Improving Web Accessibility: Computing New Web Page Design with NSGA-II for People with Low Vision

Yoann Bonavero, Marianne Huchard, Michel Meynard

Improving Web Accessibility: Computing New Web Page Design with NSGA-II for People with Low Vision

Yoann Bonavero, Marianne Huchard and Michel Meynard
LIRMM, CNRS and Université de Montpellier, Montpellier, France
Email: yoann.bonavero@lirmm.fr, marianne.huchard@lirmm.fr,
michel.meynard@lirmm.fr

***Abstract*—As society becomes increasingly aware of the need to take disabilities into account, new information technologies and intensive use of computers can be a chance or create new barriers. In the specific case of people with low vision, efforts to improve e-accessibility are mainly focused on the provision of third-party tools. Assistive technologies like screen magnifiers adapt graphical user interfaces to increase the quality of the perceived information. However, when these technologies deal with the Web, they are not able to meet all specific needs of people with low vision. In this paper, we propose an approach to make Web pages more accessible for users with specific needs. User preferences can concern font size, font family, text color, word and letter spacing, link color and decoration or even more complex features regarding brightness, relative size or contrast. We also take into account and encode the designer's graphical choices as designer preferences. Solving preferences of the user and of the designer to obtain a new Web page design is an optimization problem that we deal with Non-dominated Sorting Genetic Algorithm II (NSGA-II), a polynomial Multi-Objective Genetic Algorithm. We conducted detailed tests and evaluated the running time and quality of results of our tool on real Web pages. The results show that our approach for adapting Web page designs to specific user needs with NSGA II is worthwhile on real Web pages.**

Keywords– e-accessibility, Web page personalization, visually impaired, low vision, evolutionary algorithm, NSGA-II.

I. Introduction

In this paper, we deepen the research work presented in [1], where we developed an approach to improve Web page accessibility for people with low vision.

Many countries are adopting laws or treaties for enhancing digital accessibility. Some countries consider e-accessibility as a very important issue and even as a citizens' right. According to recent estimates, about 285 million people are considered visually impaired worldwide. 39 millions of them are blind and 246 million have low vision [2]. These figures are constantly growing, mainly because of the increased life expectancy.

ICT (Information and communication technologies) are increasingly used by everyone in everyday life. Unfortunately, this can be a double-edged issue for people with visual impairment, because these new technologies, which are able to compensate for user disabilities, can also be a new source of exclusion and discrimination. On the one hand, these technologies offer many solutions for everyday life activities. For example, they allow online purchasing, dealing with administrative documents, managing bank accounts, or locating places and finding routes. Beyond these services, ICT also bring a social dimension. They potentially offer access to information that was previously inaccessible for visually impaired people. On the other hand, many issues remain, due to the technologies used to design and develop websites.

Websites are composed of different kinds of data, including text-based documents, images, videos, and sounds. These data are displayed on pages formatted with respect to a visual style. This visual style, often given by CSS (Cascading Style Sheets) is written or used by the page author. The different choices made by the designer create the graphical context of the Web page. The graphical design of a Website reflects the brand or organization, and constitutes a landmark for people. It is also intended to influence the reader to recognize, assimilate, memorize a page and associate it with the related brand or organization. Moreover, it is intended to help users in their tasks by describing a navigation template, an information hierarchy and thus, it helps in understanding the page.

The publication language mainly used for the Web is HTML (HyperText Markup Language). This publication language is a very flexible and easy to understand language. Unfortunately, this flexibility gives us many ways to do the same basic things. For example, we can build the same (in terms of rendering) navigation menu with only list tags such as UL (Unordered List), LI (List Item), A (link to Another file), or with block tags such as DIV (DIVision) or SPAN (to span).

The W3C (World Wide Web Consortium) and other organizations publish sets of technical specifications in order to frame the development of websites. The W3C also provides a set of specifications to make accessible websites. The compliance of websites to these specifications assumes that they can be used by assistive technologies. Tools and guidelines are provided to developers and end-users, such as the WCAG 2.0 (Web Content Accessibility Guideline [3]), the UAAG (User Agent Accessibility Guideline [4]), the ATAG (Authoring tool Accessibility Guideline [5]) and the WAI - ARIA (Web Accessibility Initiative - Accessible Rich Internet Application [6]). Organizations like BrailleNet have been created to op-

erationalize the different standards of the W3C guidelines, including "AccessiWeb".

Unfortunately, e-accessibility is not a main concern of website designers and developers. It is often considered as a waste of time or an additional development cost, giving unsightly results that only target a small part of the population. Users can use third-party assistive tools to cope with visual difficulties when developers do not care about accessibility. Assistive technologies have existed for several years and are widely used by disabled people. Screen readers allow the user to get information in another communication way: vocal synthesis or braille display are used to vocalize information or display it in braille. Although this technology is designed for blind people, people with low vision also use it as a supplement to another assistive technology. Visually impaired people with low vision often use their partial sight as the principle means to access information. Screen magnifiers are applications that improve visual comfort and increase information acquisition. With these tools, it is possible to use zoom and color filters to compensate for visual issues. These tools are useful but frequently not sufficient to ensure e-accessibility because they have a general purpose, and they are not adapted to specific needs. This is mainly due to the high maintenance cost to maintain the compatibility with some applications like browsers, and to deal with after-sales technical issues. Nevertheless, e-accessibility is essential to ensure a quality of access to a large amount of Web services and contents. Recently, e-accessibility understanding is about to become a recognized professional skill for Web developers.

In this paper, we address the problems of adapting Web page design to the specific needs of a visually impaired person. Our approach proposes to replace the current pixel-level treatment process (in magnification filters) by an adaptation process based on knowledge of the HTML elements. Each HTML element has its own type and properties (color, size, position, etc.). The adaptation is performed from a set of user wishes, also called preferences. User's preferences can be font size, font family, text color, word and letter spacing, link color and decoration or even more complex wishes regarding brightness, relative size or contrast. We also take into account Web page designer's graphical choices as designer preferences. Solving these user and designer preferences to get a new Web page design is an optimization problem, that we manage with NSGA-II (Non-dominated Sorting Genetic Algorithm II), a polynomial Multi-Objective Genetic Algorithm.

In Section II, we explain how existing visual tools and assistive technologies work, and we highlight their main drawbacks regarding Web page context. We also present existing approaches that are intended to adapt user interfaces or Web pages in a personalized way. Section III presents how we represent Web page elements to be adapted as well as user (or designer) wishes. In Section IV, we describe how we are using and tuning NSGA-II. Section V reports the results that we obtained on several Web pages during our research. We conclude, in Section VI, by giving some perspectives of this work.

II. Existing work and problem statement

In this section, we explore hardware and software solutions developed to improve or provide website accessibility for visually impaired people with low vision. In all of these solutions we distinguish between two main kinds of tools and approaches. Some of them are used at development time as developer tools while others are used by end-users mainly on the client-side (filters and style-sheet redefinition). Besides, we describe more advanced proposals that try to partially automate some personalized adaptations.

A. Standards, guidelines, tools for developers

The W3C (World Wide Web Consortium) is at the origin of HTML and CSS standardization. It also works on accessibility via several initiatives, including WAI-ARIA (Web Accessibility Initiative - Accessible Rich Internet Application [6]). These standards have evolved through many versions to address the emergence of new technologies. They have two main objectives. The first is to ensure that resources can be parsed and used by external assistive technologies. The second is to provide minimal access to content for people who do not use, for different reasons, assistive technologies. One aim of these standards is to be independent of languages like HTML, JavaScript or CSS. This ensures the definition of robust standards regarding language diversity and evolution. The separation between the content and the page display style is the most important feature offered by HTML 4 and CSS 2. This separation should provide easier access to content.

In addition to the standards, guidelines like WCAG (Web Content Accessibility Guidelines [3]), UAAG (User Agent Accessibility Guidelines [4]) or ATAG (Authoring Tools Accessiblity Guidelines [5]), frameworks and tools are published to ease the use of the standards. We can mention "AccessiWeb", developed by the BrailleNet organization, which provides a simple operational interpretation of standards. WAI references a set of evaluation tools [7]. WCAG contains all standards about page content rendering, including the way the content is displayed to the user in terms of size, contrast, etc. UAAG gathers many required features of tools that browse Web pages. Finally, ATAG is concerned with tools that generate source code for Web content. The rules and standards are classified according to their importance to make websites accessible. Three increasing accessibility levels have been defined (A, AA and AAA). The first level (A) gives basic mandatory advice to ensure information accessibility. The second level (AA) provides important recommendations to be respected to avoid difficulties in accessing information. The third level (AAA) is about additional and optional ways to improve information access quality. When Web designers and developers include accessibility dimension in their websites, they mainly try to reach the intermediary AA level. Only a few very specialized websites require the highest level AAA.

Many tools exist that allow developers to make accessible websites or simply to get accessible existing websites. These tools analyze the HTML source code and either automatically rewrite it, or assist the developer to correct it, for example through suggestions in accordance with the standards [8].

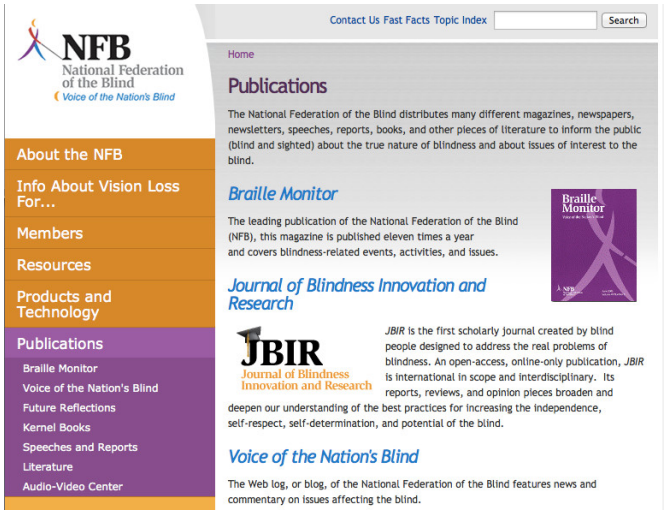

Figure 1. Original publication page of the National Federation of the Blind.

Figure 2. Applying zoom from a magnifier on the NFB publications Web page.

These tools can be separated into two categories: evaluation tools and transformation tools. The main drawback of these tools is that they do not enable adaptation for very different needs coming from various visually impaired people. Some user needs can contradict each other. Conflicts can arise due to dependencies between needs. For instance, high brightness contrast (for readability) and low light emission (for reducing the dazzle effect) can lead to conflictual needs (light emitting elements are linked to brightness contrast between them). Consequently, automated evaluation and transformation tools can only assist developers to meet a general accessibility requirement but are limited to implementation of the minimum recommended by standards.

B. Improving accessibility tools for the end user (magnification, browser options and extensions)

Some kinds of accessibility tools are available to get information from websites and report it to the user through another communication protocol. For example, for users with low vision, it is possible to retrieve information by transforming visual output with magnification applications or accessibility browser options and extensions.

