Reconciling user and designer preferences in adapting web pages for people with low vision

Comparing NSGA-II and NSGA-III evolutionary algorithms

Yoann Bonavero
LIRMM
CNRS & Montpellier University
Montpellier, France
bonavero@lirmm.fr

Marianne Huchard
LIRMM
CNRS & Montpellier University
Montpellier, France
huchard@lirmm.fr

Michel Meynard
LIRMM
CNRS & Montpellier University
Montpellier, France
meynard@lirmm.fr

ABSTRACT

The web has become a major tool for communication, services and an outstanding source of knowledge. It has also grown in complexity, and end-users may experience difficulties in reading and acquiring good understanding of some overly complex or poorly designed web pages. This observation is even more valid for people with visual disabilities. In this paper, we focus on people with low or weakening vision, for whom we propose to adapt web pages to their needs, while preserving the spirit of the original design. In this context, obtaining a web page adaptation in a very short time may be a difficult problem, because user and designer needs and preferences may contradict each other, and because there may be a large number of adaptation possibilities. Finding a relevant adaptation in a large search space can hardly be done by an algorithm which computes and assesses all possible solutions, which brings us to consider evolutionary algorithms. A characteristic of our problem is to consider a set of preferences, each being implemented by an evaluation function. This optimization problem can be dealt with multi-objective genetic algorithms, including the Non-dominated Sorting Genetic Algorithm II (NSGA-II) and its next version (NSGA-III). NSGA-III has been recently introduced to address many-objective optimization problems (having more that four objectives). We compare NSGA-II and NSGA-III performances in the context of adapting web pages in accordance to a set of preferences. The comparison is based on running time, number of generations and quality of computed adaptation (number of satisfied objectives). We also show the importance of several parameters including population size, crossover/mutation probability, and the opportunity to aggregate objective functions. From the obtained results, we conclude that the approach is feasible and effective on realistic web pages, especially with NSGA-III.

Keywords

e-accessibility, web page personalization, visually impaired, low vision, evolutionary algorithm, NSGA-II, NSGA-III

W4A '15, May 18 - 20, 2015, Florence, Italy

ACM 978-1-4503-3342-9/15/05...$15.00
http://dx.doi.org/10.1145/2745555.2746647.

1. INTRODUCTION

The world wide web is meeting a great success in increasing communication, providing services and disseminating knowledge. These opportunities are supported by the constant evolution of software and hardware technologies.

In the web world, the highly and constantly increasing bandwidth and the wide use of high-resolution screens, encourage developers and designers to enrich their websites regardless the web page complexity. In addition, designers propose more and more complex and rich graphic charts. Today, aesthetics and appearance are of great importance, sometimes even compared to website functionalities. Web browsing is becoming a visual experience for a large part of end-users. Pages are colorful, contain many images and use complex element positioning systems and organization layout.

However, this evolution may have a negative impact, especially for people with disabilities. Visually impaired people are very affected by such rich page design. Colorful websites, or web pages using color to convey information, may generate some difficulties for color-blind people. Not providing alternative text into an image containing textual data is a major barrier for blind people in accessing potentially important information. Artistic font family, and non-regular text style, may lead to reading difficulties for people with dyslexia. With very high-resolution screens, graphical interfaces may have tiny fonts which may be a problem, not only for old and disabled people.

Many countries are adopting treaties or laws in order to enhance digital accessibility. In some countries, e-accessibility is considered as a citizens' right and is a very important issue. Worldwide, about 246 million people are visually impaired and 39 million people are blind [39]. Mainly due to the increasing life expectancy, these figures are constantly growing.

Current assistive technologies, like JAWS or MAGic [6], Window-Eyes [11], VoiceOver [10], that are widely used, may address some accessibility issues. Some are included in (or a companion of) operating systems, such as Mac OS X [3], Windows 8 [2], Android [1] or Linux [7, 8]. In the low vision context, screen magnifiers are helpful to improve text readability and visualization of parts of a web page. There exist several commercial and non commercial software, such as "Enhanced desktop zoom" (Compiz accessibility plugin) or Orca for Linux, screen magnifier solution of Mac OS, and Magic or Zoomtext for Windows. They allow end-users to apply color filters on the screen, to modify the arrow and cursor, or to zoom in the screen to obtain a better view on specific parts. Nevertheless, due to high maintenance cost and very specific needs of end-users, assistive technologies only provide a set of generic tools that hardly meet the need of specific adaptation. Web browser

options have also been proposed [4, 9]. The page style can be completely redefined, which may cause the complete loss of the initial design, and these tools also propose a set of generic and global modifications.

