parallel language processors to study such concepts with regard to formal grammar and languages. Grammar systems, particularly parallel communicating grammar systems, are closely associated with language processing networks [2]. The essential notion is that any node of an underlying graph can have a language producing device (Lindenmayer system, grammar, and so on) that rewrites the strings in the node and communicates the strings to the other nodes. Strings could be successfully conveyed in the case where they pass through an input and output filter.

This approach has been modified in [1], which was motivated by cell biology. Each node's processor is an evolutionary process that is very simple. A node that can conduct relatively simple operations, such as point mutations in DNA sequence (substitution, insertion, or deletion of a couple of the nucleotides), is referred to as an evolutionary processor. More broadly, each node may be thought of as a cell with genetic information encoded in DNA sequences that can evolve through point mutations, which are local evolutionary events. Each node is dedicated to only one of such evolutionary operations. Also, data in every one of the nodes is arranged in a form of multi-sets of strings, with every one of the copies that have been processed in parallel to ensure that all of the conceivable evolution events occur. Those networks may be applied as a language (macrosets) generating devices or as computational ones, as shown in [1]. Those networks will be considered as language-generating devices, while their generative powers will be studied. Also, this study examines how extremely simple variants of such networks could be utilized to solve some other NP-complete problem. The variants that have been used here to solve the "three-colourability problem", the graph three-colorability issue is considered as a decision issue in graph representation that is asking if it might be expected to incorporate a color for each single vertex of that graph utilizing 3 colors as maximum which satisfies the criteria that each pair of laying next each other would have different color, are simpler compared to the ones used to solve "bounded Post Correspondence Issue", One of the most important variants of PCP is the bounded Post correspondence problem, which asks if we can find a match using no more than k tiles, including repeated tiles. A brute force search solves the problem in time O(2k), but this may be difficult to improve upon since the problem is NP-complete in [1].

It is worth noting that such a model is comparable in some ways to P systems. This novel computation model is inspired by hierarchical as well as modularized cell structure recently has been suggested in [9]. On the other hand, the model detailed here includes various unique features, such as a graph as the underlying structure, derivation mode, string communication mode, node filters, and so on.

2.4 Simple NEPs Formal Description

A simple NEP of an *n* size represents a construct $\Gamma = (V, N_1,N_2,...,N_n,G)$, where, V and G have an identical interpretation as for the NEPs, and for every $1 \leq i \leq n$, Ni = (Mi, FIi, POi, FOi, Ai, PIi) represents i^{th} evolutionary network node processor.
Ai and Mi from above are of the same interpretation as for evolutionary node in an NEP, however:

1. FIi and PIi are sub-sets of V that represents input filter. The input filter and output filter are defined by the conditions of the random context, PIi forms permitting context condition and FIi forms forbidding context conditions. A string $w \in V^*$, v is the set of all vertices. has the ability of passing input filter of node processor i, in the case where w contains every one of the PIi elements however, no FIi elements. It should be noted that any condition of the random context can be empty, in such case, the check of the corresponding context is omitted. We write $\rho i(w)$ = true, in the case of an ability to pass the input filter of the node processor i and, otherwise, it is written as $\rho i(w)$ = false.
2. FOi and POi can be defined as subsets of V, which represents the output filter. Similarly, a string might pass a node processor's output filter in the case where it satisfies random context conditions related to this node. Similarly, we write $\tau i(w)$ = true, in the case where there is an ability for passing input filter of a

node processor i and otherwise it is written $\tau_i(w)$ = false. The working mode of this simple NEP is only different in the step of the communication, it is namely said that configuration $C_1 = (L'_1, L'_2,...,L'_n)$ is changed directly to a C2 = (L1, L02,...,L0n) configuration through a step of communication, which is represented as C1 ⊢ C2 if for each $1 \leq i \leq n$,

$$L'_i = L_i \setminus \{w \in L_i \mid T_i(w) = \underline{\text{true}}\} \cup \bigcup_{\{N_i,N_j\} \in E} \{x \in L_j \mid \left(T_j(x) \wedge p_i(x)\right) = \underline{true}\}.$$

As far as the computation power of simple NEPs is concerned, there is no complete answer. Nonetheless, those devices have the ability for the generation of non-context-free language.

2.5 Three Color Problem

The Three-Color Problem is under what conditions can the regions of a planar map be colored in three colors so that no two regions with a common boundary have the same color, this is one of the problems tested in this thesis. One Graph K-coloring Problem: A K-coloring problem for undirected graphs is an assignment of colors to the nodes of the graph such that no two adjacent vertices have the same color, and at most K colors are used to complete color the graph [39].

As optimization problems has wide range of problems that might accept partial truth, uncertainty, and ambiguities it becomes important to use tools whose aim is to find out the optimum solutions. The 3-colorability problem may be solved in O(m+n) time with a 4 m+1 size MPNEP, where n represents number of the vertices and m represents number of input graph edges. Assuming that G = ({1,2, ⋯, n}, {e_1, e_2, ⋯, em}) a graph and assuming that $e_t = \{k_t,l_t\}, 1 \leq k_t \leq l_t \leq n, 1 \leq t \leq m$. The alphabet U = V ∪V ′ ∪T ∪A is considered, where V = {b,r,g}, A = {Aˆ$_1$,Aˆ$_2$,⋯,Aˆ$_n$} and T = {a_1,a_2,⋯,a_n}.

The following processors of a massive parallel NEP are constructed

- A generator processor:
  N_0 = {{$a_1a_2 \cdots a_n$},
  {ai → bAˆi,a_i → rAˆi, a_i → gAˆ i |1 ≤ i ≤ n},
  ϕ, {Aˆ i |1 ≤ i ≤ n}}
  Where N_0: number of processors.
  a_1... a_n: processor units.
  b,r,g: three vertices.
  such a processor is responsible for the generation of all of the potential combinations of the color, solutions or not, to the problem. And it transmits these strings to the following processors.

- For every one of the edges in graph et = {k_t,l_t}, there are four filter processors (where i = {kt,lt}):
  $N_{e_t^l}$= {φ, {gAˆi → g′ai,rAˆi → r′ai},
  {Aˆi},{g ′,r ′ }}
  $N_{e_t^2}$ = { ϕ,{gAˆ i → g ′ ai,bAˆ i → b ′ ai},

{Aˆi},{g′,b′ }}
$N_{e_t^3}$ = {ϕ, {bAˆ i → b ′ ai,rAˆ i → r ′ ai},
{Aˆi}, {b ′, r ′}}
$N_{e_t^4}$ = {ϕ, {r ′ ai → rAˆi,g ′ai → gAˆi,b ′ai → bAˆi}, {ai},{Aˆi}}

It is indicated that MPNEP can be built with previous nodes such that all of the potential colored strings are generated by N0 and after that apply processors $N_{e_t^l}$, $N_{e_t^2}$, $N_{e_t^3}$ $N_{e_t^4}$ to filter the strings for edge e_t. A valid solution to a specific problem is provided when repeating this filtering process with the remaining edges. MPNEP with the architecture mentioned above could solve 3-colourability issue of n cities that have m edges. For 1st n steps, which are evolution ones where noting has been communicated, strings remain in N_0 until no more letters in T appear in them. By the end of this process, the strings obtained encode all of the potential ways of colouring vertices, satisfying or not the problem requirements. Then, one step is required for communicating all potential solutions to the following processors. After that, for every one of the edges e_t, MPNEP keeps only the strings that encode a colourability that satisfies the condition for 2 et vertices. Which is carried out through nodes $N_{e_t^l}$, $N_{e_t^2}$, $N_{e_t^3}$ finally $N_{e_t^4}$ in 12 steps. As can be seen, the entire computational time was 12m+n+1. This proof is concluded by stating that the total number of the rules was 18 m + 3 n + 1, and all of the network parameters are of an O (m+n) size.

2.6 Cloud Computing

By enabling the complicated calculations to provide insights, knowledge, and experimental proof for scientific discovery, data analytics have demonstrated their potential to offer decision support in the administrative, financial, and scientific areas. Yet, the amount of data that must be analyzed is increasing exponentially, and experts in the technology of data analytics like machine learning and data mining frequently lack the domain knowledge needed to understand the data. As a result, academics have started developing analytic tools to help with complex data analysis. Those technologies are frequently designed for specialized data domains and may not offer ubiquitous access or scalability that is required for big data analysis. The problems can be addressed with cloud computing technology, yet a service infrastructure should be in place to provide the required resources transparently on the cloud [6]. As a result, cloud technology was developed to address this issue.

The fast technology advancement and ease of the accommodation for implementing business operations had enriched the organizations' growth. Such technology advancement had emerged as well in massive quantities of data that is obtained from multiple sources. For instance, the customer monitoring and feedback, daily transactions, web-site tracking, tweets, and blogs that are associated with the marketing activities, consumer preferences etc. [10]. Those data have different formats, such as structured and unstructured formats.

Moreover, this data is accumulated regularly at a high rate, which makes it challenging to deal with that ever-increasing data quantity using traditional data analytics methodologies [11]. The capability of the cloud analytics methodologies for mining and cross-relating different categories of data obtained from multiple places for extracting details regarding the prerequisites and preferences of users, estimations of

customers' needs, sales patterns, etc. makes organizations to plan their policies to maximize resource profits and utilization [10,12]. This could replace the intuition based on the idea of the processes of the decision making with the processes of the informed decision making that have been referred to as the Business Intelligence (BI) as well [13,14].

Analytics solutions have been classified into prescriptive, descriptive, and predictive solution types [10]. Historical data have been used for identifying hidden patterns and for producing reports in descriptive solutions. Historical and present data have been utilized for analyzing and predicting future patterns in predictive solutions. The assistance to the analysts in the decision making has been provided through the prescriptive analytic solutions in which the actions have been characterized along with the specifications of the business effect. Effective data analytics solution inclusions would provide the organizations with the capability of identifying and adapting to steadily updated customer and business requirements. Implementing on premise analytic solutions requires vast infrastructure investments, software licenses, computing resources expertise, and consultancy for organizing and integrating the data [10,11].

2.6.1 Cloud Technology

Cloud computing is the on-demand delivery of IT resources over the Internet with pay-as-you-go pricing. Instead of buying, owning, and maintaining physical data centers and servers, you can access technology services, such as computing power, storage, and databases.

Cloud computing may be defined as a new processing type that offers a scalable, virtualized infrastructure as a service, allowing users to use supercomputing capability without investing in expensive infrastructure or management. This ability to scale indefinitely has paved the way for many cloud-based data analytics systems which can handle massive amounts of data. As a result, what distinguishes cloud analytics is its ability to process big data. Newer applications for corporate intelligence, semantic web search, video surveillance search, medical image analysis, and conventional data-intensive scientific applications such as satellite image pattern matching could be handled. These apps have one thing in common: they are all incredibly parallel, and their data access bandwidth requirements outnumber other resource requirements. Over a partitioned dataset, including data-intensive applications could be simply divided into small parallel computations.

This is a great match for Google's MapReduce framework, Hadoop, Spark, Amazon, which offers a basic programming model based on map and reduce functions over the pairs of key/value which might be parallelized and run on a large machines' cluster. A well-known platform for constructing cloud data analytics applications is an open-source version of MapReduce created under the Apache Hadoop project. Large distributed clusters of inexpensive servers, along with a layer of server virtualization and parallel programming libraries, make up the fundamental architecture for cloud computing. A storage layer developed for supporting the next features is one of the significant infrastructure elements regarding cloud stack for data analytics applications: (i) scalable – storing petabytes of data, (ii) highly reliable – handling often-occurring failures in the large systems, (iii) low-cost – maintaining cloud computing economics, and (iv) sufficient – to best use resources of network and storage [15].

2.6.2 Cloud Services

When moving the computations to the cloud, it is essential to first grasp the distinctions and benefits of different cloud services. Software as a Service (SaaS), Infrastructure as a Service (IaaS) and Platform as a Service (PaaS), is the 3 cloud service models to compare. The differences between PaaS, SaaS, and IaaS must be understood.

2Table 2: Cloud Services Types

Platform Type	COMMON EXAMPLES
SaaS	DROPBOX, CISCO WEBEX, SALESFORCE, GOOGLE APPS, GOTO MEETING, CONCUR
PaaS	AWS ELASTIC BEANSTALK, WINDOWS AZURE, HEROKU, FORCE.COM, GOOGLE APP ENGINE, APACHE STRATOS, OPENSHIFT
IaaS	DIGITALOCEAN, LINODE, RACKSPACE, AMAZON WEB SERVICES (AWS), CISCO METAPOD, GOOGLE COMPUTE ENGINE (GCE), MICROSOFT AZURE

Table 3: the main differences between the SaaS, PaaS and IaaS

Platform Type	Platform's characteristics
SaaS	. Software as service. . Operating environment largely irrelevant, fully functional applications provided, e.g. CRM, ERP, email
PaaS	. Platform as a service. . Operating environment included, e.g. Windows/ .NET, Linux/j2EE, applications of choice deployed
IaaS	. Infrastructure as a service. . Virtual platform on which required operating environment and application are deployed. . Includes storage as a service offerings

The main focus of this report will be dedicated towered using the Software as a Service. The SaaS provide analytics software over the cloud on the subscription-based

model. They have evolved into a critical choice for firms looking to avoid upfront capital costs and quickly implement new business process requirements. The SaaS can be utilized for data analytics, predictive analytics, and business analytics to uncover facts and data patterns. SaaS is considered as an efficient way to deal with Big Data. It analyzes and stores knowledge from big data in scalable and cost-effective manners [16]. In general, the cloud-based systems are represented in Figure 2.

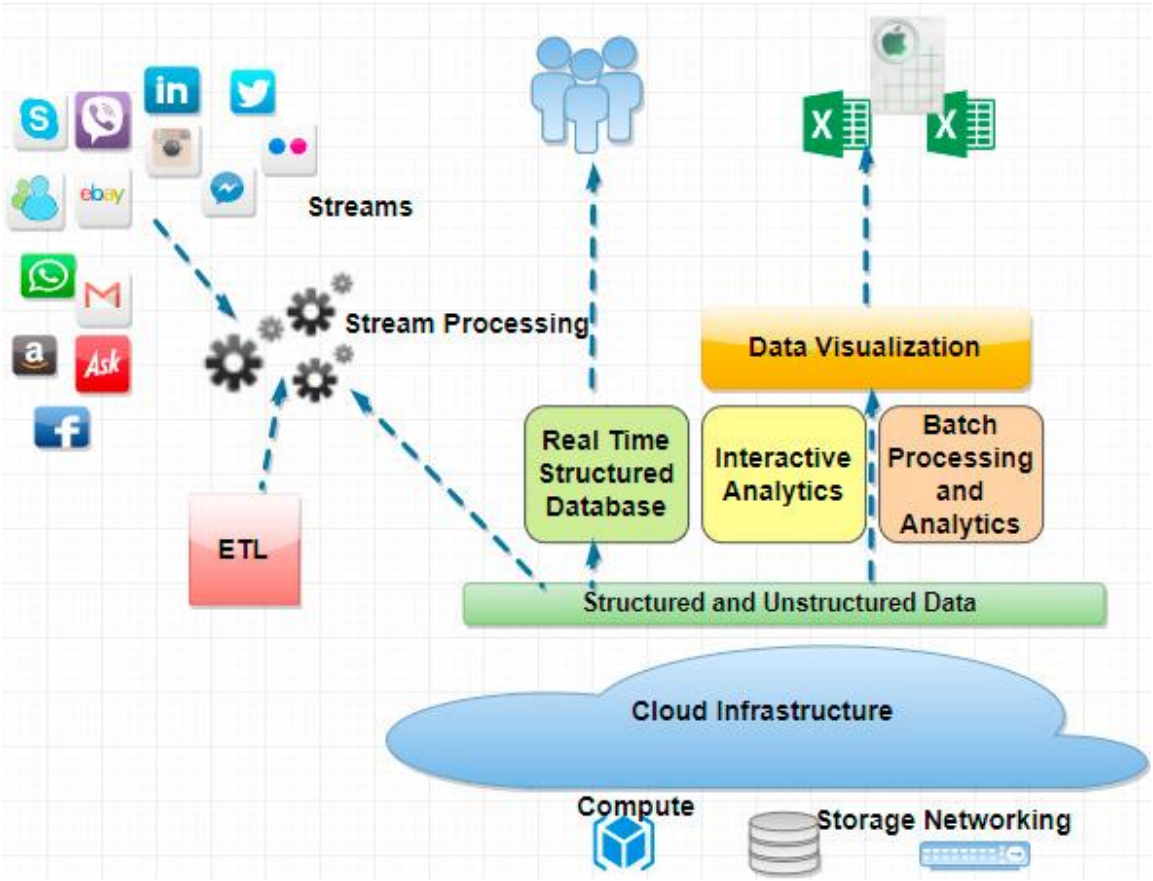


Figure 2: High-Level Architecture of Cloud-Based process used for Big Data

Figure 2 shows the high-level design of the cloud system used to handle big data processing operations. The figure clearly indicates that multiple data sources are all used to stream data, including the (ETL) extract, transform, and load data source. The stream data will all be transformed into a specific format and then sent to the cloud ecosystem. In turn, the data will be either stream after processing in the cloud as real time structured data or it would be used in an interactive analytics purpose, or it might be batched to be processed as collections and then analyzed for the final outcomes. In all representation methods, the data would be visualized in the form of tables or figures.

2.6.3 Cloud Technology Overview

According to the information as mentioned earlier regarding the necessity of using advanced tools to deal with data in large volumes and fast-streamed, it becomes necessary to use cloud technology to handle data of this kind of format. Cloud analytics can be defined as one of the service models in which at least one analytics component is implemented in the cloud. The cloud approach enables businesses to increase their analytics capabilities as their business grows. This service model is becoming more

prevalent in today's corporate intelligence platforms [17]. Based on what has been explained earlier, Cloud Analytics is a tool that helps to process Big Data characterized with all or some of the Big Data properties like Volume, Velocity, and Variety, etc. This literature review gives an extensive explanation to show the advantage of using cloud Analytics in different applications. The cloud Analytics technology has evolved.

By utilizing hardware virtualization, Cloud Computing allows for the storage of massive amounts of data over the Internet. As a result, Big Data's availability, accessibility and scalability have increased [18,19]. Furthermore, through a service that is known as Big Data as a Service (BDaaS), cloud computing provides special statistical tools for the resourceful analysis and processing of big data [9,12]. As a result, big data and cloud combine to provide value to businesses by improving elasticity, accessibility, agility, and ease of processing of cloud-based big data and lowering the costs of ownership and implementation complexity of the solutions of the big data [20].

The cloud analytics subject is a fast-growing research area. Many researchers strive to take advantage of its robustness, scalability, distribution, and high intelligence technology. Since the last few years, cloud analytics has been an active research area, particularly with the remarkable growth of data usage in all aspects of life. In the sentiment analysis, Cloud Analytics plays an important role that was reviewed by Francesco Benedetto et. al. in [21]. Similarly, by utilizing the pay-per-use model and cloud computing technologies and service models, social media firms may easily manage the users' data, improve their service quality, and lower data and infrastructure network management expenses [22].

[23] proposes a disaster response strategy based on sentiment analysis, which collects disaster data from social networks and classes it depends on the requirements of those affected individuals. For studying people's sentiments, the categorized disaster data is classified using an ML algorithm. Many features like parts of speech and lexicon are studied to choose the optimum classification approach for disaster data. The findings show that a lexicon-based technique is appropriate for evaluating people's demands throughout disasters. The real-time classification of social media big data for disaster responses and recovery is a practical use of such approach. This research aids emergency responders and rescuers in developing more efficient techniques for managing information in a fast-changing disaster scenario.

