A Supplementary material for Section 2 (Problem Statement and Modeling)

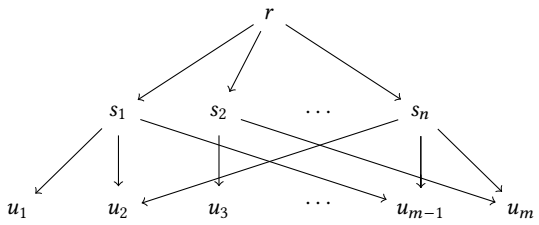


Figure 6: Graphical summarization of the graph G_{sc}

A.1 Graph Crawling Problem

Proposition 4. *Given a website graph $G = (V, E, r, \omega, \lambda)$, a subset $V^* \subseteq V$ and some $B \in \mathbb{R}^+$, determining whether there exists a crawl $T = (V', E')$ of G such that $V^* \subseteq V'$ and $\omega(T) \leqslant B$ is NP-complete; hardness holds even when ω is a constant function. This holds even when nodes in V^* do not have any outgoing links in G.*

Proof. To show NP-completeness, we must show that the problem belongs to NP and is NP-hard.

Let us start with the upper bound. Given a graph $G = (V, E, r, \omega, \lambda)$, we guess a subgraph $T = (V', E')$ of G (which is a polynomial-sized guess). In polynomial time, we check whether T is a r-rooted tree (i.e., whether it is connected, includes r, r has indegree 0 and other nodes indegree 1), we check that V' contains all nodes of V^*, and we check that $\omega(T) \leqslant B$. We accept if and only if these conditions are all satisfied. This yields a nondeterministic polynomial-time algorithm, meaning the problem is in NP.

We now move to the lower bound. Our crawling problem can be seen as a directed variant of the well-known NP-complete *Steiner Tree* [GJ79] problem. NP-hardness of the directed Steiner tree problem is mentioned in the literature (see, e.g., [WW16]), but as it is not formally shown there, we prefer for completeness of the presentation reducing from the set cover problem, a classic NP-hard problem [GJ79]. We denote $\mathcal{U} = \{u_1, \ldots, u_m\}$ a set of m elements called the universe. We also define a collection $\mathcal{S} = \{s_1, \ldots, s_n\}$ of n non-empty subsets, each of them containing some elements of $\mathcal{U}$, such that:

$$\bigcup_{s \in \mathcal{S}} s = \mathcal{U}.$$

In its decision version, the set cover consists in given such a universe and collection, given a natural integer B, determining whether there exists a *cover* $C \subseteq \{s_1, \ldots, s_n\}$ such that $|C| \leqslant B$ and:

$$\bigcup_{s \in C} s = \mathcal{U}.$$

We now propose a polynomial-time many-one reduction of the set cover problem to an instance of the graph crawling problem. We create a website graph $G_{\text{sc}} = (V_{\text{sc}}, E_{\text{sc}}, r, \omega, \lambda)$ as follows. We set V_{sc} to be $\{u_1, \ldots, u_m, r, s_1, \ldots, s_n\}$, including representations for every element of the universe $\mathcal{U}$, every set of the collection $\mathcal{S}$, as well as a distinct root r (by abuse of notation, we do not distinguish between elements of $\mathcal{U}$, $\mathcal{S}$ and the way they are represented in V_{sc}). We define E_{sc} as $\{(r, s_i) \mid i \in \{1, \ldots, n\}\} \cup \{(s_i, u) \mid u \in s_i, i \in \{1, \ldots, n\}\}$. In other words, in G_{sc} from the origin (root) r, we model each element of $\mathcal{S}$ as a vertex, that can be reached following a dedicated (directed) edge. Finally, for each new vertex $s_i \in \mathcal{S}$, we have as many outgoing edges as there are elements of $\mathcal{U}$ in s_i. Finally, we set ω to be the constant function that assigns cost 1 to every vertex, and λ to be some constant function. The result is a graph in the form of a tree of depth 2, depicted in Figure 6. We fix V^* to be $\mathcal{U}$. Observe that nodes in $\mathcal{U}$ do not have any outgoing links. We state that there exists $C \subseteq \{s_1, \ldots, s_n\}$ such that $|C| \leqslant B$ and $\bigcup_{i \in C} s_i = \mathcal{U}$ if and only if there exists a crawl T_{sc} of G_{sc} containing all elements of V^* and of total cost $\omega(T_{\text{sc}}) \leqslant |\mathcal{U}| + B + 1$.

Let us explain why this reduction is polynomial-time. In set cover, the universe $\mathcal{U}$ can be described by the number m of its elements, with representation size $\Theta(\log m)$. Each set of $\mathcal{S}$ needs to list every element within this set, so a set s_i has representation size $\Theta(\log m \times |s_i|)$. Finally, B has representation size $\Theta(\log B)$. This yields a total input size of $\Theta((\log m)\left(\sum_{i=1}^{n} |s_i| + 1\right) + \log B)$. Note that $\sum_{i=1}^{n} |s_i| \geqslant \max(m, n)$ so this is $\Omega(\max(m, n) \log m + \log B)$. But then, the construction depicted in Figure 6 can clearly be done in time polynomial in m and n (namely, in $O(m \times n)$ in the worst case where every set of the collection contains every element). The reduction is therefore polynomial-time.

We now proceed to show equivalence between the initial problem known to be NP-hard (set cover) and the graph crawling instance presented above. First, suppose that there exists $C \subseteq \{s_1, \ldots, s_n\}$ such that $|C| \leqslant B$ and $\bigcup_{i \in C} s_i = \mathcal{U}$. Then consider the crawl T_{sc} of G_{sc} formed by including r, every element of C using the edge from r to that element, and every edge from an element of C to an element of $\mathcal{U}$. Since C is a cover, this includes all elements of $\mathcal{U}$. The total cost of this crawl $\omega(T_{\text{sc}}) = 1 + |C| + |\mathcal{U}| \leqslant |\mathcal{U}| + B + 1$.

