2019 IEEE SmartWorld, Ubiquitous Intelligence & Computing, Advanced & Trusted Computing, Scalable Computing & Communications, Cloud & Big Data Computing, Internet of People and Smart City Innovation

Optimal Bandwidth Allocation for Web Crawler Systems

Weiping Zhu*, Yaodong Li*, Yi Xu†, and Xiaohui Cui‡§
*School of Computer Science, Wuhan University, P. R. China
†Department of Mathematics, Southeast University, P. R. China
‡School of Cyber Science and Engineering, Wuhan University, P. R. China
{wpzhu, yaodongli, xcui}@whu.edu.cn, yixu@seu.edu.cn

***Abstract*—Web crawler is an important tool to obtain the information from the Internet in time. In a typical web crawler system with the limited bandwidth, there are many websites required to be crawled with different time constraints. Existing works about web crawler systems have not considered the bandwidth allocation in such a complex environment and hence the time constrains may not be properly satisfied. In this study, we investigate the bandwidth allocation approaches for such a web crawler system. The approaches are designed for both the scenarios that the sequences of crawling websites can not be changed and the scenarios that can. For the former scenario, we propose approaches to control bandwidth for web crawlers to minimize the maximum complete time or minimize the sum of complete times of all web crawlers. For the latter scenario, we propose a greedy algorithm based on crawling priorities, and a optimal algorithm based on mixed integer programming. Extensive simulations are conducted for validating the proposed approaches, and the result show that our approaches achieve desirable performance.**

***Index Terms*—Bandwidth Allocation; Distributed Crawler System; Time Control**

I. Introduction

In the last decade, the amount of data in the Internet has shown explosive growth [1]. These data contains a lot of useful information, however, cause difficulties for people in obtaining the information in time [2]. For example, when a graduate seeks a job, he or she browses several dozens of websites multiple times each day to catch the career information. This leads to read a lot of redundancy or useless content, and have difficulty in obtaining the latest information at the first time.

A web crawler is a program that is capable of automatically downloading web pages from the Internet and extracting required information from them [3], [4]. A web crawler starts from one or several initial webpages, and during the process of crawling new webpages can be continuously generated from the current page until a certain stop condition is satisfied [5]. Web crawlers can be invoked quite often to obtain the latest information and notify the users. A typical web crawler system can manage hundred of websites for multiple users. Web crawler is widely used in many fields including search engines [6], software testing [7], data analysis systems [8], [9], and data mining and indexing applications [10].

Since the users often have different requirements in collecting data from the websites, different time constraints are required for the crawling tasks. To meet these time constraints and minimize the execution time, the limited bandwidth available should be allocated properly to the crawling tasks. If the website cannot be crawling simultaneously due to the bandwidth constraint, the sequence and time duration of crawling for the websites require to be further considered.

There are existing works about the bandwidth control and allocation. The work [11], [12] adapted activates the optimal numbers of connections that the crawler opens simultaneously to fully utilize bandwidth. The work [13] achieved a better utilization of the available bandwidth by attempting to maintain as many as possible different websites in the crawler system and keep the connection alive to retrieve as many pages from a given website. The work [14], [15] proposed approach to utilize limited bandwidth effectively by assigning crawling task to the node whose physical address is closer to the given website. The work [16], [17] introduced a crawling methods based on mobile agent to utilize available bandwidth. The work [18] design a migrating crawler architecture based on URL scheduling mechanism using Analytic Hierarchy Process to utilize bandwidth effectively. The work [19] proposed a task scheduling strategy based on weighted round-robin to achieve load balance of nodes in a distributed crawler systems. However, the works aforementioned have not considered the time constrains specified for different websites to be crawled. Therefore, it is highly demanded that an approach is proposed to solve this problem for a web crawler system.

In this study, we investigate the bandwidth allocation problem for a web crawler system, in which there are multiple websites on the Internet required to be collected with time constrains. The approaches are designed for both the scenarios that the sequences of crawling websites can not be changed and the scenarios that can. For the former scenario, we propose approaches to control bandwidth for web crawlers to minimize the maximum complete time or minimize the sum of complete times of all web crawlers. For

§Corresponding Author. Email: xcui@whu.edu.cn

978-1-7281-4034-6/19/$31.00 ©2019 IEEE
DOI 10.1109/SmartWorld-UIC-ATC-SCALCOM-IOP-SCI.2019.00215

the latter scenario, we propose a greedy algorithm based on the priorities of the crawling tasks, and a optimal algorithm based on mixed integer programming. We perform extensive simulations to validate our proposed approach. In summary, this study makes the following contributions:

- We formulate the bandwidth allocation problem for a web crawler system. The problem have several factors including the number of websites to be collected, the time constraints specified for each crawling tasks, the bandwidth available, whether the crawling tasks' sequence can be changed, and whether the crawling tasks can be executed in multiple time durations.
- We proposed the optimal bandwidth allocation approach for the web crawler system where the crawling tasks' sequence cannot be changed. Two objectives are considered in the approach, including minimizing the maximum complete time and minimizing the sum of complete times of all web crawlers.
- We proposed the optimal bandwidth allocation approach for the web crawler system where the crawling tasks' sequence can be changed. We propose a greedy algorithm based on the priorities of the crawling tasks, and a optimal algorithm based on mixed integer programming.
- We performed extensive simulations to validate the performance of our proposed approach. The results show that our approach achieves good performance in various scenarios.

