A Framework for Improving Web Affordability and Inclusiveness

Rumaisa Habib*, Sarah Tanveer*, Aimen Inam, Haseeb Ahmed, Ayesha Ali, Zartash Afzal Uzmi, Zafar Ayyub Qazi, Ihsan Ayyub Qazi
LUMS
Pakistan

ABSTRACT

Today's Web remains too expensive for many Internet users, especially in developing regions. Unfortunately, the rising complexity of the Web makes affordability an even bigger concern as it stands to limit users' access to Internet services. We propose a novel framework and a fairness metric for rethinking Web architecture for affordability and inclusion. Our proposed framework systematically adapts Web complexity based on geographic variations in mobile broadband prices and income levels. We conduct a cross-country analysis of 99 countries, showing that our framework can better balance affordability and webpage quality while preserving user privacy. To adapt Web complexity, our framework solves an optimization problem to produce webpages that maximize page quality while reducing the webpage to a given target size.

CCS CONCEPTS

• **Information systems → World Wide Web**.

KEYWORDS

Web, Affordability, Inclusion, User Privacy, Transcoding Service

ACM Reference Format:
Rumaisa Habib*, Sarah Tanveer*, Aimen Inam, Haseeb Ahmed, Ayesha Ali, Zartash Afzal Uzmi, Zafar Ayyub Qazi, Ihsan Ayyub Qazi. 2023. A Framework for Improving Web Affordability and Inclusiveness. In *ACM SIGCOMM 2023 Conference (ACM SIGCOMM '23), September 10–14, 2023, New York, NY, USA.* ACM, New York, NY, USA, 16 pages. https://doi.org/10.1145/3603269.3604872

1 INTRODUCTION

In 1991, Tim Berners-Lee published the first ever website and it is estimated that by November 2022, there were nearly 2 billion websites on the Internet [47]. Behind the success of the World Wide Web (WWW) lay a powerful idea: a way for people to find and share information freely across the connected humanity. Indeed, a large body of evidence shows that the Web has played an essential role in the development of economies and the socioeconomic mobility of citizens by providing access to information about markets, jobs, health, educational and financial services as well as by generating economic opportunities [26, 31, 53]. Yet, despite WWW's egalitarian roots, it remains too expensive for many Internet users, especially in developing regions.

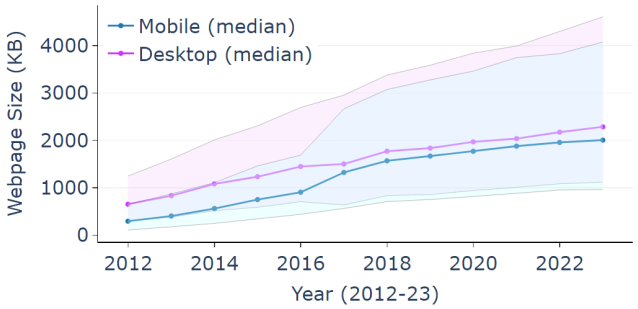


Figure 1: Evolution of landing pages of mobile and desktop versions of websites on the Internet. The figure shows the median pages along with the 25th and 75th percentile page sizes (shown by the upper and lower grey lines). Data from the HTTP Archive [5].

A key reason is that access to the Internet in many developing countries is predominantly through mobile phones where the cost of mobile broadband relative to income levels is high [48]. As a result, many citizens face difficulty paying for their mobile data usage and often restrict their data usage, which in turn limits their ability to access the Internet. A World Bank survey carried out in 11 emerging countries found that a median of 48% of respondents had difficulty paying for their mobile data usage, and 42% restrict the amount of data they use [55].

In 2018, the UN Broadband Commission for Sustainable Development set the target for affordable broadband services to be less than 2% of the monthly Gross National Income (GNI) per capita. According to the International Telecommunication Union, 94 developing countries do not meet this target for a 2 GB data-only mobile broadband plan.[1] While the average price in 51 countries is between 2–5% of GNI per capita,[2] in 20 developing countries the price exceeds 10% of GNI [30].

Concurrently, we are observing a steady increase in Web complexity as shown in Fig. 1. For example, the median mobile webpage size has increased from 145 KB to 2007 KB in nearly a decade; a 13.8× increase [27]. Partly, this growing complexity stems from the fact that the Web's design does not take into account affordability as a design consideration. Thus, such a trend can reduce users' Web accesses and make Internet services less affordable, especially for

*Co-first authors.

ACM SIGCOMM '23, September 10–14, 2023, New York, NY, USA

ACM ISBN 979-8-4007-0236-5/23/09…$15.00
https://doi.org/10.1145/3603269.3604872

[1] Countries are benchmarked according to the price of the cheapest data-only mobile broadband plan available domestically, with a minimum of 2 GB monthly data allowance and a technology of 3G or above. Mobile plans involving voice and data (at least 500 MB) were more expensive [29, 30]
[2] We use GNI to refer to GNI per capita throughout the paper.

Services	Example Data-Saving Mechanisms
Free Basics [4, 45]	Webpages cannot have JavaScript, large images, iframes, videos, Flash, and Java applets
Web Light [9, 49]	Removes all JS (except with scripts within an iframe for ads), resizes large images, and converts external CSS into inline CSS, replaces video with an image
Facebook Discover [42]	Limits responses to 1 MB, removes images or reduces resolution, and removes video, audio, and other rich content
FlyWheel [12]	Compresses images (works only for HTTP traffic)
Lite-Web [16]	Removes or blocks JavaScript that is unused or non-critical for interactive functionality.

Table 1: Some example services being used in developing countries for reducing data charges. They target an extreme design point for affordability at the cost of substantial reduction in QoE and use proxy-based solutions that have led to privacy and net-neutrality concerns.

users in developing regions. While broadband prices have reduced in the last few years, the decrease has been slow. For instance, between 2020 and 2021, the median price for mobile broadband services was reduced by only 0.02 percentage points [30]. This calls for designing solutions that can make the Web affordable by taming the complexity of the Web experienced by users from regions with high broadband prices relative to income levels.

Recent initiatives for reducing Web complexity in developing countries (e.g., Free Basics [4, 45], Google Web Light [9], Discover [42]) face two key challenges (see Table 1): (i) they target an extreme design point for affordability, which comes at a cost of substantial reduction in webpage quality. For example, Web Light reduces the median webpage size by 12× but can break the functionality of pages, rendering them unusable [49]. Similar to Web Light, Free Basics removes JavaScript (JS), iframes, large images, and other rich content and (ii) they rely on proxy-based solutions that break the end-to-end principles of TLS [16, 42, 45, 49].[3] This has raised significant data privacy and network neutrality concerns, even leading to the banning of services in some countries [34, 50].

In this work, we make a case for rethinking Web design around affordability and inclusion. To this end, we present, AW4A (Affordable Web For All) that addresses the above challenges and achieves a better balance between affordability and webpage quality by relying on two key principles: (i) Web frameworks should explicitly account for affordability constraints in their design and (ii) website operators should determine Web content adaptations to preserve user privacy and meet their business needs.

AW4A provides a systematic way for incorporating affordability constraints in Web design by relying on a new fairness metric, PAW (Price Adjusted Web access), which captures how equitable and affordable *Web accesses* are across regions with different mobile broadband prices and income levels. PAW can be used by website operators to come up with different target webpage sizes, which AW4A uses to generate multiple versions of a website with different complexities.

Given a target page size, AW4A aims to *maximize webpage quality* while ensuring the resulting page is no larger than the target size. This requires solving an optimization problem that involves selecting a set of resource optimizations for different Web objects (e.g., image, JS) that will yield the maximum quality webpage within the target size. This optimization problem is practically inefficient to solve for large webpages (§6). We design approximation algorithms for solving the optimization problem: a brute force algorithm over the discretized space of the problem and an algorithm that uses a greedy approach and can quickly scale to large webpages (§7). Our evaluation shows that our algorithms can transcode webpages such that there is minimal impact on page quality while significantly reducing page sizes (§8).

While AW4A can adapt the webpage to meet the affordability target in a region, it also offers a choice to users to view a higher quality webpage if they wish to do so. AW4A also exposes APIs to the website operators for deciding which low-complexity versions to generate and what weights to assign to different Web objects and resource optimizations.

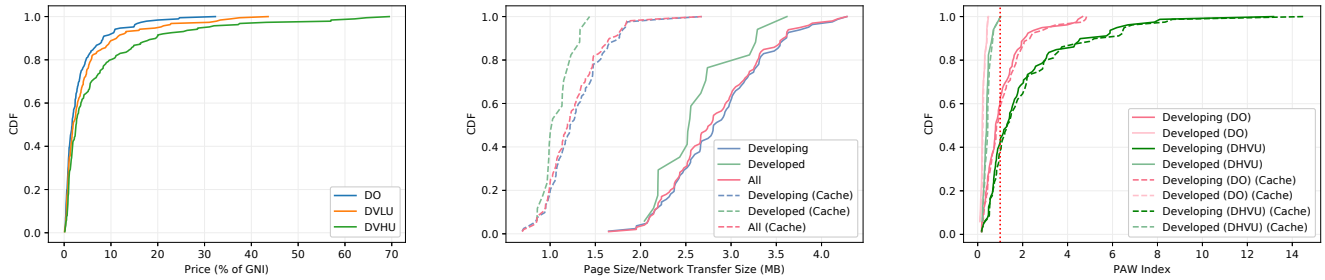
By offering more equitable Web accesses through a differentiated service offering, AW4A can bring more users online and increase access for constrained users, which in turn can lead to increased revenues for website operators. For example, our analysis shows that reducing the average webpage size by 1.5× can allow 12.1%–14.1% of the countries to meet the affordability target for Web accesses, which is achievable without significantly reducing page quality. Moreover, reducing complexity can make it viable for mobile service providers to offer smaller data plans, as users can derive more value from the same data budget.

Finally, AW4A has synergies with the increasing trend of Lite apps (e.g., Facebook Lite, Skype Lite), which are being designed for entry-level smartphones that are prevalent in developing countries [2, 39]. These apps are designed by the content providers themselves, have smaller sizes, and reduced functionality that meets the goals of the service providers. Altogether, we make the following contributions:

- We collect and analyze the landing pages of Alexa top websites accessed from 99 countries resulting in a dataset of 72,069 pages after data cleaning. Using this data, we make a case for adapting Web complexity based on geographical variations in broadband prices and average income levels across regions.
- We propose a fairness metric, PAW, which captures how equitable average webpage accesses are across countries.
- Using a simple economic model, we highlight the quality-access trade-off and show via a user study with 100 participants that users can observe significant utility gain by trading off webpage quality for increased Web accesses.
- We present AW4A; a framework that uses PAW and solves an optimization problem to generate low complexity versions of webpages with high quality.
- We carry out a cross-country analysis of 99 countries.[4] We show that different Web complexities can be achieved and highlight the trade-offs they present.
- We carry out an evaluation of HBS (our implementation of AW4A) in terms of time to run, percentage URLs that can meet the target size (as determined by PAW), and resultant webpage quality. We also compare HBS to two state-of-the-art industry solutions, Brave

[3] Some proxy-based approaches, such as FlyWheel [12], FlexiWeb [46], Traffic Guard [36], and Lite-Web [16] do not work with HTTPS, which limits their use given the near-ubiquitous adoption of end-to-end encryption [20] whereas other approaches (e.g., Web Light) leverage URL redirection to transcode content without taking website operators' consent and can expose private or personalized Web contents to third parties [34, 49].