Roger S. and colleagues presented a cloud analytics model. Project Daytona: Data Analytics as Cloud Service [24] was the name given to it. It is a cloud-based data analytics service built on Daytona, an iterative Map-Reduce runtime specialized for data analytics. In this approach, Excel and other conventional client applications offer data entry and user interaction surfaces, while Daytona offers a scalable cloud runtime for data analytics. Any analyst may utilize such data analytics services to discover and import data from the cloud, use algorithms of cloud scale data analytics for extracting information from large datasets, visualize the data, and store it back to the cloud using a spreadsheet.

On the other hand, with the popularity of cloud service, there is a remarkable increase on the needs for virtual resources to the vendors of the service. New method challenges like instant cloud resource provisioning and effective capacity planning are required with business developments. In the study that has been presented in [25], it was described a model that improved the service quality for capacity planning and the

problem of instant cloud resource provisioning. Both issues were first formulated as a generic problem of cost-sensitive prediction.

Heterogeneous and asymmetric NEPs approach for quantifying the prediction error was introduced, considering the very dynamic environment of the cloud. Finally, an ensemble prediction approach was created by combining the prediction power of a number of prediction algorithms using the suggested measure. A group of the experiments using IBM Smart Cloud Enterprise (SCE) trace data shows that this strategy has the ability for the significant improvement of the service quality through lowering the time of resource provisioning, whereas keeping cloud overhead low.

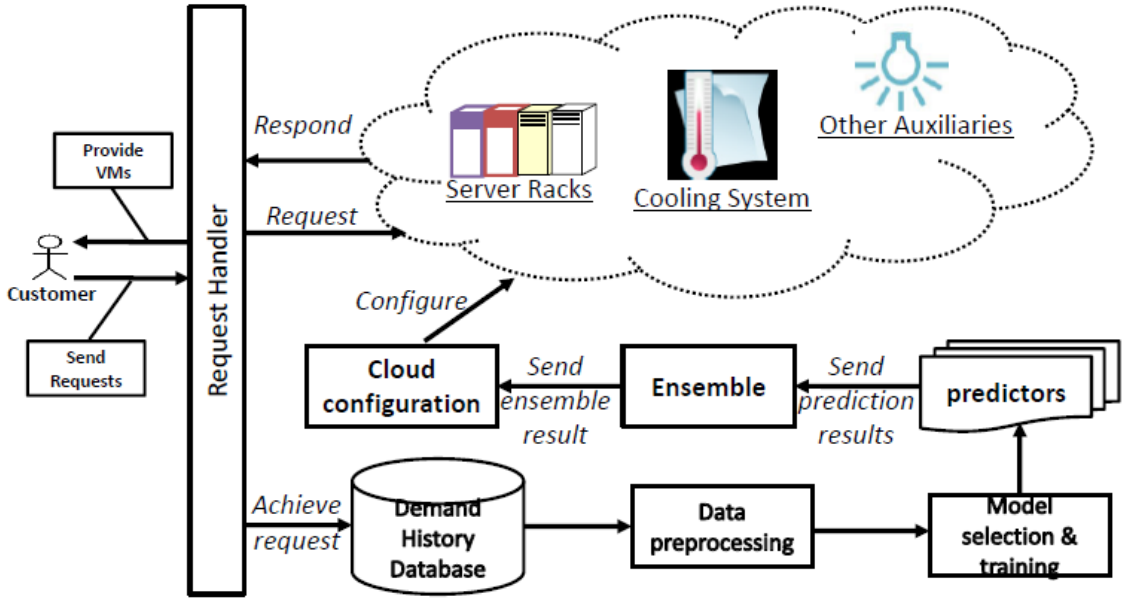


Figure 3: The system outlines of [25]

As observed in Figure 3 the request handler is used to accumulate the customer information. The user requests are sent to this terminal. In return, the request handler component will send the output feedback to the customer. The operations conducted behind the request handler can be summarized as depicted in the figure. The sent data request will be managed to be handled in the server racks and other auxiliaries. The other terminal used to handle the user requests is the demand history database, where the data mining operations such as prediction, classification and preprocessing will be conducted.

To do data analysis, you will need specialized knowledge of analytics algorithms. As a result, there are several things to consider when a production manager who is not a data analyst does analysis responsibilities. To address this issue, the research presented in [26] a manufacturing-specific algorithm template of data analytics and a cloud-based big data analytics platform for enabling the ones who are not experts in data analyses to undertake the tasks of analysis.

The manufacturing-specialized d template of the data analytics algorithm is a concept that considers data properties in relation to manufacturing challenges to systematically provide the preprocessing, data mining, and visualization algorithms needed for executing analysis tasks. In addition, a cloud-based platform of big data

analytics is meant to analyze data using algorithm templates. It can not only store and manage data per company but also create analyses models and confirm results. The platform comprises interconnected middleware, batch manager, data storage, analysis engine, and a data analytics modeler.

2.7 Hadoop Environment

Hadoop is software that is utilized for the analysis and processing of large quantities of data [44]. It is designed to be used to handle big data. Big data has a large volume, has varied data types, and it is received in high velocities. For this reason, the Hadoop technology was invented as a method that can efficiently deal with such kind of data. The Hadoop architecture is composed of two main components, which are the Map and Reduce. The map is the operation of dividing the big chunk of the data block into small portions, which are all be proposed individually. This technique makes it easier for the algorithm that is traditionally used with small amounts of data to be successfully used with big data. After dividing the data into many portions and processing them individually, it comes time to regroup the outcomes using the Reduce function.

The Apache Hadoop is considered a library allowing for the decentralized processing of large datasets amongst clusters of computers using direct programming functions. It is implemented for the reason of upgrading from few servers to thousands of nodes. Each node is considered as a local computation and storage. Instead of depending on hardware for delivering high availability, the library itself is made for detecting and addressing errors at the application layer, thereby delivering a highly-available service on top of a cluster of computers, every one of which might be vulnerable to errors.

The Hadoop architecture is composed of the master/slave idea of the distributed system. That means the processing of data is not conducted on a standalone computer. Rather it is distributed among several interconnected processing devices. That means the distributed or the decentralized architecture of the Hadoop system facilitates the processing of the Hadoop architecture is also named the Hadoop ecosystem. This system is shown in Figure 4.

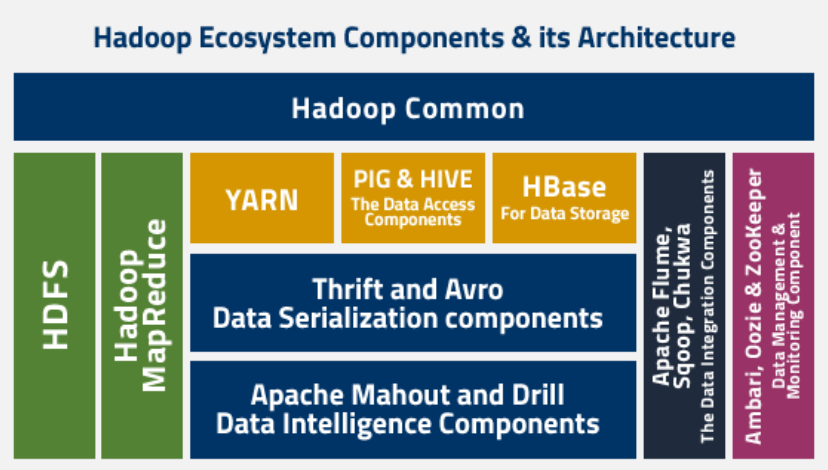


Figure 4: Hadoop ecosystem components and it is architecture

The Hadoop Distributed file system holds the input data to be processed using the two functions, which are the map and reduce. The map and reduce component, which is one of Hadoop's most important components, is responsible for dividing the data and, after processing it, group the data using the reduce. The data serialization process, the thrift and Avro is a row-oriented remote procedure call and data serialization framework developed within Apache's Hadoop project. This process is responsible for making the reduced data portions ready for further processing. The Apache Mahout. is a project of the Apache Software Foundation to produce free implementations of distributed scalable machine learning algorithms focused primarily on linear algebra and drill data components which is also one of the necessary components of the Hadoop.

It is a distributed and decentralized software that is operated on a cluster of processing units. This cluster of computers is interlinked using wireless networks. Hadoop is an open-source software that anyone can use for free. It is used to analyze, process and deal with big data. As the big data, unlike traditional data, the techniques used to handle the big data using the traditional data processing system seemed to be inefficient in comparison to the processing conducted using the Hadoop technology. Hadoop can be used in different applications that deal with large quantities of data. for example, it can be used in hospitals, businesses, education and others.

Hadoop can be built with one or a few devices, and we can add more devices if we require more storage or processing power without affecting the system that can support the scalability. It can automatically handle faults because Hadoop employs many devices and replicates data across them (replication). It will not stop working if one of them fails. Hadoop will automatically duplicate copies of data whenever a device fails. Hadoop processes the data in its locations in the cluster devices so that each computer in the cluster processes the data stored in it. All of this is done simultaneously.

2.8 Spark in Big Data

Apache Spark is one of the essential tools used to analyze data in parallel. Therefore, in this section, we will be going to explain all the operational and functional issues related to the NEPs system design. The significance of using the spark environment is highlighted in this section, where the related programming codes are thoroughly explained in its design.

Similarly, to Hadoop, Spark can be defined as an open-source and under Apache Software Foundation wing. In fact, the open-source indicates that a code may be utilized freely used by anybody. Also, it may be altered as well by anybody for producing customized versions that are aimed at certain industries or issues. The volunteer developers, who work at companies producing customized versions, continuously refine and update core software, adding more efficiencies and features. Spark has been considered by the tech experts such as programmers, software engineers, and academics in the industry as a more advanced product compared to Hadoop - it is more innovative and has been designed to be working through data processing in chunks. This indicates the fact that it transfers the data from physical, magnetic hard discs to considerably faster electronic memory in which the processing may be performed with a considerably higher speed - about 100 times faster in some of the operations.

Spark had been proven to have quite high popularity. Several large companies utilize it for massive analysis and storage of multi-petabyte data, which was partially a result of its high speed. In addition to that, Spark was proven highly suitable for the applications of Machine Learning, which is a very fast-growing and exciting computer science area, where the computers are being taught how to be spotting the patterns in the data and adapting their behavior according to the automated modelling and analyses of any type of task they attempt to carry out.

To make it available to a wider variety of businesses, numerous vendors are providing their own versions (like with the Hadoop) that have been produced toward certain industries or custom-configured for the individual clients' projects, besides related services of consultancy for the purpose of getting it operating. The Spark utilizes cluster computing for the computational (i.e. analytical) power besides its storage. This indicates that it might be using resources from several computer processors linked together for the analytics. It is a scalable solution, which means that in the case where more power is required, one can simply introduce a higher number of processors to the system. With the distributed storage, the massive datasets that have been collected for the Big Data analysis may be stored over numerous smaller separate physical hard discs, which results in speeding up the operations of read/write because the "head" reading the information from discs is of a lower physical distance for travelling over the surface of the disc. Concerning the power of the processing, there is a possibility for the addition of more storage whenever required. The fact it utilizes widely available hardware of commodity (any of the standard hard discs) keeps the costs of the infrastructure down.

In contrast to Hadoop, the Spark does not come with a dedicated file system – rather than that. It may be integrated with several file systems, which include MongoDB, Hadoop's HDFS, and Amazon's S3 system. An additional framework element is the Spark Streaming, allowing the development of the applications, performing the analytics on the streaming, real-time data – lime the automatic analysis of the social media data or videos - in real-time.

2.8.1 Overview of the Spark Architecture

Now that we have some idea about the components of the Spark ecosystem to understand the architecture behind Spark and how the components are related to each other, refer to Figure 5.

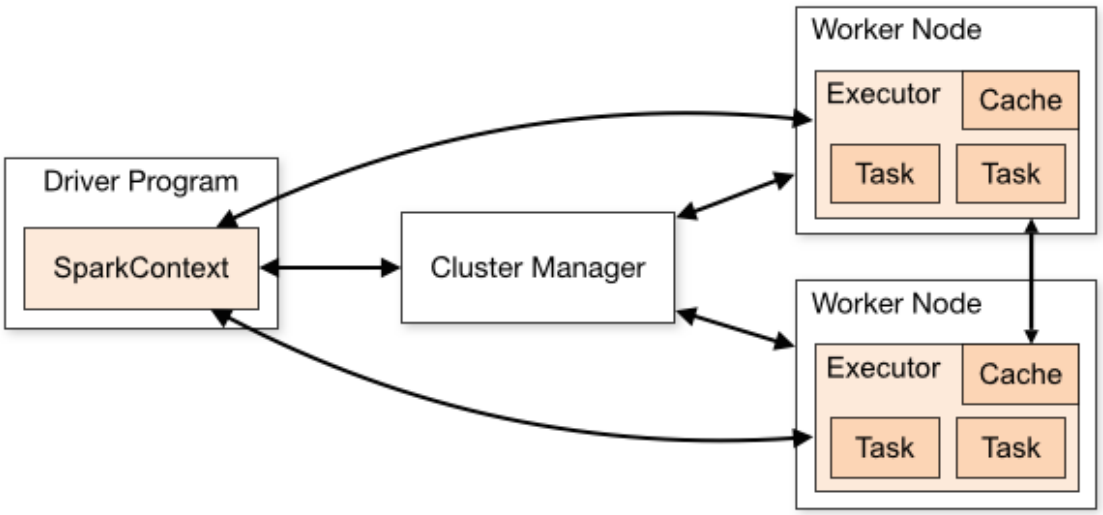


Figure 5: Spark ecosystem [45]

As you can see in Figure 5, we have a Driver Program that runs on the master node on the left. The master node is the one that is responsible for the entire flow of data and transformations across the multiple worker nodes. Usually, when we write our Spark code, the machine we deploy acts as the master node. After the Driver Program, the very first thing that we need to do is to initiate a SparkContext. The SparkContext can be considered a session using which you can use all the features available in Spark. For example, you can consider SparkContext as a database connection within your application. Using that database connection, you can interact with the database. Similarly, using SparkContext, you can interact with the other functionalities of Spark.

There is also a Cluster Manager installed that is used to control multiple worker nodes. The SparkContext that we have generated in the previous step works in conjunction with the Cluster Manager to manage and control various jobs across all the worker nodes. Whenever a job has to be executed by the Cluster Manager, it splits up the entire job into multiple individual tasks, and then these tasks are distributed over the worker nodes. This is taken care of by the Driver Program and the SparkContext. As soon as an RDD is created, it is distributed by the Cluster Manager across the multiple worker nodes and cached there.

On the right-hand side of the architecture diagram, you can see that we have two worker nodes. In practice, this can range from two to multiple worker nodes depending on the workload of the system. The worker nodes act as the slave nodes that execute the tasks distributed by the Cluster Manager. These worker nodes return the execution result of the tasks to the SparkContext. A key point to mention here is that you can increase your worker nodes such that all the jobs are distributed to each of the worker nodes, and as such, the tasks can be performed parallelly. This will increase the speed of data processing to a large extent.

2.8.2 Understanding Spark Ecosystem

From Figure 2.5, we can see that Spark is built on top of its core engine known as the "Apache Spark Core". This is the all-purpose general execution engine used to run and execute all the other functionalities within Spark. All the other components like Spark SQL, Spark Streaming, MLlib, and GraphX work in conjunction with the Spark Core engine. Spark SQL is one of the most common features of the Spark processing engine. This allows users to perform data analysis on large datasets using the standard SQL language. It also allows us to run native Hive (one of the Hadoop components) queries on the existing Hadoop environments available. Spark SQL can be used to extract and run data transformation queries as well. Spark Streaming is used to run analytic workloads on top of fast-moving streaming data is possible with the help of this unique feature in Spark. It helps to analyze large volumes of data as and when they arrive by running special operations on the data. It continuously uses the Spark Core engine to inject data in a small-scaled cluster and performs RDD (will understand later in the article) on those MLlib (Machine Learning) is one of the profound capabilities that most users desire to implement using Spark.

Running Machine Learning algorithms on top of the Spark Core engine is done with the help of MLlib. It leverages the in-memory distributed data structures for training the data models, which is relatively faster as compared to the previous version of Apache Mahout GraphX. It is a distributed graph data processing engine built using the Spark Core engine. On a very high level, it extends the functionality of the Spark RDD by creating a Resilient Distributed Property Graph. It is a simple structure that associates nodes and their properties by using vertices and edges.

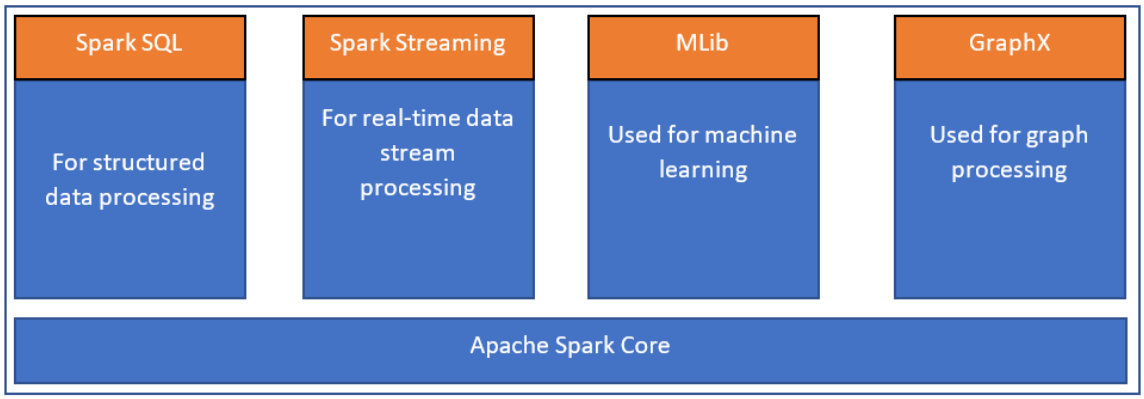


Figure 6: Apache spark components

2.9 Scalability problems

Scalability can be defined as the measure of the capacity of a parallel system for increasing the speed-up proportionately with the processors' number, and it represents the capability for handling more work with the growing size of application or computer. The scalability can be defined as the efficiency of the parallelization — i.e. the ratio between actual speed-up and optimal speed-up that is obtained in the case of utilizing A specific number of the processors.

- Increased number of processors --> reduced efficiency
- Increased problem size --> increased efficiency

A scalable parallel system keeps efficiency through the increase in the number of processors and the size of the problem simultaneously.

A scalable parallel system may always be cost-effective by adjusting the number of processors and the size of the problem, there is an importance for the evaluation of the parallel algorithm's scalability at an early development stage. Scalability is commonly utilized to describe how problem size and system sizes affect parallel computers and algorithms' performance [39]. Applications can generally be divided into strong scaling and weak scaling applications.

In case of strong scaling, the number of processors increases, whereas the size of the problem stays without change. This also leads to decreased workload for each one of the processors. Strong scaling is mostly used for long-running CPU-bound applications to find a setup that results in a reasonable runtime with moderate resource costs. The individual workload must be kept high enough to keep all processors fully occupied. The speed-up achieved by increasing the number of processes usually decreases more or less continuously.