The remainder of this paper is organized as follows. In Section II, we review the related works. The problem formulation are described in Section III. Section IV includes the system model. Then, we propose our solutions based on analysis for the optimal objective of the problem in Section V. Numeral results are presented and discussed in Section VI. Finally, conclusions are given in Section VII.

II. Related Work

In existing works, different bandwidth allocation approaches were used to improve efficiency of crawler system with bounded bandwidth. The work [11] adapts the download rate of crawlers to the difference of campus network environment at different times of one day. The download rate is controlled by controlling the number of connections that the crawler opens simultaneously. Similarly, the work [12] sequentially activates the optimal numbers of fetcher to fully utilize bandwidth, where each fetcher corresponds a URL required to be collected. The work [13] achieved a better utilization of the available bandwidth by attempting to maintain as many as possible different websites in the crawler system and keep the connection alive to retrieve as many pages from a given website. These approaches can effectively reduce the waiting time before visiting the same website again. The work [14] proposed approach to utilize limited bandwidth effectively by assigning crawling task to the node whose physical address is closer to the given website. The work [15] avoided overlap network overheads by removing the urls that had been crawled and assigning different urls to different nodes to make use of limited bandwidth effectively. The work [16], [17] introduced a crawling methods based on mobile agent to utilize available bandwidth. The mobile agent can reduce the HTTP overhead by transferring the crawler to the source of the data and compress webpages before transmitted back to local. The work [18] design a migrating crawler architecture based on URL scheduling mechanism using Analytic Hierarchy Process to utilize bandwidth effectively. The work [19] proposed a task scheduling strategy based on weighted round-robin to achieve load balance of nodes in a distributed crawler systems. However, all the above works had failed to consider time constrains specified for different websites to be crawling. Our proposed approaches achieved a high utilization of the available bandwidth in crawler system with multiple websites, where time constrains were taken into account.

III. Problem Formulation

In a web crawler system, we assume that we want to collect information from a number of websites. Then we program crawlers for these websites to get web data. Two situations is developed according to the sequences of crawling websites can not be changed or the scenarios that can. This is.

- Situation 1: the sequences of crawling websites can not be changed (sequence unchanged)
- Situation 2: the sequences can be changed (sequence changed)

The details of situation 1 and situation 2 will respectively illustrated in following.

A. Sequence Unchanged

In this situation, for each website, there is a web crawler to collect the information from it. And each crawler associate with a specify website and only crawl data from the website. The information in each website is required to be collected with a certain time constraint depending on their priorities. The sum of all allocated bandwidth is limited by the bandwidth available in the web crawler system. The objective is to find the optimal bandwidth allocation among these web crawlers to achieve the optimal time constraints.the mathematical notations used is summarized in TABLE I.

In this condition, there are two kinds of optimal time constraints for this problem. One is to minimize the maximum download time of all websites, and the other is to minimize the sum of download time of all websites. The first time constraint is to finish all download tasks as soon as possible, while the second time constraint is to minimize the total system resource consumption. We formally formulate the problem as follows.

Given

1) There are n websites whose information are required to be collected.

TABLE I
MATHEMATIC NOTATIONS USED IN THIS PAPER

Symbol	Meaning	type of data
n	number of our spiders	integer
U	net speed constraint of our project	float
D_i	bytes of our news to be crawled	integer
v_i	allocated net speed of each spiders	float
t_i^*	time limit for each spiders, 0 represent no limit	float

2) Each website has the amount of data $D_i(i = 1, 2, ...n)$.

3) Each website has a time constraint of $t_i^*(i = 1, 2, ...n)$.

4) The maximum bandwidth for the web crawler system is U.

determine the bandwidth $v_i(i = 1, 2, ...n)$ for each website in any time such that

1) $\min\max\ t_i$

or 2) $\min \sum_{i=1}^{n}(t_i)$

where t_i is the end time of downloading of website *i*

subject to

1) The time duration for collecting the information of website $j(j = 1, 2, 3 \cdots n)$ should be no more than t_j^*.

2) $\sum v_i \leq U$.

3) there exist a minimum data amount for allocation, or there exist a minimum bandwidth for allocation

each crawler to crawl the corresponding task minimizes total system resource consumption. Firstly, We want the system to complete the total amount of total tasks for the least amount of time. Secondly, The minimum sum of time for each crawler to crawl the corresponding task minimizes total system resource consumption.