Now suppose that there exists a crawl T_{sc} of G_{sc} of total cost $\omega(T_{\text{sc}}) \leqslant |\mathcal{U}| + B + 1$. Note that by definition of ω, the cost is just the number of nodes in T_{sc}, and this crawl necessarily includes the root r as well as all vertices of $\mathcal{U}$. The remaining $\leqslant B$ vertices are therefore vertices of $\mathcal{S}$. We pose C to be those. Then $|C| \leqslant B$ and since T_{sc} is a crawl, for every $u \in \mathcal{U}$, there exists at least one $s \in C$ such that the edge (s, u) is in T_{sc}, meaning that $u \in s$. We indeed have $\mathcal{U} = \bigcup_{s \in C} s$. □

A.2 Data Acquisition as Graph Crawling

Here is the full list of the 38 MIME types used to identify targets in our implementation:

```
application/csv
application/json
application/msword
application/octet-stream
application/pdf
application/rdf+xml
application/rss+xml
application/vnd.ms-excel
application/vnd.ms-excel.sheet.macroenabled.12
application/vnd.oasis.opendocument.presentation
application/vnd.oasis.opendocument.spreadsheet
application/vnd.oasis.opendocument.text
application/vnd.openxmlformats-officedocument.presentationml.presentation
application/vnd.openxmlformats-officedocument.spreadsheetml.sheet
application/vnd.openxmlformats-officedocument.wordprocessingml.document
application/vnd.openxmlformats-officedocument.wordprocessingml.template
application/vnd.rar
application/x-7z-compressed
application/x-csv
application/x-gtar
application/x-gzip
application/xml
application/x-pdf
application/x-rar-compressed
application/x-tar
application/x-yaml
application/x-zip-compressed
application/yaml
application/zip
application/zip-compressed
text/comma-separated-values
text/csv
text/json
text/plain
text/x-comma-separated-values
text/x-csv
text/x-yaml
text/yaml
```

B Supplementary material for Section 4 (Experimental Results)

B.1 Baselines

We use the following list of keywords as initial input to TRES to train its classifier:

pdf	document	statistics	download CSV
xls	report	article	download PDF
csv	publication	paper	download XLS
tar	dataset	metadata	dataset download
zip	data	fact	attached document
rar	download	download file	official documents
rdf	archive	download document	browse files
json	spreadsheet	available for download	download statistics
doc	table	access data	download article
xml	list	view report	annual report
yaml	resource	get dataset	white paper
txt	annex	data file	technical documentation
tsv	supplement	read more	technical report
ppt	attachment	resource list	raw data
ods	proceedings	get document	metadata file
dta	survey	download pulication	open data
7z	material	document archive	fact sheet
ttl	output	supporting materials	
file	content	export data	

B.2 Comparison with Baselines

In the experiments, we apply filters on MIME types and URL extensions to avoid downloading multimedia content. The MIME types in the blocklist are those for multimedia content, namely `image/*`, `audio/*`, and `video/*`. The associated URL extensions are the following:

.3g2	.asx	.dtshd	.flac	.ief	.m1v	.mjp2	.mpg	.pgm	.raw	.ts	.xif
.3ga	.avi	.dwg	.fli	.jfi	.m2a	.mka	.mpga	.pic	.rgb	.viv	.xpm
.3gp2	.avif	.dxf	.flv	.jfif-tbnl	.m2v	.mkv	.mrw	.pjpg	.rlc	.wav	.xwd
.3gp	.avifs	.ecelp4800	.fpx	.jfif	.m3a	.mmr	.mxu	.png	.rmi	.wax	
.3gpa	.bmp	.ecelp7470	.fst	.jif	.m3u	.mov	.nef	.pnm	.rmp	.wbmp	
.3gpp2	.btif	.ecelp9600	.fvt	.jpe	.m4a	.movie	.npx	.ppm	.rw2	.weba	
.3gpp	.cgm	.eol	.g3	.jpeg	.m4b	.mp2	.oga	.psd	.rwl	.webm	
.aac	.cmx	.erf	.gif	.jpg	.m4p	.mp2a	.ogg	.ptx	.snd	.webp	
.aacp	.cr2	.f4v	.h261	.jpgm	.m4r	.mp3	.ogv	.pya	.spx	.wm	
.adp	.crw	.fbs	.h263	.jpgv	.m4u	.mp4	.opus	.pyv	.sr2	.wma	
.aff	.dcr	.fh4	.h264	.jpm	.m4v	.mp4v	.orf	.qt	.srf	.wmv	
.aif	.djv	.fh5	.heic	.k25	.mdi	.mpa	.pbm	.ra	.svg	.wmx	
.aiff	.djvu	.fh7	.heif	.kar	.mid	.mpe	.pct	.raf	.svgz	.wvx	
.arw	.dng	.fh	.icns	.kdc	.midi	.mpeg	.pcx	.ram	.tif	.x3f	
.asf	.dts	.fhc	.ico	.lvp	.mj2	.mpg4	.pef	.ras	.tiff	.xbm	

Note that URL extension filters are not useful on all websites (on some, URLs don't usually include any extension, see the discussion at the beginning of Section 3.3).

Figure 7 completes Figure 4 with the eight remaining websites plots that could not fit in the paper (see Table 1 for characteristics of websites). Figures 8 to 13 present exhaustive graphical results of the hyper-parameters studies conducted on the 11 fully-crawled websites. Especially, Figures 8 and 9 study the exploration–exploitation coefficient α, Figures 10 and 11 study the choice of n in n-grams used in tag paths vector representation, and Figures 12 and 13 study the similarity threshold θ.