Optimally, an issue would be scaling linearly, which means that a program would be speeding up by a factor of N in the case where it operates on a machine with N nodes. (in fact, as $N \rightarrow \infty$ proportionality can't hold, due to the fact that the time of the communication has to be then dominating. Clearly, the aim of solving an issue scaling strongly is decreasing the time required for solving the issue with the use of a more powerful computer. Those are usually CPU-bound issues and are the hardest ones to yield something close to the linear speed-up.

The amount of time needed to complete a serial task t (1), and the amount of time that is needed for completing the same work unit with N processing elements (parallel task) is t(N), then Speedup is given as:

$$\text{Speedup} = t(1)/t(N)$$

Amdahl's law and the strong scaling in 1967[41], Amdahl stated that speed-up is bound by a fraction of the serial part of software that is not amenable to the parallelizations. This law may be represented as:

$$\text{Speedup} = 1/(s+p/N)$$

It is nearly always worthwhile measuring one's job's parallel scaling. The strong scaling measurement is carried out by testing the way the overall job's computational time scales with the number of the processing elements (either MPI processes or threads). In contrast, testing for weak scaling has been carried out by increasing the size of the job and the number of the processing elements. The parallel scaling test results provide a good indication of the number of resources needed for requesting a specific job size.

In the case of strong scaling, the number of processors increases, whereas the size of the problem stays constant. This yields as well in a reduction in the workload for each processor.

In Table 4, T(1) is the time needed to complete a serial task t (1) (here it is 1 processor). While T(Np) is the amount of time until completing the same work unit with the Np processing elements (i.e. the parallel task) (here it is Np number of different processors). N1 is number of processors needed to complete a serial task (here it is 1 processor) And Np is a number of the processors needed to complete a parallel task. Table 4 shows the experiment of strong scaling parameters and the relationship between the number of processors and execution time with Amdahl's Low (Speed up) calculation

and efficiency, we realize that every time we increase the number of processors, the execution time decreases, the speed increases and the efficiency decreases. until you reach the peak point, which is 2048 (52 knots) in the experiment, then the required time increases, and the speed up decreases due to the large number of connections that take more time than the execution time itself.

Table 4: Strong Scaling parameters [46]

#problem size (N x N) = 1600000000, where N is Matrix size and N = 40000 for all different processor numbers			
#Processors Np	#time in seconds T	(Amdahl's law) #speeedup = (T(1)/T(Np))	#efficiency = (T(1)xN1 / T(Np)xNp)
1 (1 node)	64.424242	1	1
2 (1 node)	33.901724	1.90	0.95
4 (1 node)	17.449995	3.69	0.92
8 (1 node)	8.734972	7.38	0.92
16 (1 node)	4.789075	13.45	0.84
32 (1 node)	2.749116	23.43	0.73
64 (2 nodes)	1.627157	39.59	0.62
128 (4 nodes)	1.017307	63.33	0.49
256 (7 nodes)	1.436728	44.84	0.18
512 (13 nodes)	3.689217	17.46	0.03
1024 (26 nodes)	4.709213	13.68	0.01
2048 (52 nodes)	21.462228	3.00	0.001

2.10 Conclusion

This chapter was concerned with giving an extensive theoretical background about the main concepts presented in this proposed work. Some of these concepts were used or designed in some other related work to the one proposed here in this thesis. As a quick review of the major principles mention here in this chapter is the cloud technology importance in the NEPs implementation. For instance, the way that the Hadoop Spark is distributed on the cloud has been highlighted fully in this chapter. As another matter explained in this chapter is the heterogeneous and asymmetric NEPs approach for quantifying the prediction error was introduced, considering the very dynamic environment of cloud. Finally, an ensemble prediction approach was created through the combination of prediction power of a number of prediction algorithms using the suggested measure.

CHAPTER 3. STATE OF THE ART

3.1 Background

The improvement of the NEP model and its variation follows the order introduced underneath. The Connection Machine and Logic Flow are two computational models intended for data processing. They are made of a set of processors associated with the same environment. Everyone contains neighborhood information and decides that process on the information. At the point when two of them are associated, they can share their neighborhood information. The associations do a sifting cycle to permit or disallow a few information to pass. Every one of the neighborhood information is sent and received simultaneously in an identical way. As is examined exhaustively, the overall construction and working standards are the same as the NEP model. Later on, [27] introduced a model called organizations of equal language processors. The creators were keen on growing novel thoughts from past works like language structure frameworks [28]. They were centered around the potential outcomes of parallel structures to produce formal sentence structures. The primary thought was exceptionally centered to the NEP idea. Every processor control words with specific modifying activities.

On top of that, processors are associated following a chart definition. Words can move from one processor to other in case they can pass yield/input channels in the sending/getting processor. At long last, [29] first utilized the term Network of Evolutionary Processors. The primary thought was propelled from cell science. Once more, various essential processors are set in an organization. Processors control words by fundamental tasks that remind transformations in DNA grouping, which is the principal motivation to call them "developmental". The entire model was enlivened by cell science and DNA components: words are DNA successions that can develop because of nearby tasks, and processors are cells sharing data through their layers. The first activities or rules were addition, erasure and replacement, and every hub was had practical experience in just one of them. The model expects a subjectively huge number of duplicates is accessible for each word and each standard that can be applied to a word is really applied. This last point is the beginning of the inborn parallelism of NEPs calculation. Besides, correspondence between the processors functions as in past models.

After the original work of [29] various investigations were distributed where NEPs were utilized as producing gadgets (likewise called GNEP), as tolerating gadgets (additionally called ANEPs) and as general PCs, in which case their computational force was contemplated. Various types of channels were introduced, basically two sorts. Initially, enrollment channels which characterize the yield/input channels with a set of words or an ordinary articulation [30]. The subsequent one is situated in arbitrary conditions [31], which are more conceivable and easier to execute. Furthermore, the main papers force numerous imperatives of adaptability on utilizing various types of rules or channels in similar NEPs. At the same time, later methodologies don't have any significant bearing these limitations [31]. A few researchers think about those NEPs systems and, thus, call them Hybrid Networks of Evolutionary Processors. At last, different variations have been introduced like those utilizing such rules [28,31], additionally called nets of joining processors NSP.

As referenced, there exist numerous NEP variations in the writing. Every one of them shares similar general qualities. A NEP is worked from the accompanying components:

a) A number of input items comprise the letter set of the words controlled by the processing units.
b) A number of processing units.
c) A basic chart where every vertex addresses a processor, and the edges figure out which processors are associated with trading words.
d) An underlying design characterizing which words are in every processor toward the start of the calculation.
e) at least one halting principle to end the NEP.

A transformative processor has three fundamental segments:

a) A bunch of transformative guidelines to change its words.
b) An info channel that indicates which words can be gotten from different processors.
c) A yield channel that delimits which words can pass on the processor to be shipped off others.

The variations of NEPs fundamentally contrast in their developmental standards and channels. They perform fundamental activities, such as changing the words by supplanting every one of the events of an image by another or sifting those words whose letter in order is remembered for a given arrangement of words. NEP's calculation substitutes developmental and correspondence steps: a transformative advance is constantly trailed by a correspondence step and the other way around. The calculation follows the accompanying plan: when the analysis begins, each processor has many introductory words. From the outset, a transformative advance is played out: the principles in every processor adjust the words in the processor. Then, a correspondence step drives a few words away from their processors and powers the processors to get words from the net. The correspondence step relies upon the limitations forced by the associations and the yield and information channels. The model accepts that a subjective number of duplicates of each word exists in the processors. Along these lines, every one of the principles materials to a word are really applied, bringing about another word for each standard. The NEP stops when one of the halting conditions is met, for instance, when the arrangement of words in a particular processor (the yield hub of the net) isn't unfilled. Definite, proper portrayals of NEPs can be found in [30,31]. There are various variations of NEPs, and, in this way, formal definitions can differ somewhat.

A great deal of examination exertion has been dedicated to the meaning of various groups of NEPs and to the investigation of their conventional properties, for example, their computational fulfilment and their capacity to take care of NP issues with polynomial execution. In any case, no critical exertion, aside from [32], has attempted to foster a NEP test system or any execution. Tragically, the product portrayed in this reference gives the chance of utilizing just a single sort of rules and channels and, what is more significant, disregards two of the primary standards of the model: 1) NEP's calculation ought not be deterministic and 2) transformative and correspondence steps should substitute rigorously. In fact, the product is engaged in taking care of choice issues equally, instead of mimicking the NEP model with every one of its subtleties.

In [33] jNEP framework has been proposed with a visual apparatus to plan the NEPs viable graphically. A space explicit visual language for NEPs was planned through AToM3. The AToM3's diagram language modules were taken to mechanize

some mechanical and tedious planning errands, for example, appropriately putting channels near their processors and characterizing a few sorts of standard chart structures.

In [34] a stage with which the creators carry out an overall approach to naturally plan NEPs to tackle explicit issues. CGE/AGE (another hereditary programming calculation) and jNEP (a Java NEP test system) were utilized. This work is only a proof of reasonability. All modules were connected and created the underlying populace. Building this stage is important because this system incorporates a few non-inconsequential advances, like planning a syntax, and carrying out and utilizing a test system.

Modern simulation of NEPs in [40], the author run the NEPs in cloud by mean of web services in the *blockly* platform.

In [35] jNEP, a Network of Evolutionary Processors (NEP) test system, a few representation offices have been improved. jNEPView cordially showed the organization geography and showed the total depiction of such state in each progression. Utilizing this technique, it is simpler to program, and study NEPs.

In [36] The point is to introduce the proper system where the interdisciplinary investigation of normal language is led by coordinating etymology, software engineering and science. It gives an outline of the field of exploration, passing on the principal thoughts that have affected examination in phonetics. Particularly, this work features the primary strategies for atomic figuring applied to the preparation and investigation of the construction of ordinary language. Among them, DNA registering, layer processing and NEPs are the most significant computational models that have been adjusted to represent perhaps the most obscure limits of individuals.

3.2 Equipment for Decentralized Computing

As we portray more detail in additional passages, we propose adjusting an overall philosophy to figure any normal PC to NEPs. In [37] attempts to stay away from an expert approach by explicitly dismantling the correspondences needs and utilizing best in class apparatuses to help them. A portion of these devices (protobuff, the AMQP convention and its RabbitMQ execution) are presented in the accompanying focuses.

3.2.1 Protocols Buffer

Protocol Buffer (https://code.google.com/p/protobuf/) were at first evolved at Google to manage a file worker demand/reaction buffer. They attempt to standardize the interpretation of information into a structure that could be sent between PCs. This interaction is normally named serialization. Preceding convention cushions, there was an arrangement for solicitations and reactions that pre-owned hand marshalling/ unmarshalling, which is certifiably not a truly advantageous situation. In protobuff, sufficient kinds of messages must be characterized to determine how information is serialized expressly. This is finished through proto documents Every convention cushion message is a little sensible record of data, containing a progression of name-esteem sets. Fundamental illustration of a proto record that characterizes a message containing data about an individual. A convention support compiler makes an

interpretation of the proto documents into modules fit to be connected from the application being created. For instance, the Person. proto record could be converted into a C++ Person class with techniques to get to, alter, parse and serialize information. Convention supports are subsequently adaptable, effective, computerized components for serializing organized information, like XML, yet more modest, quicker, and easier.

3.2.2 AMPQ Protocol

AMQP (http://www.amqp.org/) is the internet protocol for business messaging. The Advanced Message Queuing Protocol (AMQP) is an open standard to pass business messages between applications or organizations. It connects systems, feeds business processes with the information they need and reliably transmits the instructions that achieve their goals. AMQP enables applications to send and receive messages. In this regard, it is like How Nets of Evolutionary Processors (NEPs) Could be Simulated 59 instant messaging or email. AMQP differs enormously because it allows one to specify what messages will be received and from where and how trade-offs are made concerning security, reliability and performance.

3.2.3 RabbitMQ

RabbitMQ (http://www.rabbitmq.com/) is an informing representative, that is, a mediator for informing. It executes AMQP and gives your applications a typical stage to send and get messages and your messages a protected spot to live until got. Among the elements given by RabbitMQ, we will zero in on the binding topics system. RabbitMQ extensively utilizes lines to deal with the messages. Subjects allude to some mark you can append to your messages to recognize and order them somehow or another. RabitMQ can interface a few lines to themes presented by some confined type of customary examples.

3.3 Simulating NEPs

jNEP is a program written in Java which is fit for recreating practically any NEP in the writing. To be an important device for established researchers, it has been created under the accompanying standards: a) it thoroughly consents to the conventional definitions found in writing; b) It fills in as an overall device by permitting the utilization of the distinctive NEP variations and is prepared to adjust to future potential variations as the examination in the space progresses; c) It takes advantage of however much as could reasonably be expected the intrinsic equal/disseminated nature of NEPs. jNEP offers execution of NEPs as broad, adaptable, and thorough as. The plan of the NEP class emulates the NEP model definition. In jNEP, a NEP is made of developmental processors and a fundamental diagram (characteristic edges) that characterizes the net geography and guides the between processor cooperation. The NEP class arranges the main dynamic of the calculation and rules the processors (occurrences of the EvolutionaryProcessor class), compelling them to perform substitute transformative and correspondence steps. It additionally stops the calculation when required. The centre of the model incorporates these two classes, along with the Word class, which handles the control of words and their images. jNEP is adaptable and versatile to various NEP variations on account of its plan dependent on Java interfaces. jNEP offers three interfaces: a) StoppingCondition, which gives the technique stop to decide if a NEP article should quit by its state. b) Filter, whose technique applyFilter figures out which objects of class Word can pass it; c) EvolutionaryRule, which applies a Rule to a bunch of Words to get another set. jNEP

attempts to execute a wide arrangement of NEP's variations. Also, jNEP can broadly take advantage of the equal/disseminated nature of NEPs unexpectedly. the client can pick the equal/dispersed stage on which jNEP runs. Right now, the upheld stages are standard JVM and equal Java stages. The Sun Java Virtual Machine, which can be viewed as the standard Java, can't be run on equal stages like groups. Other parallelization devices must be utilized, for example, the Hadoop stage as proposed in this work. The test system portrayed in this segment has been created with both JVM and Apache Spark stages that will be broadly clarified in the next areas. Besides, simultaneousness is executed through two distinctive Java draws near: Threads and Processes. The primary requirements are more perplexing synchronization systems. The second uses heavier simultaneous strings. Even more, on account of the Processes choice, every processor in the net is an autonomous program in the working framework.

The correspondence between hubs is brought out through the standard info/yield surges of the program. The class NEP approaches those streams and facilitates the hubs. The required rotation of correspondence and developmental strides in the calculations of NEPs incredibly enables their synchronization and correspondence. The accompanying convention has been followed for the correspondence step: 1) NEP class makes an impression on each hub in the diagram requesting to begin the correspondence step. Then, at that point, it sits tight for their reactions; 2) Every hub completes its correspondence venture in the wake of shipping off the net the words that pass their yields channels. Then, at that point, they show the NEP class that they have completed the correspondence step; 3) The NEP class moves every one of the words from the net to the info channels of the relating hubs. The transformative advance is synchronized through an underlying message sent by the NEP class to make every one of the hubs advances. A while later, the NEP class delays until every one of the hubs finishes. The execution with Java Threads has other implications.

In this choice, every processor is an object of the Java Thread class. Hence, every processor executes its undertakings in equal as free execution lines, albeit they have a place with a similar program. Information trade between them is performed by direct admittance to memory. The standards of correspondence and coordination are equivalent to in the past alternative. The primary distinction is that as opposed to trusting that every one of the streams will complete or to send a specific message, Threads are composed through fundamental simultaneous programming instruments as semaphores, screens, and so forth All in all, jNEP is an entirely versatile device that can run in a wide range of conditions. Contingent upon the working framework, the Java Virtual Machine utilized, and the simultaneousness choice picked, jNEP will work in a somewhat unique way. The client should choose the best blend for his necessities.

3.4 Spark Environment on Cloud Setup

The XML document will be created this XML record is entered to the NEPS, Spark and cloud. The yield that comes from the cloud is to be attracted to show the outcomes. Presently, we need to establish the Spark workspace and Determine Spark as an application with SupportedProductConfig utilized in RunJobFlowRequest. The accompanying code is utilized to establish the Spark run workspace.
To start a clustered system with a Spark environment as follows:

1. Open the Amazon EMR console at https://console.aws.amazon.com/elasticmapreduce/.

2. Choose Creating clusters to utilize "Quick Options".
3. Input a Clusters name.
4. For Software Configuration, pick a "Release choice".
5. For Applications, pick the "Spark application group".
6. Select different choices as essential and afterward pick "Create cluster".

3.5 Conclusion

In this chapter we explained the most related state-of-the-art papers and thesis to our proposed work. Thus, this chapter is concentrated on giving a wide range of that paper who has implemented the NEPs architecture. Here, we highlighted the entirety of the advantages and prospects of related the design, architecture, functionality and performance of the NEPs system in different application in recent years because of the importance of this subject in terms of data processing in different applications. It is important to note that the NEPs in all of the previous works in this thesis, have not utilized the cloud technology in its design. It can be said that combining the cloud technology with the NEPs structured design might produce an effective method to handle data that needs a decentralized and distributed platform capable of doing the processing, analysis, storage and transmitting operation, simultaneously.

CHAPTER 4. PROPOSED NEPS SYSTEM

4.1 Background

The proposed method depends on the Spark environment tool of Hadoop. The Hadoop Cloud Platform system environment can be shown in Figure 7. The data acquisition is performed in the Data Storage and the Data querying. The data might be gathered from multiple sources such as the local file systems and spatial, temporal Indexes. In the next step, we choose the parallel processing using the Hadoop cloud platform environment. After that, we visualize the obtained information.

