Journal of Ambient Intelligence and Humanized Computing
https://doi.org/10.1007/s12652-020-02377-1

ORIGINAL RESEARCH

Optimal bandwidth allocation for web crawler systems with time constraints

Weiping Zhu[1] · Yaodong Li[1] · Shu Li[2] · Yi Xu[3] · Xiaohui Cui[4]

Received: 2 November 2019 / Accepted: 21 July 2020

Abstract
Web crawler is an important tool to obtain information from the Internet in a timely manner. In a typical web crawler system with limited bandwidth, many websites are crawled with different time constraints. Existing studies regarding web crawler systems do not consider the bandwidth allocation in such a complex environment; hence, the time constraints may not be satisfied. In this study, we investigate the bandwidth allocation approaches for such a web crawler system. The approaches are designed for two scenarios, i.e., when the number of websites exceeds or does not exceed the maximum number of web crawlers that the system can execute simultaneously. For the latter situation, we propose approaches to control the bandwidth for web crawlers to minimize the maximum complete time or minimize the sum of execution times of all web crawlers, considering assumptions of both sufficient and insufficient bandwidths. For the former situation, we propose a round-based reallocation approach to schedule both the sequence and bandwidth allocation of the web crawlers. Extensive simulations are conducted to validate the proposed approaches, and the results show that our approaches satisfy the time constraints well and achieve desirable execution performances in various scenarios.

Keywords Bandwidth allocation · Web crawler · Time constraint · Optimization

1 Introduction

In the last decade, the amount of data on the Internet has grown significantly (Ding and Wang 2018). The data contain a significant amount of useful information; however, it is difficult to obtain the information in a timely manner (Kumar et al. 2017). For example, when graduates seek jobs, they often browse several dozens of websites multiple times daily to obtain job-related information. Hence, they may read redundant or useless content and encounter difficulties in obtaining the latest information. In another example, people browsing the Internet may overlook information regarding food safety from multiple data sources, and this may affect their health.

A web crawler is a program that can automatically download web pages from the Internet and extract the required information from them (Wang et al. 2018b; Thelwall 2001). A web crawler starts from one or several initial web pages, and more web pages can be continuously collected until a certain stop condition is satisfied. Web crawlers can be invoked often to obtain the latest information for users. A typical web crawler system can manage hundreds of websites for multiple users. It is widely used in

✉ Xiaohui Cui
xcui@whu.edu.cn

Weiping Zhu
wpzhu@whu.edu.cn

Yaodong Li
yaodongli@whu.edu.cn

Shu Li
whu-shuli@foxmail.com

Yi Xu
yixu@seu.edu.cn

[1] School of Computer Science, Wuhan University, Wuhan, People's Republic of China

[2] School of Mathematics and Statistics, Wuhan University, Wuhan, People's Republic of China

[3] Department of Mathematics, Southeast University, Nanjing, People's Republic of China

[4] School of Cyber Science and Engineering, Wuhan University, Wuhan, People's Republic of China

many fields, including search engines (Arasu et al. 2001), software testing (Raina and Prakash Agarwal 2014), data analysis systems (Lau et al. 2012; Cai et al. 2008), data mining and indexing applications (Edwards et al. 2001; Guojun et al. 2017), and food safety management.

Because users often have different requirements when collecting data from websites, different time constraints are required for crawling tasks. To satisfy these time constraints and minimize the execution time, the limited bandwidth should be allocated suitably for the crawling tasks. If the websites cannot be crawled simultaneously owing to the bandwidth constraint, the sequence and time duration of the website crawling must be further considered.

Existing studies regarding bandwidth control and allocation mainly focus on better utilization of the available bandwidth, including the determination of the optimal number of connections dynamically and maintaining live connections to different websites (Shkapenyuk and Suel 2002; Diligenti et al. 2004; Castillo et al. 2004). Some studies investigated distributed system architectures for bandwidth control and allocation, including the determination of appropriate physical locations for system nodes (Kc et al. 2008; Yadav et al. 2008), and balancing the loads of system nodes for crawling tasks (Ge and Ding 2014). Recently, some advanced techniques including mobile agents (Singhal et al. 2011; Kausar et al. 2013), triple-stage Stackelberg game (Meng et al. 2018), constrained convex utility maximization (Das et al. 2019), genetic algorithm (Shams et al. 2017), and differential evolution (Afzal et al. 2019) have been used to solve this problem. However, these studies did not fully consider various time constraints in crawling, in particular different tightness conditions of time constraints, or different objective functions of execution time. Therefore, a better approach is necessitated to solve this problem.

In this study, we investigated the bandwidth allocation problem for a web crawler system, in which data from multiple websites on the Internet must be collected with time constraints. We considered various situations for this problem. First, the approaches were designed for two scenarios, i.e., when the number of crawling tasks exceeds or does not exceed the maximum number of crawlers that the system can execute simultaneously. For the latter scenario, we propose approaches to control the bandwidth for web crawlers to minimize the maximum complete time or minimize the sum of execution times subject to time constraints. Both sufficient and insufficient bandwidths were assumed. For the former scenario, we propose a round-based reallocation approach to solve the problem. The approaches proposed in the latter scenario were reused in individual rounds, and bandwidth reallocation was performed to optimize their succession. In summary, the contributions of the current study are as follows:

- We formulate the bandwidth allocation problem for a web crawler system with time constraints in different situations. It can accommodate different relations of crawling tasks with the maximum number of crawlers that can execute simultaneously and with different tightness of time constraints. Multiple useful execution time objectives are investigated for time-sensitive applications, system resource providers, and applications involving time constraint violations. These formulations can be used as building blocks for more complex applications
- We propose an optimal bandwidth allocation approach for a web crawler system, where the number of crawling tasks does not exceed the maximum number of crawlers. We illustrate the approach in a situation involving sufficient and insufficient bandwidths.
- We extend our study to the optimal bandwidth allocation approach for a web crawler system with a large number of crawling tasks. In addition to bandwidth allocation, the execution sequence of crawling tasks is determined.
- Extensive simulations are performed to validate the performance of our proposed approach. The results show that our approaches satisfy the time constraints well and achieve desirable execution performances in various scenarios.

The remainder of this paper is organized as follows. In Sect. 2, we review the related work. In Sect. 3, the system models used in this study are described and the problem formulation is presented. Subsequently, our solutions are proposed in Sect. 4. The evaluation results are presented and discussed in Sect. 5. Finally, the paper is concluded in Sect. 6.

This paper is based on a conference paper (Zhu et al. 2019). In this version, we extend our approach to support insufficient bandwidths and propose new evaluation metrics for the revised problem. Additional discussions and evaluation results are included herein.rics for the revised problem. Additional discussions and evaluation results are also included in this paper.

2 Related work

In existing studies, different bandwidth allocation approaches are used to improve the efficiency of web crawler systems with limited bandwidths. The work (Shkapenyuk and Suel 2002) adapted the download rate of web crawlers to a campus network environment at different times of the day. The download rate is controlled by controlling the number of connections that the web crawlers open simultaneously. Similarly, the work (Diligenti et al. 2004) sequentially activated the optimal numbers of fetchers to fully utilize the bandwidth, where each fetcher corresponds to a

URL (Uniform Resource Locator) required to be collected. The work (Castillo et al. 2004) achieved better utilization of the available bandwidth by attempting to maintain live connections to the maximum number of different websites and retrieve the maximum number of pages from a specific website. This can effectively reduce the waiting time before the same website is revisited. The work (Kc et al. 2008) proposed an approach to effectively utilize a limited bandwidth by assigning crawling tasks to a node whose physical address is closer to a specified website. The work (Yadav et al. 2008) avoided network overheads by removing URLs that had been crawled and assigned different URLs to different nodes. The work (Singhal et al. 2011; Kausar et al. 2013) introduced a crawling method based on a mobile agent to utilize the available bandwidth. The mobile agent can reduce the HTTP overhead by transferring the web crawler to the data source and compress webpages before they are transmitted back. The work (Ge and Ding 2014) proposed a task scheduling strategy based on a weighted round robin to achieve the load balance of nodes in a distributed crawler system. The work (Punj and Dixit 2017) designed a migrating crawler architecture based on a URL scheduling mechanism to utilize bandwidth effectively. Most of the aforementioned studies focused on the better utilization of available bandwidths but did not consider any specific bandwidth allocation processes, in particular those that involve complex time constraints.