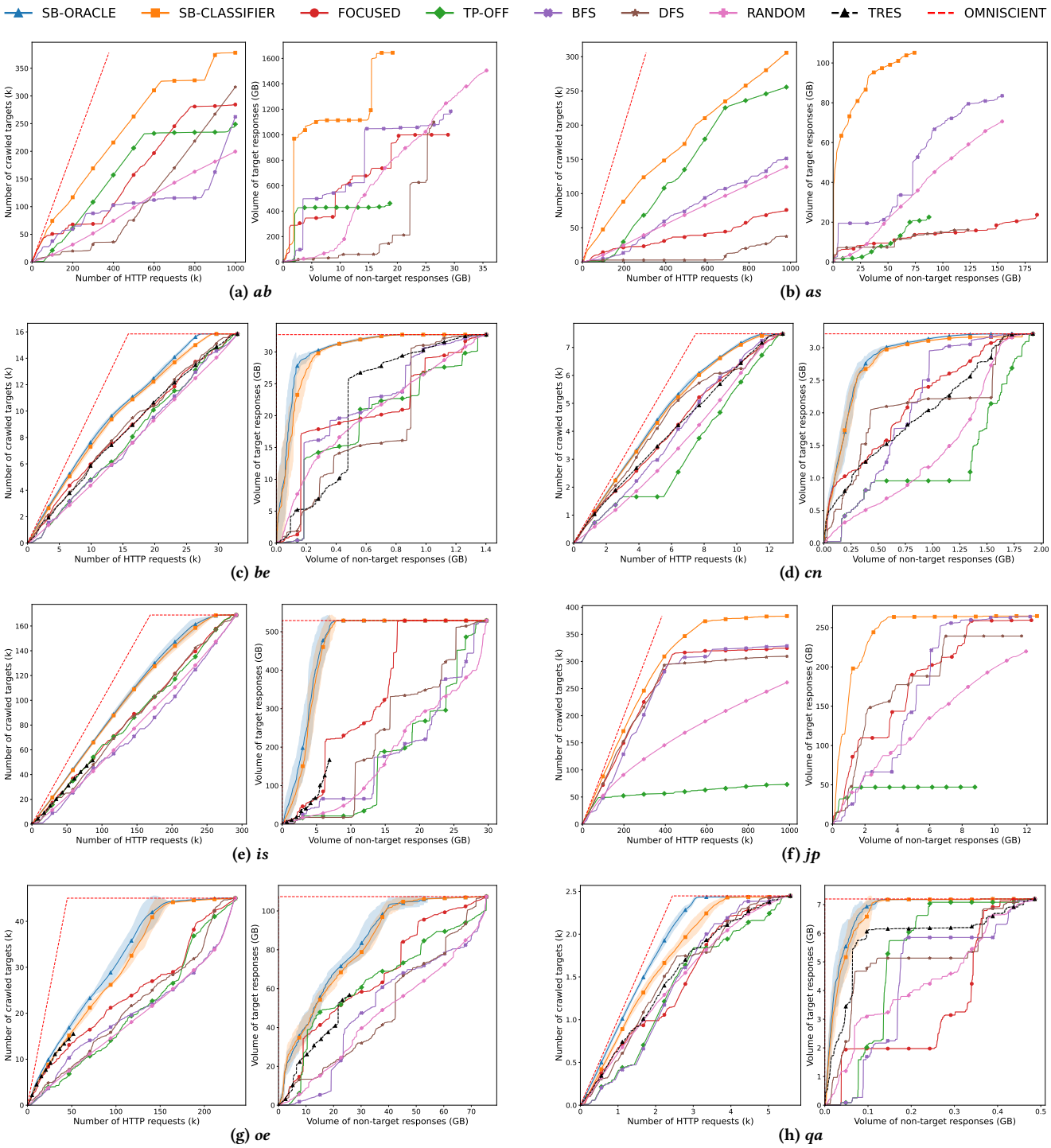


Figure 7: Comparison of different crawler performance for the 8 websites not presented in Figure 4 due to space