The basic idea of our approach is minimize crawling time. Specifically, for each crawler *i*, we predict capacity of Internet information D_i to be crawled next moment and time t_i will be cost to crawl these task D_i. So to make crawler system reach optimal performance, our goal can be object have two types:Firstly, We want the system to complete the total amount of total tasks for the least amount of time. Secondly, The minimum sum of time for each crawler to crawl the corresponding task minimizes total system resource consumption. These two expressions can be included in the following formula.

B. Sequence Changed

In this situation, the number of websites to be crawled is far greater than the number of the most crawlers that our system execute simultaneously. So the websites must be divided into batches to be crawled. In the business world, the data company should satisfied the time demands of the client in getting the data. if not, the company must compensate the loss of client based on the amount of time exceeding limit time. So we should minimize the sum of time to crawl information from the websites. We get some parameters through testing the crawler system and auxiliary software. The problem not only involve the the variables in the TABLE I but also the parameters. We formally formulate the problem as follows and the mathematical notations used is summarized in TABLE II.

TABLE II
MATHEMATIC NOTATIONS USED IN THIS PAPER

Symbol	Meaning	type of data
n	number of our spiders	integer
U	net speed constraint of our project	float
D_i	bytes of our news to be crawled	integer
v_i	allocated net speed of each spiders	float
t_i^*	time limit for each spiders, 0 represent no limit	float
v^*	allocated minimum net speed of each spiders	float
t	time of starting a crawler	float
m	the maximum number of our system can execute	integer

Given

1) There are *n* websites whose information are required to be collected.

2) Each website has the amount of data $D_i(i = 1, 2, ...n)$.

3) Each website has a time constraint of $t_i^*(i = 1, 2, ...n)$.

4) The maximum bandwidth for the web crawler system is U.

5) The minimum allocated net speed of each spiders is v_i^*.

6) The time for starting a crawler is t.

7) The maximum number of crawlers our system can execute is m.

determine the bandwidth $v_i(i = 1, 2, ...n)$ for each website in any time such that

$\min \sum_{i=1}^{n}(t_i)$

where t_i is the end time of downloading of website *i*

subject to

1) $n > m$.

2) The time duration for collecting the information of website $j(j = 1, 2, 3 \cdots n)$ should be no more than t_j^*.

3) $\sum v_k \leq U$ in any time, where k is crawlers at a certain time.

4) $v_i \geq v^*$

IV. System Implementation

Data crawling method is a foundation of studying bandwidth allocation. An efficient crawler and reasonable crawling strategy are important guarantee of testing bandwidth allocation approaches. In the collecting data more effectively and understanding crawler technology, we select the Scrapy crawler framework as our crawling tool. Details about crawling tools, crawling strategy are introduced as followings.

A. Scrapy

Scrapy is an application framework for crawling web sites and extracting structured data which can be used for a wide range of useful applications, like data mining, information processing or historical archival [20]. The architecture of the framework is clear and it contains various middleware interface. Scrapy is written in Python and uses the Twisted asynchronous network library to handle network communication. It is an integrated system that includes an engine,

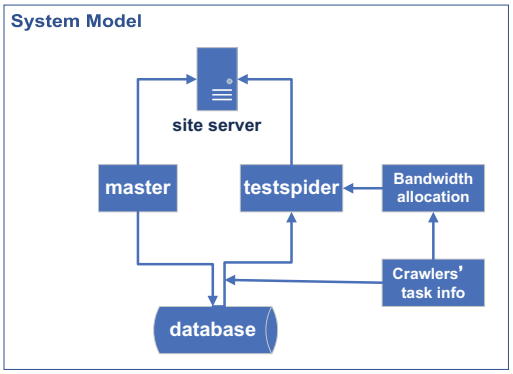

Fig. 1. System Model

a scheduler, a downloader, Item Pipeline, and Spiders [21]. Users can easily implement a crawler by only customize some modules. It is convenient for users who want to grab web content and all kinds of pictures [22].

B. System Model

In this paper, we need collect some network task to test bandwidth allocation approach. Besides, we need to implement some crawler crawling above the amount of network tasks. The architecture of the entire system is included in Fig 1.

There are mainly four components in the system model:master is some crawlers written by Scrapy, which is Responsible for collecting urls from the target site as the task amount of the test algorithm; testspider is also some crawlers written by Scrapy, which is Responsible for crawling web pages based on crawler task information and bandwidth allocation results; bandwidth allocation module is used to allocate net speed for every spider of testspider; database is used for storing links to be crawled.

V. The Solution

In this section, we study solution for the two Situations described in Section III.

A. Sequence Unchanged

In this situation, we propose our solution to meet the min-max time objective and min-sum time objective described in Section III. In these two problems, we need to set t_i^* for crawling every task. we want to reach the two goals as the following.