[4] These countries represent 7.16 B people or ∼90% of the world's population.



(a) Mobile Broadband Prices **(b) Average Page Size/Network Transfer Size** **(c) PAW index**

Figure 2: (a) Mobile broadband prices as a percentage of GNI per capita across 206 countries for three plans. 41%–52% of the countries do not meet the affordability target of 2% across the plans, (b) distribution of landing page sizes of Alexa top 1000 websites from each of the 99 countries for which the data was available (resulting in a total of 72,069 pages) (c) ratio of mean Web accesses across 96 countries—divided into developed and developing countries—relative to the affordability target and the mean global page size (2.47 MB for non-cached pages and 1.02 MB for cached pages) based on Alexa top 1000 websites globally. The vertical line at $x = 1$ shows the ratio at the global benchmark level and countries exceeding 1 do not meet the target. Note that broadband price data was not available for three countries. For PAW results of all plans, see Appendix A.3.1.

and Opera Mini. To ensure reproducibility and facilitate future work, we have made the source code of HBS publicly available at https://github.com/nsgLUMS/sigcomm2023-aw4a.

- We present a discussion, which suggests that using such a framework can bring more users online and lead to greater advertising revenues for website operators, which can serve as an incentive to offer such solutions.

2 MOTIVATION

Using data from the ITU, we first analyze the variations in mobile broadband prices across 206 countries and three broadband plans [30]. We then analyze the Web complexity across 99 countries by measuring the sizes of landing pages of Alexa top 1000 websites from *each* of the countries for which the data was available [1].[5] This resulted in a dataset of 72,069 webpages after data cleaning.[6]

In addition, we employ two strategies to quantify the effect of caching on Web complexity. We use (i) `Cache-Control` information to simulate the average 'cached' version of the webpages in our dataset and (ii) conduct real experiments on two devices with different RAM sizes.

2.1 Distribution of Mobile Broadband Prices

Fig. 2a shows the distribution of prices (normalized by GNI per capita) for three mobile broadband plans across 206 countries: These plans, defined by ITU, include: (i) a 2 GB data-only plan (DO), (ii) a hybrid plan comprising 500 MB of data and voice low-usage (DVLU), and a hybrid plan with 2 GB of data and voice high-usage (DVHU).[7] First, we observe that there are large variations in mobile broadband prices across countries for all three plans ranging from 0.07%–41%, 0.13%-38.4%, and 0.13%–56.9% for the DO, DVLU, and the DVHU plans, respectively. Second, 41%–52% of the countries do not meet the UN Broadband Commission's affordable target price of 2% when considering the three broadband plans.

This is concerning because the lack of affordability of mobile Internet has directly translated into large usage and consumption gaps in developing countries. For example, in South Asia alone, 64% of the population do not use the Internet despite having at least 3G coverage, reflecting a large usage gap. A survey carried out by the World Bank in 11 emerging countries found that a median of 48% of respondents had difficulty paying for their mobile data usage [55].

2.2 Webpage Complexity Across Regions

The growing complexity of the Web has led to steady increases in webpage sizes, which in turn, has increased the cost per access to a website. For example, in the last five years (i.e., between January 2018 and January 2023), the median mobile page size has increased from 1569 KB to 2007 KB, an increase of 27.9% [5]. Due to the differences in the popularity of websites across countries, we expect the average webpage size observed by users to also differ across countries. Thus, we collected the sizes of landing pages of Alexa top websites accessed from 99 countries, each of which had different average broadband prices. The mean (non-cached) page size in our sample was 2.83 MB ($\sigma = 0.55$ MB)[8] with some countries having a mean size greater than 4 MB and some less than 1.75 MB. We find that, in general, webpages accessed from developing countries tend to be larger in size than those accessed from developed countries, as shown in Fig. 2b.[9] In particular, the mean (non-cached) page size in developing and developed countries was 2.87 MB ($\sigma = 0.56$ MB) and 2.64 MB ($\sigma = 0.46$ MB), respectively. These variations in webpage sizes (both across countries as well as within countries) directly impact the data usage and the number of websites a user can access before running out of a data plan.

Caching. Browser caching is used to store frequently accessed objects so they do not need to be retransmitted on each website visit. Thus, the total bytes transferred to the user may not always be equivalent to the page size on the first visit. To account for this, we

[5]We use WebPageTest [11] and a popular entry-level mobile device, Nexus 5, to fetch these pages. In our analysis, we consider the network transfer size (which comprises of the compressed sizes of each object) as reported by WebPageTest.

[6]The number of URLs in our final dataset is less than 99×1000 (99,000) as some of the websites were blocked, broken, and some countries had a limited number of URLs (less than 1000) available on Alexa.

[7]DO refers to the cheapest plan in a country providing at least 2 GB of high-speed data (≥ 256 Kbps), DVLU refers to the cheapest plan providing at least 70 mins of voice, 20 SMS, and 500 MB of high-speed data (≥ 256 Kbps), and the DVHU plan refers to the plan providing at least 140 mins of voice, 70 SMS, and 2 GB of high-speed data (≥ 256 Kbps).

[8]The page size refers to the total network bytes downloaded, i.e. the compressed web objects are taken into account for both cached and non-cached page sizes.

[9]Our sample comprised data of the landing page sizes of websites from 82 developing countries (including India, Pakistan, and Ethiopia) and 17 developed countries (including USA, Germany, and Canada).

report the effect of caching using two methodologies with differing assumptions.

First, we simulate a user with infinite cache storage who visits all the webpages in our dataset once every 12 hours for 2 weeks.[10] When an object becomes stale (as determined by its `max-age` in the `Cache-Control` HTTP header), it is 'redownloaded.' In this setup, we observe that caching greatly reduces the total byte cost (Fig. 2b). Notably, the average bytes expended by a user accessing the Alexa top 1000 global sites is 1.02 MB (a 58.7% reduction of the average non-cached page size based on the same dataset). We use this methodology throughout the paper when we refer to a 'cached' webpage.

Second, to assess the impact of other significant factors on caching (such as browser cache limit and device memory), we randomly selected 25 websites from the global Alexa top 1000 sites and loaded these websites at 12-hour intervals for 2 weeks on two entry-level smartphones. The two smartphones, Nexus 5 and the Nokia 1, have varying RAM sizes (2 GB and 1 GB, respectively). We find that, in this setup, caching on the Nexus 5 and Nokia 1 provides an average page size reduction of 60.9% and 21.4%, respectively. This difference in page size reduction is attributed to the fact that the bytes expenditure, and hence the total number of accesses available to a user, also depends on the device being used. Memory and space constraints limit the possible savings through caching [44].

While these experiments provide an estimate for the impact of caching on network bytes transferred, the impact of user browsing habits (which may be region-specific), varying visit frequencies, and different browser applications on caching is not taken into account.

3 FAIRNESS IN WEB ACCESSES

We now introduce a new fairness metric for affordability and analyze the distribution of Web accesses available to users across countries. The proposed metric provides Web complexity targets to meet the affordability target set forth by the UN Broadband Commission for Sustainable Development. Next, we carry out a What-If analysis to determine the possible reduction in page sizes with different resource optimizations.

3.1 Fairness Metric

Our analysis in the previous section showed that the average page size can vary significantly across countries. Thus, for a fixed mobile broadband plan (e.g., a data-only plan), users in different countries will have different numbers of Web accesses available to them before running out of their data plan.

To capture the differences in average page sizes, mobile broadband prices, and income levels across regions as well as enable comparisons against a common benchmark, we present a new fairness metric for affordability, which we call the `PAW` index. $\texttt{PAW}_i$ captures the reduction needed in the average webpage size in a region i to achieve the affordability target for Web accesses and is computed as follows:

$$\texttt{PAW}_i = \frac{P_i}{P_T} \times \frac{W_{i,avg}}{W_{global}} \tag{1}$$

where P_T is the average price target set by UN's Broadband Commission, which currently stands at 2%, P_i is the average mobile broadband price in region i as a percentage of GNI, $W_{i,avg}$ is the average webpage size in region i, and W_{global} is the average webpage size globally. $\texttt{PAW}_i > 1$ for region i if the number of Web accesses does not meet the affordability target and $\texttt{PAW}_i \leq 1$ if it does. For example, if $W_{i,avg} = 1.5$ MB, $P_i = 5\%$ in region i, $W_{global} = 2.47$ MB, and $P_T = 2\%$, then $\texttt{PAW}_i = 1.52$.

The number of accesses, A_T, available at the target affordable price is given by D/W_{global}, where W_{global} is the average global webpage size. Therefore, the maximum number of Web accesses, A_i, available in region i to meet the affordability target is $(P_T/P_i) \times (D/W_{avg})$, where D is the data plan limit (e.g., 2 GB). To equalize accesses (i.e., $A_i/A_T = 1$), we require the average webpage size to be set to $W^T_{avg} = (P_T/P_i) \times W_{global}$.

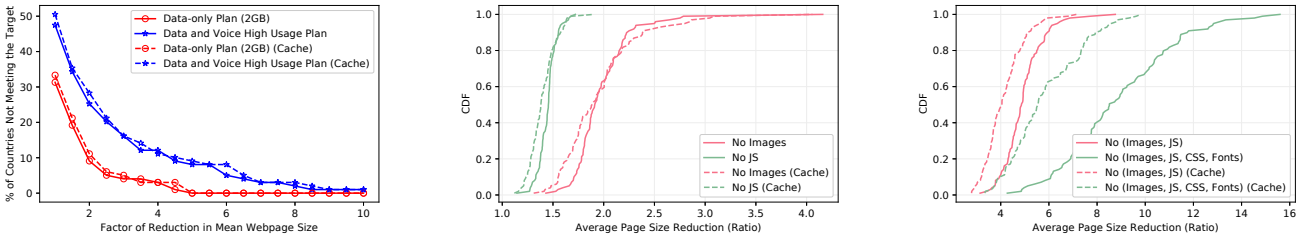
We use W_{global} to benchmark webpage sizes according to the prevailing global quality of webpages. Thus, if in region i the average page size is already lower than the average global page size, then a smaller reduction in page quality is required to equalize access relative to another region j, where the page size is larger. Therefore, despite the inherent trade-off between accesses and page quality, with this benchmarking, we are able to minimize the differences in quality across regions while equalizing accesses.

3.2 Web Accesses Across Regions

Based on the mobile broadband price and the average webpage size in each of the 99 countries, we find the expected number of Web accesses a user can afford at the target price of 2% before running out of a monthly mobile plan. Using these we compute the `PAW` index (see Fig. 2c). First, observe that 48 out of 96 countries do not meet the target Web accesses for at least one of the mobile plans.[11] The maximum `PAW` index for the DO plan is 4.7, whereas for DVHU, it is 13.2. This suggests that offering lower complexity versions (e.g., 1.5×, 2×, and 4.5×) of popular websites in these countries can bring many users within the affordability Web access budget. Second, note that there is a negligible change in the `PAW` index when the cached versions of the webpages are considered instead of the non-cached webpages. This is because caching reduces both the average page sizes in the countries (the numerator, $W_{i,avg}$ in Eq. 1) and the global average page size (the denominator, W_{global}). The implication is that even though caching increases the total number of Web accesses available in all countries, it does so (nearly) equally, and hence there is little change in the inequality in the number of accesses available across countries. `PAW` index is a comparative metric that captures this unfairness regardless of whether caching has been enabled. Third, within developing countries where an average user meets the target Web accesses, differences in income levels make it challenging for low-income users to meet this target. For example, the price of mobile broadband for users in the bottom income quintile in Pakistan is as much as 2.5% of the GNI compared to the average price of 0.96%. [40] Thus, a viable way of making the Web more affordable is to offer multiple low-complexity versions of websites while still providing users the option to view a high-quality page.