Magnification tools allow the user to zoom on windows (e.g., Figure 2 zooms on Figure 1). Some of them propose font smoothing to avoid blurred characters, and mouse pointer modification to improve tracking movements. As another example, magnification tools can apply filters on the window. Filters include “gray scale”, “one color scale”, “black and white”, or “color inversion” (see Figure 4 for three of these filters). One widely used color filter is the color inversion filter (Figure 3). Its main purpose is to considerably reduce light emission when the user is on pages with a light background.

Another way for adapting Web pages consists of using the browser options and extensions that enable us to manipulate style sheets. It is possible to completely remove style sheets or to define a unique style sheet that will be applied to all Web

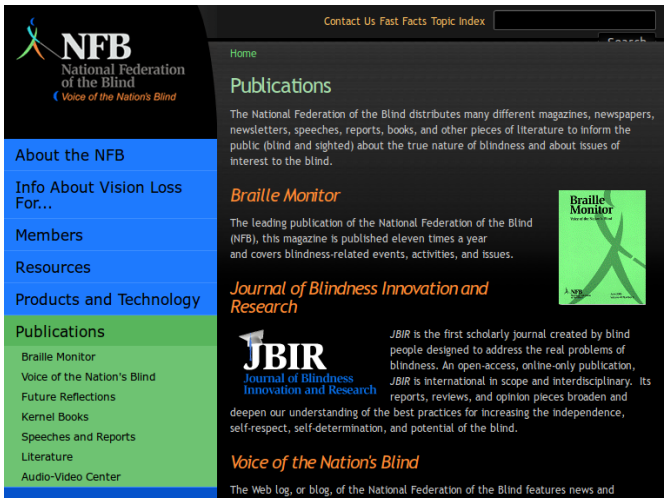

Figure 3. Applying a color inversion filter from a magnifier on the NFB publications Web page.

pages. To facilitate modifications, many browsers provide a graphical interface to help in the configuration of properties such as background color, text color, text size or link color.

With the two solutions defined previously (screen magnifier and style sheet redefinition), we can theoretically adapt almost all pages to be suitable for a large part of the impaired population. However, it is more complex in practice. These pages could be even better modified for the population, and there is a requirement for additional tools to meet more needs of visually impaired people. For various reasons, the existing solutions are not suitable for everyone. In the following, we highlight problems that occur when using filters (treatments on global images), then issues related to style sheet modification.

1) Filters: Figure 4 illustrates three filter applications. In the original *A* case, we represent a dark colored text background.

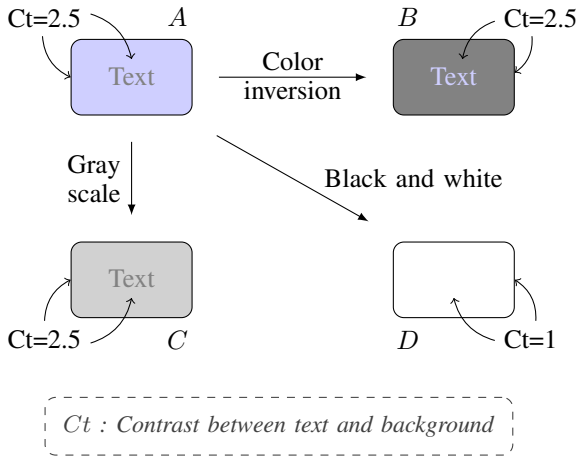

Figure 4. Filter application.

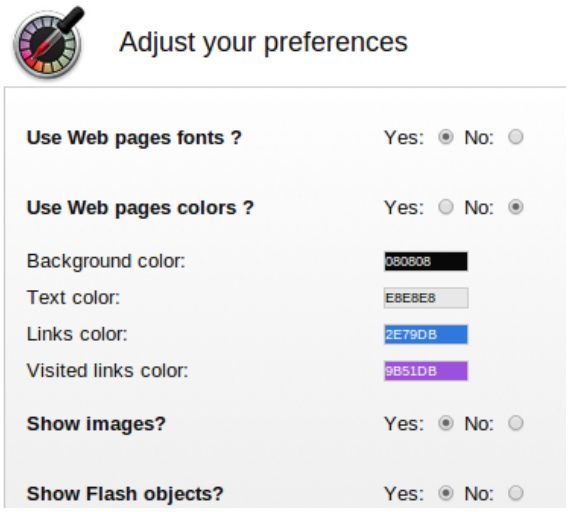

Figure 5. Chrome change colors extension.

From this first original case we apply some classical filters encountered in magnifier tools. In B, we apply to A the widely used filter inverting color of the display. We apply to A a gray scale color filter to give C, and we obtain D by applying a black and white color filter on A. For each case we associate a brightness contrast between the text color and the background color. Brightness contrast is the property that people with low vision mainly use to quantify the difference between text and direct background when they require an acceptable readability. It is measured by a ratio, which is a floating number between 1 and 21. It often ranges from 1:1 to 21:1. 1 (1:1) denotes the null contrast while 21 (21:1) denotes the highest contrast between two elements. This representation comes from the WCAG contrast computation. Regarding our filter, if the contrast is originally low (A case), filters like color inversion and gray scale filters cannot significantly improve brightness contrast and therefore the readability. Black and white filters, which are often efficient to increase the contrast, are no longer relevant in this case. These filters use a threshold to separate elements by their light emitting into two groups. Each element in the group of darkest elements is assigned a black color while each element in the second group is assigned a white color. If both text and background have two light or dark close colors (both below the threshold or both above the threshold), both elements are assigned the same color (in Figure 4, they both become white). Obviously, if both elements have the same color we lose all readability and contrast falls to 1:1. To avoid this, the filter must include a threshold, which would be customizable either automatically or manually by the end-user. However, in practice, this is not often the case, and when the threshold can be changed, this is a complex task for a non-expert user.

2) Style sheet definition: Most current browsers have extensions 5, modules or simply accessibility options to transform and adapt Web pages. These transformations are based on the manipulation of the Web page style sheet. In old browsers, options are often available to simply disable the original style sheet. This has the effect of only keeping the content without any graphical style. In more recent browsers, there are more

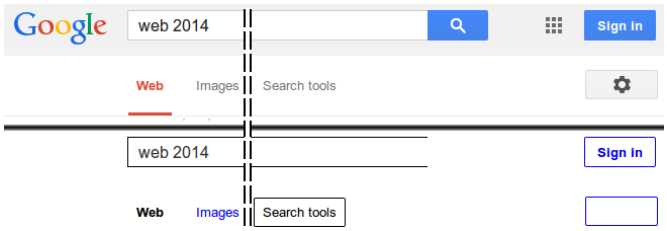

Figure 6. Google website with browser accessibility options enabled (top: original, bottom: with some accessibility options).

options to individually select to display images, keep original text font, etc. Sometimes this manipulation leads to huge information loss. Consequently, we can lose the global page context and also the brand or the organization design. This loss is unavoidable if Web pages are not properly developed in respect with the content and style separation rule given by the W3C.

An intermediary approach is also provided by browsers. Instead of disabling the entire style sheet, we can change a part of it. Rewriting a section of the page style sheet allows us to improve the readability of some elements. The end-user can define some properties like text color, text size, link color or background color. Some browsers allow more advanced users to define and provide their own style sheet. This alternative allows users to change all object properties. With style-sheet rewriting, the original context is more or less kept depending on the applied modifications. The global context (layout, colors, brand chart, etc.) is inevitably lost if there are major changes to compensate for complex disabilities.

Figure 5 shows how preferences can be adjusted in Chrome. Figure 6 presents the result on the Google result page when selecting some colors in accessibility options like background color. On the top, we have the original page, the middle of the page is removed to only show parts where alterations occurred. The result is displayed below the original. As a consequence,

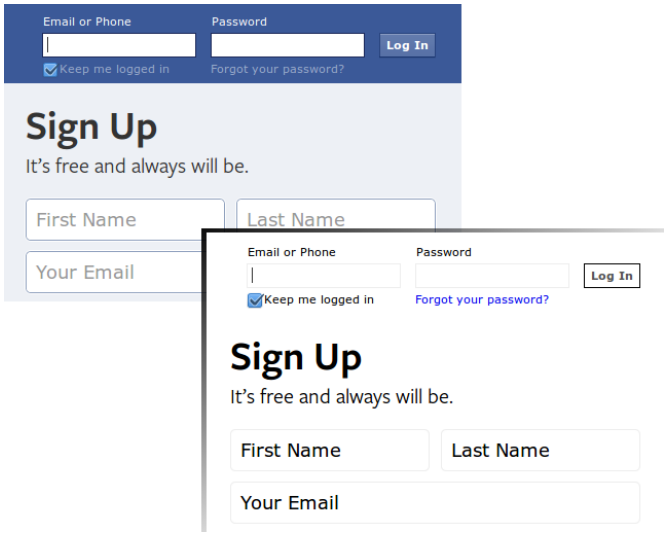

Figure 7. Facebook website with browser accessibility options enabled.

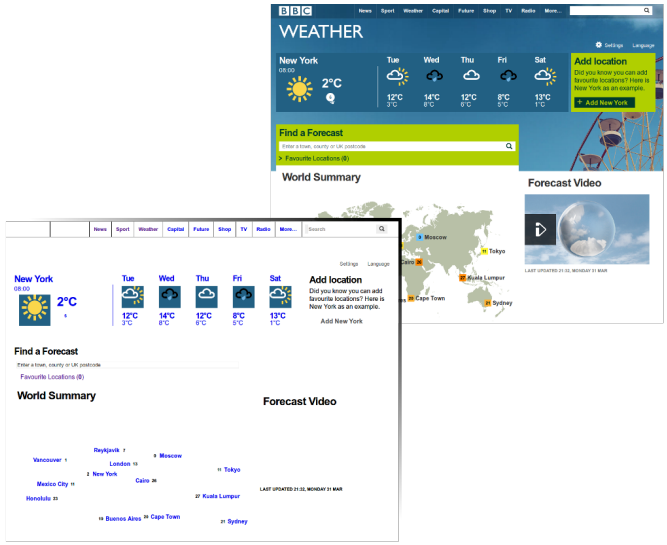

Figure 8. BBC website with browser accessibility options enabled.

buttons represented by an image included by the CSS file as a background image disappear and are replaced by the default selected color. Moreover, the main Google logo also disappears.

The Facebook sign in page (Figure 7) and the BBC (British Broadcasting Corporation) weather page (Figure 8) are deeply altered by accessibility options. Important information is lost. On the Facebook page, a recent version of HTML and styles sheets are used to make many visual effects. Main modifications concern fields of forms. Borders of login and sign in form fields become nearly invisible. Moreover, because of the removal of the top header background, login fields cannot be distinguished. Only the blinking caret is visible and allows the user to detect them. Concerning the sign in form, nearly invisible borders combined with lighter default text (text that disappears when clicking or filling the field) give to a user with low vision the impression that this text is the label of the field. Thus, the user has to click on the text to see that it is a field default text and not a label. The BBC weather page also has many alterations. This time this is not an interaction button that disappears but image justifying the position of many links. Indeed, the map at the page bottom contains text links to select a city. However, the world map seems to be inserted as a background image and not as an image. Thus, when accessibility options are set, the user is faced with a link disposition that is not logical. On this website, we also observe that the logo disappears and the search field is hard to find. Furthermore, here again, the location field appears with a nearly invisible border, which may generate ambiguity.