- Type 1:min max t_i
- Type 2:min $\sum_{i=1}^{n}(t_i)$

The basic idea and details of Type1(min-max) and Type2(min-sum) will respectively illustrated in following.

Algorithm 1: Type1: Minimize the Maximum Complete Time of All Crawlers (M-MAXT)

Input: sitelist containing information of all website and U

Output: net speed for each website

```
1  sort task that not allocated by t_i^* ascendingly;
2  calculate the average time of remaining tasks: t̄
3  if t̄ < the first t_i^* then
4  |  calculate net speed based t̄;
5  |  End;
6  else
7  |  foreach task do
8  |  |  if t̄ is lager than t_i^* and t_i^* ≠ 0 then
9  |  |  |  allocate net speed for the task according to
   |  |  |  D_i and t_i^*;
10 |  |  else
11 |  |  |  do nothing;
12 |  |  end
13 |  end
14 end
15 repeat line 1 to 14
```

1) Type 1: Min-Max: In this type, we need to minimize the max(t_i).

We consider the situation without time constraint first. In this situation, we just require to allocate v_j proportional to their data amount D_j. All web crawlers will finish their download task at the time t. It is easy to calculate the

$$t = \frac{\Sigma D_j}{U} \tag{1}$$

Then we put the time constraints $t_i^*(i = 1, 2, ...n)$ into consideration. We propose a greedy algorithm to solve this problem. We sort the tasks by the t_i^* in ascending order. The average time of the unallocated web sites $\bar{t}$ is calculated as follows:

$$\bar{t} = \frac{\Sigma D_i(i \in \textit{unallocated})}{U_{\textit{unallocated}}} \tag{2}$$

where *unallocated* denotes the unallocated web sites and $U_{\textit{unallocated}}$ is the remaining available bandwidth. The smallest $t_i^* \neq 0$ is compared with $\bar{t}$. If $t_i^* > \bar{t}$, the information of all the remaining web sites can be downloaded before their time constraints. Therefore, all the remaining web sites are assigned bandwidth proportional to their weights. Otherwise, the website i under considering is assigned the bandwidth $v_i = \frac{D_i}{t_i^*}$. This is the the minimum bandwidth required to meet its time constraint and maximal remaining bandwidth are left for other web sites. This process is repeated until all the websites are considered. The bandwidth allocation is failed if the unallocated bandwidth is less than or equal to 0. The detailed algorithm can be seen in Algorithm 1.

Algorithm 2: Type2:Minimize the Sum of Complete Times of All Crawlers (M-SUMT)

Input: sitelist containing information of all website and U

Output: net speed for each website

```
1  sort tasks unallocated by t_i^* ascendingly;
2  foreach task do
3  |  calculate t_i based equation (5);
4  end
5  if ∀t_i < t_i^* then
6  |  foreach task do
7  |  |  calculate v_i based (5);
8  |  end
9  |  End;
10 else
11 |  foreach task do
12 |  |  if t_i > t_i^* and t_i ≠ 0 then
13 |  |  |  set v_i = D_i / t_i^*;
14 |  |  else
15 |  |  |  send task to list unallocated;
16 |  |  end
17 |  end
18 end
19 repeat line 1 to 18
```

2) Type 2: Min-Sum: In this type, we need to minimize the $\sum_{i=1}^{n}(t_i)$, which is equals to find the avg $\sum_{i=1}^{n}(t_i)$. Similarly, we consider the unconstraint situation first.

According to Cauchy inequality, we have

$$(\sum \frac{D_i}{v_i})(\sum v_i) \geq (\sum \sqrt{D_i})^2 \tag{3}$$

The equal condition is hold when

$$\forall i, \frac{D_i}{v_i^2} = \lambda \tag{4}$$

where λ is a constant. Since $\sum v_i = U$, we can figure out that

$$v_i = \frac{U\sqrt{D_i}}{\sum \sqrt{D_i}} \tag{5}$$

According to Lagrange multiplier method, we assume there are *m* of *n* websites have time constraints. The Lagrange function is

$$L(v_i, \lambda_j) = \sum_{i=1}^{n} \frac{D_i}{v_i} - \lambda_j(t_i^* - \frac{D_i}{v_i}) - \mu(U - \sum_{i=1}^{n} v_i) \tag{6}$$

The optimal solution is obtained by

$$\nabla L_V = 0, \lambda_j(t_i^* - \frac{D_i}{v_i}) = 0, \lambda_j \geq 0, \frac{D_i}{v_i} \geq 0 \tag{7}$$

Then we know our v_i have only two choices,$v_i = \frac{D_i}{t_i^*}$ and $v_i = \frac{U_{remain}\sqrt{D_i}}{\sum \sqrt{D_i}}$, and the last choice is better. So we need to pick up the tasks which cannot satisfy the last choice. Every time we sort the tasks by t_i^* ascendingly. We let every task follow first choice at the beginning in order, if it cannot satisfy the constraint condition then we switch the v_i into the second choice. Then we resort them and go through this procedure again.