[10] In our dataset, we found that the median `max-age`, which is the maximum amount of time a fetched asset can be reused, of a Web object was ~2 weeks.

[11] Out of the 99 countries we considered in this analysis, the broadband price data was not available for three countries: Syria, Taiwan, and Venezuela was not available.



(a) Affordability-size trade-off **(b) Individual resource type** **(c) Multiple resource types**

Figure 3: (a) Percentage of countries that meet the target number of Web accesses for different mobile plans as a function of the reduction in the average webpage size in a country, (b) possible webpage size reductions with removals of one resource type from the pages, and (c) possible webpage size reductions with removals of multiple resource types from the pages.

Web accesses across 99 countries. Fig. 3a shows the percentage of countries meeting the Web accesses target as a function of the reduction in mean webpage size in a country across two data plans (for all plans, see Appendix A.3.2).

Observe that reducing the average (non-cached) page size by 1.5× allows 12.1%–14.1% of the countries to meet the target accesses whereas a 3× reduction will bring 27.3%–31.3% of the countries within the target across the three mobile plans. These insights can serve as useful guides for modulating website complexity.

3.3 What-If Complexity Analysis

Using the features of Alexa top 1000 websites accessed in each of the 99 developing countries, we carry out a What-If analysis on the possibilities of achieving the website complexity targets based on the `PAW` index.

Individual resource types. We find that since JS and images contribute the most bytes to webpages (former contributing 23%–47% and latter 24%–58% of non-cached webpages), removing them provides a significant reduction in page size. Fig. 3b shows the reduction in page sizes upon removing all elements of a single resource type (images and JS). For the non-cached webpages, reductions range from 1.1×–1.7× with JS and 1.4×–4.2× with images. For the cached websites, removing JS and images reduces the network bytes transferred from 1.1×–1.9× and 1.3×–4.1×, respectively.

Multiple resource types. Next, we consider removals of multiple combinations of resource types. Fig. 3c shows that, if we consider the non-cached pages, removing all JS and images from the Alexa top 1000 pages in all countries can reduce page sizes from 3.1×–8.8× and removing all four resource types reduces page sizes from 4.3×–15.6×. Removing all four Web objects from the cached webpages reduces the total bytes transferred from 3.3×–9.8×.

The above analysis provides an estimate of the possible Web complexity reductions considering the Web objects that comprise most of the network bytes (see Appendix A.3.3 for more scenarios). While results show that significant reductions in complexity are possible, these may come at the cost of a considerable decrease in page quality. Our What-If analysis also suggests that 1.5×–1.8× reductions in the average (non-cached) page sizes are possible across the 99 countries by reducing the total bytes due to images and JS by 50%.

4 QUALITY–ACCESS TRADEOFF

In this section, we present a simple economic model to capture the utility derived by users when deciding between webpage quality and the number of Web accesses. Using a user study involving 100 participants, we show that several users experience a utility gain by trading off webpage quality for more webpage accesses.

4.1 Modeling User Utility

We can achieve equity in accesses by trading off webpage size, which serves as a proxy for webpage quality. We capture this tradeoff through the Cobb-Douglas utility function [14, 54].

Let $U_i(W, A)$ by the utility of user i, where A is the expected number of website accesses available to the user and W is the average webpage size in the region. We assume that U_i is a concave function in both A and W. In particular,

$$U_i(W, A) = a \cdot log(W) + b \cdot log(A) \quad (2)$$

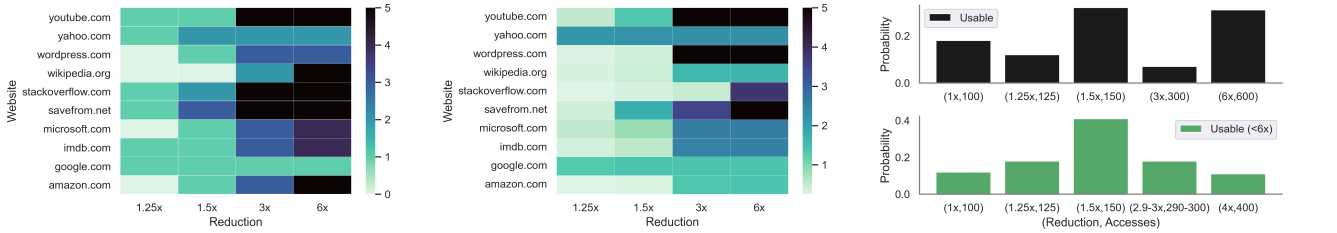
where a and b are positive constants that denote the weight of each attribute in the user's utility. The trade-off between W and A depends on the slope of this function, which is equivalent to the number of units of W that the user is willing to give up to get an additional A, while keeping utility constant. Mathematically, this is equal to the ratio of the partial derivatives of U_i with respect to A and W, that is, $\frac{a/W}{b/A}$.[12]

We can also show that for U_i to increase when W decreases and A increases, the following condition must hold: $\frac{b/A}{a/W} > \frac{dW}{dA}$. This condition implies that for a user to experience a utility gain, the willingness to give up quality to get more access must be greater than what the user would have to give up as a result of equalizing access across regions. Furthermore, users can have differing willingness to give up quality for access, depending on their current consumption of W and A, which reflects variations in the price of broadband and income levels. Thus, the utility gain (or loss) from equalizing access will also vary based on these features.

4.2 User Study

To quantify the quality-access trade-off, we conducted a user study with 100 participants for which an Institutional Review Board (IRB) approval was obtained. Users were recruited from a university campus via email and included students, staff, and faculty. A survey typically lasted between 10-15 minutes. The median age in our sample was 24 years and the median income was between USD 94 and 187 per month. To put this in perspective, the minimum monthly

[12] The slope of the utility function can be determined by taking the total derivative of U and setting $dU = 0$, as we are changing A and W such that the utility remains constant. This implies $dU = \frac{\partial U}{\partial A} dA + \frac{\partial U}{\partial W} dW = 0$. Re-arranging gives $\frac{dW}{dA} = -(\frac{\partial U}{\partial A} / \frac{\partial U}{\partial W})$.



(a) Optimizations-Reduction **(b) User Ratings** **(c) Quality-Access Trade-off**

Figure 4: (a) The heatmap shows the optimizations needed to achieve different page size reductions in terms of their impact on webpage quality on a scale of 0–5. 0 refers to optimizations that cause little or no change in quality (e.g., transcoding images into WebP format), 1 refers to reducing image quality or removing some external JS, 2 corresponds to removing all images, 3 to removing all images, some external JS, and the page usable, 4 to removing all images and external JS (page is usable), and 5 to removing all images and JS (page is unusable). (b) heatmap of user ratings on page look and content similarity with the original page (lighter shades reflect greater similarity), and (c) distribution of participant choices for page size reduction and Web accesses.

wage in the country of the user study was less than USD 94. Our sample included 31% females, 68% males, and one participant chose the 'other' category. The average monthly expenditure on mobile broadband subscription as a ratio of the monthly income in the sample was 4.1%, which is greater than the affordability target of 2% set by the UN Broadband Commission.

Webpage complexity and optimizations. We selected 10 websites that were common in the Alexa top 1000 websites list across 63 (out of 82) developing countries. All of the developing countries had at least 7 of the 10 chosen websites[13] in their Alexa top 1000 list. For each website, we created four low complexity versions of the landing pages that were 1.25×, 1.5×, 3×, and 6× smaller in size than the original version by applying different Web optimizations. While all webpages remained usable for up to 1.5× page size reduction, 8 remained functional with 3× reduction, and 5 pages with 6× reduction. Fig. 4a shows the heatmap of optimizations, with lighter shades representing ones with little or no quality impact and darker shades representing larger quality impact, as a function of page size reduction. We observe that for some websites 1.25× is achievable by just converting images into WebP format, while for other websites 1.5× reduction can be achieved by reducing image quality between 25%–75%. However, for several websites, we had to remove all images and some/all external JS to achieve 6× page size reduction.

Visual perception of pages. We showed 5 versions of each website to every participant and asked them to rate page look similarity (scale: 0–10) and page content similarity (scale: 0–10). Fig. 4b shows the heatmap of the average page look and content similarity ratings (normalized to a maximum score of 5, with 5 representing maximum dissimilarity). While almost all websites were rated to be quite similar to the original version at 1.5× reduction, some exhibited a good degree of similarity even at 6× reduction (e.g., wikipedia.org). However, some websites (e.g., youtube.com, savefrom.net) resulted in stark dissimilarities at large page size reductions due to the removal of visually important resources.

Quality-Access trade-off. Fig. 4c shows the distribution of user choices for different combinations of webpage quality (indicated by the size reduction factor) and the number of monthly Web accesses available to similar quality websites. For the 5 websites that remained usable with 6× reduction, participants chose options (1.5×,125) and (6×,600) with probabilities 0.32 and 0.31, respectively (upper plot). This indicates that some users preferred 600 accesses despite 6× lower page quality whereas others preferred higher quality pages even if it meant fewer Web accesses. For websites that were not usable with 6× reduction, the most popular combination was (1.5×,150) with a significant number of users choosing combinations with more than 2.9× quality reductions. These results suggest that a significant fraction of users are likely to observe a utility gain by trading off page quality for more Web accesses, as indicated in §4.1.

[13]These included {google, yahoo, microsoft, imdb, wordpress, amazon, stackoverflow, youtube}.com, wikipedia.org, savefrom.net.

5 FRAMEWORK DESIGN

We now introduce the Affordable Web For All (`AW4A`) framework, which provides a systematic way of transcoding webpages into low complexity versions to meet the affordability target set forth by the `PAW` index. The `AW4A` framework aims to maximize webpage quality while ensuring that the resulting page is no larger than the target size. In this section, we present `AW4A`'s key design goals and principles, and provide an overview of the framework.

5.1 Design Goals

Motivated by the insights from our analysis of geographic variations in mobile broadband prices, average income levels, variations in webpage sizes, and the need to consider affordability, we set forth the following design goals for `AW4A`:

G1 *Web complexity adaptation based on affordability and page quality*. The framework should be able to adapt the complexity of webpages based on broadband prices and income levels across geographical regions to achieve affordability targets while maximizing page quality.

G2 *User privacy and website operator consent*. The framework should preserve the end-to-end principles of TLS afforded by the existing Web. Moreover, Web complexity reductions should occur with website operator consent.

5.2 Design Principles

The above design goals lead to the following design principles for `AW4A`.

- To meet **G1**, `AW4A` explicitly accounts for affordability constraints in its design. It relies on `PAW` to provide page reduction targets and a number of metrics for capturing webpage quality.

