A Fuzzy Logic based Solution for Network Traffic Problems in Migrating Parallel Crawlers

Mohammed Faizan Farooqui[1], Mohammad Muqeem[2], Sultan Ahmad[3]*, Jabeen Nazeer[4], Hikmat A. M. Abdeljaber[5]
Department of Computer Application, Integral University, Lucknow, India[1, 2]
Department of Computer Science-College of Computer Engineering and Sciences, Prince Sattam Bin Abdulaziz University, Al-Kharj, 11942, Saudi Arabia[3*, 4]
University Center for Research & Development (UCRD)-Department of Computer Science and Engineering, Chandigarh University, Gharuan, Mohali 140413, Punjab, India[3]
Department of Computer Science-Faculty of Information Technology, Applied Science Private University, Amman, Jordan[5]

***Abstract*—Search engines are the instruments for website navigation and search, because the Internet is big and has expanded greatly. By continuously downloading web pages for processing, search engines provide search facilities and maintain indices for web documents. Online crawling is the term for this process of downloading web pages. This paper proposes solution to network traffic problem in migrating parallel web crawler. The primary benefit of a parallel web crawler is that it does local analysis at the data's residence rather than inside the web search engine repository. As a result, network load and traffic are greatly reduced, which enhances the performance, efficacy, and efficiency of the crawling process. Another benefit of moving to a parallel crawler is that as the web gets bigger, it becomes important to parallelize crawling operations in order to retrieve web pages more quickly. A web crawler will produce pages of excellent quality. When the crawling process moves to a host or server with a specific domain, it begins downloading pages from that domain. Incremental crawling will maintain the quality of downloaded pages and keep the pages in the local database updated. Java is used to implement the crawler. The model that was put into practice supports all aspects of a three-tier, real-time architecture. An implementation of a parallel web crawler migration is shown in this paper. The method for efficient parallel web migration detects changes in the content and structure using neural network-based change detection techniques in parallel web migration. This will produce high-quality pages and detection for changes will always download new pages. Either of the following strategies is used to carry out the crawling process: either crawlers are given generous permission to speak with one another, or they are not given permission to communicate with one another at all. Both strategies increase network traffic. Here, a fuzzy logic-based system that predicts the load at a specific node and the path of network traffic is presented and implemented in MATLAB using the fuzzy logic toolbox.**

***Keywords*—Web crawler; incremental crawling; fuzzy logic-based system; fuzzy logic toolbox**

I. Introduction

The Internet's numerous data sources, dynamic page production, and quick rate of change present a number of challenges for web crawling. All web crawlers are constructed using common components and must be scalable, reliable, and able to utilize available bandwidth effectively. When building a web crawler, politeness is a crucial issue that needs to be taken into consideration [1]. Crawlers should refrain from overtaxing a web server by making numerous page requests quickly [2]. Web crawlers should adhere to the guidelines set forth by website administrators and should self-identify while seeking pages. The crawlers note a delay between two requests being sent to a web server simultaneously. Request Intervals are the name given to this waiting period [3]. Typically, 30 seconds elapses between downloads. A queue shuffling mechanism is used to enforce this waiting period; the queue is shuffled into a random order and distributed evenly among URLs coming from the same web server. The other crawler, similar to Mercator, implements their URL queue as a group of sub-queues, with a queue for each domain [4]. Each search engine keeps a central database of web pages on hand. In response to the query of user, search engine creates indexes for the repository. A web crawler, sometimes known as a spider, robot, or web pot, is a program that searches the Internet and collects web pages. Beginning with the seed URLs, the web crawler downloads the web document for those URLs. It takes new links from these downloaded documents and extracts them. Next, it is determined whether or not the extracted URLs have already been downloaded. URLs are redistributed to crawlers for additional downloading if it is confirmed that the documents have not already been downloaded. Up until there are no more URLs available for download, the process is repeated. Every day, a crawler downloads millions of online pages. The scheduler, multi-threaded downloader, queue for URLs and storage for text and metadata are all components of a web crawler. The search engine database must be updated often, and such updates must be incorporated by the web crawling system. To get over the processing bottleneck, multiple instances of this component operate simultaneously connected with very high bandwidth [5]. The HTTP protocol is used by web crawlers to download and process web pages from the internet. The downloader and the processor are the two components that make up the web crawling system. The processor receives the web pages that were downloaded by the downloader and processes them further. The HTTP protocol is used throughout this operation to download online pages. To download web pages, the downloader sends HTTP-GET queries. The pages are subsequently sent to the processor for additional processing and database integration. In order to download online pages from web servers, downloader adheres to a number of best practices. TCP Connection reusability,

*Corresponding Author.

Compression, Conditional GETs, varying refresh frequency, and DNS caching are a few of the optimization strategies used on downloader to get around network constraints [6].

A. Reusability of TCP connections

To conserve time and network traffic, downloader maintain one TCP connection open for each IP address and utilize it repeatedly until the server's entire contents have been downloaded [7]. Web servers compress web pages before transmitting them to the requesting client in order to save network traffic [8]. Downloader use conditional GETs, which are defined in the HTTP protocol. Conditional GETs download the page if it changes after a certain date [9]. Different websites have different refresh rates; some are updated every few minutes, while others don't change for a year. More crucial than others are certain pages. Because of this, search engines sift through pages that are significant and likely to see frequent updates.

B. DNS Caching

DNS resolves and URL downloads are two factors that contribute to network congestion. Additional DNS server bottlenecks could occur, which would impact performance. DNS cache is used by the downloader or site crawling component to boost performance. When the size of the web expands, centralized web crawling techniques become inefficient. Only a very small portion of the web is crawled by search engines, and they frequently ignore significant pages. Because of this, simultaneous web crawlers are employed to reduce network and other bottlenecks.

The task of the processor is to gather, process, and store the downloaded web documents in the database of the search engine. It is responsible for collecting information from online documents and putting it in the database. The method involves taking keywords from a web page and ranking them. The database stores the frequency and position of keywords. In order to extract information, the processing component processes strings.

The remainder of this article is arranged accordingly. Section II reviews the related works. Section III gives a case study of crawler load, Section IV details about FIS and Fuzzy Logic. Section V gives the proposed solution, including the components. Section VI is having result discussion. Lastly, Section VII concludes the work.

II. Related Work

An illustration of a distributed publication system is the Web. Based on how they handle requests, the two models are centralized and distributed. In a distributed architecture, a central location generates the query, which is subsequently sent to a few distant locations for processing. The distant stations then relay their findings to the central location. The combined results are displayed to the searcher at a central location. Under contrast, one system in the centralized approach maintains the entire index there. The single central index is then used to test the searcher's query. When using a distributed design, the difficulty of searching each database is decreased because the index collection is spread across a number of databases. NetFind, WAIS, and Harvest are examples of the distributed model. The authors of [10] introduced NetFind, a system that addressed the problem of finding information about people on the Web by employing a distributed search technique. WAIS employed a distributed set of indexes and a vector-space retrieval paradigm. WAIS struggled with its dearth of content.

According to the developers of [11], a Digest should only be used to convey queries to servers that have answers to such queries. Each server in the system creates a digest, which is then sent to a central site, containing a portion of the content that is available there. Archie, Veronica, and all search engines on the Internet are examples of the centralized paradigm. A number of existing extensively used documents (FTP files) were indexed by Archie [11]. The Gopher system could be searched thanks to Veronica [12]. Content from participating Gopher sites was downloaded and indexed at a single location. Publishing and accessing materials became simple thanks to the web. Additionally, it made following links easier [13]. The WWW Virtual Library is a method for finding resources on the Internet. A hierarchy was used to arrange the Virtual Library. Its quality was not exceptional because it was made by humans [14]. Three searchable indexes—the RBSE index, the WWW Worm, and Jumpstation—became well-known later in 1993. Each offered a searchable interface to a database of Web sites that was created automatically. Search engines like Lycos, InfoSeek, Yahoo, AltaVista, and Excite are a few examples. To fulfill a single searcher's request, the Meta search does numerous remote search engine queries. SavvySearch and MetaCrawler are two examples. These Meta engines use servers with content duplication. Because each search engine has a few unique documents in its index, meta-search is preferred. Additionally, as each search engine refreshes pages at a different rate, certain search engines will have current web content. Better results may be obtained via Meta search engines. Meta-search engines' disadvantage is a performance slowdown. AltaVista, Fast, and Google are a few examples of search engines.