Recently, the bandwidth allocation problem has been investigated in different scenarios. The work (Fei et al. 2010) allocated a bandwidth for relay stations in IEEE 802.16j-based vehicular networks. The work (Jiang et al. 2011) proposed a hierarchical QoS-aware dynamic bandwidth allocation algorithm for wireless optical broadband access networks. The work (Amamou et al. 2012) designed a service-level agreement-aware dynamic bandwidth allocator for a virtualized cloud environment. The work (Wang et al. 2018a) proposed a queue-based bandwidth allocation method for streaming media servers in m-learning video-on-demand systems. The work (Meng et al. 2018) solved wireless bandwidth and computing resource allocation using a triple-stage Stackelberg game. The work (Das et al. 2019) formulated the bandwidth allocation in mobile cloud computing as a constrained convex utility maximization problem and proposed a cheating-resilient bandwidth distribution scheme to solve it. The works (Shams et al. 2017; Afzal et al. 2019) used a genetic algorithm and differential evolution, respectively, to allocate bandwidths for cellular IP networks to ensure an acceptable QoS level. However, none of the aforementioned studies considered various time constraints specified for different websites and different execution time objectives for users. Our proposed approaches achieved a higher utilization of available bandwidths in a web crawler system with multiple websites, where various time constraints are considered.

3 System model and problem formulation

In this study, we designed a web crawler system based on Scrapy (Wang and Guo 2012), which is a widely used application framework for crawling websites and extracting structured data. We use this system to collect information from websites related to food and health, supporting for the data analysis about food safety management. The architecture of the system is shown in Fig. 1.

The system comprises four main modules. Master is a web crawler that collects URLs to be crawled from target websites. The collected URLs are stored in the database. The task configuration module generates crawling tasks based on the URLs stored in the database and determines the time constraints for each task. Furthermore, the task configuration module predicts the amount of data to be crawled. The bandwidth allocation module is used to allocate bandwidths for the web crawlers and schedule their execution. Crawlers collect specific information from websites based on results from the task configuration module and the bandwidth allocation module.

In this web crawler system, we assume that users wish to collect information from n websites. The bandwidth in the system is limited. Moreover, owing to system constraints, the maximum number of web crawlers that can be executed simultaneously in the system is m. Two situations exist that consider different relations between n and m. When $n \leq m$, we name it *instant allocation*, where all the crawlers can be executed simultaneously and only bandwidth allocation must be considered. When $n > m$, we name it *task scheduling*, where in addition to bandwidth allocation, the sequence of websites crawled is important. We further formulate the problem in these two situations as follows.

3.1 Instant allocation

In this situation, a web crawler collects information in each website. The information in each website must be

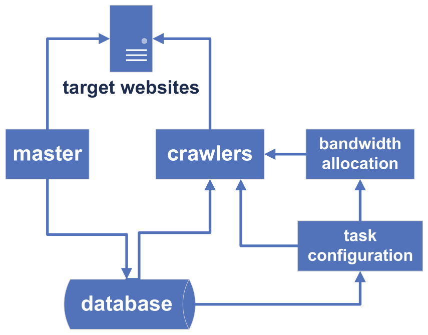

Fig. 1 System Model of the Web Crawler System

collected in a certain time limit. Furthermore, the bandwidth to these web crawlers must be allocated appropriately, and the execution time of the crawling tasks subject to the time constraints must be minimized. The formal formulation is as follows:

Given

1) n websites exist whose information must be collected.

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

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

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

5) The maximum number of crawlers that can be executed simultaneously is m $(m \geqslant n)$.

determine the bandwidth $v_i (i = 1, 2, \dots n)$ for each website such that

1) $\min \max\, t_i^{end}$ or

2) $\min \sum_{i=1}^{n} t_i^{end}$,

where t_i^{end} is the end time of crawling website i.

subject to

1) $t_j^{end} \le t_j^* (j = 1, 2, 3 \cdots n)$

2) $\sum v_i \le U$

We defined two execution time objectives for this problem. One is to minimize the maximum complete time for all websites, and the other is to minimize the sum of the execution times of all websites. The first objective aims to finish all crawling tasks as soon as possible, whereas the second one aims to minimize the total system resource consumption. Both of them are typical in various applications, and we will investigate them in this study.

This formulation requires all the time constraints of the crawling tasks to be satisfied. This can be realized if a sufficient bandwidth is available. We define the minimum bandwidth required to satisfy the time constraints of all websites $\bar{U}$ as follows:

$$\bar{U} = \sum_{i=1}^{n} \frac{D_i}{t_i^*} \qquad (t_i^* > 0). \tag{1}$$

When $\bar{U} \le U$, all websites can be crawled within the time constraints. We name this situation *sufficient bandwidth instant allocation*. When $\bar{U} > U$, one or more websites cannot be crawled within the limit constraints. We name this situation *insufficient bandwidth instant allocation*. A sufficient bandwidth instant allocation problem can be formulated as described above. For the insufficient bandwidth instant allocation, the problem formulation must be modified.

The crawler system collects information for users. If it cannot collect all information within the users' time constraints, some losses will be incurred based on the amount of time exceeding the time constraints. Therefore, we define a metric to evaluate the extent at which the time constraints were satisfied as follows:

$$\triangle t_i = \begin{cases} t_i^{end} - t_i^* & t_i^{end} > t_i^* \\ 0 & t_i^{end} \le t_i^* \end{cases} \qquad (i = 1, 2, 3 \cdots n). \tag{2}$$

Our objective is modified to minimize $\sum_{i=1}^{n} \triangle t_i$. In addition, the second constraint of the problem formulation is not required in this problem. The remainder of the problem formulation is the same as that defined previously.

3.2 Task scheduling

When the number of websites to be crawled is greater than the maximum number of crawlers that the system can execute simultaneously, the sequence of the websites's crawling must be scheduled. In other words, in addition to the bandwidth allocated to the websites, the start time of crawling the websites should be determined. Similar to instant allocation, we aim to minimize the execution time if the time constraints can be satisfied, or the time exceeding the time constraints if the time constraints cannot be satisfied. We used their weighted sum as the objective of the task scheduling problem. We formally formulate the problem as follows:

Given the same conditions as those of the instant allocation except that n is not required to be less than m,

determine the bandwidth $v_i (i = 1, 2, \dots n)$ and start time $t_i^0 (i = 1, 2, \dots n)$ for each website such that

1) $\min(\ln \max t_i^{end} + \ln \sum_{i=1}^{n} \triangle t_i)$ **or**

2) $\min(\ln \sum_{i=1}^{n} (t_i^{end} - t_i^0) + \ln \sum_{i=1}^{n} \triangle t_i)$,

where t_i^0 is the start time of crawling website i and t_i^{end} is the end time.

subject to

1) $\sum v_{k \in S} \le U$ at any time, where S denotes that the websites are executed concurrently.

Similar to instant allocation, we considered two types of evaluation metrics for the execution time, thereby minimizing the complete time $(\min(\ln \max t_i^{end}))$ and minimizing the sum of execution times $(\min(\ln \sum_{i=1}^{n} (t_i^{end} - t_i^0)))$. The other objective is to evaluate the time that exceeds the time constraints. Instant allocation is a special case of the task scheduling problem, where $n \le m$.

For convenience, we summarize all the notations in this study in Table 1.

4 Solution

Based on the problem formulated in the previous section, we propose a solution for sufficient bandwidth instant allocation, insufficient bandwidth instant allocation, and task scheduling. We will illustrate the relationship between these problems and the solutions.