Independently of tools, accessibility standards have been proposed. The W3C (World Wide Web Consortium) and other organizations publish sets of technical specifications in order to guide accessible website development. The compliance to these specifications provides a minimum level of accessibility, for people who do not use the often expensive assistive technologies. Besides, they guarantee a satisfactory access to third-party software. Specifications are divided into three main guidelines including WCAG 2.0 (Web Content Accessibility Guideline [43]), UAAG (User Agent Accessibility Guideline [41]) or ATAG (Authoring tool Accessibility Guideline [40]). WAI - ARIA (Web Accessibility Initiative - Accessible Rich Internet Application [42]) is specialized in rich and dynamic interfaces (for example Javascript web user interfaces). Nevertheless, despite the efforts of some developers to comply with the standards, many websites have a low level of accessibility because the standards are too broad [36].

Other approaches propose more personalized web page adaptation. Some are dedicated to specific deficiencies, like color-blindness in [36]. They often only consider low-level elements in the web page like [37, 13] and simple description of end-user needs or no description at all.

In this paper, we aim to consider end-user preferences, but also designer preferences, that may be more or less complex and be expressed on properties of elements of the page. For example we may want all backgrounds in the page with a similar brightness, a minimal contrast between a specific text and its background, and keep colors as close as possible to the original colors. Such preferences may contradict each other, leading to a large number of adaptation possibilities. To explore this possibly large space of solutions in a very short time, we consider evolutionary algorithms. Besides, as we consider a set of preferences, we choose to use *multi-objective* evolutionary algorithms, which produce optimal solutions that make trade-offs between the preferences. In a previous work [14], we proposed an approach based on the Non-dominated Sorting Genetic Algorithm II (NSGA-II), that we tested on a set of websites to check its feasibility. The study showed the benefits but also the limits of the approach regarding running time and the number of satisfied preferences. A variant of NSGA-II, namely NSGA-III, has been recently introduced to specifically address *many-objective* optimization problems (having more than four objectives). As we may have many preferences, we carry out a new study to investigate the potential benefits brought by NSGA-III. We compare NSGA-II and NSGA-III running time, number of generations and of satisfied objectives. We study the effect of parameters including population size, crossover and mutation probability, and the impact of aggregating objective functions.

Section 2 describes research work that aims to improve web accessibility for low vision users. In Section 3, we illustrate and outline our approach on an actual web page. Then, we present in Section 4 the principles of preference representation and of the multi-objective evolutionary algorithms. Section 5 presents how we are encoding our problem and how we are tuning the algorithms to obtain satisfactory results. Section 6 reports and discusses the results, and we conclude in Section 7 with perspectives of our work.

2. EXISTING APPROACHES

Nowadays web page adaptation is addressed by commercial assistive tools, dedicated configurations in operating systems, web browser extensions, and by research work that aims to take better account of user needs and preferences. Some approaches deal with specific deficiencies like color vision deficiency [36] [38], while in our case, we aim to propose a general solution for low vision users.

As mentioned in [34], architectures of the proposed prototypes divide into client-side [34, 36, 13], proxy-based [37], host-content-server-side, web page [12] or web-service. The tool can be intended for end-user only [36, 34, 32, 37], for the designer [32], for an assistive tool [30, 31, 25] or for multiple recipients [13, 12]. Our prototype tool is designed to be used on the client-side and targets the end-users.

The proposed approaches deal with different kinds of elements. Some only consider basic elements such as specific text, color or background [36, 37], while others focus on structure and roles of the elements, such as lists, menus, sections, fields or titles [30, 31, 25]. For example, in [25], labels are assigned to form fields by probabilistic optimization approach, in order to ease the reading and filling of the fields. Besides, more advanced approaches propose their own meta-model or language [32, 30] to be able to manipulate high-level abstractions. In our case, we let the users express their preferences at several levels of abstraction, using variables that describe pairs composed of one relevant element of any kind, and one possibly complex property. A preference is any evaluable function expressed on variables and their domains. Besides we take into account designer preferences by the mean of evaluable functions. Approaches that deal with higher level elements often need procedures to recognize them. This is done for example with pattern recognition and source code analysis in [32].