Resource	Example Optimizations
Image	Transcoding to size efficient formats (e.g., Webp, AVIF) [25], reducing image quality, image resizing (e.g., to their CSS attributes width and height) [33], image compression [12], image removal [49]
JS	Removing unused JS [52], using lighter JS frameworks [43], removing non-critical JS [18], and compression (e.g., using `gzip`)
CSS	Resizing images embedded by CSS [33], convert external CSS to inline CSS [49], minification, and compression [21]
WebFont	Removing optional metadata (e.g., font hinting, kerning), font subsetting, font compression (e.g., EOT and TTF formats) [22]
IFrames	Minification, compression [21] and removal [4]

Table 2: Some example optimizations for different Web resources.

- To meet **G2**, AW4A transcodes webpages at the *server-end* rather than at a proxy server. This is unlike existing proxy-based approaches (e.g., Free Basics and Google Web Light) that raise privacy concerns as they break the end-to-end principles of TLS. Moreover, unlike Web Light, AW4A requires website operators to create low-complexity versions of their pages to meet their business needs while improving the affordability of access for users.

5.3 AW4A Overview

A high-level view of AW4A's design is shown in Fig. 5. Given a page to optimize and a target size derived via the PAW index, AW4A aims to maximize page quality subjective to the page size constraint. To achieve this, AW4A applies a set of resource optimizations involving different Web objects (e.g., images, JS) that it applies in two stages; see Table 2.[14] In stage-1, AW4A applies optimizations that reduce the size of a page without degrading page quality (e.g., minification of JS, removing non-essential JS elements, transcoding images into data efficient formats). If the target size is not met, the page moves to stage-2, where AW4A applies a set of resource optimizations that reduce the page size but also degrade the page quality. Depending on the desired page complexity, different trade-offs emerge between page size and page quality in this stage. After stage-2, the optimized page is generated to be served to the users whenever requested.

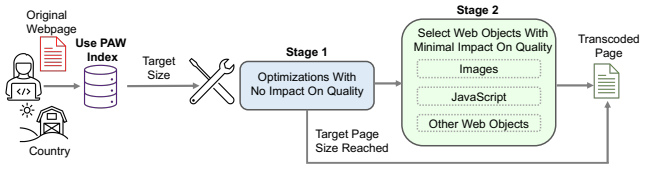


Figure 5: A high-level view of the AW4A framework, which takes a set of inputs and passes a webpage through two stages of resource optimizations to generate a low-complexity version of the page.

5.4 Developer API

AW4A provides an API to web developers to adjust parameters to suit their needs. The API's key components are as follows.

Desired page reduction. The complexity of the served webpage depends on the users' country of access, which is needed for calculating the PAW index. However, some users may not want page complexity to be determined by the country of access. Thus, the developer can input any level of desired page reduction and create low-complexity versions or 'tiers' of the original page.

Weights for Web objects. The web developer can specify which Web object types are more important for their website. For instance, for some websites reducing the resolution of images may be preferable over removing JS, so they may give image objects less weight. A web developer can determine the weights of Web objects based on domain knowledge about the website or through user studies.

Minimum image quality threshold. This parameter determines the minimum acceptable quality for images in terms of the Structural Similarity Index Measure (SSIM value) [51]. A lower threshold allows for greater page size reduction albeit at the cost of lower image quality.

[14]The AW4A imposes no restrictions on the kind of Web objects it uses for page size optimization. Thus, as new Web resource optimizations become available, they can be incorporated into AW4A.

5.5 User Perspective

With AW4A, a web developer can create low complexity versions of a webpage for different countries based on the page size required to achieve a PAW index ≤ 1. Moreover, if a user does not want their location to impact the complexity of the served page, they can specify the page complexity best suited for them from among the available options. We envision this communication between the user and web developer to occur through a browser. The user can set up their profile on the browser with the following options:

- **Country sharing:** A switch that can be activated to share the country-level location of the user with the website, which provides the version of the webpage that is considered affordable in the user's country of access.
- **Percentage savings:** The level of savings preferred by the user is shared with the accessed website. It serves the version of the webpage that provides the percentage data savings closest to the preferred level shared by the user.

The control flow for the user is shown in Fig. 6.

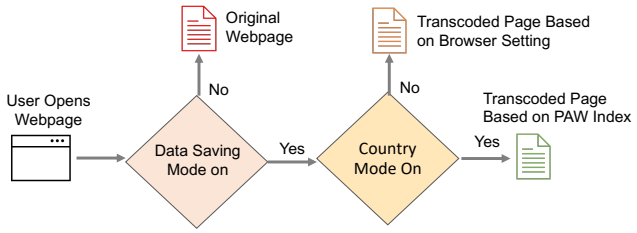


Figure 6: Control flow for the user accessing a webpage in AW4A.

6 AW4A OPTIMIZATION & ALGORITHMS

6.1 Objective Function and Constraints

AW4A's objective is to transcode a webpage such that the size of the resulting page is below the target size while maximizing the overall quality of the page. This can be achieved by solving the following

optimization problem.

$$maximize \ \frac{\sum_{i=1}^{k} w_i \times Q_i}{\sum_{i=1}^{k} w_i}, \tag{3}$$

$$page\_size = \sum_{i=1}^{k} b_i \leq target \tag{4}$$

where k is the total number of objects on the webpage (e.g., images, JS, CSS), w_i represents the weight (or priority) assigned to object i by the developer, and b_i is the number of bytes contributed by object i in the transcoded page. Thus, $page\_size \geq 0$ is the total size (in bytes) of the transcoded page which is bounded by a *target* value obtained from the PAW index or specified by the developer. The optimization function also includes an object 'quality metric' Q_i defined by the developer.[15] The objective function is connected to the optimization constraint by considering a relationship between the object quality metric Q_i and the byte contribution b_i by that object in the transcoded page, such that $Q_i = f_i(b_i)$. Intuitively, the quality metric of an image should measure how visually different it appears on the two pages: the original page and the transcoded page. Similarly, a JavaScript quality metric would quantify how the functionality of a page has changed due to the removal of a script while transcoding.

Thus, the objective function (Eq. 3) is the total quality of a page computed as the weighted average of the quality of the objects on the transcoded page. Finding a solution to this optimization problem requires us to answer the following questions: (i) *how are the object quality metrics defined? (for images, JS, etc.)*, (ii) *how can we assign priorities/weights to objects on the page?* and (iii) *how do we select the combination of transformed Web objects that will maximize the overall page quality such that the total page size is below the target size?* We address these questions by choosing quality metrics, object weights, and the target page size.

6.2 Object Quality and Priority

While our objective function (Eq. 3) provides a flexible expression for capturing the overall quality of a page, we consider only *images* and *JS* objects, both of which collectively make up the bulk of the bytes in webpages (see Fig. 7).[16] We further collate all objects of the same type (all images separately from all JS objects) and use a single holistic quality metric for all objects of the same type; QLUE Similarity Score (QSS) for images and the QLUE Functionality Score (QFS) for JS objects [24], as special cases of the generic objective function.[17]

Quality and priority for images. The quality metric QSS is based on SSIM and is used to measure the visual similarity of two webpages on a scale of 0 to 1, where 1 indicates that the webpages are identical to the human eye and 0 signifies maximum dissimilarity. It is calculated as follows:

$$\frac{\sum_{i=1}^{k} a_i \times s_i}{\sum_{i=1}^{k} a_i} \tag{5}$$

where k is the number of images on the page, a_i is the area of image i on the page, and s_i is the SSIM (a metric used for quantifying the structural similarity between two images on a scale from 0 to 1 [51]) of the reduced version of image i relative to the original one. It can readily be seen that the QSS definition maps to our generic objective function with $Q_i = s_i$ and $w_i = a_i$ when only image objects are considered. Thus, QSS computes the weighted average of the SSIM of images on a page, weighted by the area they occupy; this is because changes in the quality of larger images tend to be more noticeable to the human eye [23].

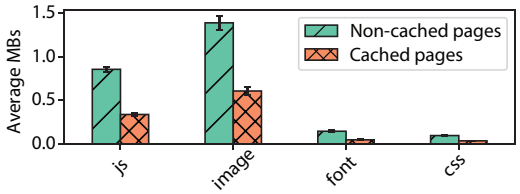


Figure 7: Average total bytes contributed by different Web objects to the total network bytes transferred (for both non-cached and cached page versions) in our dataset. The y-axis shows the average total MBs of JavaScript, images, fonts, and CSS on each webpage, considering all pages in our dataset. Error bars indicate 95% confidence intervals.

Quality and priority for JavaScript. The holistic quality metric QFS assesses the functionality of the page by using a browser interaction bot, which clicks and triggers every event on a given webpage and its reduced version, takes screenshots and compares the resultant screenshots of the original and reduced pages using SSIM. A limitation of this metric is that it only captures the functionality which results in a visual effect.[18] The QFS can also be viewed as a special case of the generic objection function with Q_i representing the average SSIM values (where each SSIM value is computed over the entire page considering it as a single image) for all underlying events in a JS object. The weight w_i in the generic objective function (Eq. 3) is the total number of the underlying events for a given JS object. A transcoded page that retains 100% functionality (at least visually) will receive a QFS score of 1.

We use both QSS and QFS and leave it up to the developer to assign weights to their weighted sum. For example, a news site may prioritize appearance over functionality and may give a higher weight to QSS than to QFS.

Optimization Structure. Considering QSS and QFS as the quality metrics in our optimization framework makes the solution to the problem practically inefficient for pages with a large number of objects. Intuitively, the intractability is due to two reasons: (i) integer constraints necessary to model page sizes (ii) non-convexity of the objective function[19] that involves the use of SSIM.[20] See Appendix A.2 for a more detailed discussion.

[15] We expect the 'quality metric' to be the same for objects of the same type (e.g., SSIM for images), though the framework allows these to be different.

[16] Even though ads may contribute a significant number of bytes to the overall page size, we do not remove them because this may be undesirable for website owners whose revenue may be largely dependent on ads. That said, our approach is flexible and allows the reduction of any combination of Web objects (which could include ads if the website owner desires).

[17] Both QSS and QFS have been evaluated using a user study where both metrics were more discerning than human evaluators.

[18] As newer quality metrics are introduced, they can be added to the optimization framework to better quantify the total quality of a webpage.

[19] Note that finding the exact global solution of a non-convex problem is known to be NP-hard [19, 32, 38].

[20] SSIM is non-monotonic as a function of image size (in bytes), specifically for image objects in JPEG format, as shown in Fig. 8.

7 APPROXIMATION ALGORITHMS

We design two approximation algorithms for solving the optimization problem for images and JavaScript. The first algorithm, Grid Search, uses a brute force approach over a discretized space of the problem. However, this approach is slow to scale for large webpages and may not be feasible for dynamic websites (e.g., news websites). Therefore, we devise a second heuristics-based algorithm, HBS, that uses a set of heuristics to rank objects and then uses a greedy approach for applying optimizations in a sequential manner. Either one of these algorithms can be used in stage-2 of the framework and is only invoked if the target size constraint is not met in stage-1 (i.e., after applying optimizations with no effect on page quality).

7.1 Grid Search

Grid Search uses a brute force approach to evaluate all possible combinations of Web resource optimizations (e.g., images and JS) to find the combination that maximizes the page quality while satisfying the target page size constraint. For images, it discretizes the SSIM values (which correspond to image resolution degradations) into uniformly spaced intervals, whereas, for each JS object, we consider only two states (0 refers to the removal/absence of a script and 1 refers to its presence). For ease of exposition, we describe Grid Search with respect to image optimizations in this section.