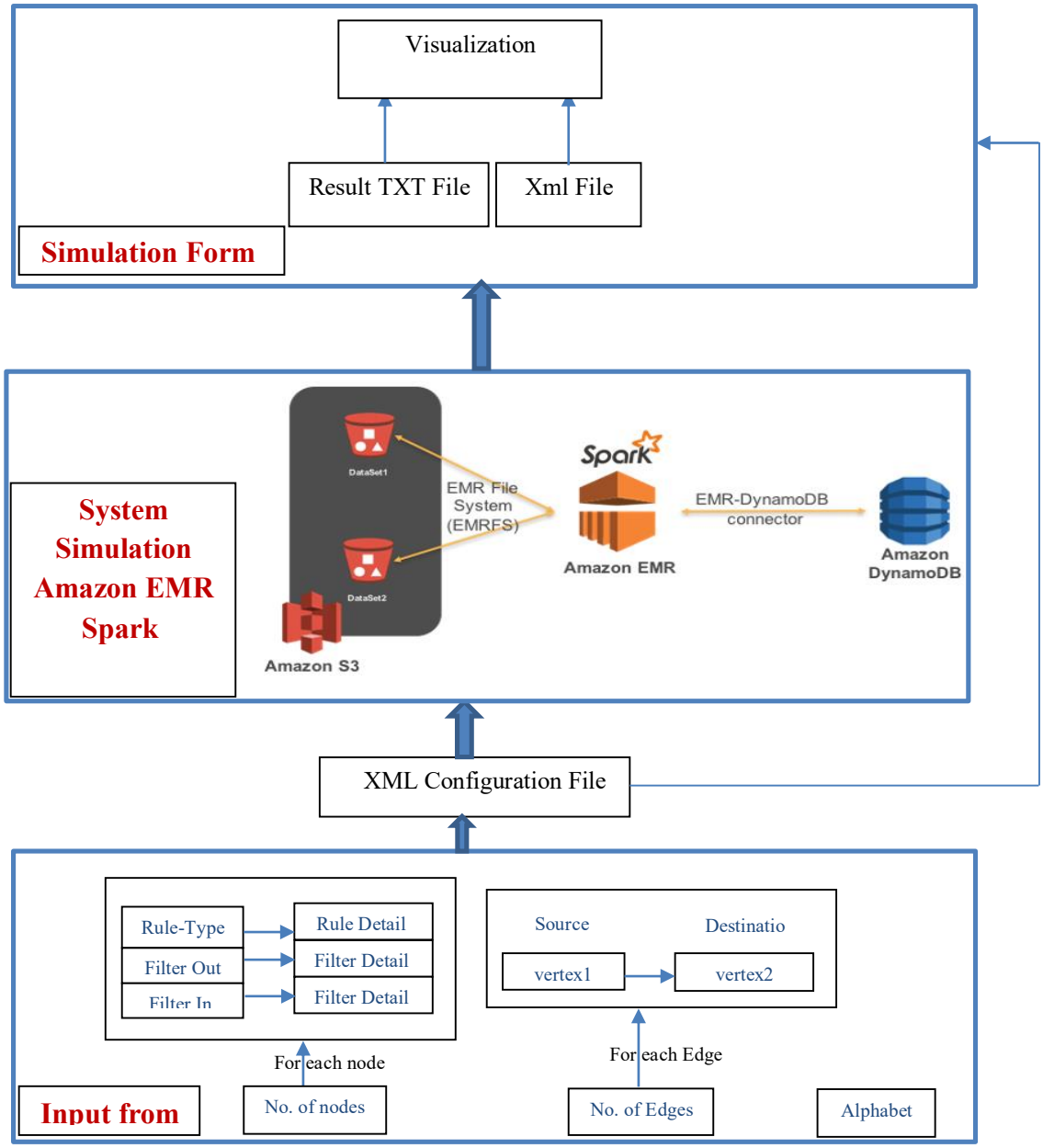


Figure 7: Hadoop Cloud Platform Environments

The following steps can summarize the main steps of the proposed method:

1- Data acquisition: we collect documents where we want to classify according to their contents according to the crime type. This classification would be helping the classification system to funnel down the search space and limit that space to thousands rather than searching in millions of items.
2- Data processing: in this stage, the data become ready to be processed using one of the techniques shown in Figure 7. Our proposal will follow the parallel processing direction due to its fast computation and its success in big data [38].
3- Visualization: is the visual representation of the obtained data.

4.2 Architecture Overview

A MapReduce implementation is considered a framework for parallelization problems, particularly combinatorial problems such as the colorability problem that takes advantage of localities of data and involves two major phases: a map and reduce phases. The MapReduce task is to split the dataset into segments of a fixed-sized, which are used by an individual map. The map phase computes the distances between each cluster centroid and an object and assigns each object to its closest cluster. One map task is created for each input split and is run by map functions for each input split record.

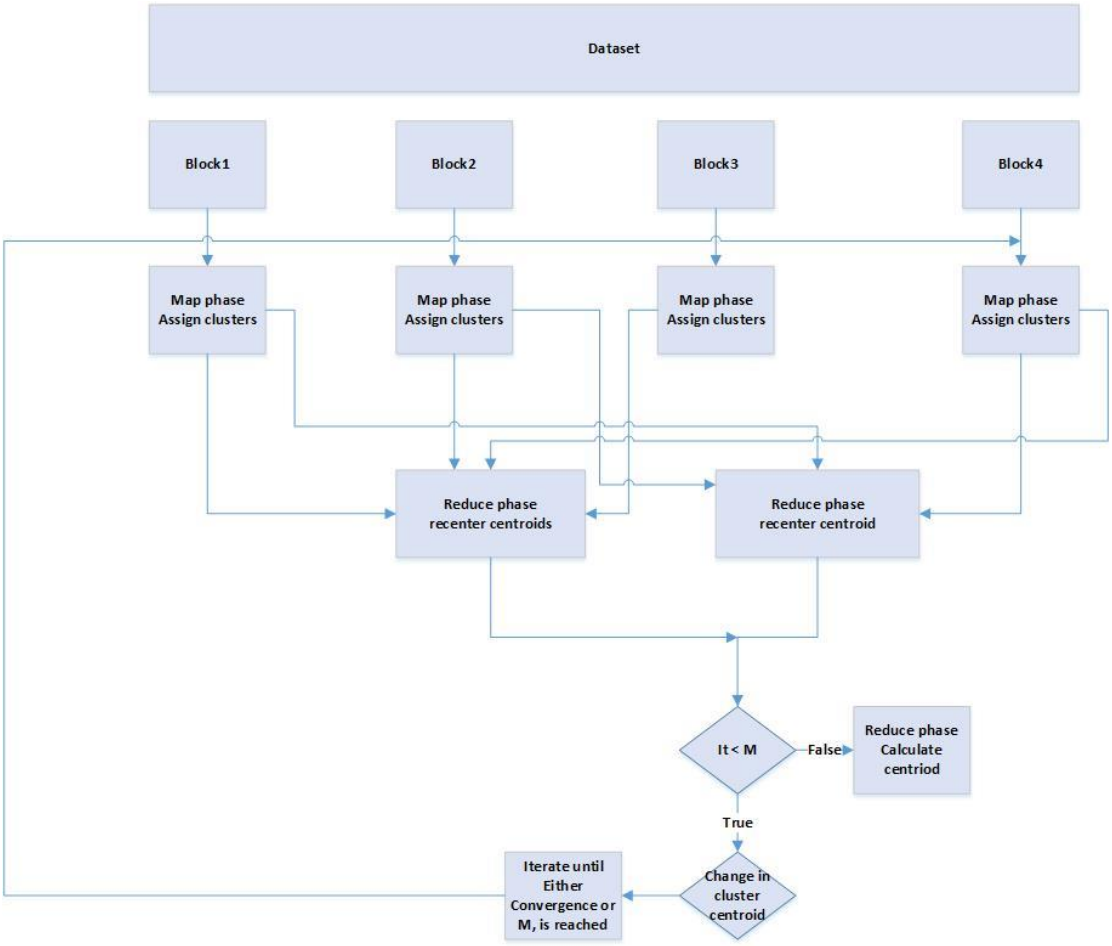


Figure 8: Hadoop environment

The system of NEPs implemented on the spark Hadoop environment can be represented in Figurer 8. It is obvious from that figure that the streamed data will be distributed on several chunks where each chunk in named a black ($block_1$... $block_n$), and after dividing the data into those blocks it will be mapped and the NEPs nodes. The reducer component then will be collecting the resulted data from the mapped NEPs nodes components. After this, the algorithms selected to be used in this research will be applied.

The first step is to make an input form that has the NEPs parameters, which are:

1- NEPs nodes: each node represents a processing unit.
2- NEPs alphabet symbols: representing the input set of strings to the system.
3- Number of channels: representing the number of edges and those edges connecting each vertex where each vertex is considered a processor.
4- The number of rules of each node represents the operations associated with each node, such as the deletion, insertion, substitution, splicing etc.
5- Filter in and filter out, which are considered the criteria are controlling the input and output flow.
6- The stopping condition, which is the criteria when is met, the processing will be halted. They type if the stopping conditions considered in our work are listed later in this chapter.

4.3 Spark Environment on Cloud Setup

The XML file will be produced. This XML file is entered to the NEPS, Spark and cloud. The output that comes from the cloud is to be drawn to show the results. Now, we must create the Spark environment. Specify Spark as an application with SupportedProductConfig used in RunJobFlowRequest. The following code is used to create the Spark run environment.

Due to the importance of the following code in Figure 9, that is focuses on creating the debugging steps, add a management system include the AWS credentials for the Amazon section and all other important libraries, the need to use this code is a must because the spark environment most first managing item candidate by this code.

```
import com.amazonaws.AmazonClientException;
import com.amazonaws.auth.AWSCredentials;
import com.amazonaws.auth.AWSStaticCredentialsProvider;
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import
com.amazonaws.services.elasticmapreduce.AmazonElasticMapReduce;
import
com.amazonaws.services.elasticmapreduce.AmazonElasticMapReduceClient
Builder;
import com.amazonaws.services.elasticmapreduce.model.*;
import com.amazonaws.services.elasticmapreduce.util.StepFactory;

public class Main {

      public static void main(String[] args) {
            AWSCredentials credentials_profile = null;
```

```
            try {
                  credentials_profile              =              new
ProfileCredentialsProvider("default").getCredentials();
        } catch (Exception e) {
            throw new AmazonClientException(
                    "Cannot load credentials from .aws/credentials
file. " +
                    "Make sure that the credentials file exists and
the profile name is specified within it.",
                    e);
        }

        AmazonElasticMapReduce                  emr                  =
AmazonElasticMapReduceClientBuilder.standard()
                  .withCredentials(new
AWSStaticCredentialsProvider(credentials_profile))
                  .withRegion(Regions.US_WEST_1)
                  .build();

        // create a step to enable debugging in the AWS Management
Console
            StepFactory stepFactory = new StepFactory();
            StepConfig enabledebugging = new StepConfig()
                  .withName("Enable debugging")
                  .withActionOnFailure("TERMINATE_JOB_FLOW")

      .withHadoopJarStep(stepFactory.newEnableDebuggingStep());

        Application spark = new Application().withName("Spark");

        RunJobFlowRequest request = new RunJobFlowRequest()
            .withName("Spark Cluster")
            .withReleaseLabel("emr-5.20.0")
            .withSteps(enabledebugging)
            .withApplications(spark)
            .withLogUri("s3://path/to/my/logs/")
                  .withServiceRole("EMR_DefaultRole")
                  .withJobFlowRole("EMR_EC2_DefaultRole")
            .withInstances(new JobFlowInstancesConfig()
                .withEc2SubnetId("subnet-12ab3c45")
                .withEc2KeyName("myEc2Key")
                .withInstanceCount(3)
                .withKeepJobFlowAliveWhenNoSteps(true)
                .withMasterInstanceType("m4.large")
                .withSlaveInstanceType("m4.large")
            );
        RunJobFlowResult result = emr.runJobFlow(request);
          System.out.println("The    cluster    ID    is    "    +
result.toString());
      }
```

Figure 9: configuration code

4.4 Spark Configuration

In order to operate the Spark system, we need to include the following configuration to our Amazon EMR (Elastic Map Reduce) system. Configuration classifications for Spark on Amazon EMR include the following:

1- spark—Sets the maximize resource allocation property to true or false. When true, Amazon EMR automatically configures spark-defaults properties based on the cluster hardware configuration. For more information, see Using maximize resource allocation.
2- spark-defaults—Sets values in the spark-defaults.conf file. For more information, see Spark configuration in the Spark documentation.
3- spark-env—Sets values in the spark-env.sh file. For more information, see the Environment variables in the Spark documentation.
4- Spark-hive-site—Sets values in the hive-site.xml for Spark.
5- spark-log4j—Sets values in the log4j.properties file. For settings and more information, see the log4j.properties. Template file on Github.
6- Spark-metrics—Sets values in the metrics.properties file. For settings and more information, see the metrics. Properties. Template file on Github, and Metrics in Spark documentation.

4.5 Conclusions

In this chapter, we presented the working conditions of the proposed NEPs method and the main concepts behind using it. On the other hand, the main components of the NEPs system are also explained with the programming relevant issues. The proposed method depends on the Spark environment tool of Hadoop. The Hadoop Cloud Platform system environment. The data acquisition is performed in the Data Storage and the Data querying. The data might be gathered from multiple sources such as the local file systems and spatial-temporal Index. In the next step, we choose the parallel processing using the Hadoop cloud platform environment. After that, we visualize the obtained information, as will be seen in the next chapter.

CHAPTER 5. TEST RESULTS

5.1 Background

In this section, we will describe the problems undertaken to be tested in the experiments of the NEPs. The proposed system can execute almost any kind of NP-hard optimization problem. However, to justify the proof of concept of this thesis that corresponds to the aims set up in chapter 1, we investigate the NEPs system performance in terms of three well-studied problems in this domain, i.e. (Adleman problem, massive-NEP and the 3-colorabilty problem. This chapter is designed to be divided into three main sections. Each section discusses one of the above-selected problems be examining the performance according to the considered evolution standard.

In this chapter, we study the proposed NEPs offline based Apache spark system's performance against that proposed in [42]. The author develope and study a full framework to simulate and research on Network of Evolutionary Processors without using the Spark technology to implement the aims supposed to be met in that work. The effect of using the Spark on the results will be highlighted in this chapter. On the other hand, the second comparison will be investigating the effects of using the online and offline cloud spark system on the same problem. This comparison will answer the research question that explores the impact of utilizing the modern technology of distributing the problem on the cloud on multiple processing units. Last, in the third comparison, the investigation of manipulating the number of nodes in the NEPs system and the effect of the scalability of the design on the performance will be given.

5.2 NEPs Components

The first step is to configure the XML, as can be demonstrated in Figure 10. In this step, we first set up the number of nodes representing the number of processors. Afterwards, we start setting up the alphabet. As a set of symbols which shape the alphabet of the words transmitted among the processors to be manipulated. The alphabet is one or more characters, i.e. a string separated by underscores or special characters. The next step is to enter the number of edges. When the number of edges is set, the form will open fields as many as the number of edges. Each one contains two blocks, the source vertex, and the destination vertex. When the number of nodes is entered into the system, the form will generate fields of rules that can be null, single or multi-rule nodes. The types of rules can be classified into the following types:

1- Deletion rule.
2- Insertion rule.
3- Substitution rule.
4- Evolutionary rule.
5- Splicing rule.
6- Splicing parsing rule
7- Splicing Choudhry rule.
8- Leftmost parsing rule.

The filters are divided into input-filters and output-filters. The input-filters will permit the word transmitted to be entered to the processor in order to configure its own rule, while the output-filter will permit the word to be transmitted to be out from this particular processor.

There are several kinds of filters used in this domain. However, three of them are implemented in the design of the current network of evolutionary processors system, which are the

- Context-Condition-Filter: This filter contains an permitting context or forbidding context one or both to pass the word transmitted between nodes interconnected by an edges
- Regular-Lang-Membership-Filter: Regular language containing words that belong to it.
- Set-Membership-Filter: Only entered words are allowed to pass

The stopping condition of the form, there are

1- Consecutive-Config-Stopping-Condition: It produces the NEP to stop if two consecutive configurations are found as communication, and evolutionary steps are performed.
   Syntax: < CONDITION type="ConsecutiveConfigStoppingCondition"/>

2- Maximum-Size-Stopping-Condition: It produces the NEP to stop after a maximum number of steps.
   Syntax: <CONDITION type="MaximumStepsStoppingCondition" maximum="[integer]"/>
3- No-Changes-Stopping-Condition,
4- Non-Empty-Node-Stopping-Condition: It produces the NEP to stop if one of the nodes is non-empty. Useful for NEPs with an output node.
5- Words-Disappear-Stopping-Condition,

5.3 NEPs Simulation using Spark Environment

To simulate the network of evolutionary processors using the Spark environment, we must first create an input form that permits the user to input the information required to create an XML file that is considered an input to the NEPs system. This file is considered as the problem space that must be resolved.

5.3.1 NEPs Modularization Form

The main input form to our proposed system can be represented as a form whose design is based on the idea of modularizing the NEPs tasks and functionalities in a modular and clear manner. This is implemented by using a set of text boxes that contain the inputs of the users. In order to further explain this system, it is important to examine Figure 10. This form has the NEPs nodes. Those nodes represent the processors used in the system, while NEPs alphabet symbols representing the words transmitted via the system. Number of edges that represented the communication channel linking each two vertices in the graph i.e. connecting the processors. The graph is a number of vertices. The vertices are two nodes linked by an edge. The number of rules of each node. Filter in and filter out. Finally, the stopping condition.

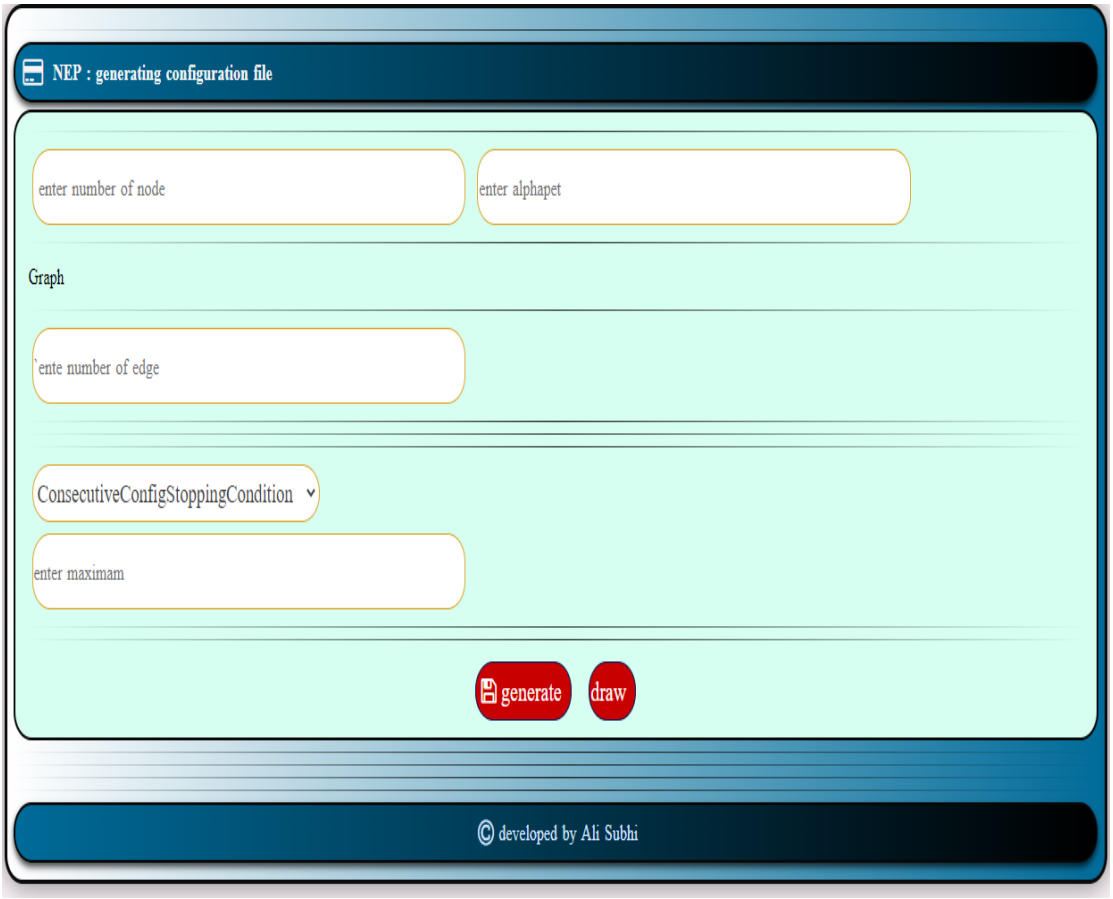


Figure 10: NEPs input form

5.4 Input Form Details

In this section, the main outcomes derived from the system explained in this thesis are given in the form of modularized sections starting from the preparation of the XML file input in the system that is designed to be processed by our NEPs module. The NEPs system first will be executed by initiating the input-form, which starts by setting up the number of nodes generating the rules and filters. As an alphabet, we will enter a set of strings separated by underscores or any other special characters. The graph will have the number of edges. The number of those edges is half the number of the input vertices. After creating the graph, fields of two vertices will be opened: one for the source and the other for the destination. And then we start to handle each rule which involves selecting the rule type, initial condition, action type, and the symbol and the new symbol. There are 8 types of rules. 5 of them are working on the explained system, which are (deletion, insertion, substitution, Left-Most-Parsing, Evolutionary) and the rest 3 types are (Splicing Choudhary, Splicing Parsing, Splicing) has the attribute (wordX, wordY, wordU, wordV) instead of action type, symbol and new symbol. Basically, each node might be having one or more rule associated to it. The rules can be shown in Figure 11.