With style-sheet modification, the user often has to set many properties, including text color, link color, visited link color, hovered link color, title level 1 color, title level 2 color, etc. As a result, he has to manipulate a set of technical terms and a lot of options, and this task often is cumbersome and time consuming. Furthermore, preferences defined through this interface are the same for all pages. They are applied on all Web pages independently of their original style. Unfortunately, a single global configuration made by the user may not be relevant for all Web pages.

C. *More advanced personalization approaches*

Beyond the modules or extensions mentioned above, some more evolved proposals provide greater physical characteristic configuration support [9]. These end-user side applications allow the user to configure text properties such as size, letter spacing, or line spacing, colors of the text, background and links. They also allow the user to configure the image display (show or hide) and table display. Once the modifications are applied, almost all information about website colors may disappear. Then the original site ambiance may be lost.

User actions on websites can be used to adapt the Web page or the navigation: In [10], the authors propose to configure Web page display according to user actions and behavior. Depending on the clicked links and interactions, content can be hidden to highlight important content. This approach also allows to module menu content for adding potentially useful content. However, this approach must be taken into account at the development time. In [11], the authors propose to personalize Web display (shopping gallery) to a specific user or user group. The analysis of user usage on existing websites allows Web pages to be shown with different structure and navigation.

Several research studies have dealt with the generation of adapted user interfaces (UI). The SUPPLE++ systems [12] targets people with low vision or motor disability. For people with low vision, they propose users to control only the visual cue size. The tool is based on an optimization algorithm to combine adaptations to both low vision and motor disability. In [13], an abstract description of the UI and an ontology modeling the context (user capabilities, devices, etc.) are used

to automatically generate adapted accessible mobile user interfaces (for ATMs — Automatic Teller Machines, information kiosk, etc.) for people with disabilities. Although we do not focus on UI, here these works may be a source of inspiration, especially [13], which uses abstract models of the UI and of user preferences.

Some approaches focus on a particular visual problem, like the modification of colors for dichromate users while preserving the contrast [14].

Other approaches concentrate on accessing the structure of the page because it helps to understand and use the content. In [15], a configuration interface helps the user to specify elements of the Web page (title, content, navigation menu) that he wants to be shown depending the platform used, namely a Personal Computer (PC) or a Personal Digital Assistant (PDA). The KAI (Accessibility Kit for the Internet) system [16] considers both the developer and the user point of view. The user is able to choose what interests him the most in the accessed Web pages. As HTML cannot be used to know the real components due to its permissive syntax, BML (Blind Markup Language), a new markup language, is provided to the developer to annotate components of the page to be used by assistive tools. This marking operation can be done automatically for existing pages. Then a new HTML code is built using the markup and the user's preferences. It is used either with a normal browser or with an audio-touch platform that helps the user to access the page structure. During the transformation process, metrics are computed to rank Web pages according their accessibility. The transformation applied to improve the Web page focuses on making the structure accessible and not on specific visual problems.

D. Discussion

Standards, guidelines and evaluation tools are very useful, but it is hard to force developers to follow their advice. Statistics tell us that less than 10% of public websites are fully accessible [17].

We saw on detailed examples that user-side tools that apply global filters on the Web page, may improve some parts of the page, at the cost of degradation of other parts and that they can be totally inefficient in some situations. Using a style-sheet redefinition approach or configuration tools does not ensure that the original Web page design is kept. Besides, accessibility of this approach is not evident, because people with low vision may also have difficulty in editing or configuring style-sheet content.

From the existing work on personalized approaches, we learn that several issues are related to the Web page adaptation: user preference elicitation, modeling structure and web page design, modeling platform context (PC/PDA) and producing a new Web page whose design is a trade-off between the respect of these user preferences and the respect of the structure and initial design of the page.

Our objective is to adapt a page in accordance with its original appearance, with its structure and with the user preferences. We aim at proposing a general method that is able to take into account various visual needs. In our current work, we do not deal with the user preference elicitation, that we suppose is described in a simple formal language, as well as initial developer choices. User preference and design choice elicitation (explicitly or via a learning phase) will be studied in a future work. We also do not take into account various platforms at this stage, and we do not address content and navigation adaptation, that we want to keep as in the original page. We apply adaptation on the client side, using the original Web page. We separately process page elements according to their semantics, rather than applying a global strategy. We propose a method for computing a compromise between user preferences and developer choices on Web page elements. In our previous paper [1], we showed that classical representation and algorithms of Preference theory [18] did not scale in our context, even with small preference sets composed of basic preferences. In this paper, we focus on use of the meta-heuristic NSGA II that demonstrated better performance.

III. Our approach

We aim to develop a global approach as independent as possible from any specific Web page and also that would be able to consider any specific user's wish to find an adaptation. The previous section highlighted some problems related to the global treatment by magnifier software or to the simplistic preferences expressed via style sheet manipulation (browser accessibility options or dynamic tools). Our approach addresses these problems by using an Artificial Intelligence based approach.

In Section II, we illustrated the problem of the global treatment by magnification tools. We have shown that filters provided by a magnification tool can only be applied on the image rendered by the graphic card. Therefore, we can only apply filters on the entire screen or on an entire window in some operating systems. To achieve a better adaptation by taking all user needs into account, our approach is based on smaller elements than the entire screen, and it considers the following four components:

- Objects and properties (HTML elements and style) of the page written in the HTML and CSS files;
- Variation points (for example the color of a specific element or the size of the second level title);
- User's wishes put forward to compensate for his disabilities;
- Algorithms for finding an adaptation, from the initial Web page, according to the user's wishes, and such that the adaptation also integrates designer initial choices.

We develop these elements in the following subsections.

A. Preference representation

Designers connect different HTML elements to build Web pages. The element organization creates a tree structure. Each tree node represents an element with its physical properties like the position, appearance (size, color) or more abstract properties describing the element type (menu, content, image, link, etc.). In HTML5, which is the direct follow up of HTML4, there is a set of new tags. These tags are used to

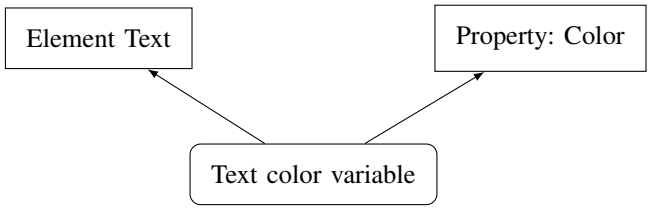

Figure 9. Association of one element and one property to make a variable.

describe element types with more semantics like navigation menu, article, section, complementary, etc. The old version of HTML (HTML4) only provides some tags without important semantics, like blocks. As a result, when a Web page is developed in HTML5, we have much information allowing us to express preferences and compute adaptations. However, if the page is in an older version of HTML, it is necessary to previously detect some important parts of the page, particularly the menu, the main content and sections, before computing an adaptation.

We define *a set of objects* (1) that represent all HTML elements that are important and useful in the page modification process. In other words, elements that will not be updated or that will not be used in computing are excluded from the object set. Excluding unnecessary elements allows us to reduce as much as possible the number of combinations and finally the search space size. HTML elements included in this set are, for example: first level titles (h1), paragraphs (p), anchors (a), images (img), articles (article), navigation bars (nav) etc.

$$Objects = \{O_1, O_2, .., O_n\} \tag{1}$$

The *variation points* are a set of variables (2) induced by properties. Properties can be either basic properties written in the HTML or CSS files, or computed from these basic properties. For example, height, width, position or color are basic properties found in the HTML or CSS files, while area is a computed property derived from both the height and width of an element. To summarize, a variable is a specific property of a specific object. An object gives rise to the creation of x variables, when x is the number of properties taken into account for this object. To be able to change the value of the red color component of an object, a variable representing this red component is created and added to the variation points. The domain of this variable is the value set of the red color component of the considered object. Thus, a variable is a pair composed of an object and a property (Figure 9).

$$VariationPoints = \{V_1, V_2, .., V_m\} \tag{2}$$

From the list of defined variables, a user can define different constraints (3). These constraints are also called *preferences* or *wishes*.

$$Constraints = \{C_1, C_2, .., C_k\} \tag{3}$$

For instance a user can say: "I prefer a dark color to a light color for titles". This preference means the user prefers to have a dark color for all titles in the page rather than a light color. This preference only concerns page titles and not the other texts in the page. With this page element segmentation, we can define different preferences for each object or each kind of object. All choices made on one or more variables constitute user preferences or user wishes. There is a main difference with existing work using user preferences. Tools using user preferences to adapt Web pages like in [9] use literal values for object properties like *red* or *blue* for the color or *14 pt* for a text size. We instead use constraints to compute such values.

We represent two different levels of preferences: basic or more complex preferences. The basic preferences, that come from Preference Theory [18], are represented as in (4).

$$V_i \ op \ x_i \ >_p \ V_j \ op \ y_j \tag{4}$$

where V_i and V_j are two variables (with possibly $V_i = V_j$), $>_p$ the preference symbol ($A >_p B$ means A preferred to B), op is a Boolean operator like $=$ and x_i (resp. x_j) is a value in the domain of V_i (resp. V_j). To represent the user's wish "I prefer black text to blue text", we use the variable c_T to represent the color of a text object T. The domain of c_T is $\{white, red, blue, black\}$. The user's wish is expressed as in (5).

$$c_T = black >_p c_T = blue \tag{5}$$

The basic representation also allows us to express conditional preferences. With conditional preferences we are able to represent preferences like "I prefer bold font to normal font if the font color is yellow". Here, we introduce a new variable w_T for representing the weight of the text concerned by the preference. We also introduce a new operator ':' to separate the condition from the remainder on the expression. A conditional wish is shown in (6). This representation was considered in our previous paper [1].

$$c_T = yellow : w_T = bold >_p w_T = normal \tag{6}$$

Here we explore more complex preferences in which we consider any complex function on variables and their domain values. This allows us to express wishes like "I would like to have a text size greater than or equal to 14 pt", "I would like to have bold text rather than regular text when text size is less than 14 pt" or "I would like to have a contrast between text and direct background greater than or equal to 50%". In this last example, the contrast is a binary function. It represents a distance between the colors of two objects: with the textual element T and the object B providing a background to T. To model this preference, we introduce two new variables, c_T that represents the text object color and c_B that represents the text background color. We define a contrast function $contrast(x, y)$ that returns the computed contrast between x and y. The result of this computation is compared to a user specified threshold. To determine the satisfaction of the preference, we evaluate (7) where l is the required threshold.

$$contrast(c_T, c_B) \geq l \tag{7}$$

B. Resolution algorithm

Different resolution algorithms exist. Choosing an optimization algorithm is justified by the search space width. Properties like contrast or brightness are not hard to compute on a given solution. Nevertheless the search space width in real cases makes impossible the use of exact algorithms. For example, with 9 color variables (our smallest experiment case), even if we drastically reduce the domain to only 27 colors (3 values for red, green and blue components), we get a search space of about 7.6×10^{12} solutions. Let us remark that with such reduction, the search space may not contain any good solution.