During the transcoding, the modified elements often are HTML [30, 37] and CSS rules [13, 34], but may also be scripts, under some conditions [34, 36]. In [31], the web content is enriched with CNR (Content Navigation Rules) by introducing ARIA statements. This allows the user to navigate using the screen reader and key presses. In [12], alternative text is automatically produced either by Optical Character Recognition from a picture or by decoding an image URL. The modifications may consider the whole web page [36], or only specific parts [20, 21, 33]. In our proposal, we modify the whole web page.

The user profile can be a set of expected values for some characteristics, like font or color [37, 36, 33]. The user profile may be predefined for user categories known in advance, as hyposight and different color blindness kinds in [36]. This user profile can be entered by the user via an interface, giving values for the characteristics [37, 34] or from visual tests [22]. In a more innovative approach [20, 21, 33], the user profile is learned from user actions and Q-learning. In this paper, we do not address the user profile acquisition, which we will consider in a future work.

Depending the form of the user profile, the adaptation step can take different forms. In [36], authors apply the predefined profile; in [33], the learned profile is applied. In Cloud4all project [5], participants work on a Global Public Inclusive Infrastructure (GPII), which will allow a user profile to be easily implemented to personalize any device and content. In our case, as our preference set may contain conflicting preferences, a trade-off has to be found between the preferences. This is done by using a meta-heuristic algorithm in order to get a good approximate solution.

Some approaches propose at the same time adaptations and metrics to judge the result [36] or to guide the process [25]. In the evolutionary algorithms, we use objective functions, that can be considered as metrics, to guide the search.

Concerning the respect of the designer preferences, we incorporate them in our constraint set, potentially increasing conflicts between preferences (let us note that such conflicts can happen even between the user preferences). In [20, 21, 33], authors make lo-

cal changes to the web page, which has the effect of keeping the designer preferences on the unaltered parts. To our knowledge, the other approaches do not consider this aspect, and transcode the web page only on the basis of the user preferences.

3. ADAPTING WEB PAGE WITH COMBINED USER AND DESIGNER PREFERENCES

In this section, we outline our approach on a realistic example inspired by the website of an e-commerce platform, this website is represented in Figure 1. This website is very colorful, and for some users, contrasts are not high enough, especially in main menu tabs.

In this colorful context, our first user preference is defined by a minimal contrast between text and background, in order to increase readability of pages. Moreover, we define a maximal distance between the original color context and the computed one in order to keep as much as possible the designer preference. We aim to compute web page adaptation that makes a trade-off between the user preference (minimal brightness contrast) and the designer preference (original colors). One of the best adaptations that our approach computes is shown in Figure 2.

Figure 1: Part of original "Lorem Ipsum" home page with "All products" tab opened.

Figure 2: Adapted "Lorem Ipsum" home page with "All products" tab opened - minimal contrast is ensured, and color context is preserved as much as possible.

The approach is based on the complete or partial knowledge of the HTML source code of the page we want to adapt. It considers low-level structural elements like paragraphs, links or titles as well as higher level structural elements like header, footer, article, menu or navigation bar. These high-level structural elements are indicated to a large extent by tags in HTML 5 and Accessibility Rich Internet Applications (ARIA [42]) standards. In addition, we associate relations between the structural element features. HTML 5 and ARIA are not required, but in the case of older versions of HTML, it is needed to add some recognition process to extract high-level structural information. If we consider the preference "if text font size is 12pt, the title font size must be at least 16pt font size", a relation is established between the font size feature of the two elements (text and title). If we consider a preference involving higher-level structural elements, such as "In articles, the summary should be highlighted to the content", a relation is established between the writing style of the summary and the main content of the same article.

When a page is loaded into the web browser, we get all HTML elements and their features concerned by the currently defined preferences. The tasks of getting high-level HTML elements when the page is loaded and applying new property values currently are manually done and doing it automatically is an on-going work. These tasks require indeed complex page decomposition with clever structure recognition algorithms that are able to capture the perception of the page by the end-user [17, 16]. In this paper, we focus on computing a relevant adaptation, given the HTML elements and their properties. Even on medium size pages (in terms of HTML elements), this is a difficult problem which can hardly be dealt with an exact solution, as we show in [15] with approaches from Preference Theory [27]. To face this complexity problem, we implement and tune multi-objective genetic algorithms. Making a trade-off between all preferences, these algorithms search for good adaptations and give results in acceptable running time. Then, we apply the new (computed) property values to the appropriate HTML elements. The next section explains our problem into more details and describes the two algorithms used, that we will compare on real web pages.