With Grid Search, the SSIM interval [SSIM_{min}, 1] is discretized into uniformly spaced values, where SSIM_{min} is the minimum desired quality of images (e.g., 0.9).[21] Grid Search then creates multiple versions of all images present on a website, one for each SSIM value. These versions are realized by reducing the quality of images and/or transcoding them into different formats (e.g., WebP) while maintaining their original dimensions. We then compute the QSS for each possible combination of images and sort these combinations by QSS. Finally, we search for the combination with the highest QSS that meets the page size constraint.

Time Complexity. The worst-case time complexity of Grid Search (if we only consider images) is in $O(v^n)$, where v is the number of possible versions of an image and n is the total number of images on a website. Consequently, Grid Search may not be a practical approach for (i) websites with a large number of images and/or scripts or (ii) dynamic websites, e.g., news websites, which would have to bear the cost of applying Grid Search every time the objects on the page change.

7.2 Heuristics-based Search

To overcome the limitations of Grid Search, we propose a heuristic-based search algorithm (HBS) for solving the optimization problem. HBS evaluates two optimization approaches to meet the target page size. In the first approach, it removes unused JS using an existing dead code elimination tool. If the target size is not achieved, HBS then applies image optimizations using a greedy approach, which we call Rank Based Reduce (RBR), by ranking images based on a set of heuristics. The page quality score is calculated using QSS and QFS. In the second approach, HBS *only* applies RBR to perform image reductions and calculates the QSS of the page. The QFS of the page will always be 1 in this case because no JS optimizations are used. The approach that meets the target is chosen. If both approaches meet the target, then the one with the higher page quality is chosen.

JavaScript Reduction. Many websites have unused JS functions, which, if removed, can help reduce the page size with minimal change in page quality. To this end, we use Muzeel [35]; a dead code elimination tool that removes JS that is not associated with the user interactive events and all their dependent events. In particular, this involves removing: (i) JS functions that only read or fetch information from the HTML and (ii) JS functions that dynamically add/modify/remove elements to/from the DOM. Unused code is identified by using a browser interaction bot that triggers every event on the page and checks for visual changes. While there are other approaches for removing non-critical JS [16, 18] which can be incorporated in the `AW4A` framework, we consider Muzeel as it aims to only remove unused code.

Image Optimization. Finding the optimal QSS given a target bytes constraint is a difficult task for two key reasons:

- The relationship between the bytes of an image and its SSIM value is not monotonic; see Fig. 8. This makes it difficult to solve the optimization problem efficiently.
- The relationship between SSIM and bytes reduction *varies* across images (Fig. 8). The exact functional form depends on several factors (e.g., characteristics of the image, software/reduction algorithm used).

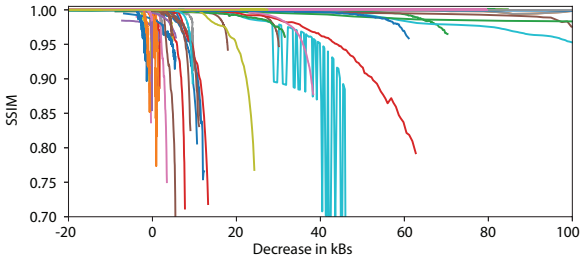


Figure 8: SSIM as a function of the decrease in image bytes for 100 different images drawn randomly from the landing pages of Alexa top 100 websites. Changes in image bytes are realized by varying the image resolution at different levels. Observe that for a given reduction in bytes, images show different changes in SSIM and some images show non-monotonic behavior with respect to SSIM.

To overcome these challenges, RBR uses a set of heuristics to rank images based on their potential for bytes savings. To facilitate this, we introduce the concept of an image's ***reducibility***. We consider image A to be more 'reducible' than image B if it is preferable to reduce A for (i) meeting the page size constraint and (ii) maximizing the resultant page quality. Thus, image A will be more reducible than image B if it gives a greater reduction in bytes for the same change in SSIM (given that both images have the same area on the page). After computing the reducibility of images, we apply a greedy algorithm to optimize images in order of their reducibility

Heuristics. To estimate reducibility, we use two heuristics:
(1) Area: We measure the area covered by images on a page in pixels. Images that cover a smaller area can withstand greater quality reduction without these changes being perceived by the user. This is similar to the concept of *viewing distance*, i.e., the further away an image is, the fewer details are visible to the human eye, and the user's quality of experience is affected less by changes in image quality [23]. Thus, images with smaller areas are ranked higher.

[21] In our evaluation, we divide the interval by 10, resulting in 11 distinct SSIM values and thus 11 versions of each image.

(2) Bytes Efficiency: We measure the *Bytes Efficiency* of an image using the following formula:[22]

$$\text{Bytes Efficiency} = \frac{\vec{\Delta}\text{bytes}}{\vec{\Delta}\text{SSIM}} \tag{6}$$

This equation gives us the change in bytes per unit change in SSIM. To compute this, we reduce the image to the minimum desired quality threshold SSIM using resolution reduction to find the largest possible reduction in bytes. A higher Bytes Efficiency indicates that the image has higher reducibility.

We use linear normalization to ensure that the heuristics values are between 0 to 1, where 1 indicates the highest reducibility. The heuristic values are then combined to give a score using the *weighted sum approach*. The greater the weighted sum score, the higher the reducibility of the image. Images with higher reducibility are reduced first. Algorithm 1 in Appendix A.1 shows the pseudocode of RBR.[23]

Image Transformations. Once the images are ranked, we use two methods from ImageMagick [28] to optimize images:

- *WebP Format:* In our dataset of 72,069 webpages, the two most popular image formats were JPEG and PNG. While JPEG is highly size-efficient, PNG images tend to have larger sizes compared to JPEGs at the same visual quality. We first transcode PNGs into the WebP [25] format to reduce its size. This allows us to retain the image transparency channels (unlike with JPEG compression) and maintain page similarity. However, this is not a blanket approach, so we only convert a PNG to WebP if the SSIM of the WebP is equal to or greater than the SSIM threshold and the Bytes Efficiency heuristic is better in the case of WebP.
- *Resolution Reduction:* If the target is not met after format conversion, we start optimizing images in order of their reducibility. We linearly reduce the resolution of the image until we reach the SSIM threshold.

Time Complexity. The time complexity of RBR depends on two factors: (i) n, the number of images on a webpage, and (ii) v, the number of possible reductions for each image. Algorithm 1 has the worst-case time complexity of $O(nv)$.[24]

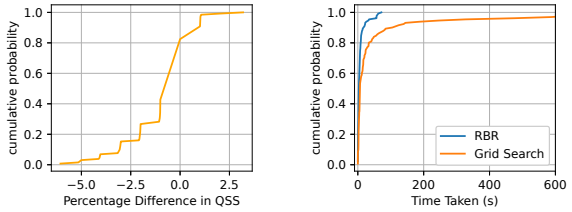
8 EVALUATION

In this section, we present the evaluation of `AW4A`. We measure (i) the QSS and time taken by RBR and Grid Search to find transcoded webpages, (ii) the number of URLs that meet the target size after country-wise reduction of page sizes using RBR for 25 developing countries, (iii) the resultant quality of webpages after reduction with HBS, and (iv) comparison with Brave [3] and Opera Mini [6].

We outline our key insights from the evaluation as follows: (i) RBR produces webpages with comparable QSS to Grid Search (with an average difference of only 0.76%) despite taking a significantly shorter time to run, (ii) a significant number of URLs can meet their country's `PAW` index target using image reductions alone, (iii) using HBS, 50% of webpages (with varying page sizes) maintain a quality of ≥ 0.98, and (iv) despite greater overall % reduction in page size, HBS produces webpages of comparable quality to Brave and Opera Mini.

8.1 RBR vs Grid Search

We find that RBR achieves QSS that is within 6.1% of the QSS obtained via Grid Search while taking 15.9× less time to complete for the same level of reduction. In our evaluation, the minimum SSIM threshold (or Q_T) for each image was set to 0.9 for both RBR and Grid Search.[25] For RBR, we gave the Area and Bytes Efficiency heuristics equal weightage. We ran all tests on a server (running Ubuntu version 20.04.5) with an Intel 2.2 GHz CPU with 40 Cores and 502 GB of available RAM. We ran both algorithms on a set of 50 unique websites (from the Alexa top sites) with varying reduction levels (5-60% overall page size reduction) for a total of 171 reduced webpages that were common in both RBR and Grid Search.[26] The number of images on the webpages ranged from 1-40.



(a) QSS **(b) Time**

Figure 9: Difference in (a) QSS of generated webpages and (b) time taken to generate the webpages by RBR and Grid Search. Positive values in (a) indicate cases in which RBR attained a larger QSS.

QSS. Fig. 9a compares the two algorithms based on the QSS of their transcoded pages, with positive values indicating cases when RBR outperformed Grid Search. While Grid Search achieved a larger QSS on average than RBR, the average difference was just −0.76%, with the largest difference being −6.1%. In 18% of the cases, RBR outperformed Grid Search, which we attribute to the precision of the discretized SSIM values.

Time. Fig. 9b shows the time taken by RBR and the Grid Search to complete. We used a timeout value of 3 hours for each test. Of the 171 tests, Grid Search timed out on 40 tests. Among the tests that did not time out, we find that the average time taken by Grid Search and RBR was 127 (σ = 929) and 8 (σ = 13) seconds, respectively. We also note that as the number of images on the webpage increases, the time it takes to run Grid Search increases drastically (resulting in more frequent timeouts).

8.2 Country-wise Page Reduction with RBR

For the 25 countries with a `PAW` index greater than 1 for a data and voice low-usage plan, we ran RBR on their respective Alexa top 1000 webpages. These webpages were reduced in inverse proportion to the `PAW` index for that country. We find that a significant percentage of the URLs can be reduced to $\frac{1}{\texttt{PAW}}$ of their original size with image optimizations alone (see Fig. 10). For instance, 91.4% of URLs in the Lebanon dataset can be brought within the target using image

[22]The formula only considers the points on the function relating bytes and SSIM which are monotonic.
[23]The configurable parameters of RBR are presented in Appendix A.1.
[24]For a detailed explanation, see Appendix A.1.
[25]According to the MOS scale, 0.9 represents Fair Quality [37].
[26]Not all webpages could be reduced to 60% due to the quality constraint. Moreover, in some cases, images made up < 60% of the page. The reduction levels we used varied by 5%. That is, we tried reducing each webpage by 5%, then 10%, then 15%, and so on for a total of 12 levels.