A distributed system can scale by adding components, duplicate or distribute services so they are always available, and choose its components carefully to lower overall costs [15]. Three areas of distributed systems research are load balancing, resource allocation, and the overall design of distributed systems [16]. The goal of distributed system research is to create platforms that are consistent, dependable, and available. Locus, V Distributed System, and Eden are early distributed system models. Amoeba and Emerald are modern examples of distributed systems. The Coign system's creators claim that their system is capable of performing the kinds of optimizations that programmers often perform. Parallel searches are carried out simultaneously on all servers, integrated, ranked, and provided to the searcher. Ninja offers a platform for creating easily configurable, scalable Web services. The objective of Ninja is to reduce the administrative burden involved in managing a sizable cluster of systems [17]. Both distributed systems at the node level and parallel systems at the processor level require load balancing. The authors of [18] proposed a number of methods for assigning labour as well as detailed models for doing so. [19], which investigated load balancing in Amoeba, found that centralized decision-making outperforms distributed techniques. The Domain Name

System (DNS) has been used to disperse queries among several servers by having name servers answer with a list of addresses for a certain name. The authors of [20] suggested a mechanism for monitoring the load of the constituent systems.

In [21], the authors suggested DNS-based approaches that return server addresses in accordance with a model of the communication cost between the client and server. These methods address the issue of availability by limiting the addresses that name servers can distribute to servers that are known to be responsive. Very few significant pages are concealed within a big number of unimportant pages. How to get a good sample is the fundamental issue in web characterization? Pages containing scant or insignificant information should be excluded. It's important to gauge a website's importance. Web pages can be sampled using either vertical sampling or horizontal sampling. Web pages are gathered via vertical sampling, which is based on domain names. At many levels of the hierarchy, vertical sampling is possible. Vertical sampling at the highest level chooses nations with top-level domains like .in, .it, .au, etc. When vertical sampling is carried out at the second level, pages created by participants in the same institution or organization are collected (e.g. integraluniversity.ac.in).

A horizontal sample is the gathering of web pages based on selection criteria other than domain names. There are two ways to gather data: first, by employing a log of transactions in a major organization's proxy; and second, by using a web crawler. It is simple to locate popular pages when using a proxy, but the revisit duration cannot be adjusted because it depends on people. When using a web crawler, the popularity of the page must be approximated, but the revisit period may be adjusted.

III. Case Study of Crawler Load

The quality of the data gathered during a crawl can always be enhanced. The sort of web graph search is determined by the ordering of the URL queue. By considering the pages' in-link factors, the queue can be sorted. The breadth first search can enhance the caliber of pages that are downloaded [22]. On the Internet, there are a lot of spam sites and indefinitely branching crawler traps whose pages are dynamically produced and made to have a very high in-link factor [23]. The various network metrics, including geographic distance and latency, are covered in this section.

A. *Definition 1: Geographic Distance*

On the Internet, there are resources that give a mapping between IP addresses and geographical data. Longitude and latitude are extracted from registrar address data by the existing Internet service [24]. Two hosts are controlled by the same ISP if their latitude and longitude are identical [25]. A pair of Internet hosts' latitude and longitude can be found, and their geographic separation can be determined by utilizing the spherical coordinates of the earth.

B. *Definition 2: Latency*

There are numerous methods for calculating the Round Trip Time between two Internet hosts. The UNIX Ping utility is used in the first technique, and the Trace route utility is used in the second way [26]. The ICMP ECHO queries used by the Ping program are occasionally restricted or altered by ISPs. Some routers may prevent the TTL-restricted UDP packets that Trace route transmits [27].

C. *Definition 3: Correlation between Metrics*

Geographic distance and latency are strongly correlated [28]. The association between distance and RTT is stronger since the observations have lower linear distance values. A minimum end-to-end RTT is implied by linear distance along a path [29]. RTT and linear distance have a stronger correlation than RTT and end-to-end distance.

The client throughput in a conventional and active network was shown in Fig. 1. The X -axis indicates throughput, bits received by the clients at each simulated time, and Y- axis displays client's requests arrival rate [30]. Crawler throughput is proportional to client throughput for active indexing with 0% overhead. This establishes the comparability of the remaining samples. The throughput rapidly declines as the systems get saturated when both simulations reach a similar throughput of roughly 222 bits per tick. 140 bits per tick remain the consistent throughput at that point [30].

The throughput of a typical network crawler was shown in Fig. 2. The X- axis shows the overall arrival request rate, and the Y-axis shows the bits amount per simulated time unit that Web crawlers are receiving. The requests are started by both crawlers and actual customers. The typical client request delay for active indexing is shown in Fig. 3. The Y- axis indicates the delay in normal client response, and the X- axis reflects the rate at which requests are created by human clients. In traditional networks with 20 or 40 percent crawler traffic, the typical client delay is higher [30].

The relationship between request delay of crawler and the overall request arrival rate is seen in Fig. 4. As shown in the Fig. 3 and Fig. 4 the two curves are comparable, suggesting that the increased crawler load has no effect on the delay experienced by crawler sites [30].

The percentage of client requests that are always fulfilled is shown in Fig. 5. When the request arrival rate is low, all requests are nevertheless completed. The completion rates of customer requests have significantly decreased, as evidenced by the 20 percent and 40 percent crawler cases [30].

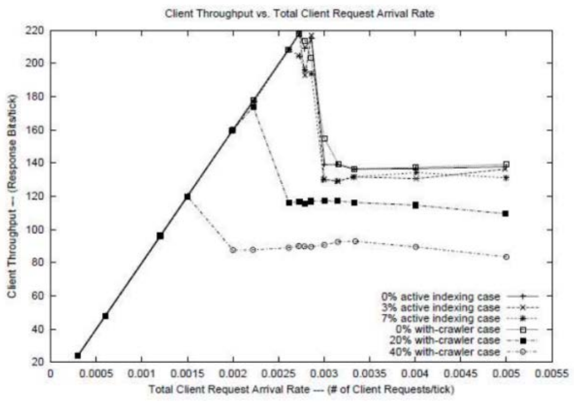


Fig. 1. Throughput of client in all cases [30].

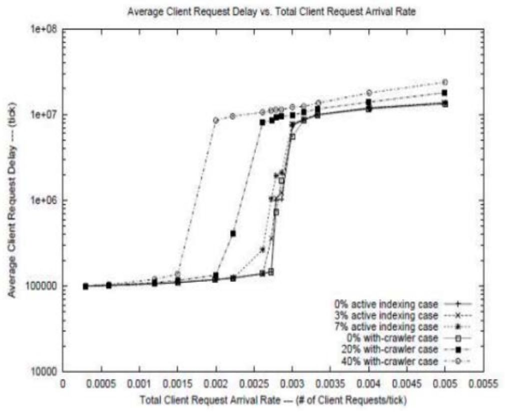


Fig. 2. Throughput of crawler.

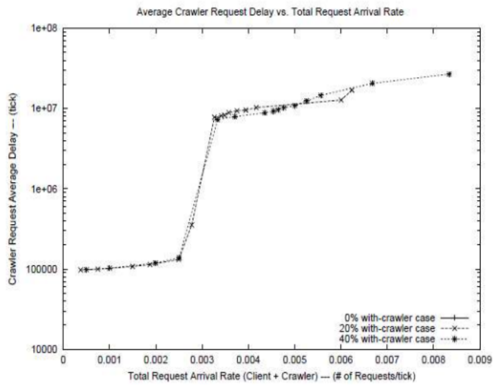


Fig. 3. Average client request delay in all cases [30].