Table 1 Notations used in this study

Symbol	Meaning
n	Number of websites to be collected
U	Maximum available bandwidth of web crawler system
m	Maximum number of crawlers that the system can execute simultaneously
D_i	Amount of data of website i
t_i^*	Time constraint of website i, where 0 denotes no time constraint
v_i	Allocated bandwidth of website i
t_i^0	Start time of website i's crawling
t_i^{end}	End time of website i's crawling
t_i	Execution time duration of website i's crawling
$\bar{U}$	Minimum bandwidth required to satisfy time constraints of all websites

4.1 Sufficient bandwidth instant allocation

As illustrated in Sect. 3, the two objectives for sufficient bandwidth instant allocation are to minimize the complete time and minimize the sum of execution times. Because all websites' crawling begins simultaneously, without loss of generality, we assume that they start at time unit 0. The bandwidth for each website must be determined. Their solutions are described as follows.

4.1.1 Minimizing the maximum complete time

First, we consider the objective min max t_i^{end} for a situation without time constraints. In this situation, for website i, we allocate bandwidth v_i to be proportional to its data amount D_i. All web crawlers will finish their crawling tasks at the time t', which is expressed as

$$t' = \frac{\Sigma_{i=1}^{n} D_i}{U}. \tag{3}$$

Subsequently, we consider the time constraints $t_i^*(i = 1, 2, \dots n)$. A greedy algorithm is proposed to solve this problem. We sort the tasks by t_i^* in the ascending order. The average complete time of the unallocated websites $\bar{t}$ is computed as

$$\bar{t} = \frac{\Sigma D_i(i \in remain)}{U_{remain}}, \tag{4}$$

where *remain* denotes the unallocated websites, and U_{remain} is the remaining bandwidth that can be allocated. The smallest $t_i^* \neq 0$ is compared with $\bar{t}$. If $t_i^* > \bar{t}$, the information of all the remaining websites can be downloaded before their time limits. Therefore, all the remaining websites are assigned a bandwidth proportional to their data amounts. Otherwise, the website i under consideration is assigned the bandwidth $v_i = \frac{D_i}{t_i^*}$. This is the minimum bandwidth required to satisfy the time constraint, and the maximal remaining bandwidth is used for other websites. This process is repeated until all websites are considered. Bandwidth allocation fails if the unallocated bandwidth is less than or equal to 0. The detailed algorithm is shown in Algorithm 1 where L contains all websites.

```
Algorithm 1: Minimize the maximum complete time for sufficient bandwidth instant allocation (M-MAXT)
Input: $L$, $U$
Output: bandwidth $v_i$ for website
        $i(i = 1, 2...n)$
1  $T_{remain}$= $L$, $U_{remain}$= $U$
2  sort websites in $T_{remain}$ by their $t^*$ in the
     ascending order
3  calculate the average complete time $\bar{t}$ of all
     websites in $T_{remain}$ using Eq.(4)
4  while $\exists t_j^* < \bar{t}$ and $t_j^* \neq 0$ do
5      get the first website $i$ in $T_{remain}$ such that
         $t_i^* < \bar{t}$
6      allocate bandwidth for website $i$ using $\frac{D_i}{t_i^*}$
7      $U_{remain} = U_{remain} - \frac{D_i}{t_i^*}$
8      remove $i$ from $T_{remain}$
9      update $\bar{t}$ of the websites in $T_{remain}$ using
         Eq.(4)
10 end
11 foreach website $i$ in $T_{remain}$ do
12     allocate $v_i$ as $\frac{D_i}{\bar{t}}$
13 end
```

4.1.2 Minimizing sum of execution times

Next, we consider the objective min $\sum_{i=1}^{n} t_i^{end}$. Similarly, we consider the unconstrained situation first. Because $t_i = \frac{D_i}{v_i}$, according to the Cauchy inequality, we have

$$\left(\sum \frac{D_i}{v_i}\right)\left(\sum v_i\right) \geq \left(\sum \sqrt{D_i}\right)^2. \tag{5}$$

The equal condition holds when

$$\forall i, \frac{D_i}{v_i^2} = \lambda, \tag{6}$$

where λ is a constant. Because $\sum v_i = U$, we have

$$v_i = \frac{U\sqrt{D_i}}{\sum \sqrt{D_i}}. \tag{7}$$

Next, we consider a situation with n time constraints. According to the Lagrange multiplier method, we have the following Lagrange function:

$$L(v, \lambda_i, \mu) = \sum_{i=1}^{n} \frac{D_i}{v_i} + \lambda_i g_i(v) + \mu h(v), \tag{8}$$

$$g_i(v) = \frac{D_i}{v_i} - t_i^*, h(v) = \sum_{i=1}^{n} v_i - U, \tag{9}$$

where $v = (v_1, \dots, v_n)$, $\lambda_i(i = 1, 2 \dots n)$ and μ are the Lagrange multiplier of the time constraints and total bandwidth constraints, respectively, and $\lambda_i, \mu \geq 0$. Therefore, the optimal solution of the optimization problem must satisfy the following KKT conditions:

$$\begin{array}{ll} \frac{\partial L(\mathrm{v},\lambda_i,\mu)}{\partial v_i}|_{v_i=v_i^*} = 0 & (a) \\ \lambda_i g_i(v^*) = 0 & (b) \\ \mu h(v^*) = 0 & (c) \\ g_i(v^*) \leq 0 & (d) \\ h(v^*) \leq 0 & (e), \end{array} \tag{10}$$

where (a) is the necessary condition for the extreme value of the Lagrange function, (b) and (c) are complementary slackness conditions, and (d) and (e) are inequality constraints.

The solution for the situation contains two cases, namely $\lambda_i > 0$ or $\lambda_i = 0$. When $\lambda_i > 0$ holds, the optimal solution of the function falls outside the feasible domain, and the time constraints apply at this time ($g_i(v^*) = 0$). Therefore, the solution is as follows:

$$v_i = \frac{D_i}{t_i^*}. \tag{11}$$

When $\lambda_i = 0$ holds, all solutions of v_i except $v_i = \frac{D_i}{t_i^*}$ are in the feasible domain. This is a special case of the problem solved at the beginning of this section. The optimal objective is achieved when the bandwidth of all crawlers is assigned proportionally to the amount of data required to be collected. Therefore, we have

$$v_i = \frac{\sqrt{D_i}}{\frac{\sum \sqrt{D_i}}{U_{remain}}}, \tag{12}$$

where U_{remain} is the remaining bandwidth of the unassigned tasks using Eq. (11). The optimal solution to the problem is obtained in these two cases, and the second one is better.

Algorithm 2: Minimize the sum of execution times for sufficient bandwidth instant allocation (M-SUMT)

```
   Input: $L$, $U$
   Output: bandwidth $v_i$ for website
           $i(i = 1, 2...n)$
1  $T_{remain} = L$, $U_{remain} = U$
2  sort websites in $T_{remain}$ by their $t^*$ in the
    ascending order
3  $preAllocate(T_{remain})$
4  while $\exists t_i > t_i^*$ and $t_i^* \neq 0$ do
5      get the first website $i$ in $T_{remain}$ such that
        $t_i^* < t_i$
6      allocate bandwidth for website $i$ using
        $v_i = \frac{D_i}{t_i^*}$
7      $U_{remain} = U_{remain} - v_i$
8      remove $i$ from $T_{remain}$
9      update $t_i$ of the websites in $T_{remain}$ using
        preAllocate($T_{remain}$)
10 end
11 foreach website $i$ in $T_{remain}$ do
12     calculate bandwidth $v_i$ by Eq.(12)
13 end
14 Function: preAllocate(List $T_{remain}$)
15     foreach website $i$ in $T_{remain}$ do
16         calculate preallocated $v_i^{pre}$ using Eq.(12)
17         obtain $t_i$ by $\frac{U_i}{v_i^{pre}}$
18     end
19 end
```

Therefore, we propose the following approach to allocate the bandwidth. U_{remain} is initialized as U. We first perform a pre-allocation of bandwidth U_{remain} for all websites according to Eq. (12). We sort the websites by their deadlines in the ascending order and consider them individually. If a website's pre-allocated bandwidth cannot satisfy the time constraint, it adopts the solution obtained using Eq. (11). Subsequently, the allocated bandwidth is reduced from U_{remain}. If U_{remain} is updated, the preallocation should be performed again for all the unallocated websites. This process is repeated until all the remaining websites' pre-allocated bandwidths satisfy their time constraints. The allocation fails if U_{remain} is less than or equal to 0 or U_{remain} is insufficient to support a website with time constraints. The detailed approach is presented in Algorithm 2.