By solving it, we have $v_i = \frac{D_i}{t_i^*}, \lambda_j \neq 0$ or $\lambda_j = 0$. When $\lambda_j = 0$ holds, all the solution of v_i except for $v_i = \frac{D_i}{t_i^*}$ are in the feasible domain. This is a special case of the problem solved at the beginning of this section. The bandwidth of all crawlers are assigned proportionally. We have

$$v_i = \frac{\sqrt{D_i}}{\frac{\sum \sqrt{D_i}}{U_{remain}}}. \tag{8}$$

where U_{remain} is remaining bandwidth after the boundary conditions are met. The minimum value must be obtained in one of the two cases for each v_i aforementioned.

So our idea is to separate the two queues to calculate vi according to the optimal solution condition. The optimal situation is definitely that there is no time limited that is the first type of optimal solution conditions is fully satisfied. If the first type of condition cannot be met at this time, then it is placed in the queue of the second type of optimal solution condition. Then we firstly do a pre-allocation that is only allocated according to the first category of conditions. For the v_i assigned, the website that cannot meet the time limited is placed in the second queue, and v_i is calculated according to the second type of condition. Since U_remain of the remaining site is reduced, the remaining sites $i+1$ to n are once again assigned according to the first type of condition. The websites that does not meet the time limit is found and assigned according to the second type of condition. The websites behind are then reassigned according to the first type of conditions...Iterative allocation like this. If all sites meet two types of conditions, then the end. The situation allocation failure is that unallocated bandwidth is less than or equal to 0 after all websites with time limited has been assigned or the remaining bandwidth is not enough to support the site to run within the time limited. The process described above can be expressed as Algorithm 2.

B. Sequence Changed

In this situation, we propose two algorithm to achieve a optimal bandwidth allocation. The first is a greedy algorithm based on crawling priority queue (CPQ), and second is a optimal algorithm based on mixed integer programming (MIP). The details of CPQ algorithm is the following.

- These tasks is sorted in ascending order, based on their limit time. A task with less limit time will has a higher priority.
- The websites ws with the same number as crawlers are popped from the queue and are assigned to the crawlers.
- For the crawler of website i, The net speed $v_i = D_i / \sum_{j \in ws} D_j U$.

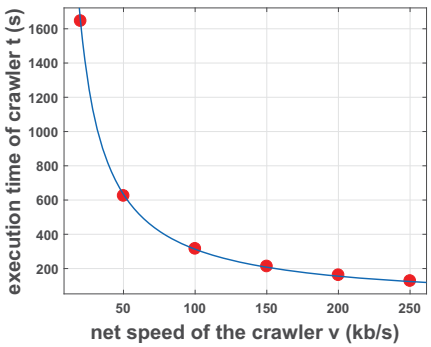

Fig. 2. The fitting relation of t-v

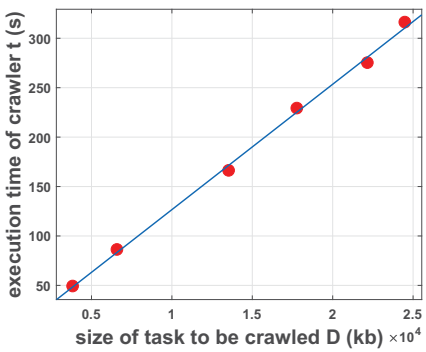

Fig. 3. The fitting relation of t-D

- repeat step 2,3 until the queue is empty.

The details of MIP algorithm is the following.

1) All websites are divide lots of batches and we assume the time for the first batch websites crawled is start time t_0.
2) The time for crawling all websites is divided into lots of fixed-size time interval t_f and start from t_0.
3) Then, we chose a set of websites w to be crawled in every time interval and the number of these websites is equal to m.
4) set two constraints in every time interval:$\sum_{j\in w}(v_j) \leq U$ and $\sum_{k\in w}(D_k) \leq (U * t_f)$.
5) get minimum $\sum_{i=1}^{n}(t_i)$ by a optimal division of websites.
6) optimal the objective with constraints by Yalmip and CLPEX integrated optimization.

VI. Experimental Results

The experimental system is implemented in pure Python. Due to limitations of equipment, This experiment simulates a distributed environment through multi-progress. NetBalancer [23] are freeware programs that are used to control the speed of downloading and uploading files and are installed in normal end clients in the experiment. Firstly, we need to verify the relation of time, net speed and size of task of crawler. We conducted two sets of experiments to find separately the relation of time and net speed and the relation of time and size of task in the same website. The size of task is the size of webpages to be crawled. Fig. 2 show the fitting result of time and net speed of the crawler. From the figure, we can see that time is inversely proportional to net speed. Fig. 3 show the fitting result of time and size of task of the crawler. The figure indicate that the time is linearly related to size of task of the crawler. So we can get the relation of time, net speed and size of task of crawler as following.

$$t = \frac{\xi D}{v} \tag{9}$$

In this experiment, ξ is constant. After that, we did the same experiment for the other websites we were going to crawl. we find the time, net speed and size of task of crawler for these websites follow the same rules. But the ξ of each website is different. So to test the algorithms described in Section V, we need set $D_i = \xi D_i$.