Figure 11: Rules of NEPs1

In the input-form, after initializing the rules, the filters now will be considered. There are a special set of filters associated to every single node. The filter is composed of filter-in and filter-out. Both have the same attribute structure. As shown in figure 12, the filer-in and filter-out fields are shown.

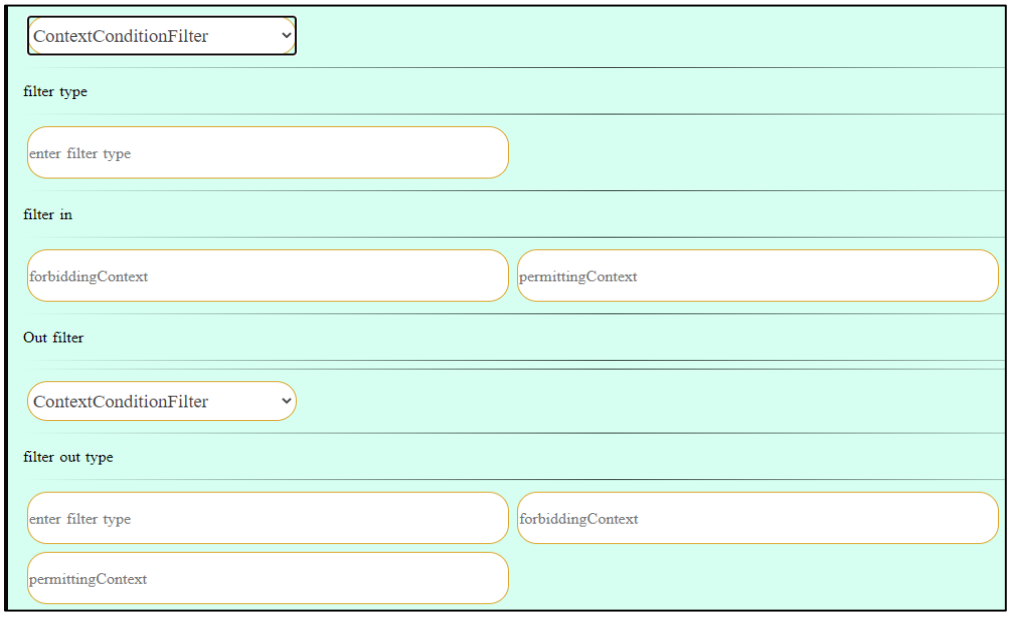


Figure 12: Filter-in and filter-out fields of NEPs

There are three types of filters available in the applications: the context conditional filter, the regular Lang membership filter, and the set membership filter. The first two has the same input fields: the filter type, an integer-based attribute, and then the forbidden context and the permitting context, all of which are string-based attributes. And the last one is the set membership filter, whose attribute is the set word, a string-based attribute. The last filed in this form is the stopping condition which has five numbers stopping conditions. The requirements are listed below:

1. Consecutive config stopping condition.
2. Maximum steps stopping condition.
3. No changes stopping condition.
4. Non-empty node stopping condition.
5. Words disappear, stopping condition.

Although there are 5 types of stopping conditions, we have used the second and the fourth ones because these stopping conditions are well-matched to our problem. The fourth differs from the second one, whose attribute represents the node ID while the second represents the number of the maximum steps.

5.5 System Simulation

This section explains the cloud environment reserved to implement our proposed work properties and the properties of the computer used for this purpose, and the comparison table between both of them in terms of the execution time. The aim of this table is to show the gain obtained after having the cloud technology incorporated in the proposed system. In this work, we are applying the NEPs to the cloud spark environment. This service can be provided by Google, Microsoft Azure and Amazon Amr. These three companies are providing the cloud service to solve problems requiring distributed decentralized environment—moreover, these companies support the use of the Apache Spark system. However, in our proposed work we decided to use the AWS environment only because of some technical issues which have nothing to do with the excitability or workability of the system.

Amazon EMR (on prior it was named Amazon Elastics Map Reduce) is a controlled clustering system simplifying the execution of big data frameworks as we use in our system the Apache Spark and Hadoop platforms on AWS for processing and analyzing huge quantities of information. One might be capable of processing data for analytical reasons and commercial intelligence data via these frameworks and associated open-sourced codes. Amazon EMR would also let us to be able to transform and transfer vast quantities of information into and out of other AWS data databases or data warehouses like using the Amazon Simple Storage Service (Amazon S3) and Amazon DynamoDB.

EMR properties of Amazon EMR Apache Spark runtime, a performance-enhanced runtime platform for Apache Spark activated on the Amazon EMR clustering system. One of the most important properties of the Amazon EMR runtime for Apache Spark is that its speed could be faster three times to the system which is not equipped with the EMR runtime as we will realize in this chapter, as we also noticed through the experiments conducted in this chapter that the used EMR environment possesses 100

% API compatibility with Apache Spark platform. This enhanced the excitability is that the data execution can be made faster and reduce the computational complexity in terms of time and cost, with no notable alterations to the system architecture. Via the directed acyclic graph (DAG) execution engines, Spark could be creating effective querying plans for data manageability. Spark would also be storing the outcomes and the intermediary data in the computer as persistent data frames that permit the processing to reach its premium execution powers. The Input and output devices costs can be eliminated. This would be a boosting factor that can remarkably enhance the performance of interactive and iterative control.

5.5.1 AWS Cloud Environment Properties

The properties of the cloud system utilized to be executing the proposed NEPs system implemented on the Apache Spark Environment are listed in Table 5.

Table 5: cloud system properties

Instance Size	vCPU	Memory (GiB)	Instance Storage (GiB)	Network Bandwidth (Gbps)	EBS Bandwidth (Mbps)
m5.large	4	16	EBS-Only	Up to 10	Up to 4,750

5.5.2 Offline Spark Execution Environment

This section is designed to be explaining the hardware and software requirements needed by the system to execute the spark environment without running the system on the cloud, as was explained in the last subsection. As can be seen in table 6, the computer specifications used in this work depends on the number of cores in the system, the memory size, and finally, the disc size that corresponds to the instance storage in gigabytes.

Table 6: Computer system properties

vCPU	Memory (GiB)	Instance Storage (GiB)
7	8	500G

5.5.3 Comparison of Running NEPs on Spark System and Offline

In Table 7 discusses a comparison between the Apache Spark-based system conducted offline and proposed in this thesis with the NEPs system that was also executed offline on [42] whose name is JNEPs. For simplicity, we name our own proposed system SNEPs. Furthermore, to specify the online and offline modes that were operated on the SNEPS, we name the offline mode as the OF-SNEPs while the online SNEPS as ON-SNEPS.

Table 7: Comparison of OF-SNEPs and jNEPs

Problem Name	OF-SNEPs				jNEPs
	Sql-Context	Map	Flat-Map	Parallelize	
Adelman	7.10477	8.8443	5.7225	4.70633	14.02100
Massive-NEPs	2.54599	3.05376	1.58296	3.91230	4.90100
3-Coloring	Not work Error: An unrecoverable stack overflow has occurred				

The measuring unit is second, as can be seen in the comparison that there is a huge different between the performance time needed by the JNEPs to execute each one of the considered problems separately.

Bearing in mind that the results obtained from both systems are the same

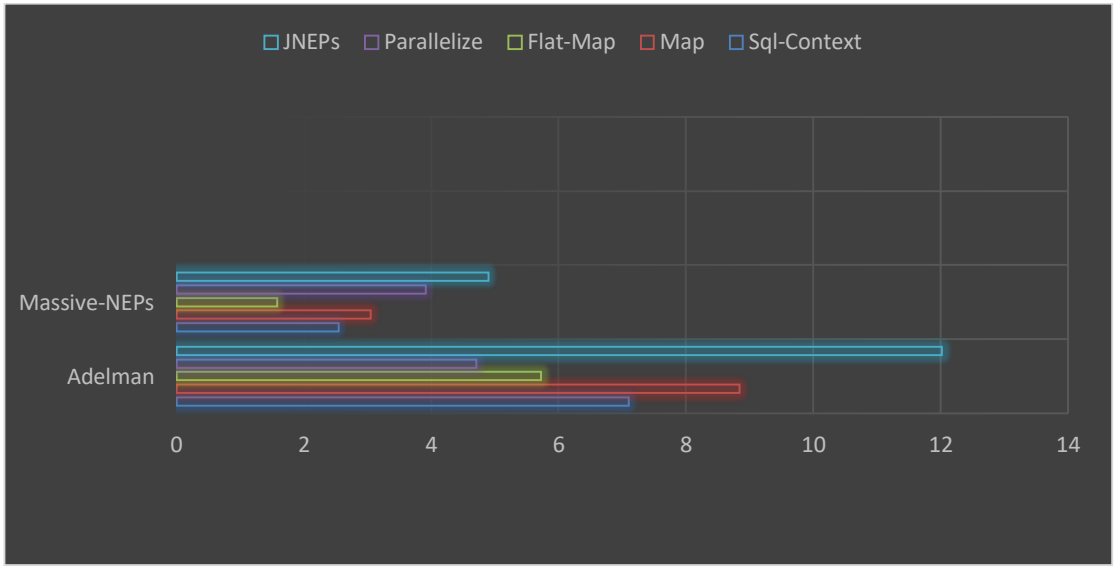


Figure 13: Performance chart of jNEPs and OF-SNEPs systems

5.5.4 Comparison of ON-SNEPs and OF-SNEPs Performance

This section discusses a comparison between the Apache Spark-based system conducted online (ON-SNEPs) and the one conducted offline (OF-SNEPs).

Table 8: Comparison of ON-SNEPs and OF-SNEPs

Problem Name	ON-SNEPs	OF-SNEPs
Adelman	2.37821	4.70633
Massive-NEPs	0. 7176	1.58296
3-Coloring	0. 6889	Not Work

As Table 8 shows, there are a huge different result of each Apache Spark-based system conducted online (ON-SNEPs) and offline (OF-SNEPs) on the three problems.

The 3-Coloring problem does not run with the previous system duo to the limitation of the memory, while in ON-SNEPs system it gave us promised result that have the less execution time.

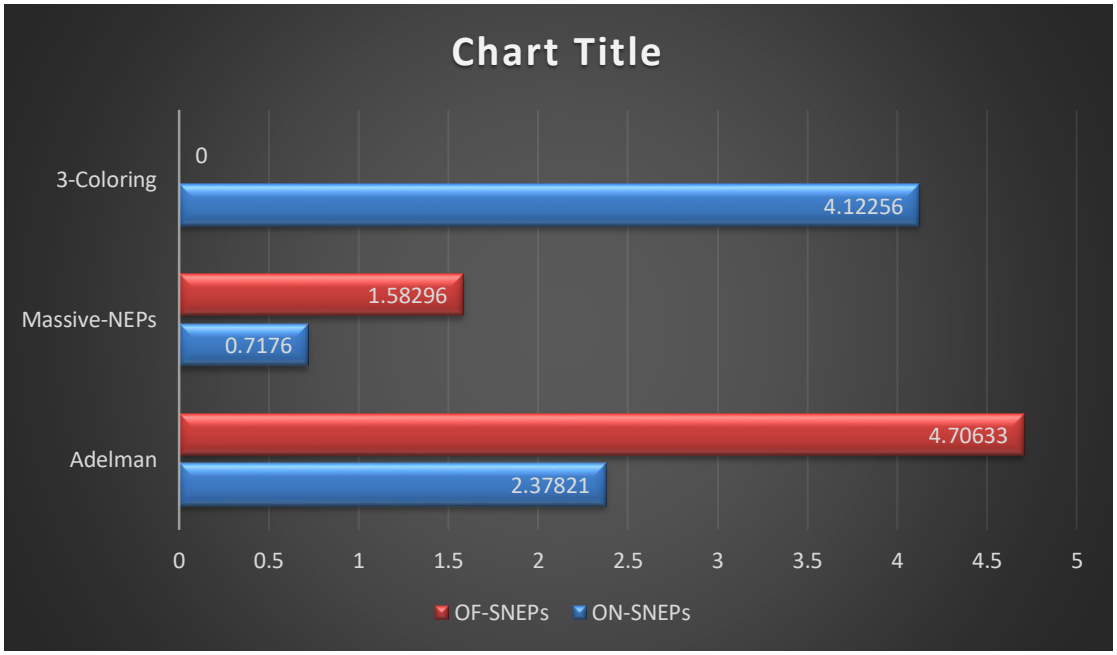


Figure 14: Performance chart of ON-SNEPs and OF-SNEPs systems

5.6 Problem Complexity Scaling

Adding the complexity of the considered problems, i.e. Adleman, 3-colourability, massive-NEP, seems to be an excellent way to show the difference between Apache Spark-based system conducted online (ON-SNEPs) and the one conducted offline (OF-SNEPs) on the three problems as we will see in this section that we will use the downscaling and upscaling to manipulate the dimensionality of the problem.

5.6.1 Downscaling

This step is important to show the performance of the method to handle the problem after and before reducing the dimensionality of the problem, applying the downscaling on 3-Coloring problem and reduce the number of nodes by 50% of the original numbers, furthermore the edges would be minimum to the halve cause the 3-Coloring problem has a full connection graph.

Table 9: Downscale comparison

Problem Name	3-Coloring Downscaled 50%	3-Coloring Original
ON-SNEPs	1.768	4.12256
JNEPs	95.70599	Not Work

5.6.2 Upscaling

This step is important to show the performance of the method to handle the problem after and before scale up the dimensionality of the problem, however the Adleman problem increased 100% than the original proposed by [42], and the massive-NEP increased by 50%.

Table 10: Scaling comparisin

Problem Name	Adleman Scaled up 100%	Massive-NEP Scaled up 50%
ON-SNEPs	4.92665	2.79695
JNEPs	79.66790	6.52900

5.7 Spark Parameter tuning

Spark has many parameters affecting its performance as excitability and interoperability. These parameters might be including (Map, Flat Map, Partitioning Map) whoever, from this work point of view, these parameters are the most effecting the current work.

5.8 Simulation Form Drawing

To visualize the test results on the output window, we must first select the XML file created in the form modularization step and then select the results derived from the emulation module, which is formatted as a text file. This file will have the same problem's name. as soon as we receive it, it will be viewed in the text area; afterwards, the "ready to draw" button will be clicked to draw the viewed file on the screen. A specific area in the window will be assigned to view the graph. The steps drop list will be activated whenever one step is selected. The number of steps will be dependent on the problem itself. Figure 15 shows the simulation drawing form.

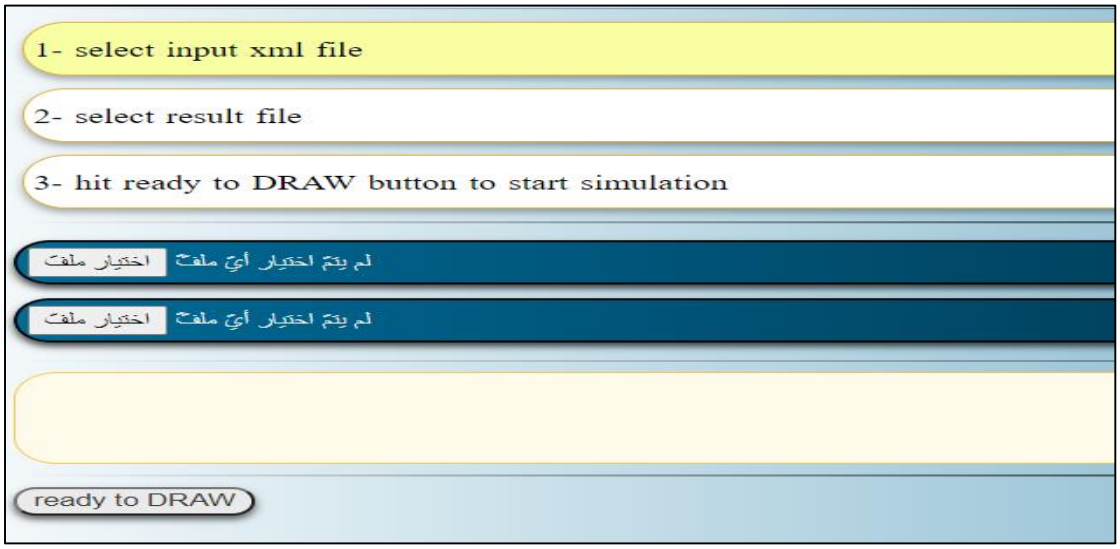


Figure 15: Simulation drawing window

For example, at the initial step, we can have a system structure as the one shown in figure 16.

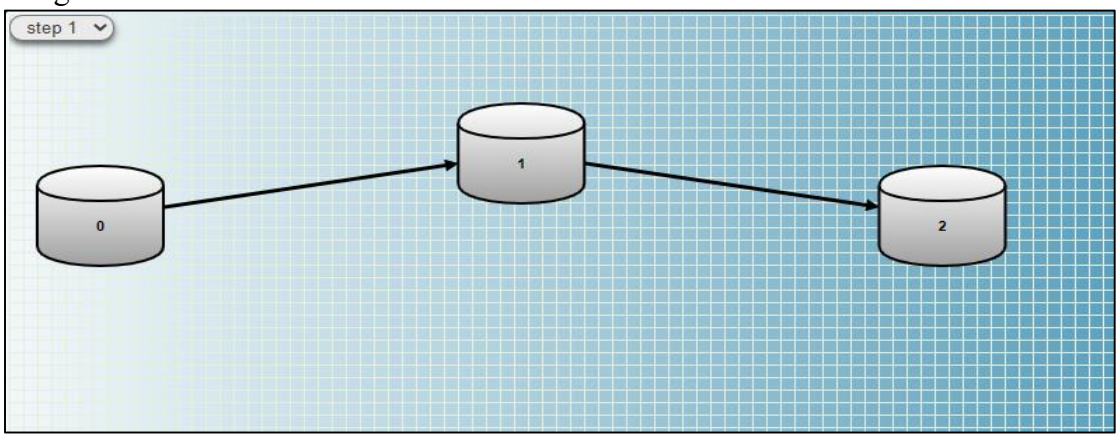


Figure 16: Initial system drawing

In Figure 17, the numbers on the cylinders shown are representing the node labels or the processor number. On the other hand, in figure 5.5, the graph represents the words transmitted in each processor. It is noteworthy to mention that the graph shown in Figure 17, can be drawn in all steps other than the initial step.