Our problem consists of choosing an adaptation that satisfies several preferences, which can be modeled as a multi-objective optimization problem. In the general case, such problems have a set of solutions, known as Pareto-optimal solutions We are interested in finding a subset of these optimal solutions if any exists, or solutions that approach optimality. For solving such problems, several multi-objective evolutionary algorithms [19], [20], [21] have been proposed as an alternative to costly deterministic methods. Among them, we choose NSGA-II (Non-dominated Sorting Genetic Algorithm-II [22]), which is popular in Search Based Software Engineering [23] due to its performances in this domain. Evolutionary algorithms (EAs) mimic the biological evolution of a population with the use of evolution operators that select, cross or mutate individuals.

NSGA-II works as follows (efficient implementation is described in [22]). The algorithm begins with an initial population P_0 of N solutions (or individuals), which can be randomly built. Figure 10 presents the evolution of the population at $t+1$ iteration. P_t corresponds to the population of step t. An offspring population Q_t of size N is created from P_t individuals using selection, crossover and mutation operators. P_t and Q_t are combined to form the population R_t. The N best individuals of R_t in terms of non-dominance and diversity are kept to form P_{t+1}, using the following principles. Several groups of solutions, called fronts, are calculated. The first non-dominated front (F_1) groups non-dominated individuals, corresponding to the best known solutions, with regard to at least one objective. A solution s_1 *dominates* another solution s_2 if: (i) s_1 is no worse than s_2 in all objectives, and (ii) s_1 is strictly better than s_2 in at least one objective. The second non-dominated front (F_2) groups the non-dominated individuals of $R_t \setminus F_1$. The third non-dominated front (F_3) groups the non-dominated individuals of $R_t \setminus (F_1 \cup F_2)$, and so on. The k first fronts whose union has less than N elements are included in the P_{t+1} population. To complete the P_{t+1} population to get N elements, individuals are selected in $k+1$ front, based on a crowding distance [24] and a binary tournament selection operator. The crowding distance helps to select solutions that have the lowest densities of surrounding solutions. For a given solution s, this is measured as the average distance of the nearest solutions (neighbors of s) along each of the objectives. The resulting crowded-Comparison operator helps to select scattered solutions. These steps are repeated until some termination criteria are satisfied, for example when a maximum number of generations has been reached.

In the next section, we present how we tuned NSGA II to solve a set of complex preferences in the context of Web pages.

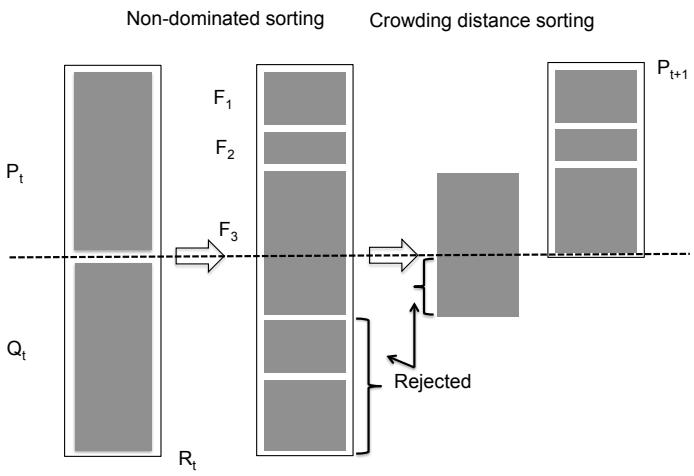

Figure 10. NSGA-II iteration, taken from [22].

IV. Experimental setup

In this section, we describe our experimental setup. As explained in the previous section, due to the nature of the optimization problem, we choose to use the Non-dominated Sorting Genetic Algorithm NSGA-II [22], which we implemented in C++. The feasibility of finding an adaptation, in accordance with user preferences, was already proved in a previous (limited) case study [1]. In the current paper, we extend our case study to five very different real Web pages. We also consider an enhanced set of preferences and we run the algorithms on a larger configuration set. In this experiment, we aim to determine cases where this approach adequately works, while giving trends about the running time, the number of generations and the number of satisfied preferences.

The NSGA-II algorithm relies on three operators. Crossover, mutation and selection operators are applied on a population to generate an offspring. The population size can be parameterized. The crossover and mutation operators are called according to a defined probability. The algorithm iterates through generations until some end criterion is satisfied.

In this setup, we configure the NSGA-II algorithm as follows (Table I). In the literature, the chosen population is commonly about 200. We decided to enlarge the population interval size to analyze the impact of this parameter on our specific problem. The population size can range from 100 to 300 individuals by steps of 50. This gives 5 population sizes for each tested website adaptation. Due to the $O(MN^2)$ complexity of this algorithm, where M is the number of objectives and N is the population size, we need to choose an upper bound for N that is not too high (300). Besides, in the algorithm, N represents the number of simultaneously treated individuals to generate the offspring. Thus, if we have an excessively low N, we considerably decrease interactions between individuals and reduce the population diversity. This is why we choose $N \geq 100$. We choose to set the probability to make a mutation rather

TABLE I. EXPERIMENT SUMMARY.

Parameter	Value(s)
Population Size	{100, 150, 200, 250, 300}
Crossover (resp. Mutation) prob.	0.94 (resp. 0.06)
Max exec time	10s
Max generations	unlimited
Preference sets	4
Maximal brightness difference	40%
Minimal color distance	40°
Minimal contrast	30%
Repeat	60 times

than making a crossover to an invariant value. This choice is based on the wish to have a more readable experimentation. We conduct some very simple tests on different input data, and we chose a probability of 94% for crossover (resp. 6% for mutation).

The non-deterministic base of NSGA-II, and of genetic algorithms in general, potentially induces local stagnation into the search space. Consequently, a part of the executions may stay blocked within a local optimum and will not return a solution in acceptable time. To avoid time explosion of an execution and disturbance in average running time, we limit the maximum execution time for an execution to 10 seconds. This time is justified by the practical end-user context. Adaptations are possibly computed each time the user changes to another page or another website. The non-terminated executions (time over or equal to 10 seconds) are included in time or generation statistics. Thus, when the number of non-terminated executions is high, the average running time tends towards 10 seconds. Besides, these non terminated executions are counted to get a terminated execution ratio and other statistics. The limited execution time is one of the two end criteria in this setup. The second end criterion is the global preference satisfaction level. For this experiment, we also decided to exit from the algorithm when we obtain a solution on which each preference is satisfied. Here we do not use the number of generations as an end criterion, we only get it for statistics. To summarize, we stop the algorithm if we find a good solution or if the execution exceeds 10 seconds.

NSGA-II allows us to have non-dominated solutions regarding several objectives. For this experiment, we define three general preferences, which correspond to practical problems of people with low vision:

- GP_1 : Uniform background color brightness.
- GP_2 : Minimal contrast between the text and its direct background.
- GP_3 : Keep original color for modified elements.

Here GP_1, GP_2 and GP_3 are general preferences. When we associate a general preference GP_i to a specific website, we obtain a set of preferences P_i. For example, for the Parempuyre website, which is part of our experiment, we have variables for the three backgrounds, including $inputBckClr$ (input background color), $leftHdBckClr$ (left header background color), and $bodyBckClr$ (body background color), and variables for text colors including $leftLkClr$ (left link color) and $hdClr$(text header color). The preference sets corresponding to the general preferences are as follows:

$$\begin{aligned}
P_1 =&\{proxBrightness(leftHdBckClr, bodyBckClr,\\
&inputBckClr)\}\\
P_2 =&\{contrast(leftHdBckClr, hdClr) >= 30,\\
&contrast(bodyBckClr, leftLkClr) >= 30, ...\}\\
P_3 =&\{distOrigClr(leftHdBckClr) < 40,\\
&distOrigClr(bodyBckClr) < 40, ...\}
\end{aligned}$$

The P_1 preference set contains a global preference given by the user to have a brightness that is comfortable for the eyes for all backgrounds on the Web page. This global preference is represented by an objective function, which refers to all backgrounds. The objective function may concern a large part of the variable set and often leads to huge dependencies between many variables. If we represent preferences and their dependencies with a graph, with only unary or binary objective functions, we may often have several connected components. When using a global objective function, we tend to group many connected components to a larger single connected component. To compute this objective function, we begin by computing the brightness of every background using indications of [3] (brightness values are in $[0, 1]$). Then we compare the difference $d = max - min$ between the minimal brightness min and maximal brightness max to a defined threshold $maxDiff$. $prefNb$ is the number of preferences used in the current preference set in a case study.

The objective function for P_1 is thus computed as follows:

$$\begin{aligned}
&If\ d >= maxDiff\\
&return\ 0 + (1 - d)/prefNb\\
&else\ return\ 1 + (1 - d)/prefNb
\end{aligned}$$

The two other preference sets are also based on colors. The P_2 preference set includes pairs composed of a text color and background color. These pairs of colors are linked to ensure minimal brightness contrast, suitable for readability. To compute the associated objective function, we compute the brightness of both colors and the contrast c using indications of [3]. The contrast is computed from the relative perceived brightness of the two colors. The WCAG computed values are in $[1, 21]$ and we normalize them to have values in $[0, 1]$. We compare the contrast to a minimum $minContrast$, which is set at 0.3. This value is slightly higher than the standard requirement in the Web Content Accessibility Guidelines (WCAG 2.0) for the AA level.

The objective function for P_2 is thus computed as follows:

$$\begin{aligned}
&If\ c >= minContrast\\
&return\ 1 + (c - minContrast)/prefNb\\
&else\ return\ 0 + (c - minContrast)/prefNb
\end{aligned}$$

The last preference set, P_3, allows the algorithm to keep as much as possible the original color context of the page after the adaptation. These preferences concern only one variable. P_3 corresponds to the designer preferences, because these preferences are concerned by the proximity between colors in adapted pages and initial colors. These preferences are binary

preferences, because they refer to two different color variables to be able to compute the contrast. For color distance preference (P_3), we compare the hue of the two colors $h_{original}$ and $h_{current}$. From the representation of the two colors in the chromatic circle, the color distance is the angle between the two hues. We set the maximal accepted angle $angleMax$ at 40°. To obtain the hue (in $[0, 360]$) of a given color from its RGB (Red Green Blue color model) components, we use the computation of the hue from the classical conversion function from RGB to HSV (Hue-Saturation-Value) color spaces [25].

The objective function for P_3 is thus computed as follows:

$$
\begin{aligned}
&hd = |h_{original} - h_{current}| \\
&if\ hd\ >= 180\ then\ hd = 360 - hd \\
&if\ hd\ <\ angleMax \\
&return\ 1 + (hd/180)/prefNb \\
&else\ return\ 0 + (hd/180)/prefNb
\end{aligned}
$$

We compose the three initial preference sets to obtain the four preference sets of our experiment.

$$S_1 = P_2 \tag{8}$$

$$S_2 = P_2 \cup P_3 \tag{9}$$

$$S_3 = P_1 \cup P_2 \tag{10}$$

$$S_4 = P_1 \cup P_2 \cup P_3 \tag{11}$$

The first preference set S_1 corresponds to only one general preference. It corresponds to a user that has issues with readability when contrast is low. The second preference set S_2 groups P_2 and P_3 preferences. In this configuration, the user possibly wants to increase contrast on the page (if the current contrast is not suitable) but he does not want to completely change the original page colors. The user may want to keep the color context close to the original one in order to recognize the browsed Web page or to avoid getting lost in the navigation (because he memorizes colors assigned to some parts of the web page). The third preference set S_3 is suitable for people who have a disease involving major light sensitivity. They need to have minimal contrast between the text and the direct background to improve readability, and also have similar background brightness to avoid dazzle. The last preference set S_4 is a complex case of adaptation, which combines the three preference sets. It also corresponds to existing real low-vision user needs.