reasons; for TRES, experiments are only shown for fully-crawled websites

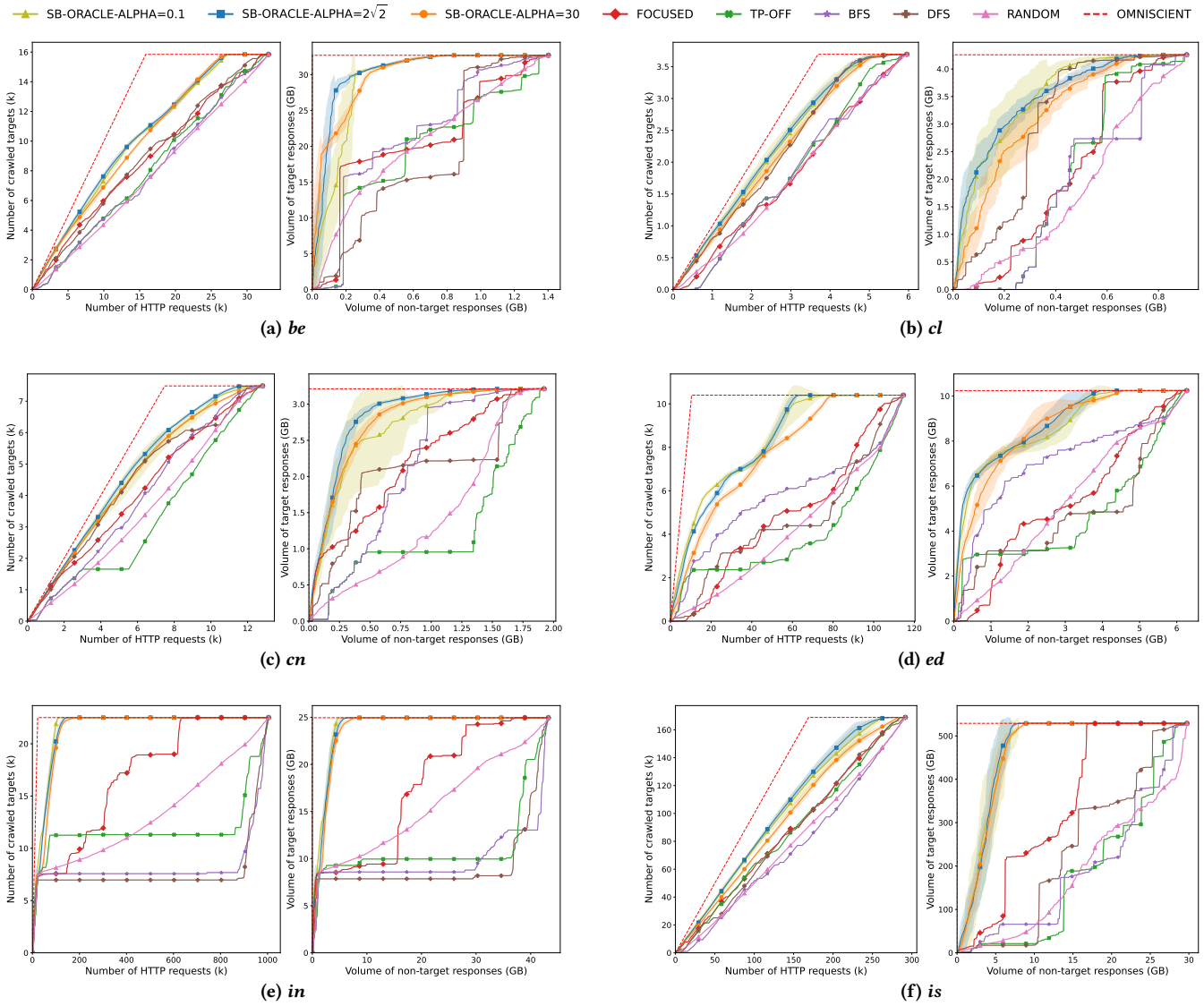
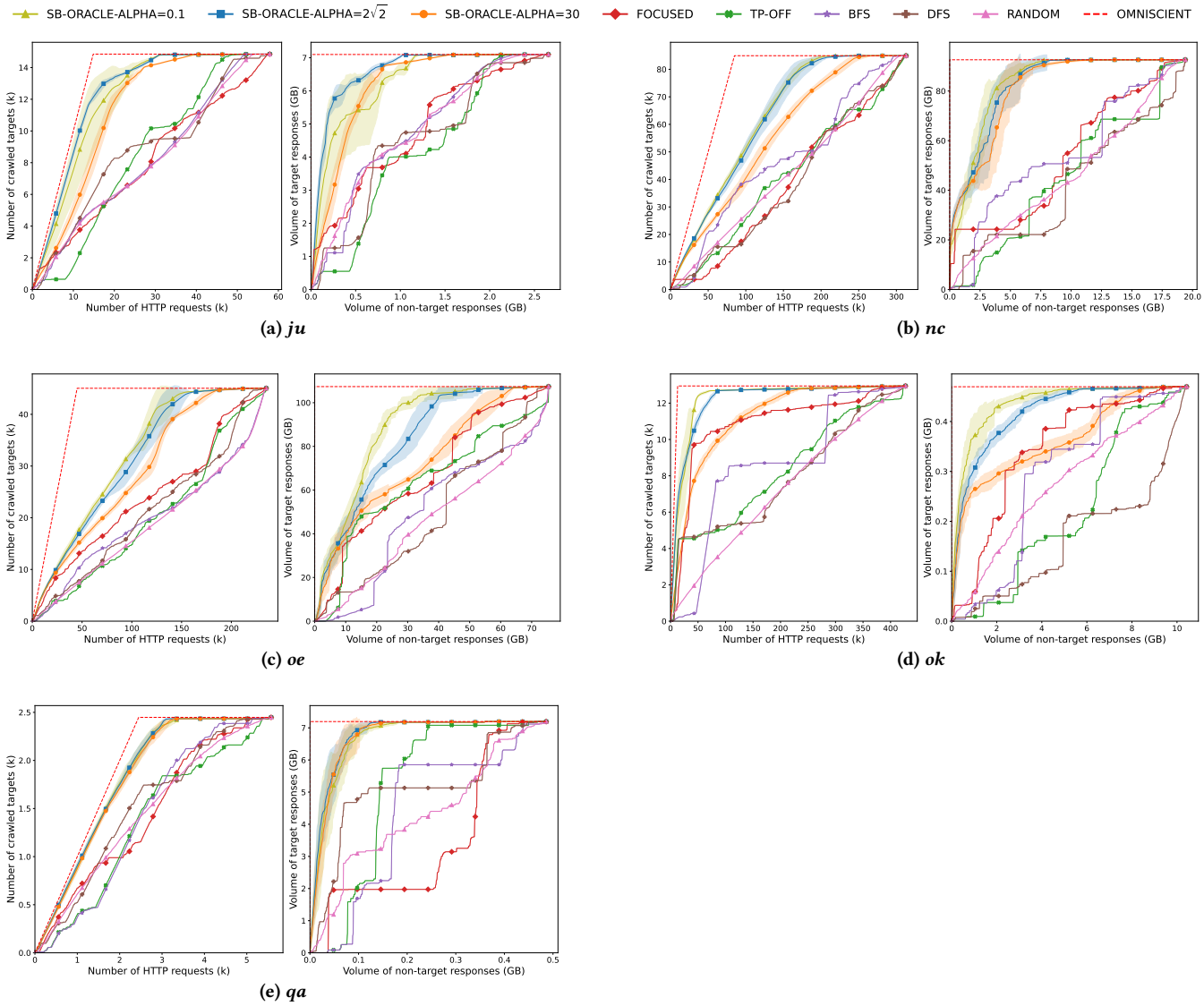