To verify the performance of our proposed algorithms, the bandwidth allocation approach proposed by (Fei et al. 2010), i.e., *PROBandwidth* can serve as a benchmark approach. For website $j (j = 1, 2, \dots n)$, we compute the required bandwidth to satisfy its time constraints as

$$\bar{v}_j = \frac{D_j}{t_j^*}. \tag{13}$$

Subsequently, we allocate v_j, which is proportional to $\bar{v}_j$ as follows:

$$v_j = \frac{D_j}{\sum_{j=1}^{n} \bar{v}_j} U. \tag{14}$$

By combining Eqs. (13) and (14), the time t_j for crawling website j can be calculated as follows:

$$t_j = \frac{D_j}{v_j} = \frac{\sum_{j=1}^{n} \frac{D_i}{t_i^*}}{U} t_i^*. \tag{15}$$

By assuming a sufficient bandwidth, we have $\frac{\sum_{j=1}^{n} \frac{D_i}{t_i^*}}{U} \leq 1$. Therefore we have $t_j \leq t_j^* (j = 1, 2, \dots n)$, which guarantees that all the time constraints are satisfied.

The second benchmark approach is referred to as *Random*. In this approach, we first allocate the bandwidth for each website based on Eq. (13). This guarantees that all the time constraints are satisfied. The bandwidth already allocated is denoted by $\bar{U}$. Subsequently, the remaining bandwidths $U_{remain} = U - \bar{U}$ are randomly allocated to the websites.

4.2 Insufficient bandwidth instant allocation

We further analyze instant allocation with insufficient bandwidth. Because the bandwidth is insufficient, all the constraints cannot be satisfied. Therefore, we attempt to minimize the time exceeding time constraints, as shown in Eq. (2). The optimal solution will be achieved when $t_i (i = 1, 2, \dots n)$ is close to t_i^*. The problem is to minimize $\sum_{i=1}^{n} t_i$ subject to $t_i \geq t_i^*$. The formulation is similar to the sufficient bandwidth instant allocation with the objective of minimizing the sum of execution times, except that $t_i \geq t_i^*$. We can revise the solution to this problem.

In our solution, the websites are stored in two queues, i.e., normal and constrained queues. The bandwidths of the websites in the normal and constrained queues are assigned using Eqs. (11) and (12), respectively. The websites were initially in the normal queue, and they were placed in the constrained queue only when their assigned bandwidths resulted in end times that exceeded the time constraints. We modified the condition in which the assigned bandwidth resulted in an end time that was less than the time constraints. We only changed line 4 and 5 of Algorithm 2 and we named the revised algorithm M-STET.

Algorithm 3: Minimize complete time exceeding time constraints for insufficient bandwidth instant allocation (M-STET)

```
Input: $L$, $U$
Output: bandwidth $v_i$ for website $i (i=1,2...n)$
1  $T_{remain}$= $L$, $U_{remain}$= $U$
2  sort websites in $T_{remain}$ by their $t^*$ in the ascending order
3  preAllocate($T_{remain}$)
4  while $\exists t_i < t_i^*$ and $t_i^* \neq 0$ do
5      get the first website $i$ in $T_{remain}$ such that $t_i^* > t_i$
6      allocate bandwidth for website $i$ using $v_i = \frac{D_i}{t_i^*}$
7      $U_{remain} = U_{remain} - v_i$
8      remove $i$ from $T_{remain}$
9      update $t_i$ of the websites in $T_{remain}$ using preAllocate($T_{remain}$)
10 end
11 foreach website $i$ in $T_{remain}$ do
12     calculate bandwidth $v_i$ by Eq.(12)
13 end
14 Function: preAllocate(List $T_{remain}$)
15     foreach website $i$ in $T_{remain}$ do
16         calculate preallocated $v_i^{pre}$ using Eq.(12)
17         obtain $t_i$ by $\frac{U_i}{v_i^{pre}}$
18     end
19 end
```

To verify the performance of our proposed algorithms, we introduce a benchmark approach, i.e., PRODataAmount. For website $j (j = 1, 2, \dots n)$ with data amount D_j, we allocate v_i proportional to their data amount D_i. The bandwidth v_j can be calculated as follows:

$$v_j = \frac{D_j}{\sum_{j=1}^{n} D_j} U. \tag{16}$$

4.3 Task scheduling

When the number of websites to be crawled n is greater than the maximum number of crawlers that the system can execute simultaneously m, we consider the bandwidth allocation a task scheduling problem. In this problem, the start time and the duration of the websites's crawling tasks must be determined. Similar to instant allocation, we aim to minimize the execution time of the websites's crawling if the time constraints can be satisfied or the time exceeding the time constraints if the time constraints cannot be satisfied. As illustrated in Sect. 3, the objectives are as follows:

$$\min(\ln \max t_i^{end} + \ln \textstyle\sum_{i=1}^{n} \triangle t_i) \text{ or}$$
$$\min(\ln \textstyle\sum_{i=1}^{n}(t_i^{end} - t_i^0) + \ln \sum_{i=1}^{n} \triangle t_i)$$

We proposed a round-based allocation approach for this problem, the details of which are as follows. We believe that websites with loose time constraints should be crawled earlier than those with tight time constraints. We can place all websites to be crawled to a priority queue, and a website with a tighter time constraint is assigned a higher priority. The first m websites are removed from the queue, and their bandwidths are allocated using the instant allocation approach. The maximum end time of these websites is denoted by t_{max}, which is the end time of this round. During crawling, once a website's crawling is completed, another website is removed from the queue and its crawling is commenced. When this round is completed at t_{max}, all remaining data and the time constraints of the websites are recomputed. Subsequently, the top m websites that have not completed the crawling compose another round's identification. Their bandwidths are reallocated based on their remaining data and the time constraints. This procedure is repeated until the queue becomes empty. This approach is illustrated in Algorithm 4

Algorithm 4: Schedule crawling for websites (ReAlloc-MAXT/ReAlloc-SUMT)

```
Input: $L$, $U$
Output: bandwidth $v_i$ and start time $t_i^0$ for
        website $i(i = 1, 2...n)$
1  sort websites in $L$ by their $t^*$ in the ascending order
2  pop $m$ websites from $L$ as $Lm$
3  while $L \neq \varnothing$ do
4      $allocateInstantBandwidth(Lm)$
5      obtain the maximum complete time of the websites in $Lm$, $t_{max}$
6      while $t_{max}$ is not reached do
7          if website $i$ is complete then
8              pop a website $j$ from $L$
9              allocate bandwidth for website $j$ using $v_j = v_i$
10             calculate the crawled data of website $j$ before $t_{max}$ using
               $\triangle D_j = v_j \times (t_{max} - t_i^{end})$
11             replace website $i$ with $j$ in $Lm$
12         end
13     end
14     remove all complete websites from $Lm$
15     foreach website $q$ in $Lm$ do
16         calculate the remaining data and time limit of website $q$ by $D_q - \triangle D_q$ and
           $t_q^* - t_{max}$, respectively
17     end
18     pop websites from $L$ and add them into $Lm$ such that $\mid Lm \mid= m$
19 end
20 Function: allocateInstantBandwidth(List $Lm$)
21     calculate $\bar{U}$ of the websites in $Lm$ using Eq.(1)
22     if $\bar{U} \leq U$ then
23         allocate bandwidths for the websites in $Lm$ using M-MAXT or M-SUMT
24     else
25         allocate bandwidths for the websites in $Lm$ using M-STET
26     end
27 end
```

As shown in lines 20–27, this algorithm is based on different instant allocation approaches. For clarity, if M-MAXT and M-STET are used, the algorithm is referred to as ReAlloc-MAXT, whereas if M-SUMT and M-STET are used, the algorithm is referred to as ReAlloc-SUMT.