We implement the selection, crossover and mutation operators to make the population evolve. The selection operator is based on the classical crowded-comparison operator. This comparison is used after applying the fast non-domination sort and with crowding distances assigned to each individual [22]. The selection operator keeps the diversity in the population.

We implement classical crossover and mutation operators. Mutation modifies part of an individual or an entire individual. Each component of the child individual has a ½ probability of getting a random value. In other ½ cases, the child keeps the parent value for this component. For an individual with x components we have a probability of $(½)^x$ either keeping all of them or changing all of them. For example, an individual with 4 components has a 0.0625 probability of completely changing or of being identical to the parent, with 8 components we have a 0.0039 probability. This probability is approximated, we do not care about the probability of getting the original value when we get a random value (in huge domains like color space, it is very low). Thereby in most cases a mutation keeps some parent components in the child.

The crossover operator also uses probabilities. However, instead of getting a random value for some components, we only get parent values. From two parents, the crossover operator generates one child. Each component of this child has a ½ probability of come-ing from the first parent. In other cases, the component value comes from the second parent. As in the mutation operator, the probability of entirely copying one parent is low when the component number is high.

Domains of variables may have several dimensions. For example, the text size variable domain generally has about 10 or 20 values. By contrast, the color variable domain can reach 2^{24} values in a true context. The number of variables depends on the type and complexity of the given user's preferences. All preferences defined in this setup are based on color variables. Each color variable is implemented with an RGB color space. This choice is based on the large size of the color domain. Each component of the implemented RGB color space is set with the sRGB (standard RGB) values of the Website. The domain of the web sRGB color space is about 16.7 million colors, but we use only 32.768 colors in the domain. Many picture editing software systems have a Web color mode. This mode contains substantially fewer colors in the domain. However, experimentation shows that limiting the color domain may have a negative impact on the resolution by the algorithm. If the number of colors is too low, some problems may have no solutions. Research also shows that the domain size has no significant effect. Increasing or reducing the search space does not change execution times because it is an optimization problem, thus it does not examine the entire solution set. We decide to have 32 values in each component (R, G and B) to not limit the number of solutions, while avoiding a huge search space. This size can be discussed and adapted to optimize color features.

To study the effect of the number of objectives, we tested two different objective function sets. We recall that the NSGA-II algorithm is a multi-objective algorithm that exploits non-domination sorting and has a complexity of MN^2, where M is the number of objectives functions. P_2 and P_3 preference sets are used in two different manners to define the objective functions. In the first case, for each preference, we make one objective function, which returns a floating point value equal to or higher than one when the preference is satisfied. In the second case, for each preference set (P_2 or P_3), we make two objective functions. Each of these two objective functions groups half of the preferences, which are randomly chosen.

Like the elementary objective functions, an aggregated objective function returns a floating value in $[0, 1 + 1/prefNb]$, which evaluate the quality of one solution accordingly to all embedded preferences. An aggregated objective function computes the average objective functions associated with the embedded preferences. The formulas of objective functions for P_1, P_2, P_3 have been designed to prevent compensation for

objective function values and prevent acceptance of a solution as a good one when one of its elementary objective functions is not satisfied.

To illustrate our chosen representation, the "Facebook" registration page will be represented by about 40 variables to implement the preference: "contrast between text and direct background higher or equal to x". For this preference we have to extract all text elements and their direct backgrounds, making color variables. For the Google search page (not the result page), we obtain about 17 variables, while for the "BBC News" home page we obtain about 200 variables. These values are rounded because large parts of websites are dynamic and regularly change (these computations were done on November, 14, 2013). The five websites were selected for the diversity of their architecture, their number of objects and the colors used.

For the first tested website, we combine the two possible variations on population size and objective function number. For the other websites, we set the population size at 250. In a $configuration$, we set a population size and a number of objective functions for a specific website. The execution of each configuration is repeated 60 times. This number of executions may seem high, but this algorithm is highly stochastic: the higher the execution number, the more precise the results are. Of course, we need to make a trade-off because a higher execution number leads to a smoother result, but it also corresponds to a very long experimentation time.

In the next section, we report and discuss the results obtained with all the previously described configurations on the selected websites.

V. Experiment results

In this section, we present the results obtained using the experimental framework presented above (Section IV), with our C++ implementation of NSGA-II. We selected websites for their diversity regarding the number of objects and colors on the page. These websites have no specific global theme or architecture. The chosen websites have been highlighted by users with visual impairment as websites with accessibility issues.

A. Parempuyre website

The "Parempuyre" website is our first example. It is a rather small website with a dozen color variables. The variables are text colors or background colors. They are selected from the previously defined preferences. We present the results for the non-aggregated case, and then for the aggregated case.

1) Parempuyre website without aggregation: For S_1 preferences that concern the minimal brightness contrast between a text color and its direct background color, we get all relevant textual elements from the Web page. One textual element associated with its foreground color property gives one variable. Other variables are also associated with background colors (three in this case). We run the algorithm with all preference sets and all configurations. The detailed results for the S_1 preference set is shown in Table II.

We recall that the presented figures are the result of the average of 60 executions. We have 3 backgrounds and 9 texts, giving 12 variables when associated with their color property. We have 9 contrast constraints. All executions terminate before 10 seconds. This means that the algorithm always finds a solution satisfying all defined preferences in the S_1 set. The population size has an impact on the number of generations needed to find these solutions: the higher the population size, the lower the number of generations. We find a satisfactory solution on average in less than 430 milliseconds. The maximal time for discovering a good adaptation is less than one second, and in the best cases only 50 milliseconds are needed. The generation number goes from 7 to 88, and is less than 37 on average. In this case study, the minimal requested contrast is 30%, which gives, for each constraint, 70% of satisfactory values in the search space.

TABLE II. "PAREMPUYRE" S_1 - EXPERIMENT FIGURES.

Variables	Prefs – obj. functions	search space
12	9 – 9	$\simeq 2 \times 10^{54}$

Pop. size	100	150	200	250	300
Exec. time (s)					
Average	0.18	0.24	0.29	0.42	0.43
Standard deviation	0.08	0.09	0.10	0.19	0.12
Min	0.05	0.06	0.12	0.18	0.22
Max	0.42	0.52	0.57	1.11	0.78
Generations					
Average	37	27	21	20	15
Standard deviation	17.1	10.5	7.0	9.1	4.4
Min	11	7	9	9	8
Max	88	58	40	54	28
Non cpltd. exec.					
Percentage	0	0	0	0	0
Satisfied obj. fct (avg.)	–	–	–	–	–
Standard deviation	–	–	–	–	–
Min	–	–	–	–	–
Max	–	–	–	–	–

On the contrary, the executions with the S_2 preference set give worse results (Table III). We still have the 12 variables representing objects associated with color properties. In addition to the minimal contrast preferences, we have a constraint in order to keep colors as close as possible to the original colors. There are 21 preferences on variables: 9 contrast preferences and 12 color closeness preferences. In this configuration, all preferences are implemented as objective functions, thus there are also 21 objective functions. With this constraint of staying close to the original colors, we considerably reduce the number of good solutions in the entire search space.

In this configuration, with these 21 objective functions, no execution returns a good solution. This is mainly due to the low number of good solutions regarding the search space. Adjusting the contrast means changing the color of one variable that also impacts its closeness to the original color. Even if the algorithm does not return a good adaptation, about 6 of the 20 objective functions are completely satisfied when we reach 10 seconds. The other objective functions (non-satisfied objective functions) may not have such bad values. This phenomenon is due to the fact that, NSGA-II is an elitist

TABLE III. "PAREMPUYRE" S_2 - EXPERIMENT FIGURES.

Variables	Prefs – obj. functions	search space
12	21 – 21	$\simeq 2 \times 10^{54}$

Pop. size	100	150	200	250	300
Exec. time (s)					
Average	10.00	10.00	10.00	10.00	10.00
Standard deviation	0.00	0.00	0.00	0.00	0.00
Min	10.00	10.00	10.00	10.00	10.00
Max	10.00	10.00	10.00	10.00	10.00
Generations					
Average	1,123	582	374	274	211
Standard deviation	14.8	13.5	12.6	12.8	15.4
Min	1,093	554	346	244	179
Max	1,156	614	409	313	256
Non cpltd. exec.					
Percentage	100	100	100	100	100
Satisfied obj. fct (avg.)	5.7	6.2	6.4	6.6	6.8
Standard deviation	1.7	1.7	1.9	1.6	1.7
Min	3	3	3	3	3
Max	11	10	11	11	11

TABLE IV. "PAREMPUYRE" S_3 - EXPERIMENT FIGURES.

Variables	Prefs – obj. functions	search space
12	10 – 10	$\simeq 2 \times 10^{54}$

Pop. size	100	150	200	250	300
Exec. time (s)					
Average	1.10	0.49	0.56	0.64	0.70
Standard deviation	2.15	0.47	0.46	0.41	0.36
Min	0.04	0.09	0.17	0.18	0.22
Max	10.00	2.29	2.58	3.49	2.72
Generations					
Average	196	53	38	30	24
Standard deviation	339.1	49.6	29.3	18.6	12.0
Min	8	10	12	9	8
Max	1,599	240	168	157	91
Non cpltd. exec.					
Percentage	5	0	0	0	0
Satisfied obj. fct (avg.)	4.7	–	–	–	–
Standard deviation	1.2	–	–	–	–
Min	3	–	–	–	–
Max	6	–	–	–	–

algorithm, it selects and keeps the best evaluated solutions for all objectives even if they are not satisfied.

In Table IV, we present figures of executions with the S_3 preference set. In this preference set, we introduce a "global" preference. By "global" preference, we mean a preference that covers all background color variables. Here this preference is an n-ary predicate involving the three background color variables. Introducing this preference concerning the uniformness of the three background color brightnesses results in adding dependencies with the contrast preferences. In this "Parempuyre" website configuration, the NSGA-II algorithm returns good solutions before 10 seconds for nearly all executions, even if the number of good solutions in the search space is reduced by the constraints.

TABLE V. "PAREMPUYRE" S_4 - EXPERIMENT FIGURES.