A. Sequence Unchanged

In this situation, we write six crawlers to crawl webpages of six different websites. The size of webpages of every website to be crawled is different. We conduct experiments on three influencing factors (i.e. bandwidth, task to be crawled and time constrains) separately in the following. (1) We specify the amount of each crawler task and test and compare the results of the two algorithms when the total network download speed is different. We designed 4 sets of experiments and the total net speed of the experiments is respectively 800,900,1000,1100; and the unit is kbps. (2) We specify the total network download speed, and test and compare the results of the two algorithms in the case where each crawler has different amounts of tasks to be crawled. We designed 4 sets of experiments and the size of webpages of same website to be crawled in each set of experiments is different. (3) We specify the amount of each crawler task and the total network download speed, and test and compare the results of the two algorithms in the case the time limited varies in degree of rigor. The degree of rigor of time limited can be measured by the $\sum \frac{D_i}{t_i^*} \quad (t_i^* \neq 0)$. The larger the value of this formula, the larger degree of rigor it is. Otherwise, the smaller degree of rigor it is. We designed 4 sets of experiments and the degree of rigor of time limited is increase.

In this section, algorithm M-MAXT and M-SUMT were tested and compared in three types of datasets mentioned in previous paragraph. Fig. 4 and Fig. 5 separately show the result of comparison of maximum crawling time between the two algorithms in the series experiments of (1). Fig. 6 and Fig. 7 separately show the comparison of sum of crawling time for all website contents between the two algorithms in the series experiments of (2). From these figures, we can

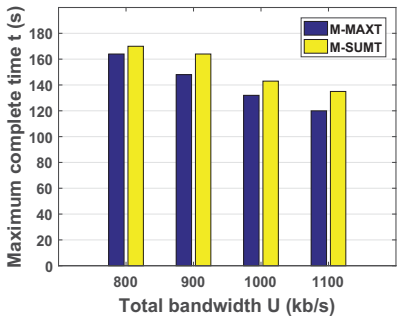

Fig. 4. comparison of maximum time for two algorithms under different bandwidths

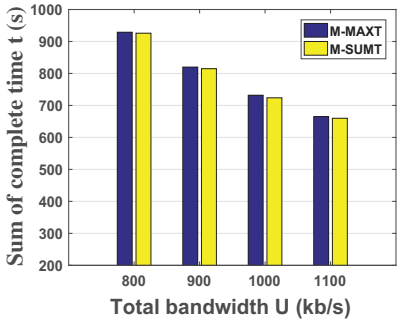

Fig. 5. comparison of sum of time for two algorithms under different bandwidths

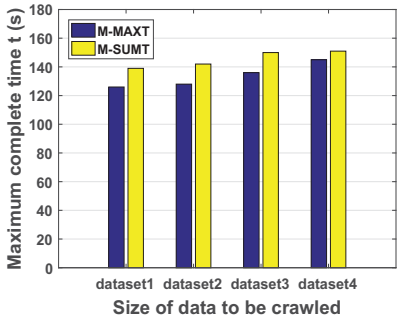

Fig. 6. comparison of maximum time for two algorithms under different sizes of webpages of each crawler to be crawled

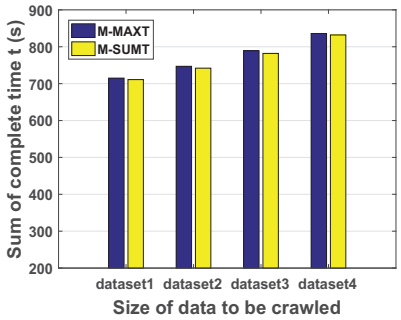

Fig. 7. comparison of sum of time for two algorithms under different sizes of webpages of each crawler to be crawled

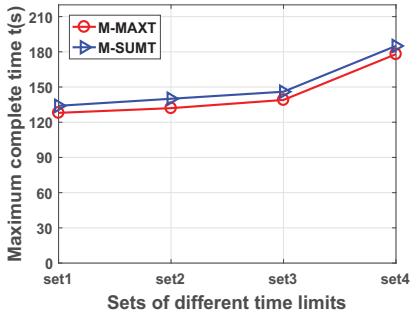

Fig. 8. comparison of maximum time for two algorithms under different limitation of time