4. PROBLEM AND RESOLUTION

We describe here the problem we want to solve, how preferences are represented and the resolution algorithms used.

4.1 Problem definition

The main purpose of the page processing is to modify a web page regarding designer and user preferences. These preferences may come from users with disabilities that express their needs through preferences or may come from non disabled persons who want to improve the page appearance (or just for fun). We get the set of HTML elements and their associated properties related to the defined preferences. As a result, the size of the built set depends on the nature of the website (in terms of architecture and number of elements) and also on the kind of defined preferences. From HTML elements and their properties, we create a set of variation points, that we call variables (Eq. (1)). Each variable associates a property with its HTML element and has a domain of values. For instance, a variation point or variable can be "color of menu background", "font size of links", or "level 3 title font family".

$$VariationPoints = \{V_1, V_2, .., V_m\} \quad (1)$$

Our aim is to find a value for each variable to satisfy at the same time all preferences. Thus preferences are represented by relations linking variables. The problem definition and the chosen resolution algorithms are driven by the desire to keep this choice as independent as possible from any specific web page or preference.

4.2 Preference representation

Preferences connect variables. A constraint is the instantiation of a preference kind on particular variables. For example the preference kind defining minimal contrast between text and its direct background is instantiated for each textual HTML element.

$$Constraints = \{C_1, C_2, .., C_k\} \quad (2)$$

All choices made on one or more variables constitute preferences or wishes. This is different from most of the existing work dealing with user preferences. For example, in [35], literal values like *10 pt* or *16pt* for a font size, *blue* or *yellow* for a color are object properties. Instead, in our approach we use constraints between variables.

We use basic preference (Eq. (3)) and conditional preference (Eq. (7)) representation from Preference Theory [27]. Furthermore we expand them to hold "more complex" preferences.

$$V_i \; op \; x_i \; >_p \; V_j \; op \; y_j \quad (3)$$

In Eq. (3), 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 an 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 green color for links to blue one", we define the variable c_L to represent the color of links (L is the link set). The c_L domain is denoted by $D(c_L)$ in Eq. (4).

$$D(c_L) = \{black, red, blue, yellow, purple, green\} \quad (4)$$

The user's wish is expressed as in Eq. (5).

$$c_L = green >_p c_L = blue \quad (5)$$

Conditional preference form is shown in Eq. (6).

$$V_k \; op \; x_k \; : \; V_i \; op \; x_i \; >_p \; V_j \; op \; y_j \quad (6)$$

For example, in Eq. (7), we model the following preference: "If the font color is yellow, I prefer bold font to regular font". To represent the weight of the text concerned by this preference, we introduce a new variable w_T. The operator ":" appears here to indicate the condition and the action to be done if the condition is satisfied. This representation was considered in our previous paper [15].

$$c_T = yellow : w_T = bold >_p w_T = normal \quad (7)$$

Furthermore, we consider any complex function on variables and their domain values. With these considerations, we can express preferences like "I would like to have a text font size greater than 12pt", "I would like to have black text if its size is less than 10pt" or "I would like to have a minimal brightness contrast of 30% between texts and their direct backgrounds". The brightness contrast is a binary function that represents the distance between color brightness of two objects. To model this last preference, we create the object T representing a textual element and the object B for its direct background. From these objects we define two variables c_T and c_B to represent the text element color and the background element color. Then, we define a brightness contrast function $contrast(x, y)$. This binary function takes as parameter the two colors and returns the value of their brightness contrast. The result of this function is compared with a threshold specified by the user. To determine the satisfaction of a constraint, we evaluate Eq. (8) where l is the desired threshold.

$$contrast(c_T, c_B) \geq l \quad (8)$$

4.3 Resolution algorithm

As said previously, the number of HTML elements in a web page multiplied by the number of considered properties can be potentially huge, and so is the number of solutions to an adaptation problem. Functions like contrast or lightness are not hard to compute for each solution, however the search space size makes impossible the use of exact algorithms. This motivated us to use an optimization algorithm. For example, in our smallest studied website, with 9 color variables, and with only 3 values for each red, green, and blue color components, we obtain about 7.6×10^{12} solutions that cannot be computed on standard computer with exact algorithms. Let us remark that this drastic domain reduction reduces the number of good solutions and may even remove all good solutions.