To verify the performance of the algorithm, we introduce the following benchmarks. First, we used PROBandwidth to replace M-MAXT or M-SUMT (line 23) and M-STET (line 25). The resulting approach is referred to as ReAlloc-PROBandwidth.

The second benchmark approach uses only the bandwidth allocation algorithm for the first m websites, and the remaining websites begin their crawling once some of the running crawling tasks are completed. The bandwidths of the websites are not reallocated. This approach is referred to as AllocOnce

The third benchmark approach is a round-based allocation approach. It first sorts all websites t_i^* in the ascending order. In the first round, the first m websites are allocated with their bandwidths proportional to their data amount. Their start and finish times are the same. Subsequently, the other m websites repeat this process until all websites are considered. The approach is referred to as ReAlloc-PRODataAmount.

5 Experimental results

We performed simulations to evaluate the performances of the proposed algorithms. Multiple web crawlers were executed on a computer, and NetBalancer (Srl 2014) was used to control the bandwidth of each web crawler in the experiment. We first present the results of instant allocation, followed by the results of task scheduling. Finally, some practical issues related to crawling are discussed.

5.1 Sufficient bandwidth instant allocation

We first compare the performances of M-MAXT and M-SUMT with other bandwidth allocation approaches, including Random, PROBandwidth (Fei et al. 2010), genetic algorithm (GA) (Shams et al. 2017), and differential evolution (DE) (Afzal et al. 2019). We developed web crawlers to collect data from six websites, including wiki (http://www.wiki.com), pear video (http://www.pearvideo.com), huanqiu (http://world.huanqiu.com), sinanews (http://news.sina.com.cn), huitu (http://www.huitu.com), and chinanews (http://www.chinanews.com). Their contents include texts, images, and videos, and their data amount satisfy 1:1.3:1.6:1.9:2.2:2.5, which simulates different practical user requirements. We conducted experiments by varying three influencing factors: bandwidth, amount of data to be crawled, and tightness of time constraints.

First, we varied the bandwidth from 800 to 1200 kbps to verify the performances of different approaches. The data amount was set to 125,000 kb, and the time constraints were the same. The results are shown in Fig. 2. It is clear that the maximum complete time and sum of execution times of all approaches decreased with increasing bandwidth. This is

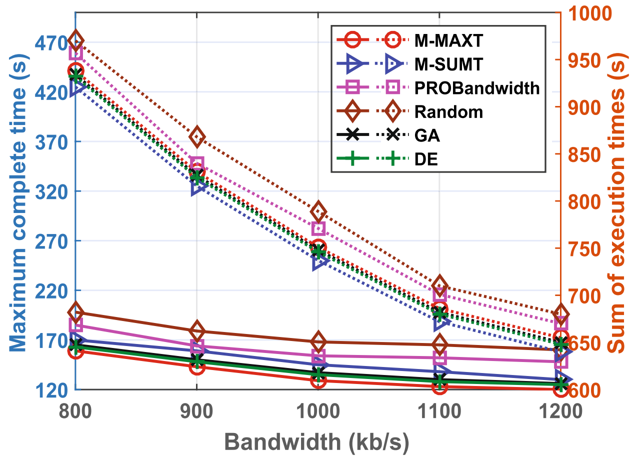

Fig. 2 Maximum complete time (left y-axis)/sum of execution times (right y-axis) vs. bandwidth (sufficient bandwidth instant allocation)

because the execution time of a website is inversely proportional to the bandwidth. Among these approaches, the maximum complete time of M-MAXT was less than that of the other approaches, whereas the sum of execution times of M-SUMT was less than that of the other approaches. This is consistent with our analysis. PROBandwidth allocated the bandwidth based on D_j/t_j^* for website j. Therefore, some websites completed crawling much earlier than their time constraints. The unutilized time constraint does not contribute to the objective but consumes the bandwidth to be used for other websites. This causes a longer crawling time and execution time for these websites. The random approach randomly allocates the remaining bandwidths after allocating the minimal required bandwidth for each website. Some websites may have small bandwidths and hence long execution times. The GA and DE performed better than PROBandwidth and Random but were affected by local optimal solutions; therefore, they could not easily achieve the minimum execution time or complete time. M-MAXT achieved the least maximum complete time because of its explicit optimization of it. After the allocation of bandwidths to all tight time constraints using D_j/t_j^* for website j, other websites shared the remaining bandwidth and ended simultaneously, thereby resulting in the least maximum complete time. M-SUMT achieved the minimal sum of execution times by allocating the bandwidths using Eq. (12). However, it did not require the websites to complete simultaneously; hence, the maximum complete time may be long.

Subsequently, we varied the amount of data to verify the performance of different approaches. The bandwidth was set to 1000 kbps, and the time constraints were set randomly such that the $\bar{U}$ of these websites was less than U. We observed that the amount of data of webpages between different websites differed, but the amount of data of webpages within the same website was similar except for some random changes. Therefore, we increased the number of webpages to be crawled to achieve a proportional increase in the data amount for all websites. The results are shown in Fig. 3. It is clear that the maximum complete time and sum of execution times of all five approaches increased with the data amount. The increase in the data amount was in fact is equivalent to the increase in the required bandwidth. Similar to the analysis when varying the bandwidth, M-MAXT and M-SUMT achieved the best performances in terms of the maximum complete time and the sum of execution times, respectively.

Finally, we varied the tightness of the time constraints to evaluate the performances of the approaches. The data amount and bandwidth were set to 125000 kb and 1000 kbps, respectively. The tightness of the time constraints was defined by $\sum \frac{D_i}{t_i^*}$ $(t_i^* \neq 0)$. A greater value of obtained using this formula denotes a tighter time constraint. The results are shown in Fig. 4. The maximum complete time and sum of execution times of all approaches increased with the tightness of the time constraints. Moreover, the increase became more evident when the tightness of the time constraints exceeded 965. This is because the increase in time constraint tightness caused the bandwidths used for other websites to decrease, and hence the required time (maximum complete time and sum of execution times) increased.

5.2 Insufficient bandwidth instant allocation

Subsequently, we evaluated the performance of the proposed approach in insufficient bandwidth instant allocation. We compared the performance of M-STET with those of PRODataAmount, PROBandwidth (Fei et al. 2010), GA (Shams et al. 2017), and DE (Afzal et al. 2019). The configurations

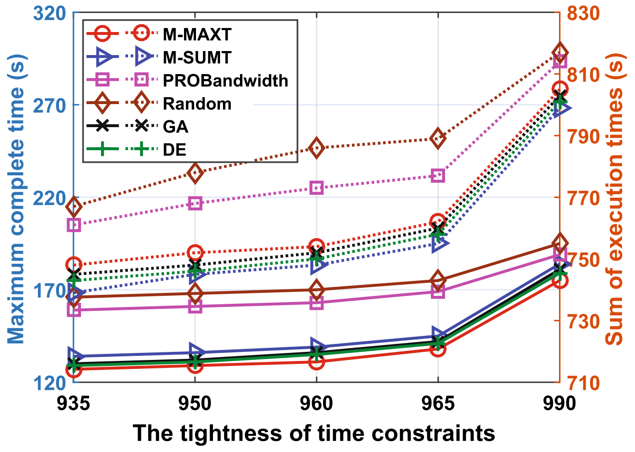

Fig. 4 Maximum complete time (left y-axis)/sum of execution times (right y-axis) vs. tightness of time constraints (sufficient bandwidth instant allocation)

of the simulations were similar to that in the sufficient bandwidth instant allocation. However, we gradually loosened the time constraints of the websites until $\bar{U} > U$ [see Eq. (12)]. We conducted experiments by varying three influencing factors: bandwidth, amount of data to be crawled, and tightness of time constraints. The results are shown in Figs. 5, 6, and 7, respectively.