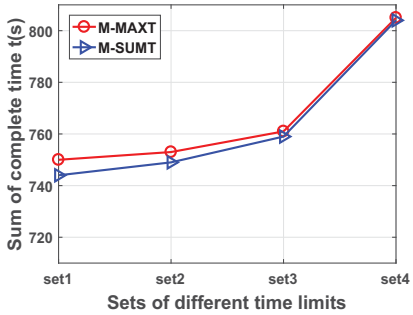

Fig. 9. comparison of sum of time for two algorithms under different limitation of time

see that the maximum crawling time of M-MAXT is less than M-SUMT, but the sum of crawling time of M-SUMT is less than M-MAXT. Fig. 8 and Fig. 9 separately describes the difference between M-MAXT and M-SUMT with the increase of the degree of rigor of time limited. the conclusion is same to the former two sets of experiments. We also observed that along with the increase of the degree of rigor of time limited, the difference between M-MAXT and M-SUMT almost constant in terms of maximum execution time of the crawler. Nevertheless, the difference between the two algorithms for sum of execution time of all crawlers decrease with the increase of the degree of rigor of time limited. The degree of rigor of time limited can be measured by the $\sum \frac{D_i}{t_i^*} \quad (t_i^* \neq 0)$. The larger the value of this formula, the larger degree of rigor it is.

B. Sequence Changed

In this situation, we consider a application scenario where there are 600 websites and a crawler system with 30 crawlers. Two simulations was implemented to verify the performance of our proposed CPQ algorithm and MIP algorithm.

In this subsection, we examine the impact of the number of bandwidth U on execution time. Fig. 10 denotes the sum of time for all crawlers of PQ-DB and MILP when U varies from 6 to 10. From Fig. 10, we can draw several observations. First, the sum of execution time of the proposed PQ-DB, as well as MILP and increases rapidly as the number of total bandwidth becomes small. The reason is that the execution time is negatively and linearly related to the number of bandwidth. Second, the performance of the proposed PQ-DB significantly outperforms the policies MILP. Then, we compare the proposed PQ-DB algorithm with the MILP alogrithm. Fig. 11 plots the comparison of end time when all websites crawled between PQ-DB and MILP with respect to the number of total bandwidth. From the results, we can see that the end time of the above two policies respectively decreases with the increasement of U. Moreover, the performance of the proposed PQ-DB is better than MILP in bandwidth allocation.

VII. Conclusion

In this paper, we have presented some approaches to allocate the bandwidth for a crawler system with different time constraints. we designed different schemes according to two different scenarios that the sequences of crawling websites can not be changed and the scenarios that can. Then different optimal objective is defined and corresponding algorithms are developed to achieve them. The detail procedures of the bandwidth allocation for crawlers are described.

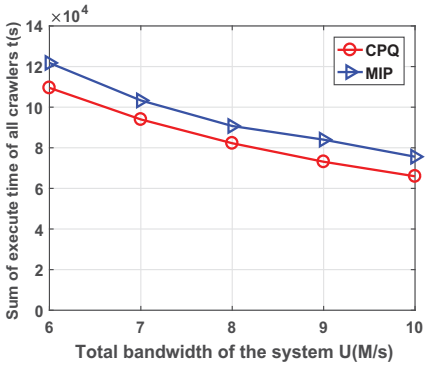

Fig. 10. comparison of sum of execution time under different policies with different bandwidth

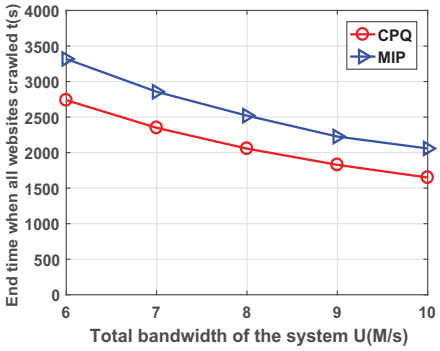

Fig. 11. comparison of end time when all websites are crawled under different policies with different bandwidth

Moreover, we perform extensive simulations to verify these approaches in terms of bandwidth time constrains and task to be crawled. The results show that our approach achieves good performance in various scenarios.

ACKNOWLEDGMENT

We thank Shu Li for developing the bandwidth allocation algorithm for the environment that the crawling sequence cannot be changed. This research is supported in part by National Key R&D Program of China No. 2018YFC1604000, Chutian Scholars Program of Hubei, China, 2018 Science and Technology Transformation Project of Grain Administration of Hubei Province "Grain and Oil Quality & Safety Assurance System Research", and 2018 English Course Project of Wuhan University "Business Intelligence".

REFERENCES

[1] R. Ding and M. Wang, "Design and implementation of web crawler based on coroutine model," in *Cloud Computing and Security*, X. Sun, Z. Pan, and E. Bertino, Eds. Cham: Springer International Publishing, 2018, pp. 427–435.