Variables	Prefs – obj. functions	search space
12	22 – 22	$\simeq 2 \times 10^{54}$

Pop. size	100	150	200	250	300
Exec. time (s)					
Average	10.00	10.00	10.00	10.00	10.00
Standard deviation	0.00	0.00	0.00	0.00	0.00
Min	10.00	10.00	10.00	10.00	10.00
Max	10.00	10.00	10.00	10.00	10.00
Generations					
Average	944	495	322	234	181
Standard deviation	149.3	76.4	47.8	31.1	26.8
Min	755	391	247	174	136
Max	1,100	600	389	280	234
Non cpltd. exec.					
Percentage	100	100	100	100	100
Satisfied obj. fct (avg.)	6.4	6.4	6.9	7.3	6.9
Standard deviation	1.5	1.5	1.7	1.8	1.6
Min	4	4	3	4	4
Max	10	11	12	12	12

Most of the executions return a good solution in less than 1 second on average. For a population of 100 individuals, we exceed 1 second, due to the 5% of non-terminated executions that add 10 s to the average time computation. The best executions return a good adaptation only in a few generations (less than 53). However, while with S_1 preference set less than 100 generations are needed in the worst execution to find a good solution, here the worst executions need up to 1,599 generations. That has an impact on the average execution time. It tends to be seven times more regarding the values obtained with S_1. Returning a good adaptation solution may take up to a second compared to less than 450 milliseconds in S_1 configuration (on average).

In the last case (with the S_4 preference set), we apply all preferences: minimal brightness color contrast, preservation of the original color context and uniformity of background brightness color (Table V). Independently of the chosen website, this configuration is the most complex configuration of our experiment. The association of many preferences may give many dependencies between variables. Here again, the global background uniform color brightness constraint tends to group many variables mainly if there are many background variables.

It is worth noting that even when we complicated the preference set compared to S_2, the number of satisfied objective functions is slightly higher. We apply three preferences instead of only two but, on average, when the maximal execution time is reached, we have satisfied one more objective function. Moreover, the worst executions discovered solutions with up to 12 satisfied objective functions among the 22 of this problem. In other words, while adding one more preference, we increase the number of satisfied objective functions. In some cases, the fact of having more preferences and finally more objective functions may help and guide the algorithm to find a good adaptation solution more rapidly. Some implicit rules are somehow given to the algorithm to converge to a good solution.

In all of the previous cases, each preference was represented

TABLE VI. "PAREMPUYRE" S_{1aggr} - EXPERIMENT FIGURES.

Variables	Prefs – obj. functions	search space
12	9 – 2	$\simeq 2 \times 10^{54}$

Pop. size	100	150	200	250	300
Exec. time (s)					
Average	0.08	0.26	0.15	0.20	0.27
Standard deviation	0.09	1.27	0.03	0.04	0.05
Min	0.03	0.07	0.09	0.07	0.17
Max	0.74	10.00	0.22	0.29	0.43
Generations					
Average	12	21	8	7	7
Standard deviation	14.1	101.7	1.6	1.4	1.2
Min	5	6	5	3	5
Max	114	803	12	11	12
Non cpltd. exec.					
Percentage	0	2	0	0	0
Satisfied obj. fct (avg.)	–	1.0	–	–	–
Standard deviation	–	0.0	–	–	–
Min	–	1	–	–	–
Max	–	1	–	–	–

TABLE VII. "PAREMPUYRE" S_{2aggr} - EXPERIMENT FIGURES.

Variables	Prefs – obj. functions	search space
12	21 – 4	$\simeq 2 \times 10^{54}$

Pop. size	100	150	200	250	300
Exec. time (s)					
Average	10.00	9.35	7.67	6.83	5.63
Standard deviation	0.00	2.43	4.08	4.49	4.67
Min	10.00	0.15	0.26	0.37	0.43
Max	10.00	10.00	10.00	10.00	10.00
Generations					
Average	1,819	900	471	292	166
Standard deviation	12.3	233.6	249.3	190.5	135.2
Min	1,797	16	18	18	15
Max	1,859	979	621	433	300
Non cpltd. exec.					
Percentage	100	94	75	67	54
Satisfied obj. fct (avg.)	1.4	1.6	1.6	1.9	1.9
Standard deviation	0.5	0.6	0.6	0.7	0.7
Min	1	1	1	1	1
Max	2	3	3	3	3

by one objective function. For the contrast constraint between the text and its direct background, each pair of text color variable and its associated background color variable corresponds to one objective function. We may obtain different results if we aggregate (see Section IV) objective functions. Instead of having one objective function for each contrast constraint, we group them into aggregated objective functions.

2) Parempuyre website with aggregation: The next tables present the same four preference sets, but now with aggregations of objective functions.

There are significant differences in the results obtained with the first preference set S_1, as shown in Table VI. The algorithm spends up to 270 milliseconds to return a good solution instead of up to 430 milliseconds. The time tends to be doubled when we do not use an aggregation configuration (with aggregated objective functions). Note that the number of generations is really stable, except for the 100 and 150 individual population. Note also that there is 2% of non-completed executions with a population of 150 individuals. The number of objective functions is reduced to only 2, which has an important role in the time needed to compute each generation. Excluding the population of 100 and 150 individuals, we only need 3 generations in the best cases and 12 generations in worst cases. In general, executions spend less time than in the non-aggregated S_1 configuration and need only a few generations.

In the S_2 preference set, the aggregation is much more useful: let us recall that in non-aggregated configurations, no execution terminates before 10 seconds.

Aggregation allows the algorithm to find a good adaptation in up to 46% of executions (with a population of 300 individuals). Whereas with the non-aggregated configuration (Table III), all executions reach 10 seconds without returning any solution that completely satisfies all preferences. With the aggregation, we highly improve the situation for the S_2 configuration. This example highlights the importance of the population size. This size plays an important role in the number of terminated executions. With a population of 300 individuals, we saw that 46% of executions returned one good solution, whereas with a population of 200 we have 25% of terminated executions, and we have 0% with a population of 100 individuals. Terminated executions return a good adaptation in about half a second.

TABLE VIII. "PAREMPUYRE" S_{3aggr} - EXPERIMENT FIGURES.

Variables	Prefs – obj. functions	search space
12	10 – 3	$\simeq 2 \times 10^{54}$

Pop. size	100	150	200	250	300
Exec. time (s)					
Average	0.05	0.08	0.11	0.16	0.19
Standard deviation	0.03	0.02	0.02	0.05	0.03
Min	0.03	0.05	0.04	0.09	0.10
Max	0.25	0.21	0.16	0.54	0.26
Generations					
Average	10	9	8	8	7
Standard deviation	6.8	2.6	1.6	2.7	1.2
Min	6	6	3	5	4
Max	52	24	12	27	10
Non cpltd. exec.					
Percentage	0	0	0	0	0
Satisfied obj. fct (avg.)	–	–	–	–	–
Standard deviation	–	–	–	–	–
Min	–	–	–	–	–
Max	–	–	–	–	–

Table VIII shows aggregation results for the S_3 preference set on the "Parempuyre" website. These executions are made with objective function aggregations.

While in the case of the aggregation of the S_1 preference set we reduced the execution time by twofold, here the impact is even more significant. On average, the aggregation allows us to reduce the execution times by threefold. The aggregation of objective functions for S_3 has a very positive impact (Table

TABLE IX. "PAREMPUYRE" S_{4aggr} - EXPERIMENT FIGURES.

Variables	Prefs – obj. functions	search space
12	22 – 5	$\simeq 2 \times 10^{54}$

Pop. size	100	150	200	250	300
Exec. time (s)					
Average	9.84	9.03	7.67	7.03	5.78
Standard deviation	1.26	2.91	4.07	4.37	4.54
Min	0.18	0.23	0.33	0.45	0.61
Max	10.00	10.00	10.00	10.00	10.00
Generations					
Average	1,676	812	440	281	161
Standard deviation	214.2	261.0	231.6	172.0	122.8
Min	34	24	22	21	21
Max	1,741	917	581	403	281
Non cpltd. exec.					
Percentage	99	90	75	69	54
Satisfied obj. fct (avg.)	2.3	2.7	2.8	2.9	3.0
Standard deviation	0.7	0.7	0.6	0.6	0.5
Min	1	2	2	2	2
Max	4	4	4	4	4

VIII).

In the case of S_4, the aggregation now allows us to have terminated executions leading to good adaptation, as is already the case for the S_2 aggregation case. Moreover, the population size seems to again have its importance in the number of terminated executions. With a population of 100 individuals, only 1% of executions are not interrupted before 10 seconds. A population of 200 individuals allows us to obtain 25% of terminated executions and we reach 46% when the population size is equal to 300 individuals. However, many executions remain not completed. In this last case, nearly 60% of objective functions are completely satisfied.

B. Godaddy website

The "Godaddy" website has more objects on the page and finally has more variables. From previously chosen preferences, 22 color variables are created to solve the problem. Table X gives results of the application of all the preference sets. For each preference set, we use a population size of 250 individuals. The population size has an impact on the computation efficiency, but this impact is not necessarily similar for all configurations. A population of 100 individuals sometimes leads to unusual results. The population of 250 seems to be a good trade-off between the execution time and the number of completed executions. We associate with each preference set its aggregation version in order to easily compare them.

For the S_1 preferences set, we can improve the computation efficiency and reach 100% of completely satisfactory solutions returned with the aggregation of objective functions. Without aggregation, not all executions give a completely satisfying solution. Some of them are interrupted by the end time criterion. When executions are not completed, on average, returned solutions satisfy a good part of all objective functions (10 out of 14). In best cases, up to 12 objective functions out of 14 are satisfied. The aggregation configuration allows us to reach 100% of executions, leading to good solutions. Moreover, the execution time is highly reduced. The algorithm returns a good adaptation solution in about 300 milliseconds instead of 5 seconds. The number of generations is closely linked to the running time. The generation number is also, like the execution time, highly reduced. We need less than 20 generations to get one good adaptation solution. Without aggregation, it takes 215 generations instead on average.

Similar to this case, with the S_3 preference set, we get a major improvement. From 18% of completed executions without any aggregation, we get 98% of completed executions with an aggregation configuration. Moreover, the running time for the executions is considerably reduced. We only need 500 milliseconds on average to get one good adaptation solution as compared to 9.67 seconds on average without an aggregation configuration. As in the previous example, the number of generations is also highly reduced on average. We need about 23 generations on average as compared to 358.

For the S_1 and S_3 preference sets, the aggregation achieves a substantial improvement in execution time and in the quality of the adaptation returned. It allows us to obtain adaptation solutions for a real-time use. Unfortunately, it does not improve all configurations. The complexity of the two other configurations is too high to get good adaptation solutions with or without aggregation.

In the S_2 preference set configuration, none of the executions gives a completely satisfactory solution even with aggregation. Without aggregation, we satisfy at least 5 of the 36 objective functions and up to 11 of the 36 objective functions in the best cases. With aggregation, we do not satisfy any objective function in the worst cases, whereas in the best cases we satisfy 3 of the 4 objective functions. Similarly, in the S_4 preference set configuration, we satisfy on average about 8 out of 37 and 1.6 out of 5 objective functions respectively, for non-aggregation and aggregation configurations. In the best cases, we satisfy 12 out of 37 objective functions without any aggregation as compared to 2 out of 5 with aggregation. Finally, in the worst cases, only 1 out of 5 objective functions are completely satisfied in the aggregation configuration and 4 out of 37 with aggregation.

In these two last cases (S_2, S_4), from the results obtained, the aggregation of objective functions seems to have no positive impact on the execution of the NSGA-II algorithm. However, these figures do not especially mean that the aggregation only improves simple configurations. Another aggregation (objective function repartition) with more aggregated objective functions for example, may give different results. In all cases, with less complex configurations, aggregation has a notable positive impact on the quantity of the solutions returned.