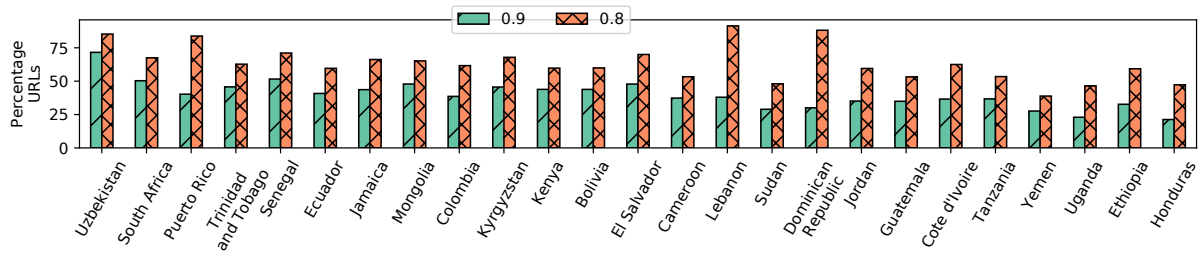
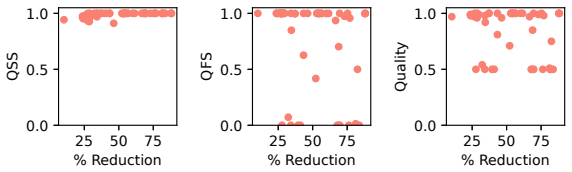


Figure 10: Percentage of URLs that can be reduced to $\frac{1}{\texttt{PAW}}$ of their original size using image reduction with RBR, given the SSIM threshold (Q_T) is 0.9 and 0.8. Countries are sorted in ascending order of PAW index.

reductions alone (with $Q_T = 0.8$). These reductions are achievable while maintaining a high level of image quality. The average QSS scores and additional results are reported in Appendix A.3.4 and A.3.5.

8.3 HBS Quality



(a) QSS **(b) QFS** **(c) Quality**

Figure 11: (a) QSS (page content similarity), (b) QFS (page functionality similarity), and (c) overall quality (average of QSS and QFS) of the resultant webpages after page size reduction by HBS.

We use a complete HBS implementation (including JavaScript and image optimizations using Muzeel and RBR, respectively) to reduce 60 unique URLs (amongst the Alexa top 1000) by 10–88% of their original page size (the median page size reduction was 43.3%).[27] The resultant webpage qualities, as captured by QSS and QFS, are shown in Fig. 11. QSS and QFS have been shown to be correlated with user-perceived similarity to the original webpage [24]. Note that 25% of the webpages maintain a webpage quality of 1. Moreover, 50% of the webpages maintain a quality of ≥ 0.98. The 10 pages with the highest reduction (77%–88%) have an average quality of 0.72. Overall, these results demonstrate the effectiveness of HBS in reducing page sizes while maintaining high page quality.

Comparison with Brave and Opera Mini. We evaluate and compare page size reductions and webpage quality of HBS with two state-of-the-art industry solutions that are widely deployed, namely Opera Mini [6] and Brave [3]. Opera Mini forwards users' Web requests to their proxy server, where the pages are first requested and then compressed before being sent to the user. Opera Mini is estimated to have about 170 million users [6]. Brave is a privacy-focused browser that automatically blocks website trackers and online advertisements by default.

- *Page Size Comparison:* In our evaluation, Opera Mini achieved an average page size reduction of 30.5% (on its High Image Quality setting). However, since Opera Mini allows only a subset of DOM events, it is prone to breaking interactive sites with JS. Some excluded events are key functionalities (such as keypress and scroll) [41]. Using the default settings (blocking ads, cookies, etc.), Brave gives an average page size reduction of 14.6%. By activating the 'block scripts' feature (that removes third-party JS), an average page size reduction of 57.3% can be achieved. The block scripts functionality (which intends to remove third-party scripts to avoid fingerprinting) breaks widgets that are loaded with third-party scripts. It uses a whitelist to avoid removing these required features. However, the scope of this approach is limited, and some widgets are still left out. We find that 4% of the pages break completely, and many others lose critical functionalities [15] (Brave and Opera Mini page size results are shown in Appendix A.3.6). A benefit AW4A provides over these approaches is that it offers more fine-grained control over the reduction of webpages. This enables it to tailor the reduction levels to the countries of access (which may have varying reduction requirements as determined by the PAW index). Moreover, the approaches adopted by Brave and Opera Mini are agnostic to the importance of each Web object on the webpage, while AW4A explicitly accounts for the relative significance assigned to each Web object by the Web developer.
- *Quality Comparison:* We conducted a user study with 35 participants to compare the qualities of the pages produced by Opera Mini, Brave, and HBS. Each user was asked to compare 5 Opera Mini and 5 Brave versions of the webpages with their HBS counterparts. Image quality was set to 'Medium' for Opera Mini and script blocking was enabled for Brave. Ad blocking was disabled in both browsers since HBS does not remove ads. The page sizes of 22 unique URLs (from amongst the Alexa global top 1000 pages) were collected on both Opera Mini and Brave. In total, there were 9 Brave and 21 Opera Mini webpages. [28] The percentage reductions achieved by both browsers (as compared to the page size on Chrome) were input into HBS to create reduced pages with similar page sizes. Due to the nature of JS reduction through Muzeel and the dynamic updating of some websites, exact page sizes could not be achieved. Instead, HBS pages tended to undergo a greater reduction than their Brave and Opera Mini counterparts. The average differences in percentage reduction achieved by HBS compared to Opera Mini and Brave were 11% and 7%, respectively. We find that, despite the greater average reduction achieved by HBS, it produces webpages with comparable quality to those produced by Opera Mini and Brave. In the Opera Mini

[27] We set a target reduction of 30%. However, several sites could not reach this target due to the SSIM constraint (Q_T) and the lack of JS optimization opportunities. Similarly, several sites overshot the target reduction due to JS reduction with Muzeel, which is not adjustable in its reduction. In the future, as JS reduction becomes more sophisticated, AW4A can adopt adjustable JS reduction strategies.

[28] Some Brave websites were heavily reduced (which broke the page) and HBS could not reach that page size target given the quality constraints.

comparison, users preferred HBS for 11 out of the 21 sites while in the Brave comparison, they preferred HBS for 5 out of the 9 sites.

9 INCENTIVES FOR STAKEHOLDERS

We now discuss incentives for different stakeholders for using the AW4A framework.

Website operators. AW4A gives website operators the option to offer their services at reduced quality, thereby creating a differentiated service offering. This new market for reduced quality services can appeal to existing users who are data constrained and new users who were previously unable to afford these services. The increase in the number of Web accesses and users visiting a website will likely increase advertising revenue due to higher click-through rates. Finally, website operators have the freedom to determine the extent to which they want to differentiate their services (by reducing quality). They can base their decisions on profit and social motives.

Mobile network operators. Data consumption is highly sensitive to market prices and service affordability [55]. An affordable Web framework can make it attractive for mobile service providers to offer smaller data plans, as users can derive more value from the same number of MBs. This can bring more users online – specifically those who were previously shut out from the market for these Web services due to affordability constraints. In the medium to long term, prices can decline if there is an increase in supply of mobile data services by network operators, for example, due to investments in new infrastructure to improve capacity and reach.

Users. Our framework will offer a choice to users between different page qualities rather than exclusively offering a separate class of Web content to users. Thus, users can have more accesses if they are willing to trade off page quality. As we expand the affordable consumption choice set for these users, their utility should increase.

10 RELATED WORK AND DISCUSSION

Related Work. There is a growing body of work on optimizing the size of Web objects [7, 12, 16, 17, 35, 42, 43, 45, 49]. In this section, we discuss the most closely related works.

Google Web Light transcodes webpages on the fly to deliver lighter pages to users on slow mobile clients. [49]. While it reduces the median webpage size by 12×, it frequently breaks page functionality. Facebook Discover [42] is similar to Web Light and only supports low-bandwidth features on websites (e.g., text and small images). Web Light and Discover both attempt to transcode every page without publishers' consent. Free Basics requires publishers to come up with a version of their website that complies with a set of guidelines (e.g., a page cannot have JS or large images) before they can be hosted on the Free Basics platform. Lite-Web [16] removes non-essential JS elements using SlimWeb [17] and JSCleaner [18]. It then uses Muzeel [35] to optimize the remaining JS elements on the page. However, Lite-Web only considers JS optimizations, which limits its effectiveness in improving the affordability of pages. All these schemes are proxy-based solutions that break the end-to-end principles of TLS. Opera Mini [6] sends the user's web requests to their proxy server. This proxy server requests the webpage and serves the end user the compressed version. This process is prone to breaking interactive websites that rely heavily on JS. Brave [3], on the other hand, is a privacy-focused web browser that automatically blocks online advertisements and website trackers in its default settings, reducing the data downloaded on the device.

Discussion. We now discuss some limitations of our work.

- *Non-landing pages and caching*. We only considered landing pages in our work, which are more frequently requested by users, but the complexity of inner pages will also impact data usage [13]. We did not consider (i) scenarios involving a user logging into a website and (ii) the impact of user browsing habits, including visit frequencies and browser application, on client-side caching, which may impact data usage for repeated visits to a website. In future work, we plan to incorporate these aspects in our evaluation.
- *Video*. While rich multimedia content, such as video, has not been the focus of works on developing countries that target affordability, we believe future trends in video compression (e.g., WebM [10], VP9 [8]) and customization of video resolutions will likely make it plausible to serve lite video content. This remains part of our future work.

11 CONCLUDING REMARKS

Web affordability is a key barrier constraining Internet use and adoption. In this work, we assembled and analyzed a dataset from 99 countries to highlight opportunities for reducing Web complexity to meet UN Broadband Commission's affordability target. We proposed a fairness metric and a novel framework for re-thinking Web architecture for affordability and inclusion. Our proposed framework systematically adapts Web complexity based on geographic variations in mobile broadband prices and income levels. Our evaluation shows that our framework can better balance affordability and web page quality while preserving user privacy.

12 ETHICAL STATEMENT

The user studies were approved by the Institutional Review Board (IRB). Participation was voluntary and informed consent was obtained from all participants. Participants could fill out *'Other/Do not know'* categories for questions, e.g., gender. All data was processed anonymously.

13 ACKNOWLEDGEMENTS

We thank Waleed Hashmi and Tofunmi Kupoluyi for their assistance in setting up QSS, QFS, and Muzeel. We thank Hafsa Akbar, Muhammad Ayain Fida Rana, and Danish Athar for their help with designing and conducting the Brave and Opera Mini comparison user study. We are also grateful to our shepherd, David Choffnes, and the anonymous reviewers for their valuable feedback.

REFERENCES

[1] Alexa - The top 500 sites on the web. http://www.alexa.com/topsites/countries.
[2] Android Go. https://www.android.com/versions/go-edition/.
[3] Brave browser moves to Chromium codebase. https://zd.net/3cZHDp7.
[4] Free Basics Platform. https://developers.facebook.com/docs/internet-org.
[5] http archive. https://httparchive.org/.
[6] Opera mini browser. http://www.opera.com/mobile/.
[7] Opera Turbo. http://www.opera.com/turbo.
[8] VP9 Overview. https://developers.google.com/media/vp9.
[9] Web Light: Faster and lighter mobile pages from search. https://support.google.com/webmasters/answer/6211428?hl=en.
[10] WebM: an open web media project. https://www.webmproject.org/.
[11] WebPageTest. https://www.webpagetest.org/.