[2] M. Kumar, R. Bhatia, and D. Rattan, "A survey of web crawlers for information retrieval," *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, vol. 7, no. 6, p. e1218, 2017.

[3] Y. Wang, Z. Hong, and M. Shi, "Research on lda model algorithm of news-oriented web crawler," in *2018 IEEE/ACIS 17th International Conference on Computer and Information Science (ICIS)*, vol. 00, June 2018, pp. 748–753.

[4] T. Y. Chun, "World wide web robots: an overview," *Online Information Review*, vol. 23, no. 3, pp. 135–142, 1999.

[5] J. Cho, "Crawling the web: Discovery and maintenance of large-scale web data," Ph.D. dissertation, Stanford, CA, USA, 2002, aAI3038076.

[6] A. Arasu, J. Cho, H. Garcia-Molina, A. Paepcke, and S. Raghavan, "Searching the web," *ACM Transactions on Internet Technologies*, vol. 1, no. 1, pp. 2–43, 2001.

[7] S. Raina and A. Prakash Agarwal, "How crawlers aid regression testing in web applications: The state of the art," *International Journal of Computer Applications*, vol. 68, no. 14, pp. 33–38, 2014.

[8] C. H. Lau, X. Tao, D. Tjondronegoro, and Y. Li, "Retrieving information from microblog using pattern mining and relevance feedback," in *Proceedings of Data and Knowledge Engineering*. Springer Berlin Heidelberg, 2012, pp. 152–160.

[9] R. Cai, J.-M. Yang, W. Lai, Y. Wang, and L. Zhang, "irobot: An intelligent crawler for web forums," in *Proceedings of the 17th International Conference on World Wide Web*. New York, NY, USA: ACM, 2008, pp. 447–456.

[10] J. Edwards, K. McCurley, and J. Tomlin, "An adaptive model for optimizing performance of an incremental web crawler," in *Proceedings of the 10th International Conference on World Wide Web*, ser. WWW '01. New York, NY, USA: ACM, 2001, pp. 106–113.

[11] V. Shkapenyuk and T. Suel, "Design and implementation of a high-performance distributed web crawler," in *Proceedings 18th International Conference on Data Engineering*, Feb 2002, pp. 357–368.

[12] M. Diligenti, M. Maggini, F. M. Pucci, and F. Scarselli, "Design of a crawler with bounded bandwidth," in *Proceedings of the 13th International World Wide Web Conference on Alternate Track Papers &Amp; Posters*, ser. WWW Alt. '04. New York, NY, USA: ACM, 2004, pp. 292–293.

[13] C. Castillo, M. Marin, A. Rodriguez, and R. Baeza-Yates, "Scheduling algorithms for web crawling," in *WebMedia and LA-Web, 2004. Proceedings*, Oct 2004, pp. 10–17.

[14] M. Kc, M. Hagenbuchner, and A. C. Tsoi, "A scalable lightweight distributed crawler for crawling with limited resources," in *2008 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology*, vol. 3, Dec 2008, pp. 663–666.

[15] D. Yadav, A. Sharma, and J. Gupta, "Parallel crawler architecture and web page change detection," *WSEAS Transactions on Computers*, vol. 7, pp. 929–940, 07 2008.

[16] N. Singhal, R. P. Agarwal, A. Dixit, and A. K. Sharma, "Information retrieval from the web and application of migrating crawler," in *2011 International Conference on Computational Intelligence and Communication Networks*, Oct 2011, pp. 476–480.

[17] S. K. S. Md. Abu Kausar, V. S. Dhaka, "Web crawler based on mobile agent and java aglets," *International Journal of Information Technology and Computer Science(IJITCS)*, vol. 5, no. 10, pp. 85–91, 2013.

[18] D. Punj and A. Dixit, "Design of a migrating crawler based on a novel URL scheduling mechanism using AHP," *IJRSDA*, vol. 4, no. 1, pp. 95–110, 2017.

[19] D. Ge and Z. Ding, "A task scheduling strategy based on weighted round-robin for distributed crawler," in *2014 IEEE/ACM 7th International Conference on Utility and Cloud Computing*, Dec 2014, pp. 848–852.

[20] J. Wang and Y. Guo, "Scrapy-based crawling and user-behavior characteristics analysis on taobao," in *2012 International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery*, Oct 2012, pp. 44–52.

[21] D. Myers and J. McGuffee, "Choosing scrapy," *Journal of Computing Sciences in Colleges*, vol. 31, pp. 83–89, 10 2015.

[22] Y. Liang, Y. Zhao, D. Que, X. Zhang, and C. Xu, "Online fake drug detection system in heterogeneous platforms using big data analysis," in *2016 7th International Conference on Cloud Computing and Big Data (CCBD)*, Nov 2016, pp. 308–311.

[23] S. Srl, "Netbalancer." [Online]. Available: http://netbalancer.com/.