C. ReasonFrance and UBOLEA websites

The "ReasonFrance" website (Table XII) and the "UBOLEA" website (Table XI) are the two largest websites in our experiment. The two tables show the results for all preference sets and configurations with a population size of 250, with and without aggregation. With 27 variables, the "UBOLEA" website returns some good adaptations for the

TABLE X. "GODADDY" ALL PREFERENCES - EXPERIMENT FIGURES.

Variables		search space
22		2×10^{99}

Prefs. set	S_1	S_{1aggr}	S_2	S_{2aggr}	S_3	S_{3aggr}	S_4	S_{4aggr}
Preferences	14	14	36	36	15	15	37	37
Objective functions	14	2	36	4	15	3	37	5
Exec. time (s)								
Average	5.28	0.32	10.00	10.00	9.67	0.50	10.00	10.00
Standard deviation	1.61	0.03	0.00	0.00	0.98	1.24	0.00	0.00
Min	3.13	0.24	10.00	10.00	5.18	0.25	10.00	10.00
Max	10.00	0.39	10.00	10.00	10.00	10.00	10.00	10.00
Generations								
Average	215	15	197	400	358	23	191	372
Standard deviation	61.7	1.5	8.3	4.2	38.0	52.5	10.1	2.5
Min	133	12	184	373	206	12	176	365
Max	398	19	222	407	407	426	232	378
Non cpltd. exec.								
Percentage	5	0	100	100	82	2	100	100
Satisfied obj. fct (avg.)	10.0	–	7.8	1.0	9.9	2.0	7.8	1.6
Standard deviation	2.8	–	1.5	0.9	3.9	0.0	1.7	0.5
Min	6	–	5	0	3	2	4	1
Max	12	–	11	3	14	2	12	2

S_1 case. For the other preference sets, we do not get any good adaptations. In the S_1 case, the aggregation provides substantial improvement. It allows us to increase the number of terminated executions by up to 86%, whereas without any aggregation it is about 6%. Moreover, it allows us to considerably reduce the execution time from nearly 10 seconds to less than 2 seconds. The execution times that enable good solutions is around the minimal execution time (about 500 milliseconds). The minimal execution time in the best cases is about 7 seconds, but only about 350 milliseconds when aggregation is used. In the same way, the average number of generations is reduced from 323 to 77 generations. The minimal number of generations is also highly reduced when we use aggregation. The maximal number is still high because at least one execution is not completed and gives a high number of generations.

The other three preference sets (S_2, S_3, S_4) do not return any good adaptation in less than 10 seconds. Nevertheless, aggregation gives a slight improvement in the number of satisfied objective functions. For instance, in the S_2 preference set, on average, 8.9 of the objective functions are satisfied out of the 48 objective functions in the non-aggregated configuration. This ratio becomes 0.7 out of 4 satisfied objective functions when we aggregate them. The number of satisfied objective functions is thus equivalent from about 18.5% to 17.5%. In the S_3 preference set, the number of satisfied objective functions is increased from about 16.8% to 33%, and it is increased from 18.8% to 26% in the S_4 preference set case.

For the "ReasonFrance" website, the behavior of executions with or without any aggregation is quite similar. The results are improved for only the first preference set S_1 when using aggregation, allowing us to obtain up to 60% of terminated executions. A good solution is returned in less than 5 seconds with aggregation and some terminated executions provide a good solution in less than 1 second. In best cases, this time is about 750 milliseconds. In all preference sets (except S_2), the aggregation allows us to really improve the number of satisfied objective functions even if no good solution is found.

D. NFB website

The last studied website is presented in Section II. It corresponds to the publications Web page of the National Federation of the Blind (NFB, nfb.org/publications, may 2014).

1) Execution results: We applied the same preference sets, again with a population size of 250. Table XIII presents figures on all of these executions.

Good adaptations usually correspond to aggregated configurations. Without aggregation, in the most complex problems (S_2 and S_4), the algorithm gives a few solutions (S_2) or none (S_4). In two other cases, with or without aggregation, all executions terminate before 10 seconds.

The S_1 preference set gives quite good adaptations in less than 200 milliseconds. The aggregation configuration for the S_1 preference set only achieves negligible improvement. In this second configuration, the execution time falls to 100 milliseconds. Furthermore, the average number of generations and the maximal number of generations is about twofold lower. On the same website, the S_3 preference set gives good results. Without any aggregation, a good adaptation is computed on average in about 260 milliseconds versus 130 milliseconds with aggregation. The aggregated configuration thus brings an improvement, while also reducing the average number of generations and the maximal number of generations by about twofold. In the aggregated configuration or in the non-aggregated configuration, all executions lead to a good adaptation in less than 400 milliseconds.

Solving the S_2 preference set is highly facilitated by aggregating objective functions. When only six percent of the

TABLE XI. "UBOLEA" ALL PREFERENCES - EXPERIMENT FIGURES.

Variables		search space
27		8×10^{122}

Prefs. set	S_1	S_{1aggr}	S_2	S_{2aggr}	S_3	S_{3aggr}	S_4	S_{4aggr}
Preferences	21	21	48	48	22	22	49	49
Objective functions	21	2	48	4	22	3	49	5
Exec. time (s)								
Average	9.91	1.70	10.00	10.00	10.00	10.00	10.00	10.00
Standard deviation	0.45	3.26	0.00	0.00	0.00	0.00	0.00	0.00
Min	6.74	0.34	10.00	10.00	10.00	10.00	10.00	10.00
Max	10.00	10.00	10.00	10.00	10.00	10.00	10.00	10.00
Generations								
Average	323	77	176	375	250	417	171	351
Standard deviation	24.0	147.6	7.1	2.8	24.3	7.6	6.3	5.9
Min	242	16	161	368	210	365	161	310
Max	364	459	208	381	346	425	196	358
Non cpltd. exec.								
Percentage	94	14	100	100	100	100	100	100
Satisfied obj. fct (avg.)	7.4	1.0	8.9	0.7	3.7	1.0	9.2	1.3
Standard deviation	4.2	0.0	1.6	0.6	1.4	0.3	1.4	0.6
Min	1	1	6	0	2	0	6	0
Max	19	1	13	2	7	2	12	3

TABLE XII. "REASONFRANCE" ALL PREFERENCES - EXPERIMENT FIGURES.

Variables		search space
59		3×10^{266}

Prefs. set	S_1	S_{1aggr}	S_2	S_{2aggr}	S_3	S_{3aggr}	S_4	S_{4aggr}
Preferences	41	41	100	100	42	42	101	101
Objective functions	41	2	100	4	42	3	101	5
Exec. time (s)								
Average	10.00	4.51	10.00	10.00	10.00	10.00	10.00	10.00
Standard deviation	0.00	4.49	0.00	0.00	0.00	0.00	0.00	0.00
Min	10.00	0.74	10.00	10.00	10.00	10.00	10.00	10.00
Max	10.00	10.00	10.00	10.00	10.00	10.00	10.00	10.00
Generations								
Average	271	175	198	317	312	348	193	293
Standard deviation	18.6	173.9	11.1	1.6	15.8	5.9	14.0	1.8
Min	230	29	171	313	279	307	167	289
Max	318	397	225	321	360	356	229	297
Non cpltd. exec.								
Percentage	100	40	100	100	100	100	100	100
Satisfied obj. fct (avg.)	4.7	1.0	16.6	0.0	8.0	1.3	16.8	0.9
Standard deviation	1.6	0.2	2.3	0.1	3.2	0.7	2.3	0.4
Min	1	0	12	0	2	0	13	0
Max	9	1	22	1	17	2	23	2

executions without aggregation lead to good adaptations in nearly 10 seconds, the aggregation configuration achieves 70% of terminated executions in less than 4 seconds. In addition, the average number of generations is 156 in the aggregated configuration, while it is 325 generation without. Solving the last preference set (S_4) is also highly facilitated when using aggregation. Aggregation allows us to increase by up to 60% the number of terminated executions before 10 s, compared to 0% without any aggregation.

2) Visual examples of computation: We randomly chose solutions returned by our implementation of NSGA-II algorithm and we manually applied them on the NFB website. We recall that the original appearance of this website is shown in Figure 1. We hereafter discuss the displayed results.

The first preference set S_1 includes preferences that guarantee minimal brightness contrast between the text and its direct background. The minimal required contrast is slightly over the contrast recommended by the Web Content Accessibility Guidelines (WCAG 2.0). Figure 11a is the application of one computed adaptation. In the left menu, in top navigation bar and the content, the contrast is not especially high, but it satisfies the minimal requested contrast. The first part of the menu is somewhat light green and the second part yellow. These two colors are far from the original colors. The same

TABLE XIII. "NFB" ALL PREFERENCES - EXPERIMENT FIGURES.

Variables		search space
9		4×10^{40}

Prefs. set	S_1	S_{1aggr}	S_2	S_{2aggr}	S_3	S_{3aggr}	S_4	S_{4aggr}
Preferences	5	5	14	14	6	6	15	15
Objective functions	5	2	14	4	6	3	15	5
Exec. time (s)								
Average	0.16	0.10	9.73	3.49	0.26	0.13	10.00	4.66
Standard deviation	0.05	0.02	1.17	4.37	0.05	0.02	0.00	4.61
Min	0.07	0.05	2.54	0.21	0.13	0.07	10.00	0.30
Max	0.30	0.14	10.00	10.00	0.39	0.20	10.00	10.00
Generations								
Average	8	5	325	156	13	7	325	184
Standard deviation	2.3	1.1	37.3	192.7	2.5	1.2	13.9	181.3
Min	4	3	112	11	7	4	298	15
Max	16	8	385	447	20	11	367	424
Non cpltd. exec.								
Percentage	0	0	94	30	0	0	100	40
Satisfied obj. fct (avg.)	–	–	8.3	2.1	–	–	7.9	2.9
Standard deviation	–	–	1.8	0.4	–	–	1.6	0.8
Min	–	–	5	1	–	–	5	2
Max	–	–	12	3	–	–	13	4

applies for the color of the top navigation bar and the color of the main content. The computed colors only satisfy the minimal brightness contrast constraint.

When we add to the minimal brightness contrast constraint and the color closeness constraint, we reduce the possible colors for each object. Figure 11b shows the application of an adaptation computed using the S_2 preference set. If we consider the left menu, the new colors of the background and of the text allow us to have the minimal required contrast. Moreover, the background colors are not too far from the original colors chosen by the designer. We have a problem with the top navigation menu. This is normal regarding our implementation of the distance between two colors. We will explain this effect in the application of the S_4 preference set.

In Figure 11c, instead of adding the color closeness constraint, we add the uniform background brightness constraint. This constraint allows us to have near light emission for all backgrounds on the page to avoid dazzle. Like in the first example (Figure 11a), colors are randomly chosen during the computation process. However, the minimal required contrast is respected, and all backgrounds are dark. Thus, the uniform background brightness constraint is also satisfied.