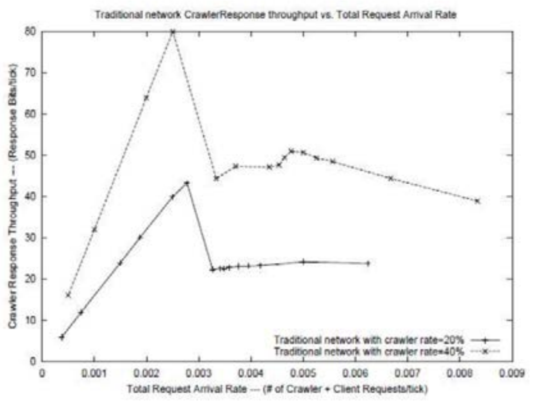


Fig. 4. Total request arrival time vs. average crawler request delay [30].

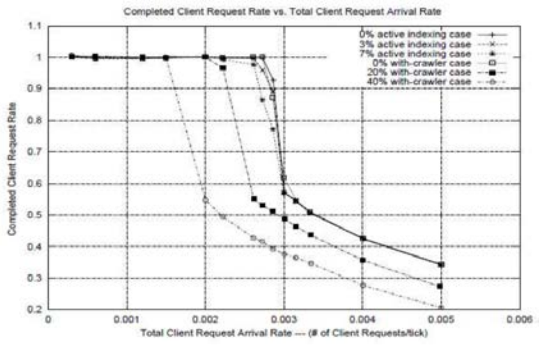


Fig. 5. Completed client request rates in all cases [30].

IV. FIS AND FUZZY LOGIC

A FIS makes inferences from a knowledge base using a fuzzy inference engine (FIE). The FIE, represents the methods required for formalizing results and reasoning with the data in the knowledge database, is comparable to the brain of expert systems[31]. Boolean algebra's adaptation to deal with insufficient truth is fuzzy logic. Fuzzy logic illustrates the degree of truth of propositional logic. Everything in Boolean algebra may be stated in terms of binary numbers, or zero and one. Boolean algebraic values are replaced with the degree of truth in fuzzy logic.

The erroneous patterns of thinking are recorded using the level of truth. In an environment of ambiguity and uncertainty, this method of thinking is crucial to how humans make decisions. The membership function in fuzzy sets is comparable to the indicator function in classical set theory. Curves represent membership functions. According to member functions, every input space point is translated to a value lies between 0 and 1. A membership function has a triangular, bell-shaped, and trapezoidal shape. The universe of discourse is the name of the input space.

A fuzzy inference system is simpler to construct and has a very simple conceptual foundation. Three steps make up a fuzzy inference system: the input stage, the output stage, and the processing stage [32]. Input is translated into membership functions in the input step. At the processing stage, the appropriate rule is triggered, and each rule's result is generated before being combined. The output stage next transforms the outcome into output. The inference engine is the level of processing. The foundation of an inference engine is a set of IF-THEN logic rules. The THEN sub-statement is "consequent" if the IF sub-statement is "antecedent." A knowledge database contains n number of rules that are specific to fuzzy inference subsystems. The steps of the FIS are as follows:

- Fuzzification.
- Application of Fuzzy operators.
- Implication.
- Output Aggregation.
- Defuzzification.

Fuzzification of inputs is the process of assessing an input's degree of membership in its fuzzy sets using membership functions [33]. Fuzzy sets are used as the input and crisp values are produced during the defuzzification process. In fuzzy systems, there are two popular inference techniques. Ebrahim Mamdani proposed the first approach, the Mamdani fuzzy inference method, in 1975. Takagi-Sugeno-Kang proposed the second method, the Takagi-Sugeno-Kang fuzzy inference method, in 1985[34]. Numerous aspects of these approaches are comparable, such as the fuzzying of the inputs and fuzzy operators. While Mamdani's inference uses fuzzy sets as its output membership functions. Sugeno's approach is computationally effective and functions well with adaptive and optimization strategies and uses output membership functions that are linear and constant. It also functions well when analysed mathematically.

The crawling procedure keeps the quality up. The web crawling is carried out using one of the two methods listed below: either the web crawlers are permitted to communicate with one another or they are not. Both methods add to the load on the network [35]. Here, a fuzzy logic-based method that predicts the load at a specific node and the path of network traffic is presented and implemented in MATLAB using the fuzzy logic toolbox.

V. Proposed Solution

These are the major steps of the proposed solution.

a) Using a Fuzzy Inference System to Fix the Network Traffic Issue with Parallel Crawler Migration.

b) Using the membership function editor.

c) Using the Rule Editor to specify rules for a fuzzy inference system.

d) Rule Evaluation.

e) Adding up the results of the rule.

f) Removing fuzziness from the output value.

1) FIS to Solve Network Traffic problem in migrating parallel Crawlers.

The fuzzy set is the foundation of fuzzy logic theory. Every point in the input space is assigned a membership value between 0 and 1 that is defined by the curve known as the membership function. A fuzzy set is one that lacks a definite crisp border. Fuzzy Logic Toolbox includes the following tools for creating and customizing fuzzy inference systems:

a) Fuzzy Inference System (FIS) Editor.

b) Membership Function Editor.

c) Rule Editor.

d) Rule Viewer.

e) Surface Viewer.

The Mamdani method is employed since it is well-liked for gathering knowledge. It enables us to speak more humanely when describing the expertise [36].

2) Defining FIS variables and fuzzification of the input variables using membership function editor.

a) gaussmf: The built-in membership function for the Gaussian curve in the fuzzy toolbox is known as gaussmf. y = gaussmf(x,[sig c])[37] gives the syntax (Fig. 6). The fuzzy toolbox's symmetric Gaussian function is dependent on the two parameters and as stated by.

$$f(x;\sigma,c) = e^{\frac{-(x-c)^2}{2\sigma^2}}$$

For example if y=gaussmf(x,[2 5]).

plot(x,y).

xlabel('gaussmf, P=[2 5]').

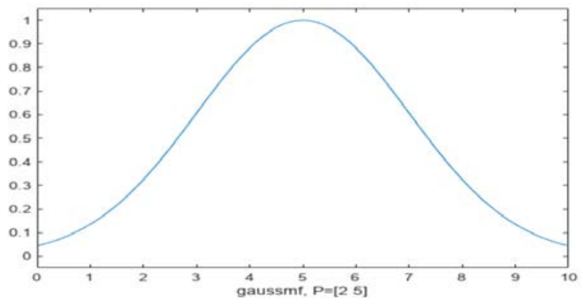


Fig. 6. Gaussmf Curve.

b) Trimf: In the fuzzy toolbox, trimf is the built-in membership function with a triangular shape (Fig. 7). The triangular curve is a function of a vector x and depends on three parameters when the syntax is y = trimf(x,params); if y = trimf(x,[a b c]):

$$f(x;a,b,c) = \begin{cases} 0, x \le a \\ \frac{x-a}{b-a}, a \le x \le b \\ \frac{c-x}{c-b}, b \le x \le c \\ 0, c \le x \end{cases}$$

Or,

$$f(x;a,b,c) = \max\left(min\left(\frac{x-a}{b-a},\frac{c-x}{c-b}\right),0\right)$$

The triangle's base is indicated by first parameter a and third parameter c, while the triangle's peak is shown by second parameter b [38]. For example:

x=0:0.1:10;

y=trimf(x,[3 6 8]);

plot(x,y)

xlabel('trimf, P=[3 6 8]')

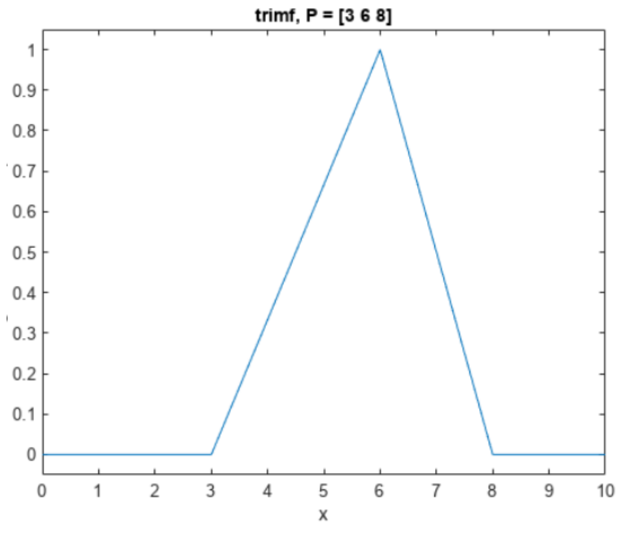
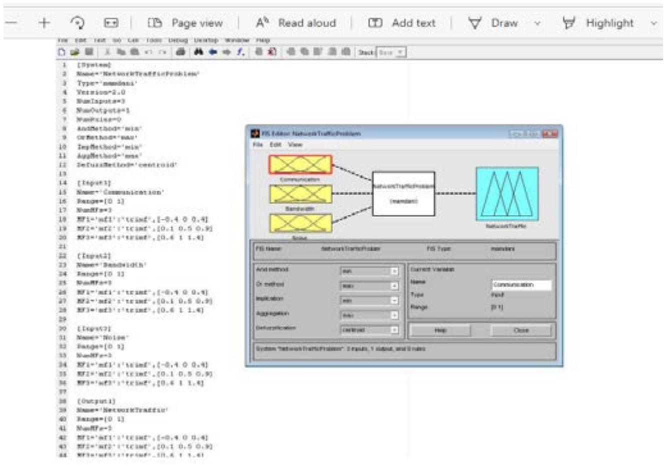