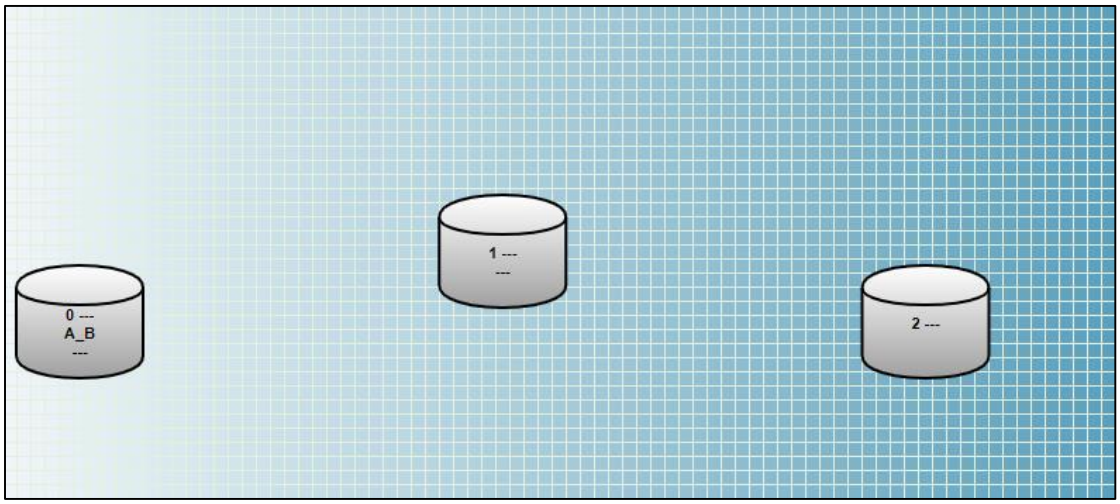


Figure 17: Evolutionary steps visualization

5.9 Conclusion

In this chapter we investigated the outcomes of the proposed NEPs based Apache flash framework. We created and considered a full system to recreate and investigate on Network of Evolutionary Processors utilizing the Spark to execute the points expected to be met in this work. The impact of utilizing the different parameters of the spark on the outcomes is also given in this chapter. Then again, the subsequent correlation will examine the impacts of utilizing the on the web and disconnected cloud spark framework on a similar issue. This examination will address the research question that investigates the effect of using the cutting-edge innovation of appropriating the issue on the cloud on different preparing units. Last, in the third test, the test of controlling the number of nodes in the NEPs framework.

CHAPTER 6. CONCLUSION AND FUTURE WORK

6.1 Background

In this chapter, we give some recommendations about the proposed method. This chapter helps future research to be developed or enhanced as an addition to this work. As discussed in the chapter mentioned above, the NEPs system is one of the most effective methods deployed to handle parallelism in data and task management in real-world examples. During the experimentation derived from the previous. chapter 5, the following recommendations, and future works are provided in scalability, parameter tuning and time requirement. It becomes essential to include the concluding remarks extracted from the current position. Moreover, the future work will also be listed in this chapter to elaborate this work in terms of the same criteria that will be mentioned in the recommendation section, i.e. performance, efficiency, time requirement and scalability.

In the first chapter, we explained the aims of the current work. For this, we designed a method of how to operate NEPs in parallel to get the best results using the Apache Spark Platform. This incorporated the scalability of the NEPs system and made the parameter tuning which is considered as one of the most vital objectives in our proposed system. The main achievements of this work can be concluded in the following steps:

1- The simulation of a NEP was successfully conducted to solve three optimizations problem types using the Spark Apache Platform .
2- We are analyzing the parameters of the system. The parameters tuning is the process that examines every single factor affecting the performance of the NEPs individually . Thus, in this work, the tuning was successfully made.
3- As the last objective, the Scaling of the system to incorporate further processing units has also been investigated. Scaling has been examined in two ways: downscaling and up-scaling. This has been done to verify the system's capability of handling more complex problems, such as increasing number of processors, number of connections, size and number of words transmitted.

6.2 Discussion

This section gives some detailed recommendations about the proposed NEPs system considering the earlier explained summarized points in the last section.

1- Using NEPs on the decentralized cloud eco-system can be an effective method to handle data of different formats and execute optimization problems such as Adelman, 3-colorabilty and Massive-NEP problems. Moreover, this scheme is also robust and adaptable to handle data that might be scaled up to be big data characterized by its volume and heterogeneity. In this context, heterogeneity might be referring to collecting data from multiple sources in different time-lapses.
2- Due to the importance of conducting parameters study related to any system, in our proposed method, parameters have been tuned to let the system perform in its best execution manner.

3- using the spark environment as a platform to operate the NEPs system has it is advantages. This environment is characterized by its fast task of handing chunks of data to Hadoop architecture used to implement the spark system, mainly based on the map and reduce functions. For this reason, the task is distributed on a decentralized NEPs system using the cloud system, which has made it possible to obtain the realistic result in all the three examples investigated and examined in our proposed method.

6.3 Future Work

In this section, we give some future ideas and concepts that assist future researchers in continuing researching in this domain and develop the present work explained in this thesis:

1- It is possible to use the system in real-world institutions such as hospitals, universities, schools, or the government to handle big data that does not make sense to deal with it in the old traditional ways to take advantage of time and size.
2- Using more technologies other than cloud technology, such as fog technology, to further distribute tasks among the processing units can be future work because of the financial issue in Amazon, and to get more advantages.
3- Doing a complexity analyses study that theoretically assists the performance of the proposed NEPs system other than using the present examples would give more approved evidence of its usability in the future.

REFERENCES

[1] Zhang, Yunquan, et al. "Parallel processing systems for big data: a survey." Proceedings of the IEEE 104.11 (2016): 2114-2136.

[2] Montasari, Reza, Richard Hill, Victoria Carpenter, and Amin Hosseinian-Far. "The Standardised Digital Forensic Investigation Process Model (SDFIPM)." In Blockchain and Clinical Trial, pp. 169-209. Springer, Cham, 2019.

[3] W. Anwar, I. S. Bajwa, M. A. Choudhary, and S. Ramzan, "An Empirical Study on Forensic Analysis of Urdu Text Using LDA-Based Authorship Attribution," IEEE Access, vol. 7, pp. 3224-3234, 2019.

[4] Saravanan, P., et al. "Survey on Crime Analysis and Prediction Using Data Mining and Machine Learning Techniques." Advances in Smart Grid Technology. Springer, Singapore, 2020. 435-448.

[5] Sreedhar, Chowdam, Nagulapally Kasiviswanath, and Pakanti Chenna Reddy. "Clustering large datasets using K-means modified inter and intra clustering (KM-I2C) in Hadoop." Journal of Big Data 4.1 (2017): 1-19.

[6] Rosal García, Emilio del. "Real life applications of bio-inspired computing models: EAP and NEPs." (2013).

[7] Rosal García, Emilio del, et al. "Simulating NEPs in a cluster with jNEP." International Journal of Computers, Communications & Control (2008).

[8] Navarrete, Carmen Navarrete, et al. "Parallel simulation of NEPs on clusters." 2011 IEEE/WIC/ACM International Conferences on Web Intelligence and Intelligent Agent Technology. Vol. 3. IEEE, 2011.

[9] S. Wolfram."Cellular Automata and Complexity: Collected Papers". CRC Press, 2018.

[10] M. D. Assunção, R. N. Calheiros, S. Bianchi, M. A. Netto & R. Buyya. “Big Data computing and clouds: Trends and Future Directions,” Journal of Parallel and Distributed Computing, 79 -80 (2015), pp. 3-15.

[11] X. Sun, B. Gao, Y. Zhang, W. An, H. Cao, C. Guo, W. Sun, “Towards delivering analytical solutions in cloud: Business models and technical challenges,” In Proceedings of the 8th IEEE International Conference on e-Business Engineering (ICEBE 2011), IEEE Computer Society, Washington, USA, 2011, pp. 347-351.

[12] D. Talia. “Toward Cloud-based Big-data Analytics,” IEEE Computer Science, (2013), pp. 98-101.

[13] C. Elena. “Business Intelligence”, Journal of Knowledge Management, Economics and Information Technology, 1(2), 2011, pp. 1-12.

[14] J. Ranjan. “Business Intelligence: Concepts, Components, Techniques and Benefits,” Journal of Theoretical and Applied Information Technology, 9(1), 2009, pp. 60-70.

[15] Zulkernine, F., Martin, P., Zou, Y., Bauer, M., Gwadry-Sridhar, F., & Aboulnaga, A. (2013). "Towards Cloud-Based Analytics-as-a-Service (CLAaaS) for Big Data Analytics in the Cloud". 2013 IEEE International Congress on Big Data.

[16] Madaan, Aman, et al. "Hadoop: Solution to Unstructured Data Handling." Big Data Analytics. Springer, Singapore, 2018. 47-54.

[17] Skourletopoulos, Georgios, et al. "Big data and cloud computing: a survey of the state-of-the-art and research challenges." Advances in mobile cloud computing and big data in the 5G Era. Springer, Cham, 2017. 23-41.

[18] Dabbèchi, Hichem, Ahlem Nabli, and Lotfi Bouzguenda. "Towards cloud-based data warehouse as a service for big data analytics." International Conference on Computational Collective Intelligence. Springer, Cham, 2016.

[19] Cardoso, Abílio, and Paulo Simões. "Cloud computing: Concepts, technologies and challenges." International Conference on Virtual and Networked Organizations, Emergent Technologies, and Tools. Springer, Berlin, Heidelberg, 2011.

[20] Kakhani, Manish Kumar, Sweeti Kakhani, and S. R. Biradar. "Research issues in big data analytics." International Journal of Application or Innovation in Engineering & Management 2.8 (2015): 228-232.

[21] Gandomi, Amir, and Murtaza Haider. "Beyond the hype: Big data concepts, methods, and analytics." International journal of information management 35.2 (2015): 137-144.

[22] Kitchin, Rob. "Big Data, New epistemologies and paradigm shifts." Big data & society 1.1 (2014): 2053951714528481.

[23] Jin, Xiaolong, et al. "Significance and challenges of big data research." Big Data Research 2.2 (2015): 59-64.

[24] Dean, Jeffrey, and Sanjay Ghemawat. "MapReduce: Simplified data processing on large clusters." (2004).

[25] Ananthanarayanan, Rajagopal, et al. "Cloud analytics: Do we really need to reinvent the storage stack?" HotCloud. 2009.

[26] https://geekflare.com/cloud-service-models, Accessed, August 2021.

[27] Csuhaj-Varjú, Erzsébet, and Arto Salomaa. "Networks of parallel language processors." New Trends in Formal Languages. Springer, Berlin, Heidelberg, 1997. 299-318.

[28] Csuhaj-Varj´ u, E., Dassow, J., Kelemen, J., Paun, G.: “Grammar Systems. Gordon and Breach”, London (1993).

[29] Castellanos, J., Mart´ın-Vide, C., Mitrana, V., Sempere, J.M.: “Solving NP-complete problems with networks of evolutionary processors”. In: Mira, J., Prieto, A.G. (eds.) IWANN 2001, Part I. LNCS, vol. 2084, p. 621. Springer, Heidelberg (2001).

[30] Castellanos, Juan, et al. "Networks of evolutionary processors." Acta informatica 39.6 (2003): 517-529.

[31] Martín-Vide, Carlos, and Victor Mitrana. "Networks of evolutionary processors: results and perspectives." Molecular Computational Models: Unconventional Approaches. IGI Global, 2005. 78-114.

[32] Díaz, Miguel Angel, et al. "Networks of evolutionary processors (NEP) as decision support systems." International Conference «Information Research & Applications»-i. Tech. 2007.

[33] Jimenez, Antonio, Emilio del Rosal, and Juan de Lara. "A visual language for modelling and simulation of networks of evolutionary processors." Trends in Practical Applications of Agents and Multiagent Systems (2010): 411-418.

[34] del Rosal, Emilio, Marina de la Cruz, and Alfonso Ortega de la Puente. "Towards the automatic programming of NEPs." International Work-Conference on the Interplay Between Natural and Artificial Computation. Springer, Berlin, Heidelberg, 2011.

[35] del Rosal, Emilio, and Miguel Cuéllar. "jNEPView: A graphical trace viewer for the simulations of nEPs." International Work-Conference on the Interplay Between Natural and Artificial Computation. Springer, Berlin, Heidelberg, 2009.

[36] Bel-Enguix, Gemma. "Computing Natural Language with Biomolecules: Overview and Challenges." International Work-Conference on the Interplay Between Natural and Artificial Computation. Springer, Berlin, Heidelberg, 2009.

[37] JIM ENEZ, Javier. "Energy transfer and constrained simulations in isotropic turbulence." CTR Annu. Res. Briefs (1993): 171-186.

[38] Joseph, S. Iwin Thanakumar, and Iwin Thanakumar. "Survey of data mining algorithms for intelligent computing system." Journal of trends in Computer Science and Smart technology (TCSST) 1.01 (2019): 14-24.

[39] Lu, Weijia. "Improved K-means clustering algorithm for big data mining under Hadoop parallel framework." Journal of Grid Computing (2019): 1-12.

[40] Sami Nayyef Al-Dabbagh, Bashar. Desarrollo de entorno online de programación para computación natural. MS thesis. 2019.

[41] G. M. Amdahl. “Validity of the single processor approach to achieving large scale computing capabilities.” In Proc. Spring Joint Computer Conference, pages 483–485, New York, NY, USA, 1967. ACM.

[42] Rosal García, Emilio del. "Real life applications of bio-inspired computing models: EAP and NEPs." (2013).

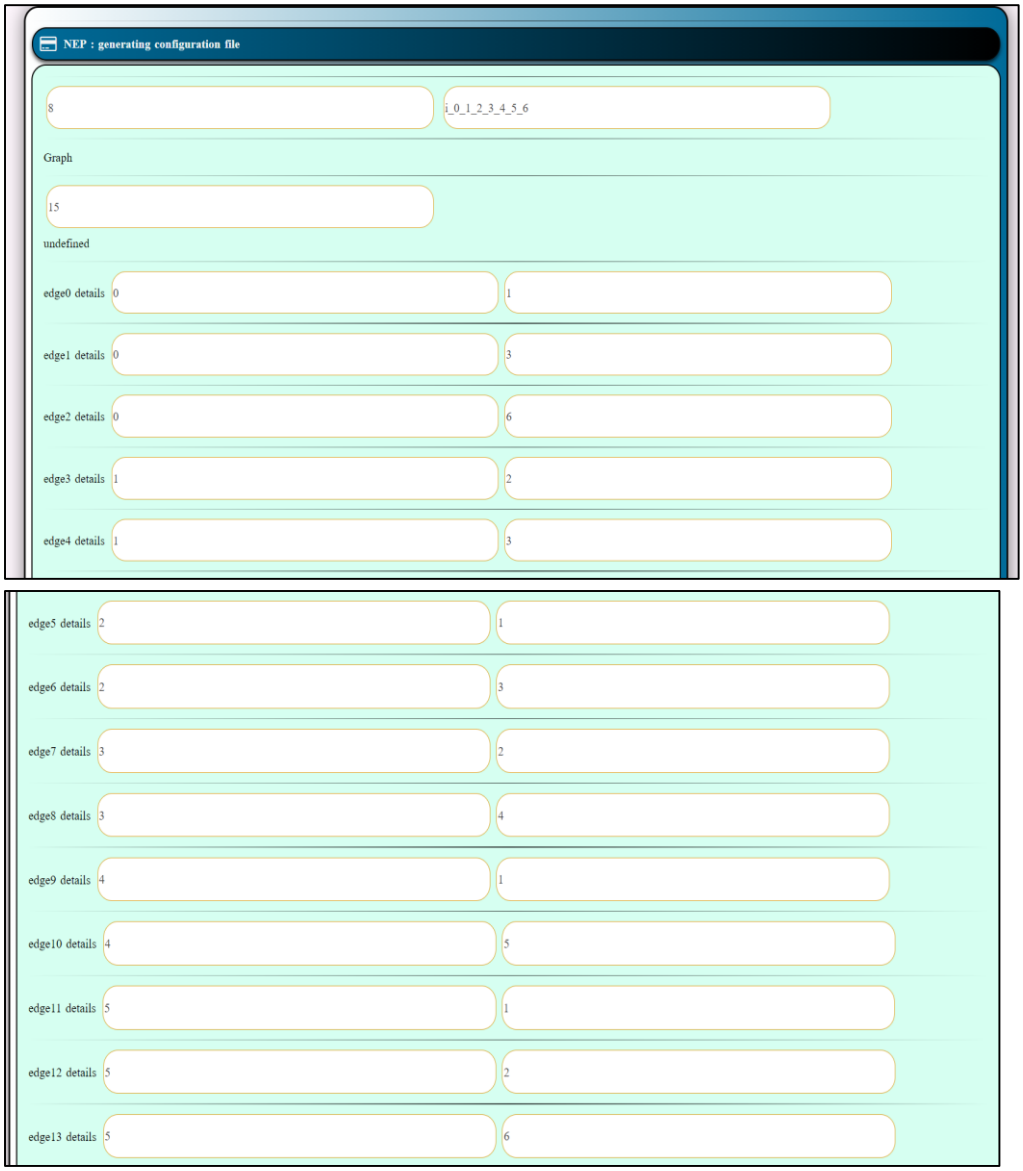
[43] Eiben, Agoston E., and James E. Smith. “Introduction to evolutionary computing.” Vol. 53. Berlin: springer, 2003.

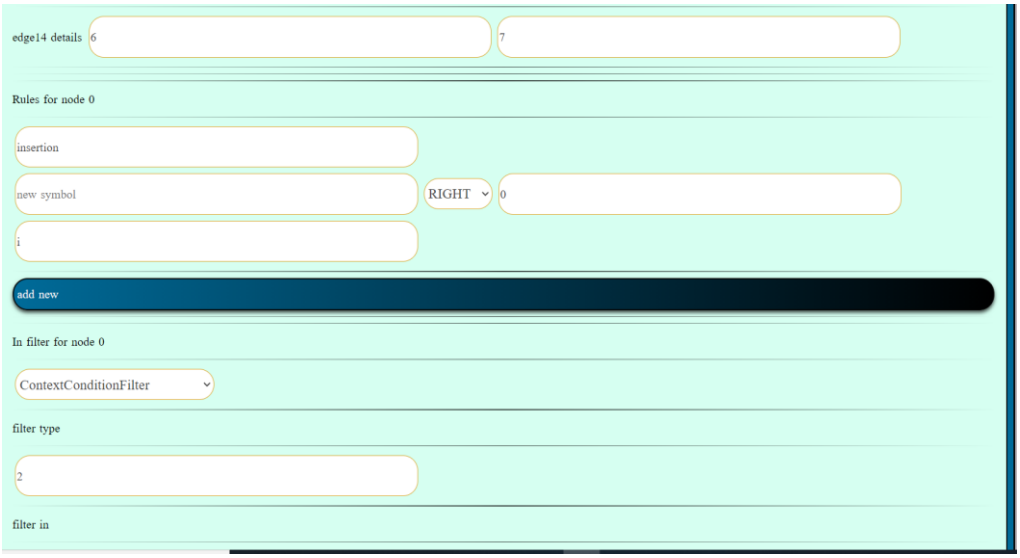
[44] Anuradha, J. "A brief introduction on Big Data 5Vs characteristics and Hadoop technology." Procedia computer science 48 (2015): 319-324.

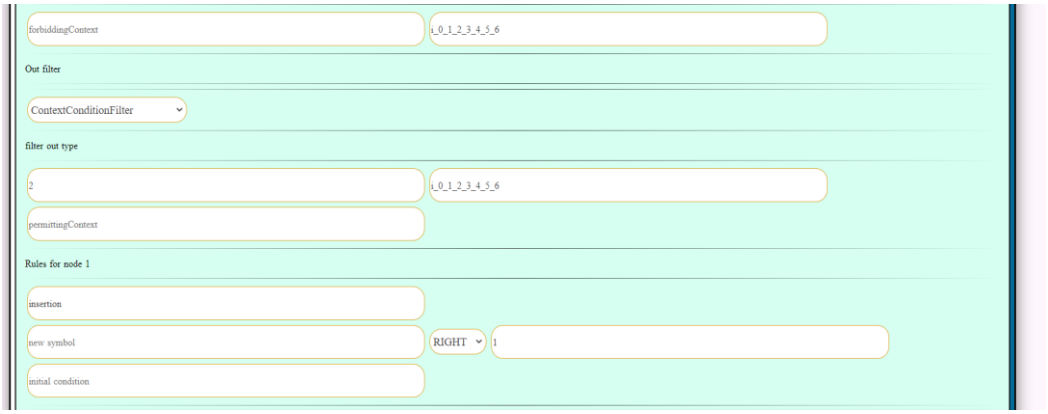
[45] https://spark.apache.org/docs/latest/cluster-overview.html, Accessed, August 2021.
[46] https://hpc-wiki.info/hpc/Scaling_tutorial#Scaling_tests, Accessed, August 2021.