With all preferences (S_4 preference set), the result is not totally satisfactory. The minimal required contrast is respected. The near background brightness is also satisfied. Unfortunately, the closeness to the original colors is not very good. As already noted in the S_2 example, some colors are far from the original colors. This is especially the case for grey-based colors, including black and white. The color difference is revealed by comparison of the hue of the two colors. The hue of the black, the white and all grey colors is actually equal to 0 (there is no hue). Colors have hues from 0 to 360 degrees, thus, colors with a degree of around 0 are considered close to all grey scale colors. The white background of the main content can thus to be transformed into a dark colored background. In a future work, we will study how to change the implementation of our color distance function to avoid such effects.

E. Threats to validity

1) Construct validity: We have considered several configurations to apply the NSGA-II algorithm, especially for analyzing the effect of the population size and the potential benefit or drawback of aggregated objective functions. During the experiment, we collected information, including the execution time, the number of generations, and the number of satisfied objective functions (aggregated or not). The execution time tells us whether all objective functions could be satisfied in less than 10 seconds and indicates if the approach can be used in practice. The number of generations allows us to analyze the effect of the population size regarding the elapsed time. For the same time, a larger population can correspond to fewer generations (this refers to the complexity of NSGA-II). For the non-terminated executions, we collect the number of satisfied objective functions to get a quality indication. For each preference set, we compare the non-aggregated implementation with the aggregated one. The case without aggregation is easy to interpret because it easily shows quality results for each preference. However, the aggregated configuration allows us to reduce the number of objective functions, which plays an important role in the complexity of the algorithm. Unfortunately, with aggregation, it is harder to know the impact of changes on each preference. Furthermore, in aggregation, we chose to associate two objective functions to a general preference, but we could associate more objective functions. However, with two objective functions, we study configurations, which are very different from the non-aggregated configurations, while keeping a multi-objective approach. Other values could be

(a) S_1 on the NFB website

(b) S_2 on the NFB website

(c) S_3 on the NFB website

(d) S_4 on the NFB website

Figure 11. Adaptations of the NFB website.

chosen for the mutation (versus crossover) probability. The analysis of the results would have been more complex, without giving any substantial benefit.

2) Internal validity: The extraction of variables from the websites was done manually. Two persons participated in gathering this information. All colors were obtained from Web pages using two different software systems: the "gcolor2" on Linux system and the digital colorimeter included in the MAC OS platform (OS X 10.8.5). Even if colors can differ from one screen to another, such software systems, by getting values from the "display manager" functionalities, are not concerned by this issue. The defined model (variables, constraints) simplifies pages by only encoding variables and constraints concerned by the currently used preferences.

3) External validity: We have considered five very different websites in terms of size, architecture and colors used. Websites were also chosen for noticed accessibility problems for visually impaired people. People who reported accessibility problems have a visual impairment (low or high). We focused the study on the contrast problem and brightness problem because they are the most difficult problems to solve today with existing assistive technologies. Several dimensions of the search space and variables included in the constraints were considered with the aim of studying variations in the resolution. We feel that the variability in the results obtained between NSGA-II executions is significant on this kind of input data.

4) Conclusion validity: We have considered several measures, including min, max, standard deviation and average execution time, generation number and number of satisfied objective functions. We considered 60 executions for each configuration. In our previous paper [1], we compared results of NSGA-II with results obtained with an exact algorithm from preference theory. The exact algorithm appeared to be totally unusable (on a standard computer) when we had a search space of 10^5. We also checked if there are cases where our algorithm returns a good solution in the random initialization phase.

VI. Conclusion and Future Work

Widely used assistive technologies now partially meet their objective of helping people with low vision. However, their general purpose often leads to rather simple adaptations that do not provide relevant improvement for specific visual impairment. Moreover, they may also radically change the appearance of websites: part of the website design may be lost, while navigation marks (deliberately placed or not) helping internet users may disappear. As there is a substantial website diversity in terms of architecture, number of objects, colors used or even in font styles, we propose and test a new approach to adapt Web pages. This approach is based on real user needs (modeled by preferences) to find an adaptation that fits for these preferences.

To address scalability issues when we work with real websites, our approach is based on a multi-objective optimization algorithm (NSGA-II). We describe and carry out an experimentation on five very different websites. The chosen websites have been noticed by users with some impairment as websites with accessibility issues. Research has shown that this approach works fine for small and some medium websites with contrast or color distance preferences. We also learn lessons about limitations of the algorithm regarding running time and the number of considered objects (variables). This experiment also highlights the significant benefits provided by the aggregation of objective functions.

As future work, we plan to investigate other resolution approaches by modeling our problem as a CSP (Constraint Satisfaction Problem). The evaluation of our CSP modeling on existing constraint solvers will be compared with the results obtained with NSGA-II. The main and original target is to help people with low vision, thus we will test our approach in real conditions with users with visual impairment. This evaluation will be conducted on frequently visited websites by previously defining quality indicators. Testing websites with visually impaired people may help us to get information on how to facilitate the expression of preferences by the concerned end user. In parallel, we will work on object or structure recognition from Web pages. This phase is important to automate the whole process. Object and structure recognition is especially complex and requires refined heuristics for HTML4 Web pages because they are not explicitly defined in the source code.

ACKNOWLEDGMENT

The authors would like to thank Berger-Levrault, which supported this work with a grant, and Rémi Coletta for his valuable remarks about the way of reporting the evaluation.

REFERENCES

[1] Y. Bonavero, M. Huchard, and M. Meynard, "Web page personalization to improve e-accessibility for visually impaired people," in Proceedings of the Second International Conference on Building and Exploring Web Based Environments (WEB 2014,), 2014, pp. 40–45.

[2] "Visual impairment and blindness, fact sheet n°282," World Health Org., http://www.who.int/mediacentre/factsheets/fs282/en, Oct. 2013, accessed: 2014-11-09.

[3] Web Content Accessibility Guidelines, World Wide Web Consortium, http://www.w3.org/TR/WCAG20/, accessed: 2014-11-09.

[4] User Agent Accessibility Guidelines, World Wide Web Consortium, http://www.w3.org/TR/UAAG20/, accessed: 2014-11-09.

[5] Authoring tools Accessibility Guidelines, World Wide Web Consortium, http://www.w3.org/TR/ATAG20/, accessed: 2014-11-09.

[6] Web Accessibility Initiative - Accessible Rich Internet Applications, World Wide Web Consortium, http://www.w3.org/WAI/intro/aria, accessed: 2014-11-09.

[7] "References on evaluation tools by WAI," http://www.w3.org/WAI/ER/existingtools.html, accessed: 2014-11-09.

[8] M. Y. Ivory, J. Mankoff, and A. Le, "Using automated tools to improve Web site usage by users with diverse abilities," Information Technology and Society, vol. 3, no. 1, 2003, pp. 195–236.

[9] J. T. Richards and V. L. Hanson, "Web accessibility: a broader view," in WWW '04: Proceedings of the 13th international conference on World Wide Web. ACM Press, 2004, pp. 72–79.

[10] C. Domshlak, R. I. Brafman, and S. E. Shimony, "Preference-based configuration of Web page content," in Proceedings of International Joint Conference on Artificial Intelligence (IJCAI), 2001, pp. 1451–1456.

[11] B. Mobasher, R. Cooley, and J. Srivastava, "Automatic personalization based on Web usage mining," Com. ACM, vol. 43, no. 8, 2000, pp. 142–151.

[12] K. Z. Gajos, J. O. Wobbrock, and D. S. Weld, "Automatically generating user interfaces adapted to users' motor and vision capabilities," in Proceedings of the 20th Annual ACM Symposium on User Interface Software and Technology (UIST), C. Shen, R. J. K. Jacob, and R. Balakrishnan, Eds. ACM, 2007, pp. 231–240.

[13] J. Abascal, A. Aizpurua, I. Cearreta, B. Gamecho, N. Garay-Vitoria, and R. Miñón, "Automatically generating tailored accessible user interfaces for ubiquitous services," in Proceedings of the 13th International ACM SIGACCESS Conference on Computers and Accessibility (ASSETS), K. F. McCoy and Y. Yesilada, Eds. ACM, 2011, pp. 187–194.

[14] A. Mereuta, S. Aupetit, and M. Slimane, "Improving Web accessibility for dichromat users through contrast preservation," in Proceedings od the 13th International Conference ICCHP (International Conference on Computers Helping People with Special Needs), ser. Lecture Notes in Computer Science, K. Miesenberger, A. I. Karshmer, P. Penáz, and W. L. Zagler, Eds., vol. 7382. Springer, 2012, pp. 363–370.

[15] L. Balme, A. Demeure, G. Calvary, and J. Coutaz, "Sedan-Bouillon: a plastic Web site," 2005, plastic Services for Mobile Devices (PSMD), Workshop held in conjunction with Interact'05, Rome, 12 Sept. 2005.

[16] M. Macías, J. González, and F. Sánchez, "On adaptability of Web sites for visually handicapped people," in Proceedings of the second International Conference on Adaptive Hypermedia and Adaptive Web-Based Systems (AH), ser. Lecture Notes in Computer Science, P. D. Bra, P. Brusilovsky, and R. Conejo, Eds., vol. 2347. Springer, 2002, pp. 264–273.

[17] K. Cullen, L. Kubitschke, T. Boussios, C. Dolphion, and I. Meyer, "Web accessibility in European countries: level of compliance with latest international accessibility specifications, notably WCAG 2.0, and approaches or plans to implement those specifications," http://ec.europa.eu/digital-agenda/en/news/study-report-web-accessibility-european-countries-level-compliance-latest-international, 2009, accessed: 2014-11-09.

[18] S. Kaci, Working with Preferences: Less Is More, ser. Cognitive Technologies. Springer, 2011, iSBN:978-3-642-17279-3.

[19] J. Horn, N. Nafpliotis, and D. Goldberg, "A niched Pareto genetic algorithm for multiobjective optimization," in Proceedings of the First IEEE Conference on Evolutionary Computation. IEEE, 1994, pp. 82–87.

[20] J. Knowles and D. Corne, "The Pareto archived evolution strategy: A new baseline algorithm for Pareto multiobjective optimisation," in Proceedings of the Congress on Evolutionary Computation, vol. 1. IEEE, 1999, pp. 98–105.

[21] E. Zitzler and L. Thiele, "Multiobjective evolutionary algorithms: a comparative case study and the strength Pareto approach," IEEE Trans. Evolutionary Computation, vol. 3, no. 4, 1999, pp. 257–271.

[22] K. Deb, S. Agrawal, A. Pratap, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," IEEE Trans. Evolutionary Computation, vol. 6, no. 2, 2002, pp. 182–197.

[23] M. Harman, S. A. Mansouri, and Y. Zhang, "Search-based software engineering: trends, techniques and applications," ACM Comput. Surv., vol. 45, no. 1, Dec. 2012, pp. 11:1–11:61. [Online]. Available: http://doi.acm.org/10.1145/2379776.2379787

[24] M. Laumanns, L. Thiele, K. Deb, and E. Zitzler, "Combining convergence and diversity in evolutionary multiobjective optimization," Evolutionary computation, vol. 10, no. 3, 2002, pp. 263–282.

[25] "Conversion from RGB to HSV," http://en.wikipedia.org/wiki/HSL_and_HSV, accessed: 2014-11-09.