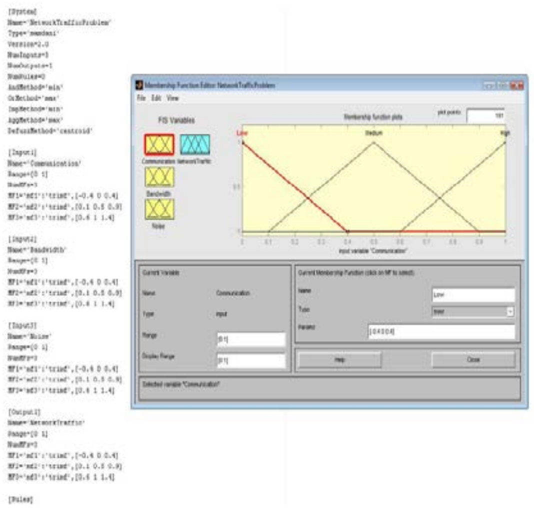
Fig. 7. Trimf function.



```
[System]
Name='NetworkTrafficProblem'
Type='mamdani'
Version=2.0
NumInputs=3
NumOutputs=1
NumRules=0
AndMethod='min'
OrMethod='max'
ImpMethod='min'
AggMethod='max'
DefuzzMethod='centroid'

[Input1]
Name='Communication'
Range=[0 1]
NumMFs=3
MF1='mf1':'trimf',[-0.4 0 0.4]
MF2='mf2':'trimf',[0.1 0.5 0.9]
MF3='mf3':'trimf',[0.6 1 1.4]

[Input2]
Name='Bandwidth'
Range=[0 1]
NumMFs=3
MF1='mf1':'trimf',[-0.4 0 0.4]
MF2='mf2':'trimf',[0.1 0.5 0.9]
MF3='mf3':'trimf',[0.6 1 1.4]

[Input3]
Name='Noise'
Range=[0 1]
NumMFs=3
MF1='mf1':'trimf',[-0.4 0 0.4]
MF2='mf2':'trimf',[0.1 0.5 0.9]
MF3='mf3':'trimf',[0.6 1 1.4]

[Output1]
Name='NetworkTraffic'
Range=[0 1]
NumMFs=3
MF1='mf1':'trimf',[-0.4 0 0.4]
MF2='mf2':'trimf',[0.1 0.5 0.9]
MF3='mf3':'trimf',[0.6 1 1.4]
```

FIS Name: NetworkTrafficProblem — FIS Type: mamdani
And method: min
Or method: max
Implication: min
Aggregation: max
Defuzzification: centroid
Current Variable — Name: Communication; Type: input; Range: [0 1]
System "NetworkTrafficProblem": 3 inputs, 1 output, and 0 rules

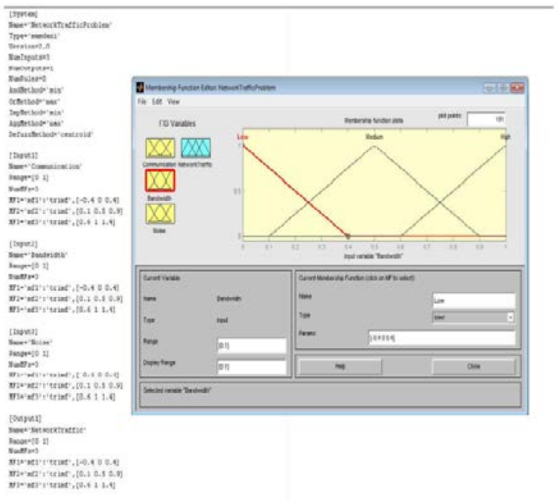
Fig. 8. FIS editor for network traffic problem.



```
[System]
Name='NetworkTrafficProblem'
Type='mamdani'
Version=2.0
NumInputs=3
NumOutputs=1
NumRules=0
AndMethod='min'
OrMethod='max'
ImpMethod='min'
AggMethod='max'
DefuzzMethod='centroid'

[Input1]
Name='Communication'
Range=[0 1]
NumMFs=3
MF1='mf1':'trimf',[-0.4 0 0.4]
MF2='mf2':'trimf',[0.1 0.5 0.9]
MF3='mf3':'trimf',[0.6 1 1.4]

[Input2]
Name='Bandwidth'
Range=[0 1]
NumMFs=3
MF1='mf1':'trimf',[-0.4 0 0.4]
MF2='mf2':'trimf',[0.1 0.5 0.9]
MF3='mf3':'trimf',[0.6 1 1.4]

[Input3]
Name='Noise'
Range=[0 1]
NumMFs=3
MF1='mf1':'trimf',[-0.4 0 0.4]
MF2='mf2':'trimf',[0.1 0.5 0.9]
MF3='mf3':'trimf',[0.6 1 1.4]

[Output1]
Name='NetworkTraffic'
Range=[0 1]
NumMFs=3
MF1='mf1':'trimf',[-0.4 0 0.4]
MF2='mf2':'trimf',[0.1 0.5 0.9]
MF3='mf3':'trimf',[0.6 1 1.4]

[Rules]
```

Current Variable — Name: Communication; Type: input; Range: [0 1]; Display Range: [0 1]
Current Membership Function (click on MF to select) — Name: Low; Type: trimf; Params: [-0.4 0 0.4]
Selected variable "Communication"

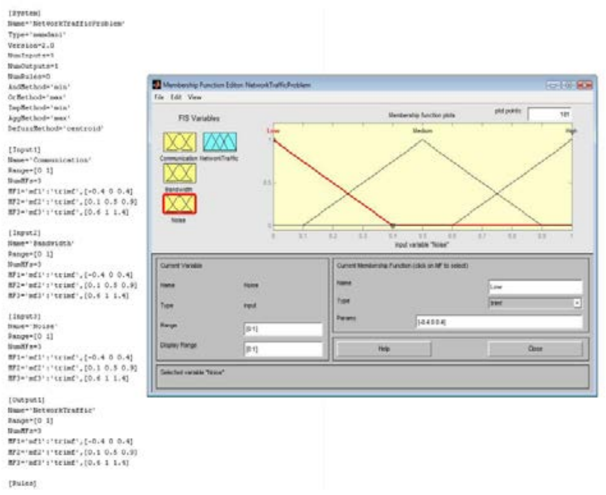
Fig. 9. FIS variable communication.



```
[System]
Name='NetworkTrafficProblem'
Type='mamdani'
Version=2.0
NumInputs=3
NumOutputs=1
NumRules=0
AndMethod='min'
OrMethod='max'
ImpMethod='min'
AggMethod='max'
DefuzzMethod='centroid'

[Input1]
Name='Communication'
Range=[0 1]
NumMFs=3
MF1='mf1':'trimf',[-0.4 0 0.4]
MF2='mf2':'trimf',[0.1 0.5 0.9]
MF3='mf3':'trimf',[0.6 1 1.4]

[Input2]
Name='Bandwidth'
Range=[0 1]
NumMFs=3
MF1='mf1':'trimf',[-0.4 0 0.4]
MF2='mf2':'trimf',[0.1 0.5 0.9]
MF3='mf3':'trimf',[0.6 1 1.4]

[Input3]
Name='Noise'
Range=[0 1]
NumMFs=3
MF1='mf1':'trimf',[-0.4 0 0.4]
MF2='mf2':'trimf',[0.1 0.5 0.9]
MF3='mf3':'trimf',[0.6 1 1.4]

[Output1]
Name='NetworkTraffic'
Range=[0 1]
NumMFs=3
MF1='mf1':'trimf',[-0.4 0 0.4]
MF2='mf2':'trimf',[0.1 0.5 0.9]
MF3='mf3':'trimf',[0.6 1 1.4]
```