[12] V. Agababov, M. Buettner, V. Chudnovsky, M. Cogan, B. Greenstein, S. McDaniel, M. Piatek, C. Scott, M. Welsh, and B. Yin. Flywheel: Google's data compression proxy for the mobile web. 2015.
[13] W. Aqeel, B. Chandrasekaran, A. Feldmann, and B. M. Maggs. On landing and internal web pages: The strange case of jekyll and hyde in web performance measurement. In *Proceedings of the ACM Internet Measurement Conference*, IMC '20, page 680–695, New York, NY, USA, 2020. Association for Computing Machinery.
[14] J. Biddle. Retrospectives: The introduction of the cobb-douglas regression. *Journal of Economic Perspectives*, 26(2):223–36, May 2012.
[15] Brave. Brave: How do i use shields while browsing?, 2022.
[16] M. Chaqfeh, R. Asim, B. AlShebli, M. F. Zaffar, T. Rahwan, and Y. Zaki. Towards a world wide web without digital inequality. *Proceedings of the National Academy of Sciences*, 120(3):e2212649120, 2023.
[17] M. Chaqfeh, M. Haseeb, W. Hashmi, P. Inshuti, M. Ramesh, M. Varvello, F. Zaffar, L. Subramanian, and Y. Zaki. To block or not to block: Accelerating mobile web pages on-the-fly through javascript classification, 2021.
[18] M. Chaqfeh, Y. Zaki, J. Hu, and L. Subramanian. Jscleaner: De-cluttering mobile webpages through javascript cleanup. In *Proceedings of The Web Conference 2020*, WWW '20, page 763–773, New York, NY, USA, 2020. Association for Computing Machinery.
[19] M. Danilova, P. Dvurechensky, A. Gasnikov, E. Gorbunov, S. Guminov, D. Kamzolov, and I. Shibaev. Recent theoretical advances in non-convex optimization, 2020.
[20] A. P. Felt, R. Barnes, A. King, C. Palmer, C. Bentzel, and P. Tabriz. Measuring HTTPS adoption on the web. In *26th USENIX Security Symposium (USENIX Security 17)*, pages 1323–1338, Vancouver, BC, Aug. 2017. USENIX Association.
[21] I. Grigorik. *Optimizing Encoding and Transfer Size of Text-Based Assets*, 2019. https://cutt.ly/EmqpXJU.
[22] I. Grigorik. *Reduce WebFont Size*, 2019. https://web.dev/reduce-webfont-size/.
[23] K. Gu, M. Liu, G. Zhai, X. Yang, and W. Zhang. Quality assessment considering viewing distance and image resolution. *IEEE Transactions on Broadcasting*, 61(3):520–531, 2015.
[24] W. Hashmi, M. Chaqfeh, L. Subramanian, and Y. Zaki. Qlue: A computer vision tool for uniform qualitative evaluation of web pages. In *Proceedings of the ACM Web Conference 2022*, WWW '22, page 2400–2410, New York, NY, USA, 2022. Association for Computing Machinery.
[25] K. Hempenius. *Fast load times*, 2018. https://web.dev/serve-images-webp/.
[26] J. Hjort and L. Tian. The economic impact of internet connectivity in developing countries. INSEAD Working Paper 2021/68/EPS, November 2021.
[27] HTTPArchive. *Report: State of the Web*, 2021. https://httparchive.org/reports/state-of-the-web.
[28] ImageMagick Studio LLC. Imagemagick.
[29] ITU. *ITU price data collection rules*, 2019. https://www.itu.int/en/ITU-D/Statistics/Documents/ICT_Prices/ICT%20Price%20Basket%20rules_E.pdf.
[30] ITU. *Measuring Digital Development: ICT Price Trends 2020*, 2020. https://www.itu.int/en/ITU-D/Statistics/Pages/ICTprices/default.aspx.
[31] W. Jack and T. Suri. Risk Sharing and Transactions Costs: Evidence from Kenya's Mobile Money Revolution. *American Economic Review*, 104(1):183–223, January 2014.
[32] S. Kameshwaran and Y. Narahari. Nonconvex piecewise linear knapsack problems. *European Journal of Operational Research*, 192(1):56–68, 2009.
[33] C. Kelton, M. Varvello, A. Aucinas, and B. Livshits. Browselite: A private data saving solution for the web. In *Proceedings of the Web Conference 2021*, WWW '21, page 305–316, New York, NY, USA, 2021. Association for Computing Machinery.
[34] B. Kondracki, A. Aliyeva, M. Egele, J. Polakis, and N. Nikiforakis. Meddling middlemen: Empirical analysis of the risks of data-saving mobile browsers. In *2020 IEEE Symposium on Security and Privacy (SP)*, pages 810–824, 2020.
[35] J. Kupoluyi, M. Chaqfeh, M. Varvello, R. Coke, W. Hashmi, L. Subramanian, and Y. Zaki. Muzeel: Assessing the impact of javascript dead code elimination on mobile web performance. In *Proceedings of the 22nd ACM Internet Measurement Conference*, IMC '22, page 335–348, New York, NY, USA, 2022. Association for Computing Machinery.
[36] Z. Li, W. Wang, T. Xu, X. Zhong, X.-Y. Li, Y. Liu, C. Wilson, and B. Y. Zhao. Exploring cross-application cellular traffic optimization with baidu trafficguard. In *13th USENIX Symposium on Networked Systems Design and Implementation (NSDI 16)*, pages 61–76, Santa Clara, CA, Mar. 2016. USENIX Association.
[37] A.-N. Moldovan, I. Ghergulescu, and C. H. Muntean. A novel methodology for mapping objective video quality metrics to the subjective mos scale. *2014 IEEE International Symposium on Broadband Multimedia Systems and Broadcasting*, pages 1–7, 2014.
[38] K. G. Murty and S. N. Kabadi. Some np-complete problems in quadratic and nonlinear programming. *Mathematical Programming*, 39:117–129, 1987.
[39] U. Naseer, T. A. Benson, and R. Netravali. Webmedic: Disentangling the memory-functionality tension for the next billion mobile web users. In *Proceedings of the 22nd International Workshop on Mobile Computing Systems and Applications*, HotMobile '21, page 71–77, New York, NY, USA, 2021. Association for Computing Machinery.
[40] P. B. of Statistics. The Global Social Mobility Report 2020 Equality, Opportunity and a New Economic Imperative., 2020. https://www.pbs.gov.pk/sites/default/files/pslm/publications/hies2018-19/hies_2018-19_writeup.pdf.
[41] Opera Mini. Opera mini and javascript, 2022.
[42] L. Pei, B. S. Olgado, and R. Crooks. Market, testbed, backroom: The redacted internet of facebook's discover. In *In CHI Conference on Human Factors in Computing Systems*, CHI '21, 2021.
[43] I. A. Qazi, Z. A. Qazi, T. A. Benson, G. Murtaza, E. Latif, A. Manan, and A. Tariq. Mobile web browsing under memory pressure. *SIGCOMM Comput. Commun. Rev.*, 50(4):35–48, Oct. 2020.
[44] F. Qian, K. S. Quah, J. Huang, J. Erman, A. Gerber, Z. Mao, S. Sen, and O. Spatscheck. Web caching on smartphones: Ideal vs. reality. In *Proceedings of the 10th International Conference on Mobile Systems, Applications, and Services*, MobiSys '12, page 127–140, New York, NY, USA, 2012. Association for Computing Machinery.
[45] R. Sen, S. Ahmad, A. Phokeer, Z. A. Farooq, I. A. Qazi, D. Choffnes, and K. P. Gummadi. Inside the walled garden: Deconstructing facebook's free basics program. *SIGCOMM Comput. Commun. Rev.*, 47(5):12–24, Oct. 2017.
[46] S. Singh, H. V. Madhyastha, S. V. Krishnamurthy, and R. Govindan. Flexiweb: Network-aware compaction for accelerating mobile web transfers. In *Proceedings of the 21st Annual International Conference on Mobile Computing and Networking*, MobiCom '15, page 604–616, 2015.
[47] Statista. Internet statistics 2023: Facts you need-to-know, 2022.
[48] Statista. Percentage of mobile device website traffic worldwide from 1st quarter 2015 to 4th quarter 2022, 2022.
[49] A. Tahir, M. T. Munir, S. M. Malik, Z. A. Qazi, and I. A. Qazi. Deconstructing google's web light service. In *Proceedings of The Web Conference 2020*, WWW '20, page 884–893, 2020.
[50] J. Vincent. *Facebook's Free Basics Service has been banned in India. The Verge*, 2016. https://www.theverge.com/2016/2/8/10913398/free-basics-india-regulator-ruling.
[51] Z. Wang, A. Bovik, H. Sheikh, and E. Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE Transactions on Image Processing*, 13(4):600–612, 2004.
[52] web.dev. *Remove unused JavaScript*, 2020. https://web.dev/unused-javascript/.
[53] L. Wheeler, R. Garlick, E. Johnson, P. Shaw, and M. Gargano. Linkedin(to) job opportunities: Experimental evidence from job readiness training. *American Economic Journal: Applied Economics*, 2022. forthcoming.
[54] Wikipedia. *Cobb–Douglas production function*, 2019. https://en.wikipedia.org/wiki/Cobb%E2%80%93Douglas_production_function.
[55] World Bank. World Development Report 2021 : Data for Better Lives., 2021. https://openknowledge.worldbank.org/handle/10986/35218.

A APPENDIX

Appendices are supporting material that has not been peer-reviewed.

A.1 RBR Pseudocode & Time Complexity

Algorithm A.1 describes the pseudocode of RBR.

Algorithm 1 Rank Based Reduce (RBR).
S_C: Current total image bytes
S_T: Target image bytes
Q_T: Minimum quality threshold of SSIM for each image.
PQ: A priority queue of images, where the priority of each image is its weighted sum (or reducibility) score

```
if S_C <= S_T then
    return True
end if
while PQ is not empty do
i = PQ.dequeue()
    while SSIM(i, reduce(i)) ≥ Q_T do
     i = reduce(i)
        if S_C <= S_T then return True
        end if
    end while
end while
```

Configurable Parameters. RBR has the following configurable parameters.

- *Weights of heuristics*. The weights of the two heuristics, Area and Bytes Efficiency, can be adjusted by the web developer according to the types and sizes of the images on a webpage. The default setting gives them both equal weight.
- *SSIM threshold*. The web developer can set Q_T, the minimum quality threshold for SSIM. A smaller value for Q_T will increase the number of possible levels of reduction on the page. The quality may or may not decrease as a result, depending on the type of images. It is possible that when the Q_T for each image is lower, fewer images may need to be transcoded. However, the average quality of a transcoded image will be reduced. A larger value for Q_T will result in a higher QSS, but a lower reduction in bytes may be possible given the tighter constraint on quality.
- *Resolution granularity:* A web developer can set the desired granularity for resolution reduction. A smaller granularity will increase the time taken by the algorithm but will likely produce a better QSS since the resolution reduction will be more fine-grained. A larger granularity will decrease the time taken but may produce a lower QSS. In some cases, it may not find an answer because it may miss a more precise resolution which gives a better SSIM value with a greater difference in bytes that does not invalidate the quality threshold.

Time Complexity. The time complexity of RBR depends on two factors: (i) n, the number of images on a webpage, and (ii) v, the number of possible reductions for each image. Algorithm 1 has the worst-case time complexity of $O(nv)$.[29] Algorithm 1 iterates over each image (in order of their reducibility), so the first for loop runs n times in the worst-case. For each image, there is a certain number of valid versions, v, that need to be checked iteratively until the target image bytes are reached, or it is the last valid version of that image. The value v depends on the resolution granularity on which the image versions are being considered. A version of an image is considered valid if the conversion does not increase the size (as compared to an old quality level),[30] or gives a valid SSIM (greater than Q_T).