APPENDIX A. INPUT FORM

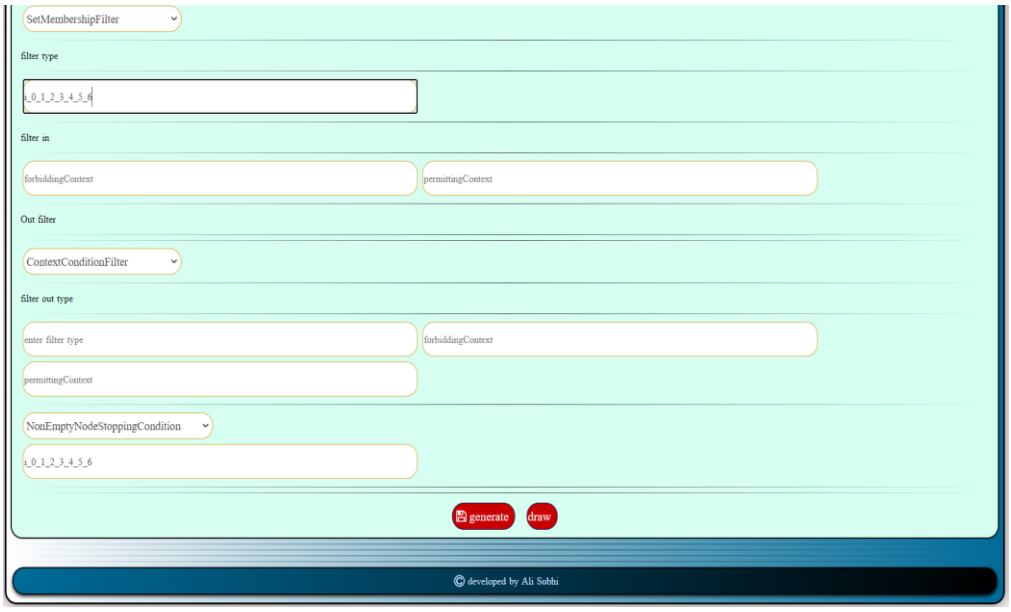
The Adleman problem have been introduced in [42] using manual writing of XML configurations. In this appendix, we introduce the input Form implementation.







Complete all rules and filters in the same way, and then



Setting up the last filter (SetMemberShipFilter) and the stopping condition, then when we finish just click on generate bottom to save the XML file, the XML file will be as follows:

```
<?xml version="1.0"?>

<!-- NEP Config file-->

<!-- The character "_" is reserved since it is used to separate symbols within words or within a set of symbols-->

<NEP nodes="8">

 <ALPHABET symbols="i_0_1_2_3_4_5_6"/>

 <GRAPH>

  <EDGE vertex1="0" vertex2="1"/>

  <EDGE vertex1="0" vertex2="3"/>

  <EDGE vertex1="0" vertex2="6"/>

  <EDGE vertex1="1" vertex2="2"/>

  <EDGE vertex1="1" vertex2="3"/>

  <EDGE vertex1="2" vertex2="1"/>

  <EDGE vertex1="2" vertex2="3"/>

  <EDGE vertex1="3" vertex2="2"/>

  <EDGE vertex1="3" vertex2="4"/>

  <EDGE vertex1="4" vertex2="1"/>

  <EDGE vertex1="4" vertex2="5"/>

  <EDGE vertex1="5" vertex2="1"/>

  <EDGE vertex1="5" vertex2="2"/>

  <EDGE vertex1="5" vertex2="6"/>

  <EDGE vertex1="6" vertex2="7"/>

 </GRAPH>

 <EVOLUTIONARY_PROCESSORS>
```

<NODE initCond="i">

<EVOLUTIONARY_RULES>

<RULE ruleType="insertion" actionType="RIGHT" symbol="0" newSymbol=""/>

</EVOLUTIONARY_RULES>

<FILTERS>

<INPUT type="2" permittingContext="i_0_1_2_3_4_5_6" forbiddingContext=""/>

<OUTPUT type="2" permittingContext="" forbiddingContext="i_0_1_2_3_4_5_6"/>

</FILTERS>

</NODE>

<NODE initCond="">

<EVOLUTIONARY_RULES>

<RULE ruleType="insertion" actionType="RIGHT" symbol="1" newSymbol=""/>

</EVOLUTIONARY_RULES>

<FILTERS>

<INPUT type="2" permittingContext="i_0_1_2_3_4_5_6" forbiddingContext=""/>

<OUTPUT type="2" permittingContext="i_0_1_2_3_4_5_6" forbiddingContext=""/>

</FILTERS>

</NODE>

<NODE initCond="">

<EVOLUTIONARY_RULES>

<RULE ruleType="" actionType="RIGHT" symbol="" newSymbol=""/>

</EVOLUTIONARY_RULES>

<FILTERS>

<INPUT type="2" permittingContext="i_0_1_2_3_4_5_6" forbiddingContext=""/>

<OUTPUT type="2" permittingContext="i_0_1_2_3_4_5_6" forbiddingContext=""/>

</FILTERS>

</NODE>

<NODE initCond="">

<EVOLUTIONARY_RULES>

<RULE ruleType="insertion" actionType="RIGHT" symbol="3" newSymbol=""/>

</EVOLUTIONARY_RULES>

<FILTERS>

<INPUT type="2" permittingContext="i_0_1_2_3_4_5_6" forbiddingContext=""/>

<OUTPUT type="2" permittingContext="i_0_1_2_3_4_5_6" forbiddingContext=""/>

</FILTERS>

</NODE>

<NODE initCond="">

<EVOLUTIONARY_RULES>

<RULE ruleType="insertion" actionType="RIGHT" symbol="4" newSymbol=""/>

</EVOLUTIONARY_RULES>

<FILTERS>

<INPUT type="2" permittingContext="i_0_1_2_3_4_5_6" forbiddingContext=""/>

<OUTPUT type="2" permittingContext="i_0_1_2_3_4_5_6" forbiddingContext=""/>

</FILTERS>

</NODE>

<NODE initCond="">

<EVOLUTIONARY_RULES>

<RULE ruleType="insertion" actionType="RIGHT" symbol="5" newSymbol=""/>

</EVOLUTIONARY_RULES>

<FILTERS>

<INPUT type="2" permittingContext="i_0_1_2_3_4_5_6" forbiddingContext=""/>

<OUTPUT type="2" permittingContext="i_0_1_2_3_4_5_6" forbiddingContext=""/>

</FILTERS>

</NODE>

<NODE initCond="">

<EVOLUTIONARY_RULES>

<RULE ruleType="insertion" actionType="RIGHT" symbol="6" newSymbol=""/>

</EVOLUTIONARY_RULES>

<FILTERS>

<INPUT type="2" permittingContext="i_0_1_2_3_4_5_6" forbiddingContext=""/>

<OUTPUT type="2" permittingContext="i_0_1_2_3_4_5_6" forbiddingContext=""/>

</FILTERS>

</NODE>

<NODE initCond="">

<EVOLUTIONARY_RULES>

<RULE ruleType="" actionType="RIGHT" symbol="" newSymbol=""/>

</EVOLUTIONARY_RULES>

<FILTERS>

```
<INPUT type="SetMembershipFilter" wordSet="i_0_1_2_3_4_5_6"/>
</FILTERS>
</NODE>
</EVOLUTIONARY_PROCESSORS>
<STOPPING_CONDITION>
<CONDITION type="NonEmptyNodeStoppingCondition" nodeID="i_0_1_2_3_4_5_6"/>
</STOPPING_CONDITION>
</NEP>
```

This XML file or configuration file that describe the problem will enter the Simulation Form to be handle.

The below code for the input form in java script:

```
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<link href="style/css.css" rel="stylesheet" type="text/css">
<link href="style/print.css" rel="stylesheet" media="print" type="text/css">
<link href="style/admin.css" rel="stylesheet" type="text/css">
<link href="style/font-awesome/css/font-awesome.min.css" rel="stylesheet" type="text/css">
<title> </title>
<script type="text/javascript" src="js/jquery.min.js"></script>
<script type="text/javascript" >
var source='<?xml version="1.0"?>\n<!-- NEP Config file-->\n<!-- The character "_" is reserved since it is used to separate symbols within words or within a set of symbols-->';
var rulesstrxml='';
var filterstrxm='';
```

```
var edxml='';

var outputxml=source;

function genrules()

 {

        $("#rulandfl").empty();

        $("#rulandfl").empty();

        var no_nodes=document.getElementById("no_node").value;

        no_nodes_int= parseInt(no_nodes);

        var rules;

        var filter;

        for (let index = 0; index < no_nodes_int; index++)

{

rules='<hr>Rules for node '+index+'<hr><input ind="'+index+'" onchange="modifyrule(this);" list="type" id="'+'rule'+index;

rules+='"placeholder="select rule type"><datalist id="type"><option value="deletion"></option>';

rules+='<option value="insertion"></option>';

rules+='<option value="Evolutionary"></option>';

rules+='<option value="LeftMostParsing"></option>';

rules+='<option value="SplicingChoudhary"></option>';

rules+='<option value="SplicingParsing"></option>';

rules+='<option value="Splicing"></option>';

rules+='<option value="Substitution"></option>';

rules+='</datalist>';

rules+='<div id="onchangeevent" ind="'+index+'"><input list="text" id="'+'rulenews'+index+'"" placeholder="new symbol">';

rules+='<select id="'+'ruleper'+index+'"" placeholder="action type"><option>RIGHT</option><option>LEFT</option><option>ANY</option></select>';
```

```
rules+='<input list="text" id="'+'rulesden'+index+'" placeholder="symbol">';

rules+='<input list="text" id="'+'rulesini'+index+'" placeholder="initial condition"></div>';

filter='<hr><div class="tips" id="rulefornodeiii'+index+'" ondblclick="addnewrule(this)"> add new </div><hr><hr>In filter for node '+index+'<hr><select ind="'+index+'" onchange="setm(this)" id="'+'filter11'+index+'">';

filter+= '<option value="2">ContextConditionFilter</option>';

filter+= '<option value="1">RegularLangMembershipFilter</option>';

filter+='<option value="3">SetMembershipFilter</option></select>';

filter+='<hr> filter type <hr> <div id="setmf'+index+'"><input list="text" id='+'filter'+index+' placeholder="enter filter type"></div>';

filter+='<hr> filter in <hr> <input list="text" id='+'filterin'+index+' placeholder="forbiddingContext">';

filter+='<input list="text" id='+'filterini'+index+' placeholder="permittingContext">';

filter+='<hr>Out filter <hr>';

filter+='<hr><select id="'+'filter12'+index+'">';

filter+= '<option value="2">ContextConditionFilter</option>';

filter+= '<option value="1">RegularLangMembershipFilter</option>';

filter+='<option value="3">SetMembershipFilter</option></select>';

filter+='<hr> filter out type <hr> <input list="text" id='+'filter1'+index+' placeholder="enter filter type">';

filter+='<input list="text" id='+'filterout'+index+' placeholder="forbiddingContext">';

filter+='<input list="text" id='+'filterouti'+index+' placeholder="permittingContext">';

filter+='';

$("#rulandfl").append(rules);

$("#rulandfl").append(filter);

}

}

function genedges()
```

```
{
        $("#loadedg").empty();
        var no_ed_int=document.getElementById("noofver").value
        no_ed_int= parseInt(no_ed_int);
        var edstr;
        for (let index = 0; index < no_ed_int; index++)
        {
        edstr+='<hr>edge'+index+' details ';
edstr+='<input list="text" id='+'edstr'+index+' placeholder="vertex1">';
        edstr+='<input list="text" id='+'edstrend'+index+' placeholder="vertex2">';
        }
        $("#loadedg").append(edstr);
}
function gen()
{
        var no_ed_int=document.getElementById("noofver").value
        no_ed_int= parseInt(no_ed_int);
        var edstr;
outputxml+="\n"+'<NEP nodes="'+
document.getElementById("no_node").value+'">';
outputxml+="\n"+' <ALPHABET symbols="'+
document.getElementById("alphapet").value+'"/>';
        var no_ed_int=document.getElementById("noofver").value
        no_ed_int= parseInt(no_ed_int);
        var edstr;
        outputxml+="\n"+' <GRAPH>';
        for (let index = 0; index < no_ed_int; index++)
        {
```

```
                var      ids12='edstr'+index;
                ids12=document.getElementById(ids12).value;
                var      ids13='edstrend'+index;
                ids13=document.getElementById(ids13).value;
outputxml+="\n"+'  <EDGE vertex1="'+ids12+'" vertex2="'+ids13+'"/>';
        }
        outputxml+="\n"+' </GRAPH>';
        var no_nodes=document.getElementById("no_node").value;
        no_nodes_int= parseInt(no_nodes);
        var rules11="";
        var filter11="";
        //outputxml+=rulesstrxml;
        outputxml+="\n"+' <EVOLUTIONARY_PROCESSORS>';
        for (let index = 0; index < no_nodes_int; index++)
        {
                var ids1='rule'+index;
                rules11="";
                filter11="";
                ids1=document.getElementById(ids1).value;
                var ids4_11='filter11'+index;
                ids4_11=document.getElementById(ids4_11).value;
                var ids1_1='rulesini'+index;
                ids1_1=document.getElementById(ids1_1).value;
                var ids2='ruleper'+index;
                ids2=document.getElementById(ids2).value;
                var ids3='rulesden'+index;
                ids3=document.getElementById(ids3).value;
```

```
if(ids4_11!="3" )
{
var ids4='filter'+index;
ids4=document.getElementById(ids4).value;
}
if(ids4_11!="3" )
{
var ids4_1='filter1'+index;
ids4_1=document.getElementById(ids4_1).value;
var ids5='filterin'+index;
ids5=document.getElementById(ids5).value;
var ids51='filterini'+index;
ids51=document.getElementById(ids51).value;
var ids6='filterout'+index;
ids6=document.getElementById(ids6).value;
var ids61='filterouti'+index;
ids61=document.getElementById(ids61).value;
}
var ids66='rulenews'+index;
ids66=document.getElementById(ids66).value;
if(ids1=="SplicingChoudhary" | ids1=="Splicing" | ids1=="SplicingParsing")
{
var ids3_1='rulesdenv'+index;
ids3_1=document.getElementById(ids3_1).value;
rules11+='\n <NODE initCond="'+ids1_1+'">\n <EVOLUTIONARY_RULES>\n
<RULE ruleType="'+ids1+'" wordX="'+ids2+'" wordY="'+ids3+'" wordU="'+ids66+'"
wordV="'+ids3_1+'"/>\n </EVOLUTIONARY_RULES>';
}
```

```
else

rules11+='\n <NODE initCond="'+ids1_1+'">\n <EVOLUTIONARY_RULES>\n <RULE ruleType="'+ids1+'" actionType="'+ids2+'" symbol="'+ids3+'" newSymbol="'+ids66+'"/>\n </EVOLUTIONARY_RULES>';

outputxml+="\n"+rules11;

if(ids4_11=="3" )

{

var ids4_11_1='filtewordSet'+index;

ids4_11_1=document.getElementById(ids4_11_1).value;

filter11+='\n <FILTERS>\n <INPUT type="SetMembershipFilter" wordSet="'+ids4_11_1+'"/>\n </FILTERS>\n </NODE>';

}

else

{

filter11+='\n <FILTERS>\n';

if(ids4!="" | ids51!="" | ids5!="" )

{

filter11+=' <INPUT type="'+ids4+'" permittingContext="'+ids51+'" forbiddingContext="'+ids5+'"/>';

}

if(ids4_1!="" | ids61!="" | ids6!="" )

{

filter11+='\n <OUTPUT type="'+ids4_1+'" permittingContext="'+ids61+'" forbiddingContext="'+ids6+'"/>';

}

filter11+='\n </FILTERS>\n </NODE>';

}

outputxml+="\n"+filter11;

}
```

```
outputxml+="\n"+' </EVOLUTIONARY_PROCESSORS>';

outputxml+="\n"+' <STOPPING_CONDITION>';

if(document.getElementById("stop").value=="NonEmptyNodeStoppingCondition" )

{

outputxml+="\n"+'                                                    <CONDITION
type="'+document.getElementById("stop").value+'"
nodeID="'+document.getElementById("exstop").value+'"/>';

}

else

        outputxml+="\n"+'                                            <CONDITION
type="'+document.getElementById("stop").value+'"
maximum="'+document.getElementById("exstop").value+'"/>';

outputxml+="\n"+' </STOPPING_CONDITION>';

outputxml+="\n"+'</NEP>';

downloadxml("out.XML",outputxml);

}


function modifyrule(p11)

{

if(p11.value=="SplicingChoudhary"  |  p11.value=="Splicing"  |
p11.value=="SplicingParsing")

        {

                $("#onchangeevent").empty();

                var index=p11.ind;

                var ty="";

ty+='<div  id="onchangeevent"><input  list="text"  id="'+'rulenews'+index+'"'
placeholder="wordX">';

ty+='<input list="text" id="'+'ruleper'+index+'" placeholder="wordY">';

ty+='<input list="text" id="'+'rulesden'+index+'" placeholder="wordU">';
```

```
ty+='<input list="text" id="'+'rulesdenv'+index+'" placeholder="wordV">';
ty+='<input list="text" id="'+'rulesini'+index+'" placeholder="initial condition">';
                $("#onchangeevent").append(ty);
        }
}
function setm(p11)
{
        var idth=p11.indx;
        var index=p11.getAttribute('ind');
        if(p11.value=="3" )
        {
                var idxs="#setmf"+index;
                $(idxs).empty(idxs);
                var ty="";
ty+='\n  <input list="text" id='+'filtewordSet'+index+' placeholder="wordSet">';
                $(idxs).append(ty);
        }
}
function runch(p11)
{
        if(p11.value=="NonEmptyNodeStoppingCondition" )
        {
                $("#stnon").empty();
var ty='\n  <input list="text" id='+'exstop  placeholder=" enter node id">';
                $("#stnon").append(ty);
        }
}
```

```
function downloadxml(fil, txtInput)
{
	var element = document.createElement('a');
element.setAttribute('href','data:text/plain;charset=utf-8,' +
encodeURIComponent(txtInput));
	element.setAttribute('download', fil);
	document.body.appendChild(element);
	element.click();
}

function addnewrule(pointerx)
{
	var no_node_requierd = prompt("Please enter required nodes", "1");
	var toaddid=pointerx.getAttribute('id');
	var toid="#"+toaddid;
	no_node_requierd= parseInt(no_node_requierd);
	var rules =null;
	for (let index = 0; index < no_node_requierd; index++)
	{
rules='<hr>additional rule '+index+'<hr><input ind="'+index+'"
onchange="modifyrule(this);" list="type" id="'+'rule'+index;
rules+='"placeholder="select rule type"><datalist id="type"><option
value="deletion"></option>';
		rules+='<option value="insertion"></option>';
		rules+='<option value="Evolutionary"></option>';
		rules+='<option value="LeftMostParsing"></option>';
		rules+='<option value="SplicingChoudhary"></option>';
		rules+='<option value="SplicingParsing"></option>';
```

```
                rules+='<option value="Splicing"></option>';

                rules+='<option value="Substitution"></option>';

                rules+='</datalist>';

rules+='<div id="onchangeevent" ind="'+index+'"><input list="text" id="'+'rulenews'+index+'"" placeholder="new symbol">';

rules+='<select id="'+'ruleper'+index+'"" placeholder="action type"><option>RIGHT</option><option>LEFT</option><option>ANY</option></select>';

rules+='<input list="text" id="'+'rulesden'+index+'" placeholder="symbol">';

rules+='<input list="text" id="'+'rulesini'+index+'" placeholder="initial condition"></div>';

                $(toid).append(rules);

        }

}

</script>

</head>

<body>

<div class="bod">

<hr>

<div class="tips" id="3dtext"><i class="fa fa-credit-card fa-lg"></i> <b> NEP : generating configuration file </b><B id="bal" ></B></div>

<div class="adminback">

<div class="Auth"><hr>

<input id="no_node" required accept="number" onchange="genrules();" pat="^[0-9]$"placeholder=" enter number of node ">

<input type="text" id='alphapet' list="place" placeholder="enter alphapet">

<hr>

Graph

<hr>
```

```
<input type="text" id='noofver' onchange="genedges();"  placeholder="`ente number of edge">

<div id='loadedg'></div>

<hr>

<hr>

<div id="rulandfl"></div>

<hr>

<select id="stop" onchange="runch(this);">

<option value="ConsecutiveConfigStoppingCondition">ConsecutiveConfigStoppingCondition</option>

<option value="MaximumStepsStoppingCondition">MaximumStepsStoppingCondition</option>

<option value="NoChangesStoppingCondition">NoChangesStoppingCondition</option>

<option value="NonEmptyNodeStoppingCondition">NonEmptyNodeStoppingCondition</option>

<option value="WordsDisappearStoppingCondition">WordsDisappearStoppingCondition</option>

</select>

<div id="stnon">

<input type="text" placeholder="enter maximam" id="exstop">

</div>

<hr>

<hr>

<center>

<button  onclick="gen();"><i class="fa fa-floppy-o"></i>generate </button>
```

 <button> <type="draw">draw</button>

<hr><hr>

<ul id="listview">

<hr><hr>

developed by Ali Subhi

<hr> اقفاء <hr>

<hr>

</body>

</html>

APPENDIX B. SYSTEM SIMULATION

To make this code work we need Eclipse IDE, you must install latest eclipse stable version not a beta release to avoid errors. Then open the Eclipse IDE, choose Help and Install New Software.