Current Variable — Name: Bandwidth; Type: input; Range: [0 1]; Display Range: [0 1]
Current Membership Function (click on MF to select) — Name: Low; Type: trimf; Params: [-0.4 0 0.4]
Selected variable "Bandwidth"

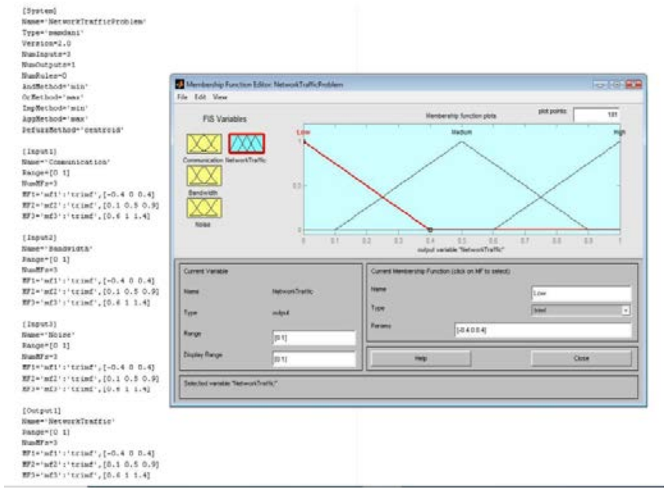
Fig. 10. FIS variable bandwidth.



```
[System]
Name='NetworkTrafficProblem'
Type='mamdani'
Version=2.0
NumInputs=3
NumOutputs=1
NumRules=0
AndMethod='min'
OrMethod='max'
ImpMethod='min'
AggMethod='max'
DefuzzMethod='centroid'

[Input1]
Name='Communication'
Range=[0 1]
NumMFs=3
MF1='mf1':'trimf',[-0.4 0 0.4]
MF2='mf2':'trimf',[0.1 0.5 0.9]
MF3='mf3':'trimf',[0.6 1 1.4]

[Input2]
Name='Bandwidth'
Range=[0 1]
NumMFs=3
MF1='mf1':'trimf',[-0.4 0 0.4]
MF2='mf2':'trimf',[0.1 0.5 0.9]
MF3='mf3':'trimf',[0.6 1 1.4]

[Input3]
Name='Noise'
Range=[0 1]
NumMFs=3
MF1='mf1':'trimf',[-0.4 0 0.4]
MF2='mf2':'trimf',[0.1 0.5 0.9]
MF3='mf3':'trimf',[0.6 1 1.4]

[Output1]
Name='NetworkTraffic'
Range=[0 1]
NumMFs=3
MF1='mf1':'trimf',[-0.4 0 0.4]
MF2='mf2':'trimf',[0.1 0.5 0.9]
MF3='mf3':'trimf',[0.6 1 1.4]

[Rules]
```

Current Variable — Name: Noise; Type: input; Range: [0 1]; Display Range: [0 1]
Current Membership Function (click on MF to select) — Name: Low; Type: trimf; Params: [-0.4 0 0.4]
Selected variable "Noise"

Fig. 11. FIS variable noise.


Membership Function Editor: NetworkTrafficProblem
File Edit View
FIS Variables
Communication
NetworkTraffic
Bandwidth
Noise
Membership function plots
plot points: 181
Low
Medium
High
output variable "NetworkTraffic"
Help
Close

```
[System]
Name='NetworkTrafficProblem'
Type='mamdani'
Version=2.0
NumInputs=3
NumOutputs=1
NumRules=0
AndMethod='min'
OrMethod='max'
ImpMethod='min'
AggMethod='max'
DefuzzMethod='centroid'

[Input1]
Name='Communication'
Range=[0 1]
NumMFs=3
MF1='mf1':'trimf',[-0.4 0 0.4]
MF2='mf2':'trimf',[0.1 0.5 0.9]
MF3='mf3':'trimf',[0.6 1 1.4]

[Input2]
Name='Bandwidth'
Range=[0 1]
NumMFs=3
MF1='mf1':'trimf',[-0.4 0 0.4]
MF2='mf2':'trimf',[0.1 0.5 0.9]
MF3='mf3':'trimf',[0.6 1 1.4]

[Input3]
Name='Noise'
Range=[0 1]
NumMFs=3
MF1='mf1':'trimf',[-0.4 0 0.4]
MF2='mf2':'trimf',[0.1 0.5 0.9]
MF3='mf3':'trimf',[0.6 1 1.4]

[Output1]
Name='NetworkTraffic'
Range=[0 1]
NumMFs=3
MF1='mf1':'trimf',[-0.4 0 0.4]
MF2='mf2':'trimf',[0.1 0.5 0.9]
MF3='mf3':'trimf',[0.6 1 1.4]
```

Current Variable — Name: NetworkTraffic; Type: output; Range: [0 1]; Display Range: [0 1]
Current Membership Function (click on MF to select) — Name: Low; Type: trimf; Params: [-0.4 0 0.4]
Selected variable "NetworkTraffic"

Fig. 12. FIS output variable network traffic.

The FIS editor for the Network Traffic Problem is shown in Fig. 8. The FIS variable Communication is shown in Fig. 9. The FIS variable Bandwidth is shown as a Fig. 10. The FIS variable Noise is shown in Fig. 11. The FIS output variable Network Traffic is shown in Fig. 12.

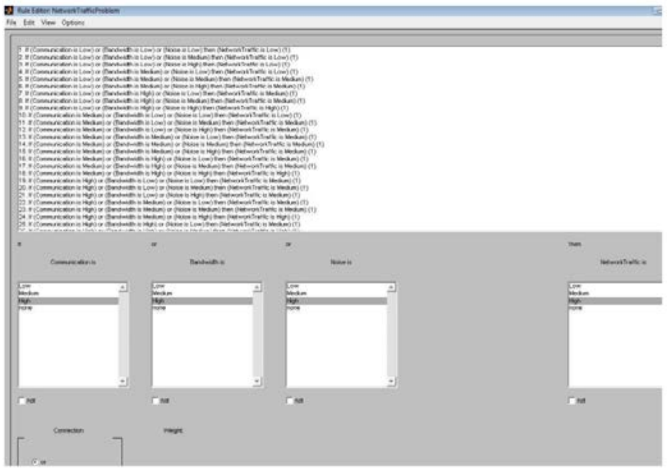
Lemma 1. When n is the total number of network nodes, the technique has a worst-case time complexity of O(n).

Proof. Steps 1 and 3 of the method take a fixed amount of time. In the worst case situation, a node may have m number of gateways within its communication range, hence Step 2 can be finished in O(m) time. The node selects a CH in Step 4 in O(n) time and O(m) processing time after resetting its backup set (Step 4.2) in Step 4. (Step 4). The worst-case execution time for Step 4 is O(m) + O(n), or O, if n > m. (n). Step 4 shows that the worst-case processing time of the algorithm is O (n).

Lemma 2. The algorithm's worst-case message exchange complexity is O(1) per node or O(n) across the network's n nodes.

Proof. During the cluster creation phase, a node calculates the cost values of the CHs within its communication range and sends a join request to the selected CH. Nodes that cannot communicate with any CHs, however, broadcast a HELP request message (step 4.1). It will send a join request message to join the cluster using multi-hop communication if it finds a node that can help. Therefore, in the worst scenario, a node only needs to send two messages for the cluster to form. As a result, each node's message complexity, O, is constant (1). The network's overall message exchange complexity is O as a result (n).

3) Using the Rule Editor to specify rules for a fuzzy inference system to solve the network traffic issue when migrating parallel crawlers.