RBR Output. Theoretically, if an answer exists, RBR will always return an answer if we assume a *monotonic* relationship between SSIM and image bytes, and the resolution granularity is infinitely small. This is because RBR considers the case in which the total bytes of images are at a minimum, which occurs when every image is reduced to the SSIM quality threshold. If the target image bytes are less than the minimum image bytes, then the answer cannot exist. However, as we observed in Fig. 8, the relationship between SSIM and images bytes is not monotonic in the case of JPEGs. Thus, the assumption that the minimum bytes are achieved when all images are at the SSIM threshold may not be true in general. Moreover, we cannot have an infinitely small resolution granularity so the optimal resolution for that image can be missed by RBR.

[29] If we assume monotonicity between SSIM and images bytes, we can use binary search to find the needed reduction, which will reduce the complexity to $O(nlog(v))$.
[30] This may occur in the case of JPEG images which are already compressed.

A.2 Hardness of `AW4A` Optimization

The general (transcoding) optimization problem stated in Equations 3 and 4 directly maps to the *Knapsack problem*. We demonstrate it by separately mapping the constraint and the objective function to the standard Knapsack cases. For simplicity, assume that the page being transcoded only consists of image objects. First, we consider the constraint:

$$\text{For Knapsack:} \quad \sum_{i=1}^{k} w_i x_i \leq W, \qquad x_i \in \mathbb{W} \tag{7}$$

$$\text{Transcoding:} \quad \sum_{i=1}^{k} b_i \leq T \tag{8}$$

Noting that b_i is the number of bytes of image i in the transcoded page. If B_i denotes the number of bytes of the same image on the original page, then $b_i \leq B_i$. We now show that the constraint in Eq. 8 maps to that in Eq. 7. Considering the byte-level granularity of images, we have $b_i = \alpha_i B_i$, where α_i can only take a discrete number of values and represents the fractional reduction of image i in the transcoded page. To proceed with the mapping, we define $B = \prod_{i=1}^{k} B_i$, and then $y_i = \alpha_i B \in \mathbb{W}$. Note that higher values of y_i indicate less reduction of the image i in the transcoded page. Thus, the constraint for the page transcoding problem can be written as:

$$\sum_{i=1}^{k} \frac{B_i}{B} y_i \leq T$$

which maps to the Knapsack constraint. Next, we consider the objective function (we only consider the numerator because the denominator does not affect the optimization):

$$\text{For Knapsack:} \quad \max \sum_{i=1}^{k} v_i x_i, \qquad x_i \in \mathbb{W} \tag{9}$$

$$\text{Transcoding (general):} \quad \max \sum_{i=1}^{k} w_i Q_i \tag{10}$$

$$\text{Transcoding (images):} \quad \max \sum_{i=1}^{k} a_i s_i \qquad \text{(the QSS value)} \tag{11}$$

Even a linear relationship between the SSIM value s_i for image i and y_i maps the objective function of the transcoding problem (with just images) to that of the bounded Knapsack problem (BKP). In general, the SSIM value for an image object i would be a non-linear and, even worse, a non-monotonic function of y_i, which will render the transcoding optimization a non-convex Knapsack problem.

A.3 Additional Results

A.3.1 `PAW` *index for all plans with caching disabled and enabled.* Fig. 12 shows the ratio of mean Web accesses across 96 total countries (divided into developed and developing countries) relative to the affordability target and the mean global page size for all plans.

A.3.2 Affordability-size trade-off for all plans. Fig. 13 shows the percentage of countries meeting the Web accesses target as a function of the reduction in mean webpage size in a country across all data plans.

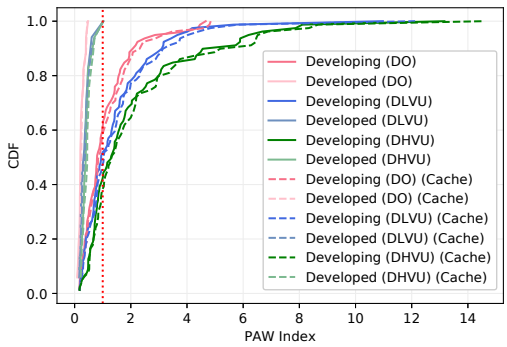


Figure 12: PAW index.

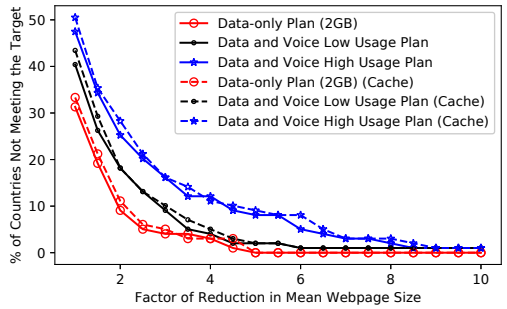


Figure 13: Affordability-size trade-off

A.3.3 Possible webpage size reductions with removals of more objects. We observe the differences in page size when removing different web objects using non-cache and cached bytes.

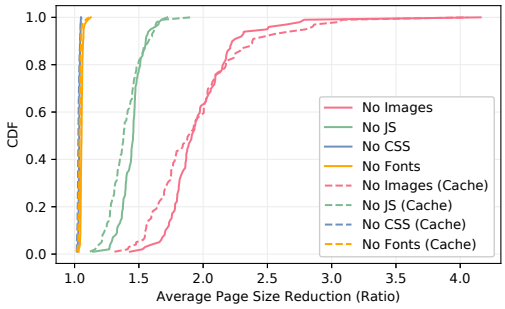
Individual resource types. Fig. 14a shows the reduction in page sizes upon removing all elements of a single resource type. We observe minimal differences between the non-cache and cached page sizes.

Multiple resource types. Fig. 14b shows the reduction of page size of multiple combinations of resource types on the first visit and the cached visit.

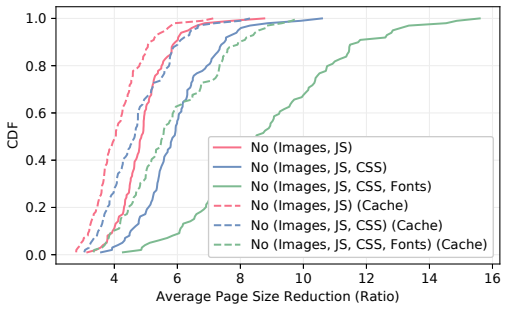
A.3.4 QSS after country-wise reduction using RBR. Table 3 shows the average QSS value of the Alexa top 1000 webpages of each country when reducing them to $\frac{1}{\text{PAW}}$ of their original page size.

A.3.5 Reducing all images to a quality threshold. Fig. 15 shows the percentage of URLs in 25 countries that can be reduced to $\frac{1}{\text{PAW}}$ of the page size if all images on webpages were reduced to 0.9 SSIM threshold quality level. 50 sites were sampled from each country's Alexa top 1000 sites. There are 418 unique URLs in the dataset (across all countries). The mean bytes reduction that this blanket approach gives is 23%. The average QSS is 0.94.

A.3.6 Brave and Opera Mini. We collected the page data of 50 randomly sampled websites from amongst the Alexa top 1000 websites on three browsers: (i) Brave, (ii) Opera Mini, and (iii) (Google) Chrome. For Brave, we collected the page data using both the default settings and the 'block scripts' setting. Comparisons with Chrome are shown in Fig. 16a and Fig. 16b.



(a) Individual resource type



(b) Multiple resource types

Figure 14: (a) The possible webpage size reductions with removals of one resource type from the pages, and (b) possible webpage size reductions with removals of multiple resource types from the pages.

Country	Q_T =0.9	Q_T =0.8
Uzbekistan	0.98	0.97
South Africa	0.97	0.94
Puerto Rico	0.97	0.93
Trinidad and Tobago	0.97	0.97
Senegal	0.96	0.95
Ecuador	0.96	0.93
Jamaica	0.96	0.94
Mongolia	0.96	0.94
Colombia	0.96	0.95
Kyrgyzstan	0.95	0.94
Kenya	0.96	0.94
Bolivia	0.96	0.93
El Salvador	0.95	0.87
Cameroon	0.96	0.93
Lebanon	0.95	0.95
Sudan	0.96	0.86
Dominican Republic	0.95	0.96
Jordan	0.96	0.94
Guatemala	0.96	0.94
Cote d'Ivoire	0.94	0.93
Tanzania	0.95	0.94
Yemen	0.94	0.91
Uganda	0.95	0.93
Ethiopia	0.95	0.93
Honduras	0.94	0.92

Table 3: Average QSS of Alexa top 1000 webpages (each country) after reduction (using RBR) to $\frac{1}{\text{PAW}}$ of the original page size. SSIM thresholds of 0.9 and 0.8 were used.

Summary statistics for the page size reductions are given in Table 4. Most notably, for some sites, Brave Blocked reduces the page size

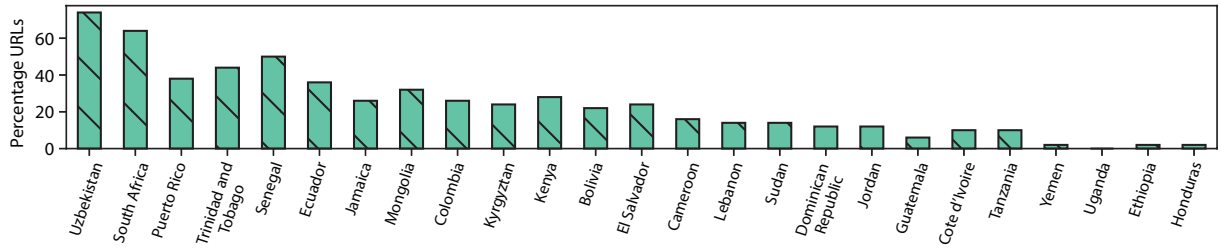
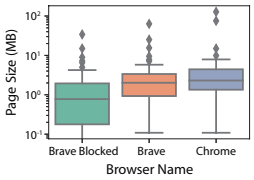
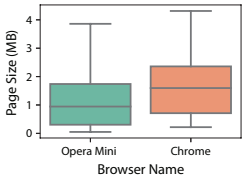


Figure 15: The percentage URLs that can be reduced to meet $\frac{1}{\text{PAW}}$ of their original page size when all images are reduced to 0.9 SSIM (maintaining Fair page quality).



(a) Brave (the y-axis in is in the log scale)



(b) Opera Mini

Figure 16: Distribution of webpage sizes on Google Chrome, (a) Brave Browser (with and without scripts blocked), and (b) Opera Mini. Note that the y-axis in (a) is in the log scale.

by 100% (rendering the page completely broken). Moreover, all of the selected transcoding browsers sometimes increase the page size (hence the negative percentage reduction in page size).

Browser	Mean	St. dev	Median	Min	Max
Opera Mini	30.5	48.9	41.9	-86.1	99.1
Brave	14.6	25.6	8.4	-91.1	80.3
Brave Blocked	57.3	34.1	61.9	-2.1	100

Table 4: Summary statistics for the percentage reduction in page sizes (as compared with Chrome versions of the same sites) of 50 unique URLs using three transcoding browsers.