Figure 8: Crawler performance for hyper-parameter study on exploration–exploitation coefficient α, for websites *be*, *cl*, *cn*, *ed*, *in*, and *is*



(a) *ju* (b) *nc*

(c) *oe* (d) *ok*

(e) *qa*

Figure 9: Crawler performance for hyper-parameter study on exploration–exploitation coefficient α, for websites *ju*, *nc*, *oe*, *ok*, and *qa*

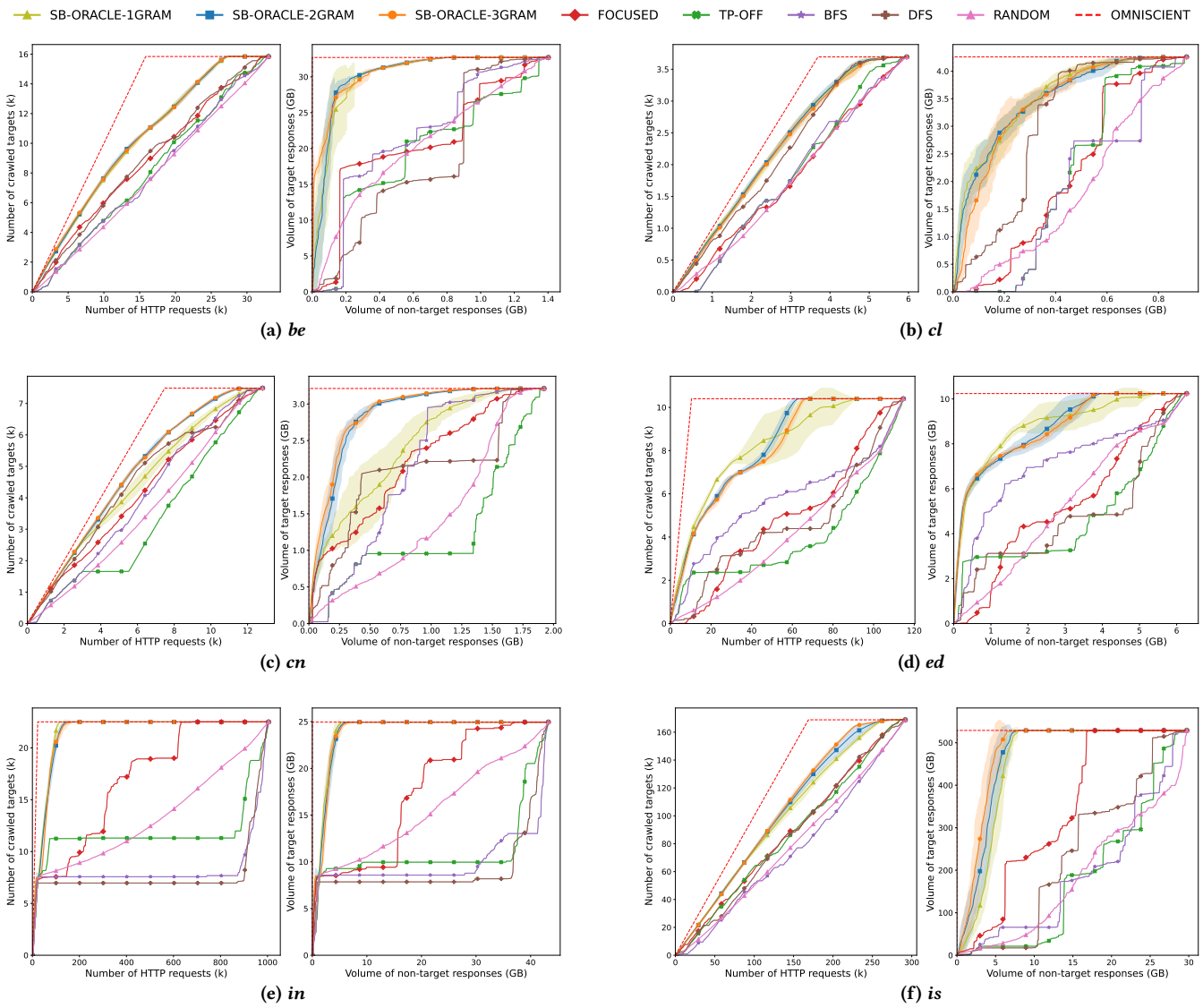
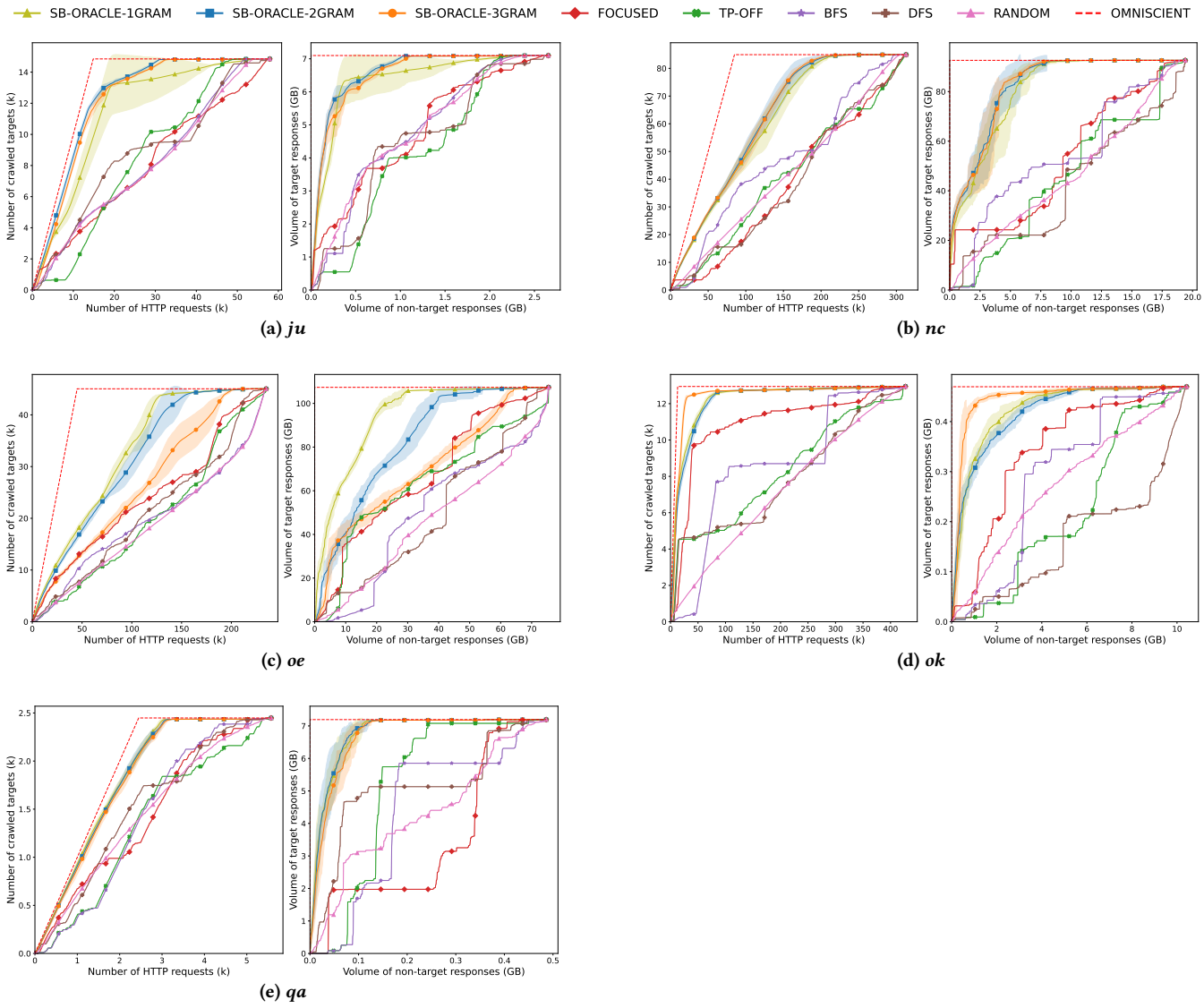


Figure 10: Crawler performance for impact study of the choice of n in n-grams used in tag path vector representation, for websites *be*, *cl*, *cn*, *ed*, *in*, and *is*



(a) *ju* (b) *nc* (c) *oe* (d) *ok* (e) *qa*

Figure 11: Crawler performance for impact study of the choice of n in n-grams used in tag path vector representation, for websites *ju*, *nc*, *oe*, *ok*, and *qa*