Rule Editor: NetworkTrafficProblem

1. If (Communication is Low) or (Bandwidth is Low) or (Noise is Low) then (NetworkTraffic is Low) (1)
2. If (Communication is Low) or (Bandwidth is Low) or (Noise is Medium) then (NetworkTraffic is Low) (1)
3. If (Communication is Low) or (Bandwidth is Low) or (Noise is High) then (NetworkTraffic is Low) (1)
4. If (Communication is Low) or (Bandwidth is Medium) or (Noise is Low) then (NetworkTraffic is Low) (1)
5. If (Communication is Low) or (Bandwidth is Medium) or (Noise is Medium) then (NetworkTraffic is Medium) (1)
6. If (Communication is Low) or (Bandwidth is Medium) or (Noise is High) then (NetworkTraffic is Medium) (1)
7. If (Communication is Low) or (Bandwidth is High) or (Noise is Low) then (NetworkTraffic is Medium) (1)
8. If (Communication is Low) or (Bandwidth is High) or (Noise is Medium) then (NetworkTraffic is Medium) (1)
9. If (Communication is Low) or (Bandwidth is High) or (Noise is High) then (NetworkTraffic is High) (1)
10. If (Communication is Medium) or (Bandwidth is Low) or (Noise is Low) then (NetworkTraffic is Low) (1)
11. If (Communication is Medium) or (Bandwidth is Low) or (Noise is Medium) then (NetworkTraffic is Medium) (1)
12. If (Communication is Medium) or (Bandwidth is Low) or (Noise is High) then (NetworkTraffic is Medium) (1)
13. If (Communication is Medium) or (Bandwidth is Medium) or (Noise is Low) then (NetworkTraffic is Medium) (1)
14. If (Communication is Medium) or (Bandwidth is Medium) or (Noise is Medium) then (NetworkTraffic is Medium) (1)
15. If (Communication is Medium) or (Bandwidth is Medium) or (Noise is High) then (NetworkTraffic is Medium) (1)
16. If (Communication is Medium) or (Bandwidth is High) or (Noise is Low) then (NetworkTraffic is Medium) (1)
17. If (Communication is Medium) or (Bandwidth is High) or (Noise is Medium) then (NetworkTraffic is Medium) (1)
18. If (Communication is Medium) or (Bandwidth is High) or (Noise is High) then (NetworkTraffic is High) (1)
19. If (Communication is High) or (Bandwidth is Low) or (Noise is Low) then (NetworkTraffic is Medium) (1)
20. If (Communication is High) or (Bandwidth is Low) or (Noise is Medium) then (NetworkTraffic is Medium) (1)
21. If (Communication is High) or (Bandwidth is Low) or (Noise is High) then (NetworkTraffic is Medium) (1)
22. If (Communication is High) or (Bandwidth is Medium) or (Noise is Low) then (NetworkTraffic is Medium) (1)
23. If (Communication is High) or (Bandwidth is Medium) or (Noise is Medium) then (NetworkTraffic is Medium) (1)
24. If (Communication is High) or (Bandwidth is Medium) or (Noise is High) then (NetworkTraffic is High) (1)
25. If (Communication is High) or (Bandwidth is High) or (Noise is Low) then (NetworkTraffic is Medium) (1)

Fig. 13. Rules editor for network traffic problem.

TABLE I. RULES IN FIS (WHERE L=LOW, M=MEDIUM, H=HIGH)

Communication	Bandwidth	Noise Network	Traffic
L	L	L	L
L	L	M	L
L	L	H	L
L	M	L	L
L	M	M	M
L	M	H	M
L	H	L	M
L	H	M	M
L	H	H	H
M	L	L	L
M	L	M	M
M	L	H	M
M	M	L	M
M	M	M	M
M	M	H	M
M	H	L	M
M	H	M	M
M	H	H	H
H	L	L	M
H	L	M	M
H	L	H	M
H	M	L	M
H	M	M	M
H	M	H	H
H	H	L	M
H	H	M	H
H	H	H	H

4) Rule evaluation, rule output aggregate, and output value defuzzification.

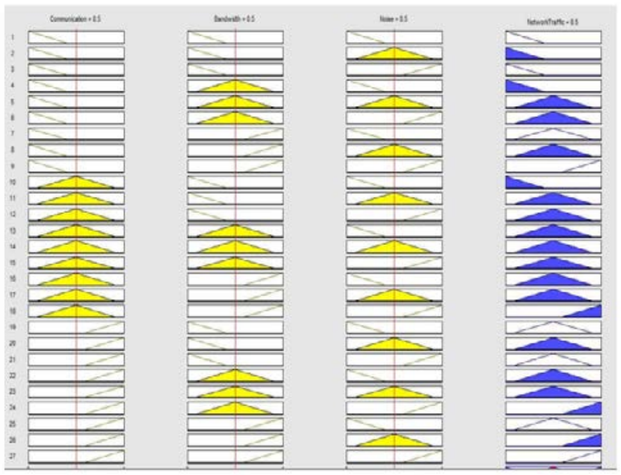
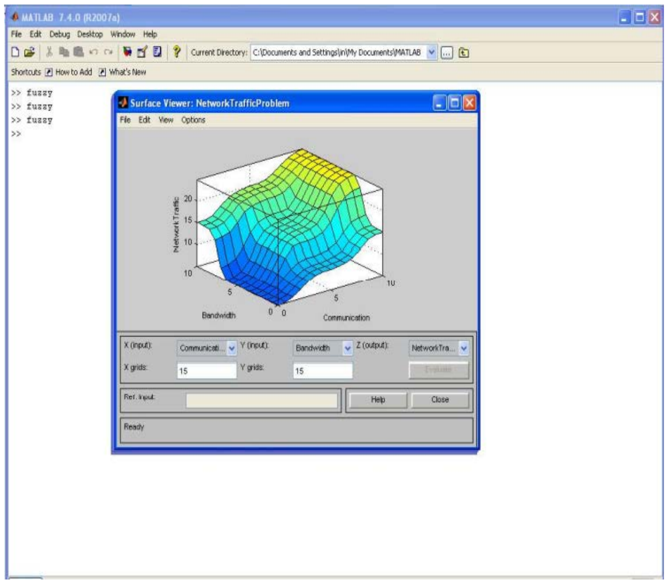


Fig. 14. Rule evaluation aggregation of the rule output.


MATLAB 7.4.0 (R2007a)

Current Directory: C:\Documents and Settings\n\My Documents\MATLAB

```
>> fuzzy
>> fuzzy
>> fuzzy
>>
```

Surface Viewer: NetworkTrafficProblem

X (input): Communicati... Y (input): Bandwidth Z (output): NetworkTra...
X grids: 15 Y grids: 15

Ready

Fig. 15. Surface viewer for network traffic problem.

The FIS rules are in Table I. The Rules Editor for Network Traffic Problem is shown in Fig. 13. The Rule Evaluation Aggregation of the rule output is shown in Fig. 14. The Surface Viewer for Network Traffic Problem is shown in Fig. 15.

VI. Result and Discussion

The algorithm is integrated with the aforementioned module. The MATLAB Compiler is used to generate the code. The Implementation is tested against current web crawlers and designed to function on active websites.

TABLE II. Load Caused using Conventional Crawler

	Page 1	Page 2	Page 3	Total Load (KB)
visit 1	185	185	185	555
visit 2	193	196	195	
visit 3	188	189	199	
visit 4	200	201	205	
visit 5	188	199	188	
load caused	954	970	972	2896
visit 6	188	189	188	
visit 7	198	198	189	
visit 8	178	176	189	
visit 9	189	187	189	
visit 10	199	189	198	
load caused	1906	1906	1925	5740

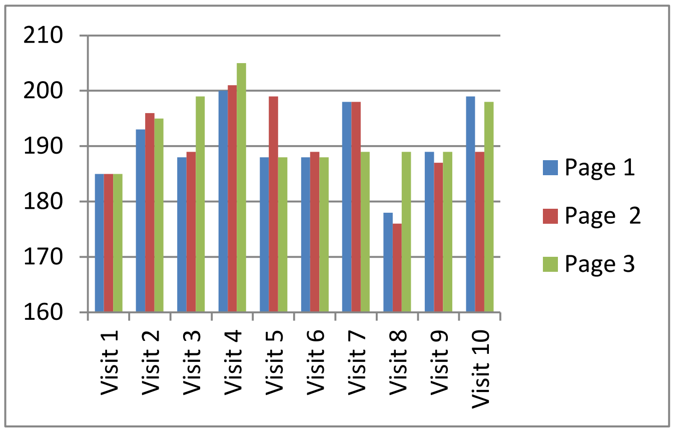


Fig. 16. Load caused using conventional crawler.