As shown, M-STET outperformed PROBandwidth and PRODataAmount in these simulations. This is because in PROBandwidth and PRODataAmount, some websites's execution times were less than their time constraints, thereby causing the available bandwidths of the remaining websites to decrease and their execution times to increase. Therefore, the sum of complete times exceeding the time constraints increased. GA and DE performed better than PROBandwidth and PRODataAmount; however, the time constraints

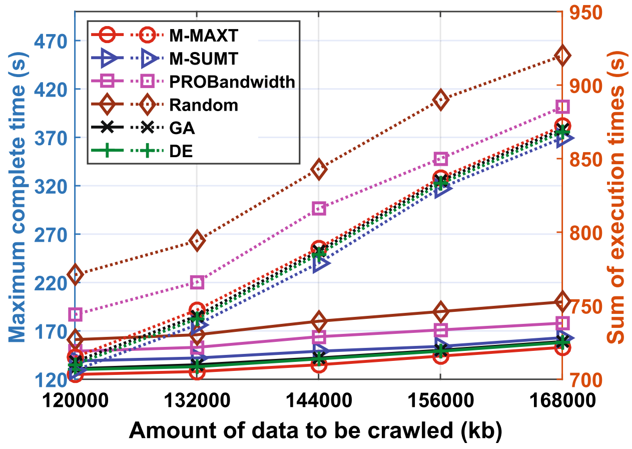

Fig. 3 Maximum complete time (left y-axis)/sum of execution times (right y-axis) vs. amount of data to be crawled (sufficient bandwidth instant allocation)

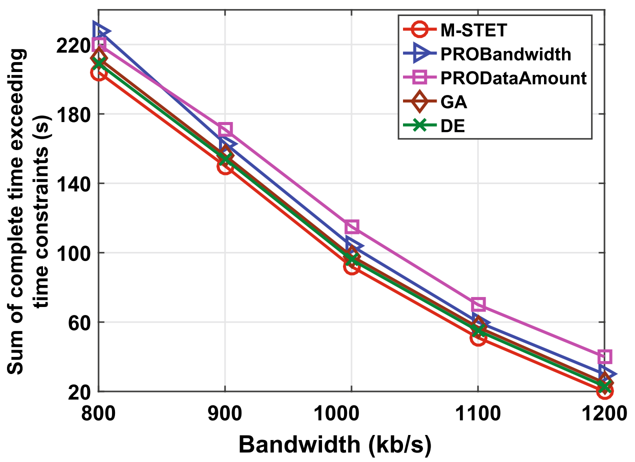

Fig. 5 Sum of complete time exceeding time constraints vs. bandwidth (insufficient bandwidth instant allocation)

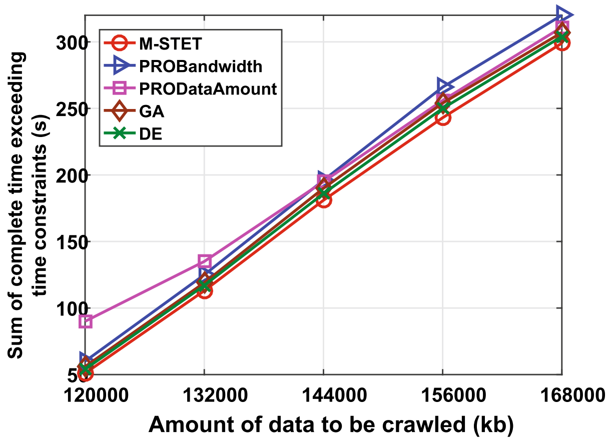

Fig. 6 Sum of complete time exceeding time constraints vs. amount of data to be crawled (insufficient bandwidth instant allocation)

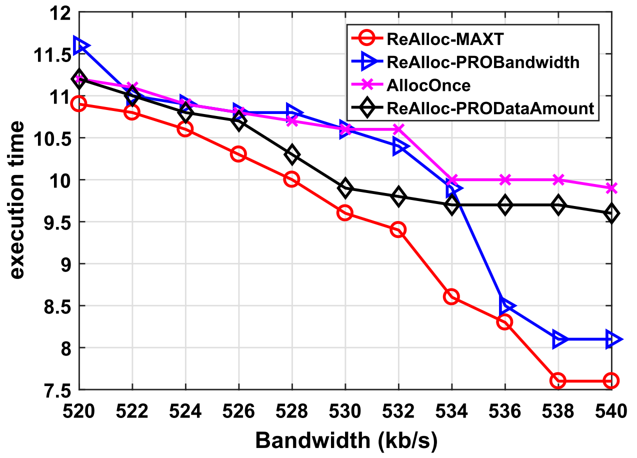

Fig. 8 Execution time measured by $\ln \max t_i + \ln \sum_{i=1}^{n} \triangle t_i$ vs. bandwidth (task scheduling)

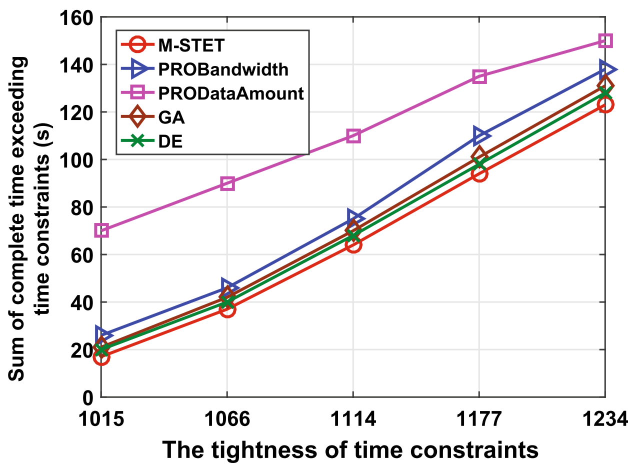

Fig. 7 Sum of complete time exceeding time constraints vs. tightness of time constraints (insufficient bandwidth instant allocation)

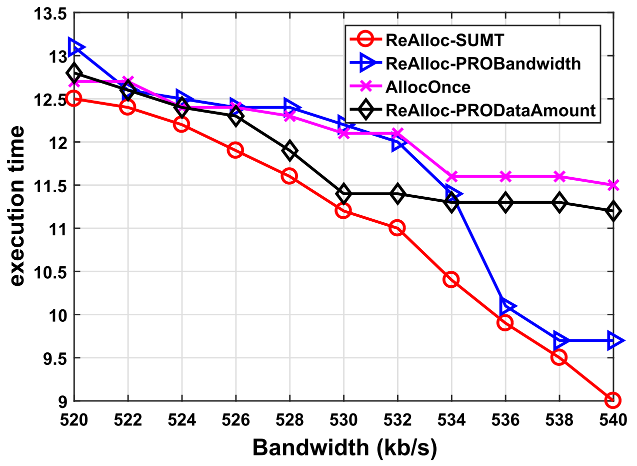

Fig. 9 Execution time measured by $\ln \sum_{i=1}^{n}(t_i - t_i^0) + \ln \sum_{i=1}^{n} \triangle t_i$ vs. bandwidth (task scheduling)

were not fully utilized. In M-STET, the execution times of all websites were greater than or equal to the time constraints, rendering the execution times of the websites to be similar to the time constraints; hence, the time constraints were fully utilized. Therefore, M-STET achieved the least sum of complete time that exceeded the time constraints.

5.3 Task scheduling

Next, we consider the method to allocate the bandwidth for many websites to be crawled and evaluate the performances of our proposed approaches. We consider a scenario involving 50 websites, in which the maximum number of crawlers that can be executed simultaneously is 5. Simulations were conducted to verify the performances of our proposed ReAlloc-MAXT and ReAlloc-SUMT; subsequently, they were compared with those of ReAlloc-PROBandwidth, Alloc-Once, and ReAlloc-PRODataAmount.