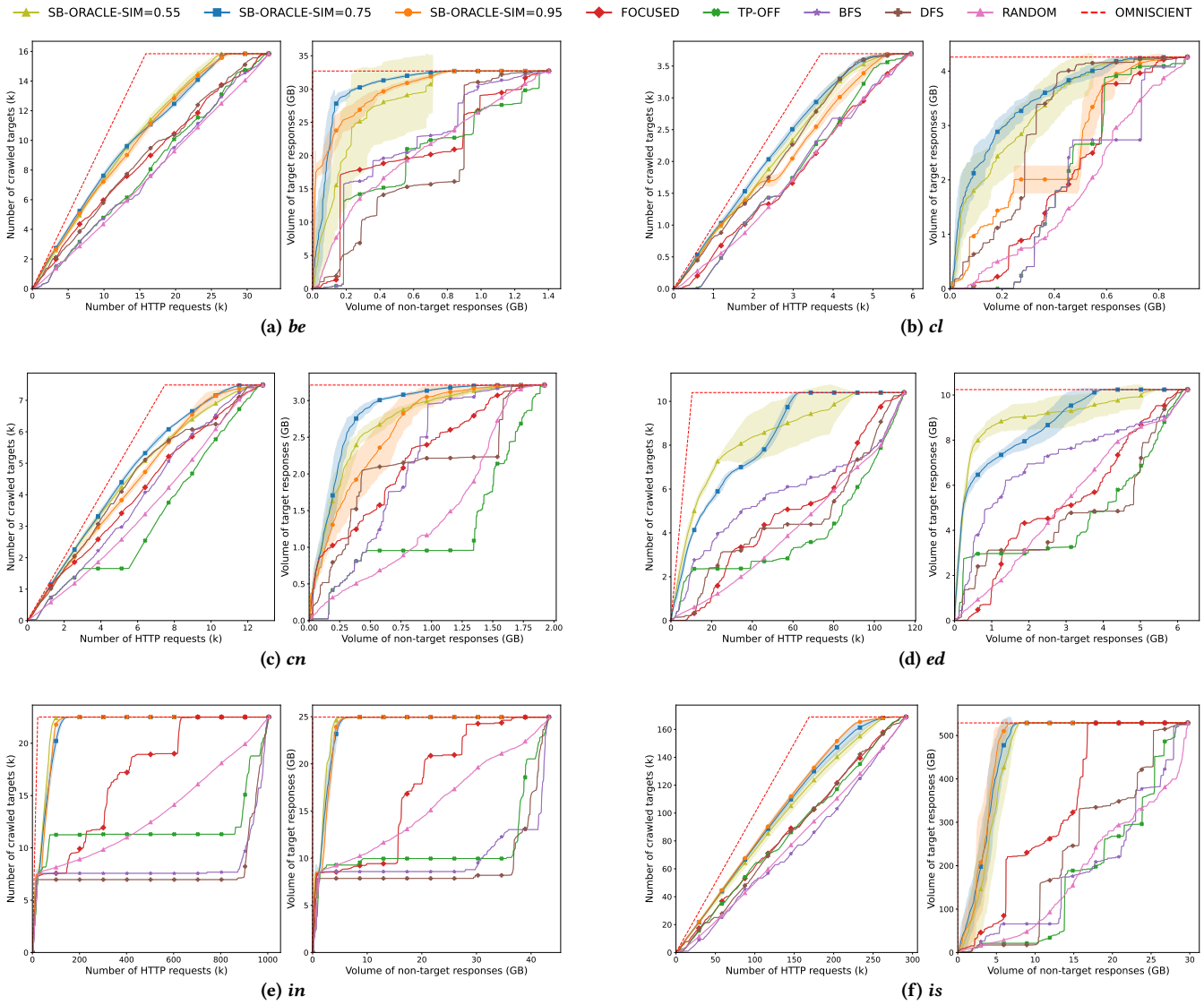
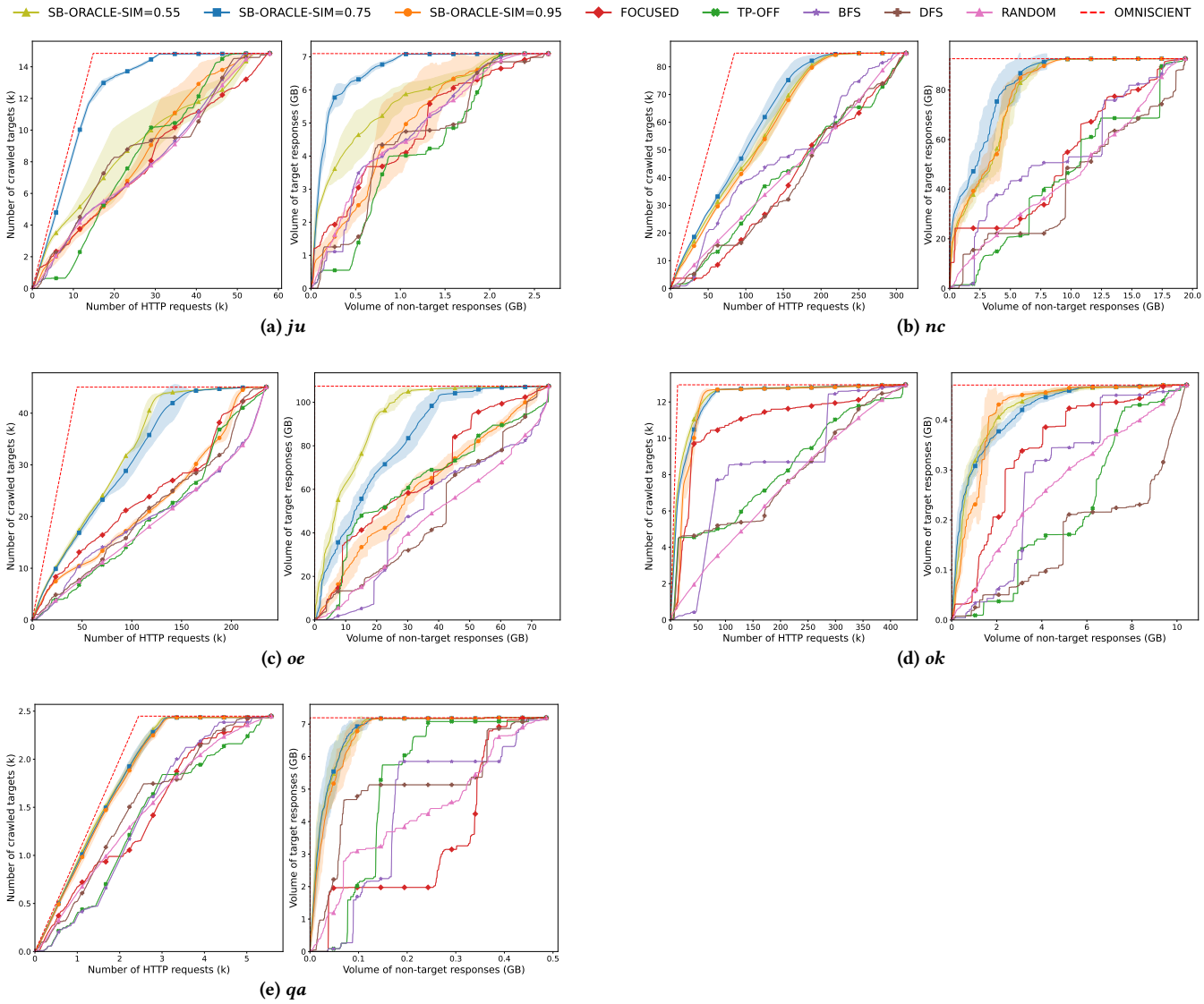


Figure 12: Crawler performance for impact study on similarity threshold θ, *cl*, *cn*, *ed*, *in*, and *is*



(a) *ju* (b) *nc*

(c) *oe* (d) *ok*

(e) *qa*

Figure 13: Crawler performance for impact study on similarity threshold θ, for websites *ju*, *nc*, *oe*, *ok*, and *qa*

B.3 Effectiveness of SB Learning

A typical tag path containing reference to IAP in the website *nc* is:

`/html/body/div.nces/div.container.w-iap/div.iap-content/div.row/div.col-md-6/h4/a`

and to SIEF in the website *wo*:

`/html/body/div.wp-page-body.container.default-wrapperpage-collections.collections-sief/div.body-content-wrap.theme-nada-2/div.about-collection/div.repository-container/div.body/div/p/a`

Examples of tag paths including keywords related to target retrieval are

`/html/body.path-node.page-node-type-ressource/div.dialog-off-canvas-main-canvas/div.layout-container/main#main-content/div.layout-content/div.region.region-content/div.block.block-system.block-system-main-block#block-open-theme-contenudelapageprincipale/article.mj-resource.mj-break-word.node--promoted/div.fr-container.mj-ressource__content/div.fr-grid-row.fr-grid-row--gutters.mj-content__corps/div.fr-col-lg-8.fr-col-offset-lg-2.fr-col-12.content-container__paragraph/section.fr-downloads-group.fr-downloads-group--multiple-links/ul/li/a.fr-linkfr-link--download`

for *ju*, and

`/html/body.home.page-template.page-template-onecolumn-page.page-template-onecolumn-page-php.page.page-id-7/main.content/div.container/div.row/div.maincol-md-12/article.post.post-7.page.type-page.status-publish.hentry#post-7/div.entry-content/div#stcpDiv/div/strong/a`

for *ok*.

Many other websites however have typical target-rich tag paths that cannot be interpreted by humans: for instance on *jp* we have

`/html/body.top/div#wrapper/div#groval_navi/ul#groval_menu/li.menu-item-has-children/ul.sub-menu/li/a`

on *il* we have

`/html/body/div.container.s-lib-side-borders.pad-top-med#s-lg-side-nav-content/div.row/div.col-md-9/div.s-lg-tab-content/div.tab-pane.active#s-lg-guide-main/div.row.s-lg-row/div.col-md-12#s-lg-col-1/div.s-lg-col-boxes/div.s-lg-box-wrapper-17054143#s-lg-box-wrapper-17054143/div.s-lib-box-container#s-lg-box-14451639-container/div.s-lib-box.s-lib-box-std#s-lg-box-14451639/div#s-lg-box-collapse-14451639/div.s-lib-box-content/div.clearfix#s-lg-content-31285074/ul/li/a`

and on *ed* we have

`/html/body.cf-rtm/div.accueil-portail#page/div.container#main-ermes-container/main#main/div#readspeaker-container/div#portal/div.row/div.col-md-12.cms-inner-layout#layout-2/div.row/div.col-md-6.cms-inner-zone#zone-4/div#frame-224/div.frame.frame-portalcarouselwebframefactory.hidden-print/div.frame-standard.panel.panel-front.webframe-ermes-carousel/div.panel-body/div.ermes-frame-html#carousel-4AD04CE72B556A3CCF5DFCF7FB7096…/div.rsItem/div.modele_13.model-html/table/tbody/tr/td/a.vide`

B.4 URL Classification Quality

Table 7: Confusion matrix of the URL classifier, on the 11 fully-crawled websites (average over 15 runs)

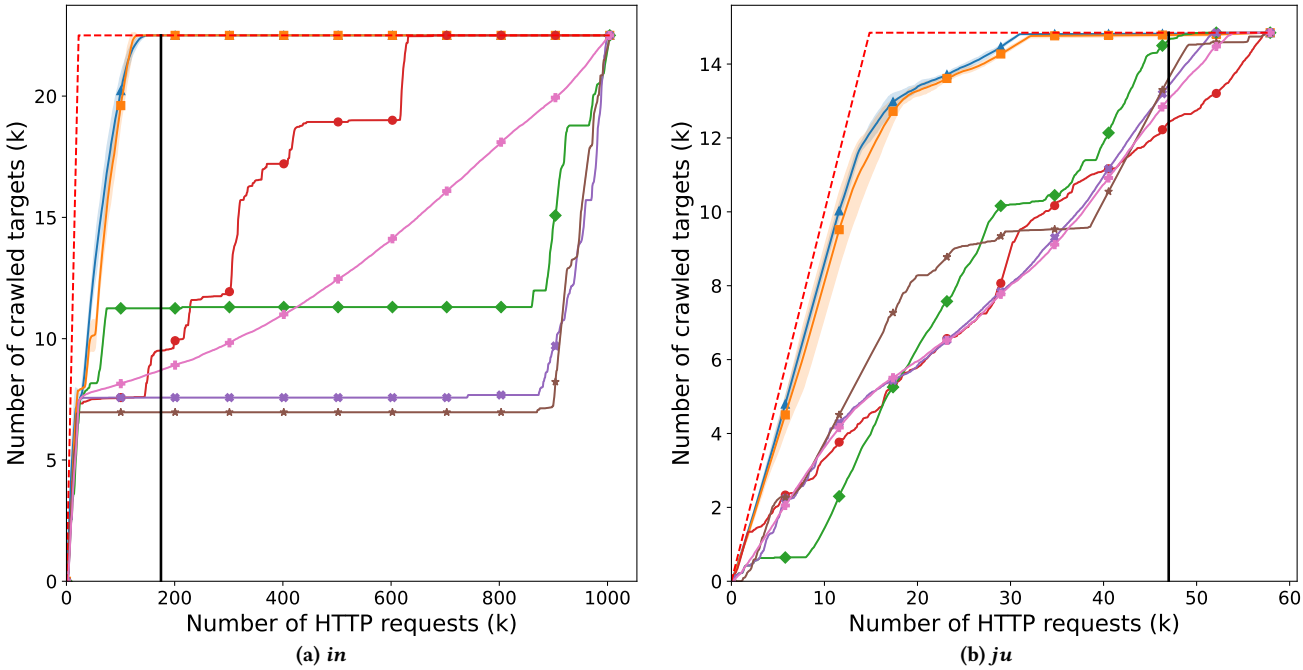
True \ Predicted	HTML (%)	Target (%)	Neither (%)
HTML	58.04	1.37	0.00
Target	0.75	32.19	0.00
Neither	5.34	2.41	0.00

The confusion matrix of our URL classifier, averaged over 15 runs, for all fully-crawled websites, is presented in Table 7.

We observe that **classification errors are extremely marginal on "HTML" and "Target" URLs**. Never classifying as "Neither" leads to some classification errors (discussed in Sec. 3.3), ultimately responsible for the difference between the number of requests made by SB-CLASSIFIER and SB-ORACLE on the 11 fully-crawled websites. However, since the classifier oracle is unfeasible in practice, the performance of SB-CLASSIFIER is satisfactory.