TABLE III. Load Caused using Single Threaded Crawler

	Page 1	Page 2	Page 3	Total Load (KB)
visit 1	78	87	98	263
visit 2	87	89	98	
visit 3	76	98	98	
visit 4	87	98	87	
visit 5	87	998	89	
load caused	415	470	470	1355
visit 6	87	89	87	
visit 7	78	98	98	
visit 8	98	76	98	
visit 9	87	97	98	
visit 10	78	98	87	
load caused	843	928	938	2709

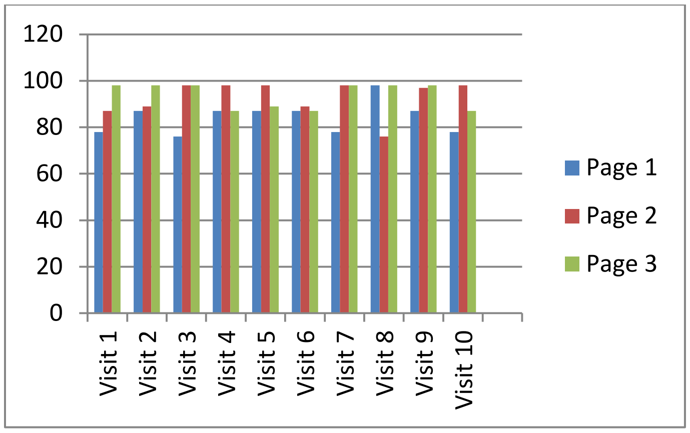


Fig. 17. Load caused using Single threaded crawler.

TABLE IV. Load Caused using Agent based Crawler

	Page 1	Page 2	Page 3	Total Load (KB)
visit 1	35	35	37	107
visit 2	36	37	37	
visit 3	43	36	45	
visit 4	34	45	57	
visit 5	34	43	43	
load caused	182	196	219	597
visit 6	43	53	43	
visit 7	43	34	34	
visit 8	45	54	43	
visit 9	34	43	45	
visit 10	34	34	45	
load caused	381	414	429	1224

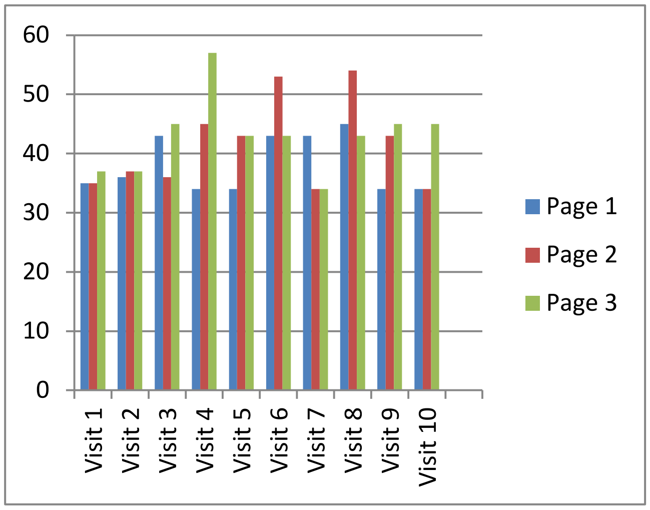


Fig. 18. Load caused using agent based crawler.

The load produced by a conventional crawler is shown in Table II. The load produced by a single threaded crawler is shown in Table III. The load produced by an agent-based crawler is seen in Table IV. The load created by migrating parallel web crawlers is shown in Table V. The graph in Fig.

16 to Fig. 19 depict the network load created by various methods. Three websites are used in the analysis and comparison of the methods. Since an HTML page typically weighed 205 KB, the network traffic produced by the conventional centralised crawling strategy was 555 KB. While in our method, the pages were compressed on the server, and the traffic load that was discovered was 70 KB. As seen in the above Fig. 20, the load incurred after five visits to the pages was 2896 KB, 1355 KB, 597 KB, and 379 KB, respectively, and after 10 visits, the load was 5740 KB, 2709 KB, 1224 KB, and 774 KB, respectively. Additionally, this resulted in less network traffic.

TABLE V. LOAD CAUSED USING MIGRATING PARALLEL WEB CRAWLER

	Page 1	**Page 2**	**Page 3**	**Total Load (KB)**
visit 1	23	23	24	70
visit 2	24	24	24	
visit 3	24	28	27	
visit 4	27	26	27	
visit 5	24	27	27	
load caused	122	128	129	379
visit 6	27	27	27	
visit 7	26	26	26	
visit 8	26	26	27	
visit 9	27	25	27	
visit 10	25	26	29	
load caused	253	258	263	774

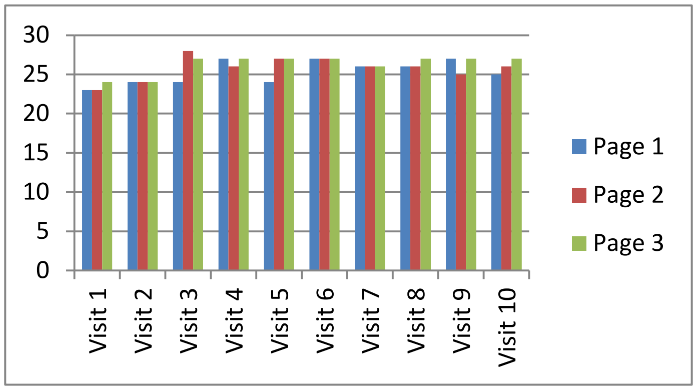


Fig. 19. Load caused using migrating parallel web crawler.

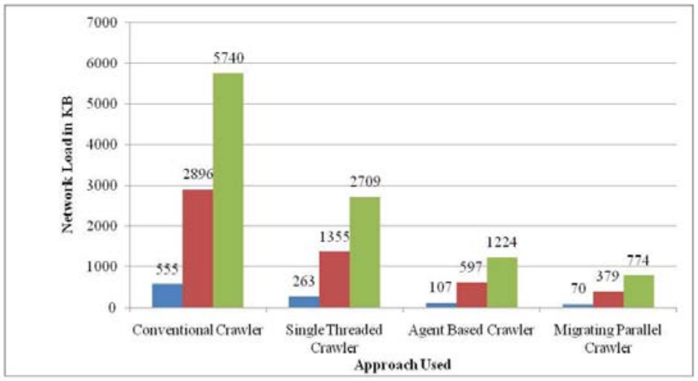


Fig. 20. Graph showing network load caused in various approaches.

VII. CONCLUSION

This paper discusses the crawling process using one of the two following approaches: either allowing crawlers to communicate among themselves freely or forbidding them from doing so altogether. Both approaches increase network traffic. Here, a fuzzy logic-based method that predicts the load at a specific node and the path of network traffic is presented and implemented in MATLAB using the fuzzy logic toolbox. The experimental findings demonstrate that the network demand is decreased when a parallel web crawler is migrated.

ACKNOWLEDGMENT

We thank the Deanship of Scientific Research, Prince Sattam Bin Abdulaziz University, Alkharj, Saudi Arabia for help and support.

REFERENCES

[1] M. Haleem, M. F. Farooqui, and M. Faisal, “Tackling Requirements Uncertainty in Software Projects: A Cognitive Approach,” Int. J. Cogn. Comput. Eng., vol. 2, pp. 180–190, 2021.

[2] S. Khalil and M. Fakir, “RCrawler: An R package for parallel web crawling and scraping,” SoftwareX, vol. 6, pp. 98–106, 2017.

[3] Jha S, Sultan A, Alharbi M, Alouffi B, Sebastian S. Secured and provisioned access authentication using subscribed user identity in federated clouds. International Journal of Advanced Computer Science and Applications. 2021;12(11).