First, we varied the bandwidth from 520 to 540 kbps to verify the performances of different approaches. The result of the maximum complete time is shown in Fig. 8, and the sum of execution times is shown in Fig. 9. The trends in these two figures are similar. As shown, the execution times of ReAlloc-MAXT and ReAlloc-SUMT decreased almost linearly as the bandwidth increased. The execution times of ReAlloc-PROBandwidth decreased slowly first and rapidly when the bandwidth exceeded 534 kb/s. The execution times of AllocOnce and ReAlloc-PRODataAmount decreased relatively slower than those of the approaches above. ReAlloc-MAXT and ReAlloc-SUMT always outperformed their benchmark approaches. In this situation, the objective include two parts: the maximum complete time (or the sum of execution times) and the complete time exceeding time constraints. ReAlloc-MAXT and ReAlloc-SUMT optimized these objectives in every reallocation. They used multiple reallocations to achieve shorter execution times compared with other approaches. AllocOnce did not perform bandwidth reallocation, and its execution time was almost the largest because its initial allocation was not optimal. ReAlloc-PRODataAmount and ReAlloc-PROBandwidth did not perform well because their allocations were not optimal in each round. The performance of ReAlloc-PROBandwidth can be improved only when the available bandwidth is large. These results are consistent with the analysis in Sect. 4.3.

We further verified the complete time exceeding the time constraints and the number of websites that satisfied their time constraints under different bandwidths. Figure 10 shows the results when the available bandwidth varied from 520 to 540 kb/s. As shown, the complete times exceeding the time constraints of all approaches decreased as the bandwidth increased. This is because more rounds existed when the minimum required bandwidth $\bar{U}$ [see Eq. (1)] was satisfied. In addition, the execution times of individual rounds decreased, which caused a decrease in the complete time exceeding the time constraints. ReAlloc-SUMT and ReAlloc-MAXT performed better than the other approaches. This shows the effectiveness of the optimization in these approaches. The results of websites that satisfied their time constraints further verified our analysis above. As shown, more websites satisfied their time constraints as the bandwidth increased. Among the approaches, ReAlloc-SUMT and ReAlloc-MAXT achieved faster satisfaction with time constraints.

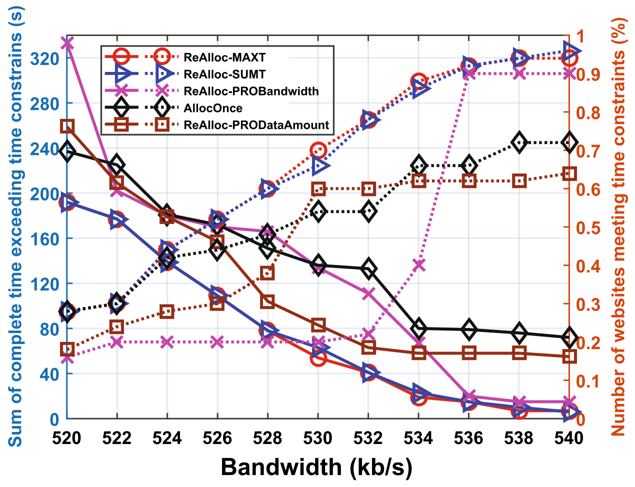

Fig. 10 Sum of complete time exceeding time constrains (left y-axis)/ number of websites satisfying time constraints (right y-axis) vs. bandwidth (task scheduling)

5.4 Practical crawling issues

In the previous sections, we assumed that for website i, $t_i = \frac{D_i}{v_i}$. We further verified the relation among the execution time, bandwidth, and data amount of the crawler. We tested many websites, and the results based on three websites, i.e., wiki (http://www.wiki.com), pear video (http://www.pearvideo.com), and huitu (http://www.huitu.com) are presented herein. Their contents were mainly texts, videos, and images, respectively. First, we varied the bandwidth such that the crawler can verify its execution time, where the data amount was fixed. Subsequently, we set a constant bandwidth for the crawler and recorded the crawling time required for crawling different amounts of data, and the results are shown in Figs. 11 and 12, respectively. According to the figures, the execution time was inversely proportional to the bandwidth

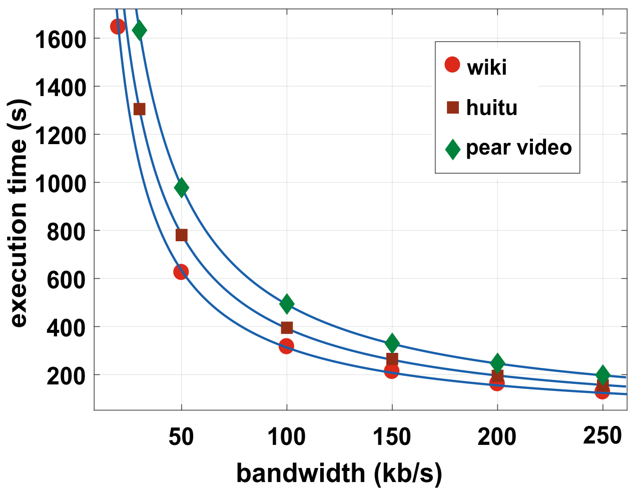

Fig. 11 Execution time of crawler vs. bandwidth

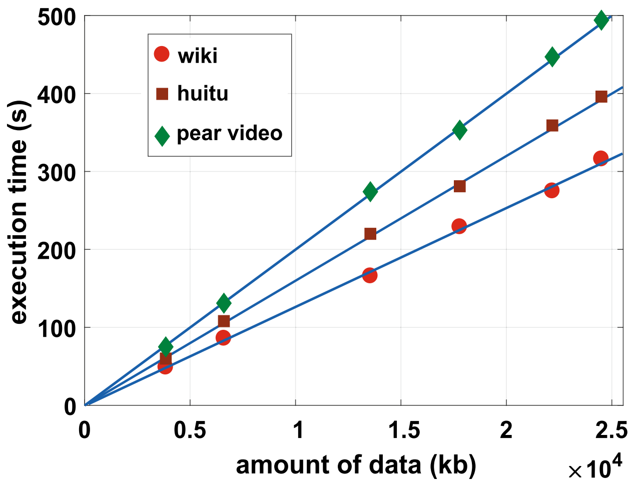

Fig. 12 Execution time of crawler vs. data amount

and proportional to the data amount. However, the relations for different websites differed. We demonstrate this using the following equation,

$$t_i = \frac{\xi_i D_i}{v_i}(i = 1, 2, \dots n), \tag{17}$$

where ξ_i is a constant that differs for websites. We can conduct experiments for the websites to be crawled and obtain their constant. Subsequently, we can use $\xi_i D_i$ to replace D_i, which renders the proposed approaches adapted to real situations.

6 Conclusion

In this study, we investigated the bandwidth allocation problem for a web crawler system with time constraints. We formulated the problem in different situations and with different objective functions. Specifically, we first considered the bandwidth allocation of a small number of crawling tasks with sufficient or insufficient bandwidths. For a sufficient bandwidth, the objective can minimize the maximum complete time or minimize the sum of execution times. For an insufficient bandwidth, the objective is based on the time exceeding the time constraints. We further extended our study to the scheduling of large-scale crawling tasks. The sequence and bandwidth of the crawling tasks were determined. We performed extensive simulations to verify these approaches. The results demonstrated that our approach satisfactorily satisfied the time constraints and achieved desirable performances in terms of corresponding objectives in various scenarios.

Acknowledgements This research is supported in part by National Key R&D Program of China no. 2018YFC1604000.

Compliance with ethical standards

Conflict of interest The authors declare that there is no conflict of interest regarding the publication of this paper.

Ethical approval This article dose not contain any studies with human participants or animals performed by any of the authors.

Informed consent Informed consent was obtained from all individual participants included in the study.