In the Work with: field, type http://download.eclipse.org/releases/kepler or the path that matches the version number of your Eclipse IDE.

In the items list, choose Database Development and Finish, restart Eclipse when prompted.

Next, install the Toolkit for Eclipse to make the helpful, pre-configured source code project templates available.

To install the Toolkit for Eclipse, open the Eclipse IDE, choose Help and Install New Software.

In the Work with: field, type https://aws.amazon.com/eclipse, in the items list, choose AWS Toolkit for Eclipse and Finish, restart Eclipse when prompted, next, create a new AWS Java project, copy the code below and paste it in the AWS java project to run.

This is a Spark application we wrote to simulate the NEP, it takes the XML file generated from the input form or anywhere to handle and save the result in TXT file and show the execution time it takes to handle the problem:

```
package com.amazonaws.ayman;

import java.io.BufferedReader;

import java.io.IOException;

import java.io.InputStreamReader;

import com.amazonaws.AmazonClientException;

import com.amazonaws.auth.AWSCredentials;

import com.amazonaws.auth.AWSStaticCredentialsProvider;

import com.amazonaws.auth.profile.ProfileCredentialsProvider;

import com.amazonaws.regions.Regions;

import com.amazonaws.services.elasticmapreduce.AmazonElasticMapReduce;

import
com.amazonaws.services.elasticmapreduce.AmazonElasticMapReduceClientBuilder;

import com.amazonaws.services.elasticmapreduce.model.*;
```

```
import com.amazonaws.services.elasticmapreduce.util.StepFactory;

import com.amazonaws.AmazonServiceException;

import com.amazonaws.regions.Regions;

import com.amazonaws.services.s3.AmazonS3;

import com.amazonaws.services.s3.AmazonS3ClientBuilder;

public class main

 {

        public static void dealswiths3(String args)

{

final String USAGE = "\n" +", supply the name (key) of an S3 object, the bucket name\n" +"that it's contained within, and the bucket to save it to.\n" +"\n" +"Ex: \n";

if (args == "") {System.out.println(USAGE);

System.exit(1);

      }

      String object_key = args;

String from_bucket ="clusterrunning/output/dataset1" ;

String to_bucket = "clusterrunning/output";

System.out.format("Copying object %s from bucket %s to %s\n",

object_key, from_bucket, to_bucket);

final AmazonS3 s3 = AmazonS3ClientBuilder.standard().withRegion("us-east-2").build();

      try

{

         s3.copyObject(from_bucket, object_key, to_bucket, object_key);

      }

catch (AmazonServiceException e) {

System.err.println(e.getErrorMessage());

System.exit(1);
```

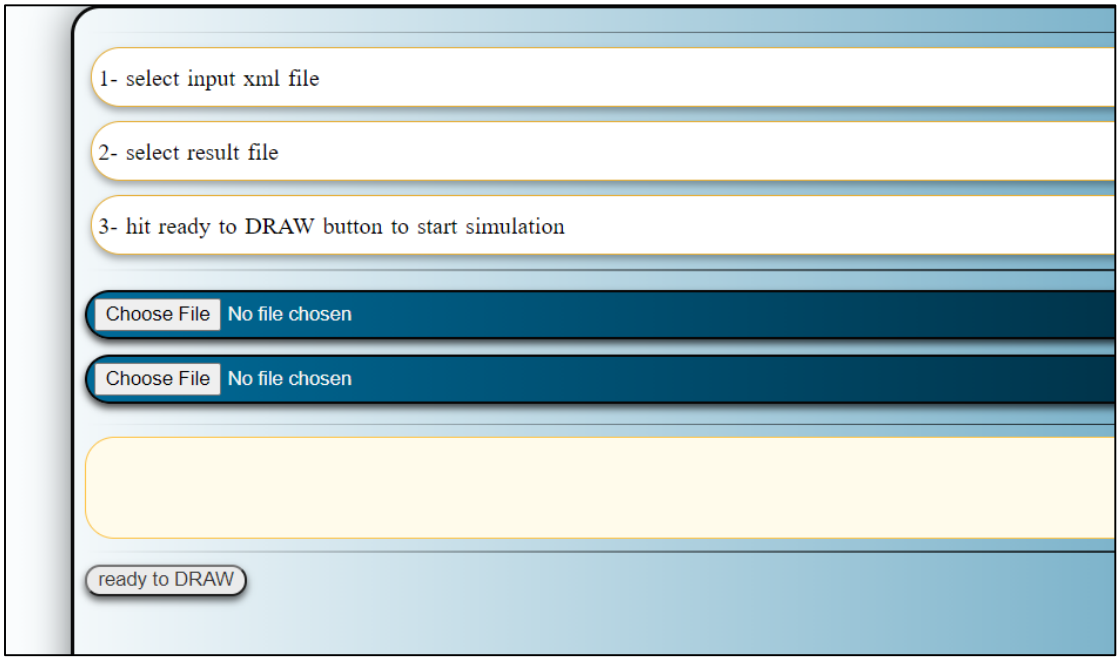
```
        }
        System.out.println("Done!");
    }
public static void main(String[] args) {
long startTime = System.nanoTime();
AWSCredentials credentials_profile = null;
try
{
credentials_profile = new ProfileCredentialsProvider("default").getCredentials();
}
catch (Exception e)
{
        throw new AmazonClientException(
        "Cannot load credentials from .aws/credentials file. " +
"Make sure that the credentials file exists and the profile name is specified within it.",
e);
}
AmazonElasticMapReduce emr = AmazonElasticMapReduceClientBuilder.standard()
.withCredentials(new AWSStaticCredentialsProvider(credentials_profile))
.withRegion(Regions.US_WEST_1).build();
StepFactory stepFactory = new StepFactory();
StepConfig runBashScript = new StepConfig()
.withName("Run a bash script")
.withHadoopJarStep(stepFactory.newScriptRunnerStep("s3://us-east-
2.elasticmapreduce/libs/script-runner/script-runner.jar"))
.withActionOnFailure("CONTINUE");
HadoopJarStepConfig hadoopConfig1 = new HadoopJarStepConfig()
```

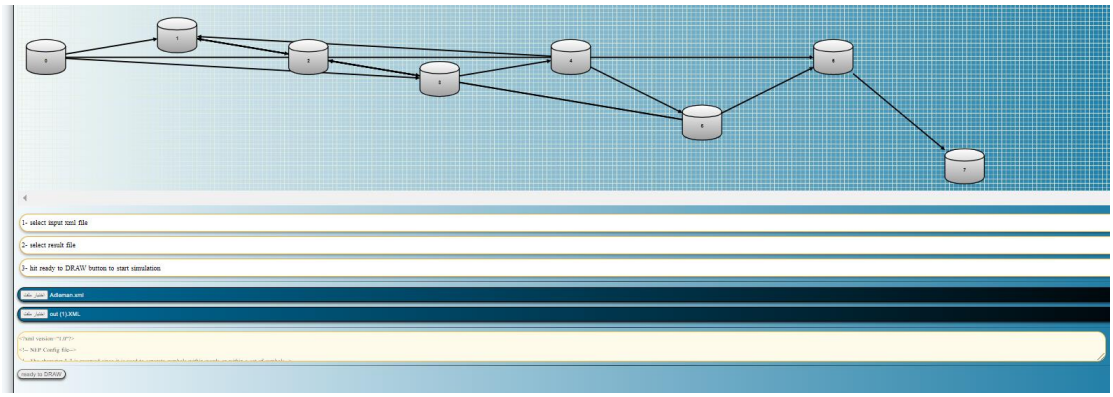
```
.withJar("s3://clusterrunning/build/classes/net/e_delrosal/jnep") // replace with the location of the jar to run as a step

.withMainClass("NEP") // optional main class, this can be omitted if jar above has a manifest

.withArgs("--verbose"); // optional list of arguments to pass to the jar

System.out.println(emr);

System.out.println(runBashScript);

StepConfig myCustomJarStep = new StepConfig("RunSpark", hadoopConfig1);

String input="3Coloring.xml";

String output="3Coloring.txt";

String cmd = " java -cp build/classes net.e_delrosal.jnep.NEP \r\n"+input

+ "\r\n";

Runtime run = Runtime.getRuntime();

Process pr = null;

try

{

        pr = run.exec(cmd);

        System.out.println(pr);

}

catch (IOException e)

{

        e.printStackTrace();

}

try

{

        pr.waitFor();

}

catch (InterruptedException e)
```

```
{
        e.printStackTrace();
}
BufferedReader buf = new BufferedReader(new
InputStreamReader(pr.getInputStream()));
String line = "output successfully for s3";
try
{
        while ((line=buf.readLine())!=null) {
        System.out.println(line);
}
}
catch (IOException e)
{
        e.printStackTrace();
}
System.out.println(myCustomJarStep);
long elapsedTime = System.nanoTime() - startTime ;
System.out.println("Total execution time to run  "+ myCustomJarStep +"    "
+ elapsedTime/1000000);
dealswiths3(output);
}
}
```

APPENDIX C. SIMULATION FORM DRAWING

In the last step we draw all the possible steps that the execution run with beginning from the initial step to the end, and make the user choose the step that he prefers to see, first to choose the input file and the result file and then click on Ready to Draw bottom to see the result as follows:





The below code in java script to do the drawing form of course with other CSS files supported the design:

<html>

<head>

<script src="../js/jquery.min.js">

```
</script>
<title>Animation example for mxGraph</title>
<link href="../style/css.css" rel="stylesheet" type="text/css">
<link href="../style/admin.css" rel="stylesheet" type="text/css">
<link href="../style/font-awesome/css/font-awesome.min.css" rel="stylesheet" type="text/css">
<script type="text/javascript">
mxBasePath = '../src';
</script>
<script type="text/javascript" src="../src/js/mxClient.js"></script>
<style type="text/css">
.flow
{
	stroke-dasharray: 8;
	animation: dash 0.5s linear;
	animation-iteration-count: infinite;
}
@keyframes dash {
to
{
stroke-dashoffset: -16;
}
}
</style>
<script type="text/javascript">
var Processor=0;
var arrayoft=[];
```

```
var working_area_pr=[];
var vertexarry1=[];
var vertexarry2=[];
function main(container,labels){
spss();
var graph = new mxGraph(container);
graph.setEnabled(false);
var parent = graph.getDefaultParent();
var vertexStyle = 'shape=cylinder;strokeWidth=2;fillColor=#ffffff;strokeColor=black;'
+
'gradientColor=#a0a0a0;fontColor=black;fontStyle=1;spacingTop=14;';
graph.getModel().beginUpdate();
try
{
        var formal15;
        var ts;
        var counter=1;
        for(j=0;j<parseInt(Processor);j++)
{
                ts=j;
                counter++;
                var x=j * 300 + 20;
                var y=j * 50;
                if(labels.length>0)
                {
                        var entire=labels [j];
                        var ch15=entire.substr(23,50);
```

```
                formal15=ch15;
                ts=ch15;
            }
            if (counter==3)
            {
                counter=1;
arrayoft.push(graph.insertVertex(parent, null, ts, x, y, 90, 80,vertexStyle));
            }
            else
arrayoft.push(graph.insertVertex(parent, null, ts, x, 100, 90, 80,vertexStyle));
            }
            var e1=[];
            for(vx=0;vx<vertexarry1.length;vx++)
            {
                var v1ind=vertexarry1[vx];
                var x=vx * 100;
                var y=vx + 100;
                var v2ind=vertexarry2[vx];
                var v1=arrayoft[v1ind];
                var v2=arrayoft[v2ind];
                e1.push (graph.insertEdge(parent, null, '', v1, v2,
'strokeWidth=3;endArrow=block;endSize=2;
endFill=1;strokeColor=black;rounded=1;'));
                graph.orderCells(true, [e1[vx]]);
            }
        document.getElementById("graphContainer").style.display="block";
}
```

```
finally
{
        graph.getModel().endUpdate();
}
};
function spss()
{
      var text=$("#result").val();
      var pre1=text.replace(/JS/g,"***************","");
      var pre2=pre1.replace(/JS/g,'---',"");
      var stp=getmultipeocur('TOTAL STEPS:', pre2);
$("#sel").empty();
for(iter=0; iter<stp.length;iter++)
{
var iter1=iter+1;
        var listr="<option value="'+iter+'"> step "+iter1+" </option>";
        $("#sel").append(listr);
        var bstr="TOTAL STEPS:";
        var element=stp[iter];
        var startind=parseInt(element)+bstr.length;
        var endind=startind+3;
        var curentstp=pre2.substring(startind,endind);
        var st="st="+ startind+"ind="+endind + "cur="+curentstp;
        var next=iter+1;
        var working_area=pre2.substring(endind, stp[next]);
        working_area=          working_area.replace("***************","");
        working_area_pr.push(working_area);
```

```
}
}
var readXml=null;
function loadxmfile(){
event.preventDefault();
var selectedFile = document.getElementById('input').files[0];
console.log(selectedFile);
var reader = new FileReader();
reader.onload = function(e)
{
        readXml=e.target.result;
        console.log(readXml);
        var parser = new DOMParser();
        var doc = parser.parseFromString(readXml, "application/xml");
        var nep=doc.getElementsByTagName("NEP");
        Processor=nep[0].getAttribute('nodes');
        var txt=null;
        var edge=doc.getElementsByTagName("EDGE");
        for (i = 0; i < edge.length; i++)
        {
                vertexarry1.push(edge[i].getAttribute('vertex1'));
                vertexarry2.push(edge[i].getAttribute('vertex2'));
                txt +="vetex"+ edge[i].getAttribute('vertex2') + "<br>";
                txt += edge[i].getAttribute('vertex1') + "<br>";
        }
        console.log();
}
```

```
reader.readAsText(selectedFile);
}
function getmultipeocur(searchStr, str, caseSensitive)
{
var searchStrLen = searchStr.length;
if (searchStrLen == 0)
{
        return [];
    }
    var startIndex = 0, index, indices = [];
    if (!caseSensitive)
{
        str = str.toLowerCase();
        searchStr = searchStr.toLowerCase();
    }
    while ((index = str.indexOf(searchStr, startIndex)) > -1) {
    indices.push(index);
    startIndex = index + searchStrLen;
  }
  return indices;
}
function extractsteps(working_area)
{
    var lab=[];
    for(j=0;j<parseInt(Processor);j++)
    {
        var bstr1=" Evolutionary Processor ";
```

```
                bstr1+=j;
                var  DH= parseInt(j)+1;
                var bstr2= " Evolutionary Processor " + DH;
                var ep1=working_area.indexOf(bstr1)
                var ep2=working_area.indexOf(bstr2)
                 if(j!=parseInt(Processor)-1)
                {var curentstp1=working_area.substring(ep1,ep2);
                lab.push(curentstp1);
        }
        else
        {
                 DH= parseInt(j);
                 bstr2= " Evolutionary Processor " + DH;
                 console.log(bstr2);
                ep2=working_area.indexOf(bstr2);
                var str3=working_area.substring(ep2);
                ep3=str3.indexOf("----------------------");
                var curentstp1=str3.substring(0,ep3);
                lab.push(curentstp1);
        }
        }
        main(document.getElementById('graphContainer'),lab);
}
function drawnep()
{
        var ft=document.getElementById("sel").value;
        ft=parseInt(ft);
```

```
            extractsteps(working_area_pr[ft]);
}
var readXml1=null;
function loadxmfile1()
{
            event.preventDefault();
            var selectedFile = document.getElementById('input1').files[0];
           console.log(selectedFile);
            var reader = new FileReader();
            reader.onload = function(e) {
            readXml1=e.target.result;
document.getElementById("result").value=readXml1;
    }
    console.log();
    reader.readAsText(selectedFile);
    }
</script>
</head>
<!-- Page passes the container for the graph to the program -->
<body class="bod" >
<!-- Creates a container for the graph with a grid wallpaper -->
<div id="graphContainer"
style="display:                                                                    none;
position:relative;overflow:scroll;width:100%;height:500px;background:url('editors/im
ages/grid.gif');cursor:default;">
<select onchange="drawnep();" class="headlink aa"  id="sel"></select>
</div>
<hr>
```

```
<ul>

<li> 1- select input xml file</li>

<li> 2- select result file</li>

<li>3- hit ready to DRAW  button to start simulation</li>

</ul>

<hr>

<input onchange="loadxmfile();"  class="tips" id="input" type="file" value="input xml file to start">

<input onchange="loadxmfile1();"  class="tips" id="input1" type="file" value="choose file to start">

<hr>

<textarea  onchange="spss();" id="result" placeholder="enter result here" >

</textarea>

<hr>

<button class="headlink aa" onclick="main(document.getElementById('graphContainer'),[])"> ready to DRAW</button>

<br>

<script>

</script>

</body>

</html>
```