[4] S. Singh and N. Tyjagi, “A Novel Architecture of Mercator: A Scalable, Extensible Web Crawler with Focused Web Crawler,” Int. J. Comput. Sci. Mob. Comput., vol. 2, no. 6, pp. 244–250, 2013.

[5] Gopi R, Mathapati M, Prasad B, Ahmad S, Al-Wesabi FN, Abdullah Alohali M, Mustafa Hilal A. Intelligent DoS attack detection with congestion control technique for vanets. Computers, Materials & Continua. 2022;72(1):141-56.

[6] Ahmad S, Jha S, Alam A, Alharbi M, Nazeer J. Analysis of Intrusion Detection Approaches for Network Traffic Anomalies with Comparative Analysis on Botnets (2008–2020). Security and Communication Networks. 2022 May 12;2022.

[7] J. Kim, H. Kim, and J. Rexford, “Analyzing traffic by domain name in the data plane,” in Proceedings of the ACM SIGCOMM Symposium on SDN Research (SOSR), 2021, pp. 1–12.

[8] R. Palacios, A. F. Fernández-Portillo, E. F. Sánchez-Úbeda, and P. García-De-Zúñiga, “HTB: A Very Effective Method to Protect Web Servers Against BREACH Attack to HTTPS,” IEEE Access, vol. 10, pp. 40381–40390, 2022.

[9] R. P. Kasturi et al., “Mistrust Plugins You Must: A Large-Scale Study Of Malicious Plugins In WordPress Marketplaces,” in 31st USENIX Security Symposium (USENIX Security 22), 2022, pp. 10–12.

[10] M. F. Schwartz and C. Pu, “Applying an information gathering architecture to Netfind: a white pages tool for a changing and growing Internet,” IEEE/ACM Trans. Netw., vol. 2, no. 5, pp. 426–439, 1994.

[11] M. Li et al., “Bringing Decentralized Search to Decentralized Services,” in 15th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 21), 2021, pp. 331–347.

[12] B. E. É. R. Oliveira, “Web Search Engines-A study on the evolution of user interfaces,” 2021.

[13] E. F. Pettersen et al., “UCSF ChimeraX: Structure visualization for researchers, educators, and developers,” Protein Sci., vol. 30, no. 1, pp. 70–82, 2021.

[14] L. Heng, G. Yin, and X. Zhao, “Energy aware cloud - edge service placement approaches in the Internet of Things communications,” Int. J. Commun. Syst., vol. 35, no. 1, p. e4899, 2022.

[15] Uddin M. Y, Ahmad S. A review on edge to cloud: paradigm shift from large data centers to small centers of data everywhere. In2020 International Conference on Inventive Computation Technologies (ICICT) 2020 Feb 26 (pp. 318-322). IEEE.

[16] I. K. Aksakalli, T. Çelik, A. B. Can, and B. Tekinerdoğan, "Deployment and communication patterns in microservice architectures: A systematic literature review," J. Syst. Softw., vol. 180, p. 111014, 2021.

[17] C. S. Long et al., "California Needs Clean Firm Power, and So Does the Rest of the World: Three Detailed Models of the Future of California's Power System all show that California needs Carbon-Free Electricity Sources that don't Depend on the Weather," Clean Air Task Force, 2021.

[18] A. M. Fathollahi-Fard, L. Woodward, and O. Akhrif, "Sustainable distributed permutation flow-shop scheduling model based on a triple bottom line concept," J. Ind. Inf. Integr., vol. 24, p. 100233, 2021.

[19] K. Oshima, D. Yamamoto, A. Yumoto, S.-J. Kim, Y. Ito, and M. Hasegawa, "Online machine learning algorithms to optimize performances of complex wireless communication systems," Math. Biosci. Eng., vol. 19, no. 2, pp. 2056–2094, 2022.

[20] G. L. Golewski, "Evaluation of fracture processes under shear with the use of DIC technique in fly ash concrete and accurate measurement of crack path lengths with the use of a new crack tip tracking method," Measurement, vol. 181, p. 109632, 2021.

[21] L. Csikor, H. Singh, M. S. Kang, and D. M. Divakaran, "Privacy of DNS-over-HTTPS: Requiem for a Dream?," in 2021 IEEE European Symposium on Security and Privacy (EuroS&P), 2021, pp. 252–271.

[22] A. Garg, K. Gupta, and A. Singh, "Survey of Web Crawler Algorithms.," Int. J. Adv. Res. Comput. Sci., vol. 8, no. 5, 2017.

[23] A. Goswami and A. Kumar, "Online Social Communities," Digit. Bus., pp. 289–341, 2019.

[24] M. Mansoori and I. Welch, "How do they find us? A study of geolocation tracking techniques of malicious web sites," Comput. Secur., vol. 97, p. 101948, 2020.

[25] X. Mi et al., "Resident evil: Understanding residential IP proxy as a dark service," in 2019 IEEE symposium on security and privacy (SP), 2019, pp. 1185–1201.

[26] I. Pelle, T. Lévai, F. Németh, and A. Gulyás, "One tool to rule them all: A modular troubleshooting framework for SDN (and other) networks," in Proceedings of the 1st ACM SIGCOMM Symposium on Software Defined Networking Research, 2015, pp. 1–7.

[27] I. D. Oladipo, M. AbdulRaheem, J. B. Awotunde, A. K. Bhoi, E. A. Adeniyi, and M. K. Abiodun, "Machine Learning and Deep Learning Algorithms for Smart Cities: A Start-of-the-Art Review," IoT IoE Driven Smart Cities, pp. 143–162, 2022.

[28] M. Haleem, M. F. Farooqui, and M. Faisal, "Cognitive impact validation of requirement uncertainty in software project development," International Journal of Cognitive Computing in Engineering, vol. 2. pp. 1–11, 2021, doi: 10.1016/j.ijcce.2020.12.002.

[29] J. Jiang et al., "Via: Improving internet telephony call quality using predictive relay selection," in Proceedings of the 2016 ACM SIGCOMM Conference, 2016, pp. 286–299.

[30] X. Yuan, M. H. MacGregor, and J. Harms, "An efficient scheme to remove crawler traffic from the internet," in Proceedings. Eleventh International Conference on Computer Communications and Networks, 2002, pp. 90–95.

[31] S. Sweta and K. Lal, "Personalized adaptive learner model in e-learning system using FCM and fuzzy inference system," Int. J. Fuzzy Syst., vol. 19, no. 4, pp. 1249–1260, 2017.

[32] H. Uğuz, "Adaptive neuro-fuzzy inference system for diagnosis of the heart valve diseases using wavelet transform with entropy," Neural Comput. Appl., vol. 21, no. 7, pp. 1617–1628, 2012.

[33] M. Taki and Y. Omid, "A new fuzzy based joint DF relay selection and link adaptation," in 2015 International Conference on Communications, Signal Processing, and their Applications (ICCSPA'15), 2015, pp. 1–6.

[34] V. Jain and S. Raheja, "Improving the prediction rate of diabetes using fuzzy expert system," IJ Inf. Technol. Comput. Sci., vol. 7, no. 10, pp. 84–91, 2015.

[35] G. Sun, R. Liang, H. Qu, and Y. Wu, "Embedding spatio-temporal information into maps by route-zooming," IEEE Trans. Vis. Comput. Graph., vol. 23, no. 5, pp. 1506–1519, 2016.

[36] G. Improta, V. Mazzella, D. Vecchione, S. Santini, and M. Triassi, " Fuzzy logic‐based clinical decision support system for the evaluation of renal function in post - Transplant Patients," J. Eval. Clin. Pract., vol. 26, no. 4, pp. 1224‐1234, 2020.

[37] A. Hajian and P. Styles, "Application of Neuro-Fuzzy Systems in Geophysics," in Application of Soft Computing and Intelligent Methods in Geophysics, Springer, 2018, pp. 417–484.

[38] G. Robins, T. Snijders, P. Wang, M. Handcock, and P. Pattison, "Recent developments in exponential random graph (p*) models for social networks," Soc. Networks, vol. 29, no. 2, pp. 192–215, 2007.