References

Afzal Z, Shah PA, Awan KM, ur Rehman Z, (2019) Optimum bandwidth allocation in wireless networks using differential evolution. J Ambient Intell Humaniz Comput 10(4):1401–1412. https://doi.org/10.1007/s12652-018-0858-4

Amamou A, Bourguiba M, Haddadou K, Pujolle G (2012) A dynamic bandwidth allocator for virtual machines in a cloud environment. In: Proceeding of 2012 IEEE consumer communications and networking conference (CCNC), Las Vegas, NV, USA, January 14–17, 2012, pp 99–104. https://doi.org/10.1109/CCNC.2012.6181065

Arasu A, Cho J, Garcia-Molina H, Paepcke A, Raghavan S (2001) Searching the web. ACM Trans Internet Technol 1(1):2–43. https://doi.org/10.1145/383034.383035

Cai R, Yang JM, Lai W, Wang Y, Zhang L (2008) Irobot: an intelligent crawler for web forums. In: Proceeding of the 17th international conference on world wide web (WWW), ACM, New York, NY, USA, pp 447–456. https://doi.org/10.1145/1367497.1367558

Castillo C, Marin M, Rodriguez A, Baeza-Yates R (2004) Scheduling algorithms for web crawling. In: Proceeding of joint conference 10th Brazilian symposium on multimedia and the web & 2nd Latin American Web Congress (WebMedia & LA-Web), 12–15 October 2004, Ribeirao Preto, SP, Brazil, pp 10–17. https://doi.org/10.1109/WEBMED.2004.1348139

Das S, Khatua M, Misra S (2019) Cheating-resilient bandwidth distribution in mobile cloud computing. IEEE Trans Cloud Comput 7(2):469–482. https://doi.org/10.1109/TCC.2016.2638909

Diligenti M, Maggini M, Pucci FM, Scarselli F (2004) Design of a crawler with bounded bandwidth. In: Proceeding of the 13th international conference on world wide web (WWW), May 17–20, 2004, ACM, New York, NY, USA, pp 292–293. https://doi.org/10.1145/1013367.1013441

Ding R, Wang M (2018) Design and implementation of web crawler based on coroutine model. In: Proceeding of cloud computing and security—4th international conference (ICCCS), Haikou, China, June 8–10, 2018. Springer International Publishing, Cham, pp 427–435. https://doi.org/10.1007/978-3-030-00006-6

Edwards J, McCurley K, Tomlin J (2001) An adaptive model for optimizing performance of an incremental web crawler. In: Proceeding of the 10th international conference on world wide web (WWW), ACM, New York, NY, USA, pp 106–113. https://doi.org/10.1145/371920.371960

Fei R, Yang K, Ou S (2010) A qos-aware dynamic bandwidth allocation algorithm for relay stations in ieee 802.16j-based vehicular networks. In: Proceeding of 2010 IEEE wireless communications and networking conference (WCNC), Proceedings, Sydney, Australia, 18–21 April 2010, pp 1–6. https://doi.org/10.1109/WCNC.2010.5506376

Ge D, Ding Z (2014) A task scheduling strategy based on weighted round-robin for distributed crawler. In: Proceeding of the 7th IEEE/ACM international conference on utility and cloud computing (UCC), London, United Kingdom, December 8–11, 2014, pp 848–852. https://doi.org/10.1109/UCC.2014.138

Guojun Z, Wenchao J, Jihui S, Fan S, Hao Z, Jiang L (2017) Design and application of intelligent dynamic crawler for web data mining. In: Proceeding of the 32nd youth academic annual conference of Chinese Association of Automation (YAC), pp 1098–1105. https://doi.org/10.1109/YAC.2017.7967575

Jiang L, Fu M, Le Z (2011) Hierarchical QOS-aware dynamic bandwidth allocation algorithm for wireless optical broadband access network. In: Proceeding of 2011 international conference on electronics, communications and control (ICECC), pp 4329–4332. https://doi.org/10.1109/ICECC.2011.6066634

Kausar MdA, Dhaka SKSVS (2013) Web crawler based on mobile agent and java aglets. Int J Inf Technol Comput Sci 5(10):85–91. https://doi.org/10.5815/ijitcs.2013.10.09

Kc M, Hagenbuchner M, Tsoi AC (2008) A scalable lightweight distributed crawler for crawling with limited resources. In: Proceeding of the 2008 IEEE/WIC/ACM international conference on web intelligence and international conference on intelligent agent technology—workshops, 9–12 December 2008, Sydney, NSW, Australia, vol 3, pp 663–666. https://doi.org/10.1109/WIIAT.2008.234

Kumar M, Bhatia R, Rattan D (2017) A survey of web crawlers for information retrieval. Wiley Interdiscip Rev Data Min Knowl Discov 7(6):e1218. https://doi.org/10.1002/widm.1218

Lau CH, Tao X, Tjondronegoro D, Li Y (2012) Retrieving information from microblog using pattern mining and relevance feedback. In: Proceedings of data and knowledge engineering. Springer, Berlin, pp 152–160. https://doi.org/10.1007/978-3-642-34679-8_15

Meng S, Wang Y, Miao Z, Sun K (2018) Joint optimization of wireless bandwidth and computing resource in cloudlet-based mobile cloud computing environment. Peer-to-Peer Netw Appl 11(3):462–472. https://doi.org/10.1007/s12083-017-0544-x

Punj D, Dixit A (2017) Design of a migrating crawler based on a novel URL scheduling mechanism using AHP. Int J Rough Sets Data Anal 4(1):95–110. https://doi.org/10.4018/IJRSDA.2017010106

Raina S, Prakash Agarwal A (2014) How crawlers aid regression testing in web applications: the state of the art. Int J Comput Appl 68(14):33–38. https://doi.org/10.5120/11651-7153

Shams R, Khan FH, Abbass S, Javaid R (2017) Bandwidth allocation for wireless cellular network by using genetic algorithm. Wirel Pers Commun 95(2):245–260. https://doi.org/10.1007/s11277-016-3890-8

Shkapenyuk V, Suel T (2002) Design and implementation of a high-performance distributed web crawler. In: Proceeding of the 18th international conference on data engineering, San Jose, CA, USA, February 26–March 1, 2002, pp 357–368. https://doi.org/10.1109/ICDE.2002.994750

Singhal N, Agarwal RP, Dixit A, Sharma AK (2011) Information retrieval from the web and application of migrating crawler. In: Proceeding of the 2011 international conference on computational intelligence and communication networks, pp 476–480. https://doi.org/10.1109/CICN.2011.99

Srl S (2014) Netbalancer. http://netbalancer.com/. Accessed 25 June 2020

Thelwall M (2001) A web crawler design for data mining. J Inf Sci 27(5):319–325. https://doi.org/10.1177/016555150102700503

Wang J, Guo Y (2012) Scrapy-based crawling and user-behavior characteristics analysis on taobao. In: Proceeding of 2012 international conference on cyber-enabled distributed computing and knowledge discovery (CyberC), Sanya, China, October 10–12, 2012, pp 44–52. https://doi.org/10.1109/CyberC.2012.17

Wang J, Zhao H, Liu F, Zhang J (2018a) A queue-based bandwidth allocation method for streaming media servers in m-learning VOD systems. In: Proceeding of the 12th international conference, edutainment 2018: e-learning and games, Xi'an, China, June 28–30, 2018, Springer, lecture notes in computer science, vol 11462, pp 107–114. https://doi.org/10.1007/978-3-030-23712-7_15

Wang Y, Hong Z, Shi M (2018b) Research on lDA model algorithm of news-oriented web crawler. In: Proceeding of 17th IEEE/ACIS international conference on computer and information science (ICIS), Singapore, June 6–8, 2018, pp 748–753. https://doi.org/10.1109/ICIS.2018.8466502

Yadav D, Sharma A, Gupta J (2008) Parallel crawler architecture and web page change detection. World Sci Eng Acad Soc Trans Comput 7:929–940. https://doi.org/10.5555/1457973.1457982

Zhu W, Li Y, Xu Y, Cui X (2019) Optimal bandwidth allocation for web crawler systems. In: Proceeding of 2019 IEEE SmartWorld, ubiquitous intelligence & computing, advanced & trusted computing, scalable computing & communications, cloud & big data computing, internet of people and smart city innovation (SmartWorld/SCALCOM/UIC/ATC/CBDCom/IOP/SCI), Leicester, United Kingdom, pp 1146–1153. https://doi.org/10.1109/SmartWorld-UIC-ATC-SCALCOM-IOP-SCI.2019.00215