B.5 Early-Stopping

Figure 14 presents a visualization of the early-stopping mechanism applied to two websites: *in* and *ju* (right). Both fall under the first behavior described in Sec. 4.8, illustrating that smaller websites take longer (in %) to stop after the first signs of target convergence. For *in*, stopping occurs shortly after the point where a human would likely have cut, but for *ju*, the delay is more important. Despite early signs of convergence, stopping is postponed due to the fixed $\kappa \cdot \nu = 15\,000$ threshold, which represents a much larger fraction of the site for *ju* (60k pages) than for *in* (1M pages).



(a) *in* **(b) *ju***

Figure 14: Visualization of early-stopping time (black rule) for websites *in* and *ju*

C Supplementary material for Section 5 (Related Work)

Web crawlers. We discuss other relevant literature on Web crawling apart from focused and incremental crawling.

(1) *Distributed* or *parallel* Web crawlers use multiple crawlers concurrently to speed up page retrieval. Key challenges lie in resource (especially, network) management. [CG02] introduced general architectures for parallel crawling, while [BCSV04] proposed UbiCrawler, a decentralized and distributed system using consistent hashing for domain partitioning. Other works [CPWF07, BOM+12] focus on social network crawling. These works aim to efficiently crawl a given set of pages, whereas we focus on minimizing that set. The two could be combined to improve crawling speed and reduce network load.

(2) *Web crawlers targeting specific website types* (such as forums, blogs, or CMS) are built to leverage specific and common structural patterns within these sites. For instance, [GLZZ06, CYL+08] are focusing on forums. While effective, these crawlers are less general, relying on assumptions about site structure. In contrast, our approach adapts to each website dynamically, and needs no assumptions (see Sec. 2.1).

(3) *Hidden-* or *deep*-Web crawlers explore content not reachable via standard hyperlinks, often requiring user interactions such as filling out forms. A comparative study of state-of-the-art deep-Web crawler appears in [HRR19]. In contrast, our work focuses on acquiring specific targets from official websites. In our context, form-based interfaces serve mainly as filters, which are not useful for large-scale retrieval. Moreover, on many institutional websites (see Sec. 4.1), such data is accessible through standard navigation, not portals. Extending our approach to deep-Web crawling remains a natural direction for future work (see Sec. 6).

MAB algorithms. While UCB [ACBF02] and its variants are the state of the art for solving MAB problems, simpler approaches like ε-greedy [SB98] randomly select actions with probability ε and otherwise choose the highest scoring action. We also considered Bayesian Bandits, especially TS, a probabilistic method based on posterior distribution estimation. We excluded those approaches to ensure our crawler's *stability*, i.e., ability to give the same output if run several times independently of how the site can change from one crawl to another. Also, Bayesian approaches require prior distributions, unavailable to us due to lack of prior knowledge about the websites. We also did not adopt *linear* or *kernel*-based bandit methods, despite their generalization capabilities. These approaches require stable, qualitative feature representations, extracted from webpages, which are difficult to define in our context. Indeed, while we hypothesize that links appearing in similar tag paths lead to similar content, this assumption does not guarantee the existence of consistent features extracted from the *content* (not structure) of the pages. We would also need to choose *universal* features, i.e., computable for any page of any website we want our crawler to visit. Moreover, most feature-based methods are language-dependent, which conflicts with our aim of being language-independent. Given these limitations, and the good performance we already observed, we focused on AUER [KNS08], the sleeping variant of UCB.

References for the Appendix

[ACBF02] Peter Auer, Nicolò Cesa-Bianchi, and Paul Fischer. Finite-time Analysis of the Multiarmed Bandit Problem. *Machine Learning*, 47(2):235–256, 2002.
[BCSV04] Paolo Boldi, Bruno Codenotti, Massimo Santini, and Sebastiano Vigna. Ubicrawler: a scalable fully distributed web crawler. *Softw. Pract. Exp.*, 34(8):711–726, 2004.

[BOM+12] Matko Bosnjak, Eduardo Oliveira, José Martins, Eduarda Mendes Rodrigues, and Luís Sarmento. Twitterecho: a distributed focused crawler to support open research with twitter data. In Alain Mille, Fabien Gandon, Jacques Misselis, Michael Rabinovich, and Steffen Staab, editors, *Proceedings of the 21st World Wide Web Conference, WWW 2012, Lyon, France, April 16-20, 2012 (Companion Volume)*, pages 1233–1240. ACM, 2012.

[CG02] Junghoo Cho and Hector Garcia-Molina. Parallel crawlers. In David Lassner, David De Roure, and Arun Iyengar, editors, *Proceedings of the Eleventh International World Wide Web Conference, WWW 2002, May 7-11, 2002, Honolulu, Hawaii, USA*, pages 124–135. ACM, 2002.

[CPWF07] Duen Horng Chau, Shashank Pandit, Samuel Wang, and Christos Faloutsos. Parallel crawling for online social networks. In Carey L. Williamson, Mary Ellen Zurko, Peter F. Patel-Schneider, and Prashant J. Shenoy, editors, *Proceedings of the 16th International Conference on World Wide Web, WWW 2007, Banff, Alberta, Canada, May 8-12, 2007*, pages 1283–1284. ACM, 2007.

[CYL+08] Rui Cai, Jiang-Ming Yang, Wei Lai, Yida Wang, and Lei Zhang. irobot: an intelligent crawler for web forums. In Jinpeng Huai, Robin Chen, Hsiao-Wuen Hon, Yunhao Liu, Wei-Ying Ma, Andrew Tomkins, and Xiaodong Zhang, editors, *Proceedings of the 17th International Conference on World Wide Web, WWW 2008, Beijing, China, April 21-25, 2008*, pages 447–456. ACM, 2008.

[GJ79] M. R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman, 1979.

[GLZZ06] Yan Guo, Kui Li, Kai Zhang, and Gang Zhang. Board forum crawling: A web crawling method for web forum. In *2006 IEEE / WIC / ACM International Conference on Web Intelligence (WI 2006), 18-22 December 2006, Hong Kong, China*, pages 745–748. IEEE Computer Society, 2006.

[HRR19] Inma Hernández, Carlos R. Rivero, and David Ruiz. Deep web crawling: a survey. *World Wide Web*, 22(4):1577–1610, 2019.

[KNS08] Robert D. Kleinberg, Alexandru Niculescu-Mizil, and Yogeshwer Sharma. Regret bounds for sleeping experts and bandits. In Rocco A. Servedio and Tong Zhang, editors, *21st Annual Conference on Learning Theory - COLT 2008, Helsinki, Finland, July 9-12, 2008*, pages 425–436. Omnipress, 2008.

[SB98] Richard S. Sutton and Andrew G. Barto. *Reinforcement learning - an introduction*. Adaptive computation and machine learning. MIT Press, 1998.

[WW16] Dimitri Watel and Marc-Antoine Weisser. A practical greedy approximation for the directed steiner tree problem. *J. Comb. Optim.*, 32(4):1327–1370, 2016.