When there is no solution satisfying all constraints, we are interested in finding approximate solutions. Regarding the nature of the problem and its multi-objective dimension (several preferences and then constraints), we considered multi-objective evolutionary algorithms [24, 28, 44]. We firstly used the Non-dominated Sorting Genetic Algorithm II (NSGA-II), which gave us acceptable results on some websites. Recently, a new version has been proposed (NSGA-III), which may be more adapted to our specific case, where we have many objective functions (connected to many preferences). NSGA-III is still in early stage, while NSGA-II is popular in search based software engineering [23].

NSGA-II [18] considers an initial (possibly randomly built) population P_0 of N solutions (or individuals). P_t is step t population. 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 merged to constitute R_t population. The N best individuals of R_t in terms of non-dominance and diversity are kept to form P_{t+1} as follows. Several fronts are calculated that group solutions. The first non-dominated front (F_1) groups non-dominated individuals (that correspond to the best known solutions) with regard to at least one objective. We state that a solution s_1 *dominates* another solution s_2 if: (i) s_1 is better 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) gathers the non-dominated individuals of $R_t \setminus F_1$. The third non-dominated front (F_3) gathers the non-dominated individuals of $R_t \setminus (F_1 \cup F_2)$, etc. The P_{t+1} population is composed of the k first fronts whose union has less than N elements. Then to get N elements, the P_{t+1} population is filled with individuals selected in $k+1$ front, based on a crowding distance [29] and a binary tournament selection operator. The crowding distance is used 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 is used to select scattered solutions. These steps are repeated until some stopping criteria are satisfied.

NSGA-III [19, 26] differs from NSGA-II in the selection procedures and the point where they occur in the algorithm.

In NSGA-II, cross-over and mutation operators are applied to a selected part of the current population. This selection is made by binary tournaments. Two individuals are repeatedly randomly selected in the current population and the best in terms of crowding distance and rank is kept. In NSGA III, all individuals of the current population have equal chance to be crossed or mutated (no prior selection). When the current population has N individuals, application of cross-over and mutation operators produces N individuals (offspring population). The obtained $2N$ individuals (current population and offspring population) are sorted in non-dominated fronts and the k first fronts whose union has less than N elements are included in the next population (as in NSGA II). If the new popula-

tion did not meet N individuals, the remaining individuals have to be picked in the $k+1$ front (denoted by F_l for "Last Front"). The underlying idea is to favor diversity (choosing individuals that are isolated in the objective function value space). While in NSGA II it is based on the crowding distance between F_l individuals, NSGA-III uses a different strategy for preserving diversity which is based on reference points on the whole population.

The reference points are uniformly distributed on the M dimensional space where M is the number of objectives functions that are normalized. The number of reference points depends on the number of divisions that are chosen for each axis. The number of divisions is a parameter of the algorithm. A systematic method can consider the M dimensional space where each axis represents the values of one normalized objective function. Normalization consists here to bring the values between 0 and 1. We build the hyper-plane by connecting the 1 value of each objective function (axis). The reference points are uniformly placed on this hyper-plane. In the very simplified case of 3 objective functions and 2 divisions, we obtain a triangle where vertices have coordinates (1, 0, 0), (0, 1, 0) and (0, 0, 1), Reference points are obtained by dividing each triangle segment in two parts. This gives 6 reference points (one for each apex and one for each segment center). With 3 divisions, we obtain 10 reference points because the triangle surface also has reference points uniformly placed on it. Individuals are then associated to a reference point as follows. The half straight line connecting the coordinate marker origin to a reference point is called a reference line. We compute the perpendicular distance between an individual and each reference line. The individual is associated with the reference point corresponding to its closest reference line. For each reference point, the algorithm counts the number of individuals of the k first fronts that are associated with it. Each individual of last front F_l is associated to a reference point as a potential individual for the selection. Then for choosing an individual in F_l for filling the next population, the algorithm looks for the first found reference point which has the smallest individual count. If the reference point has at least one associated individual in F_l, the closest individual is chosen and added to the population. This is repeated until the next population (under construction) has N individuals.

5. COMPARISON SETUP

In this section, we describe the experimental setup that we will use to compare NSGA-II and NSGA-III .

The websites included in the dataset have been chosen to keep as much as possible a diversity in terms of number of HTML elements, number of used colors and because they may cause readability issues for people with disabilities. Website element number varies from dozen to hundreds of elements.

Due to the non determinism of the studied algorithms (crossover and mutation operators are indeed stochastic techniques), we repeat many times each configuration in order to smooth random effects. Configurations include variation of the population size, aggregation of objective functions or not, stopping criteria, preference sets, etc.

In order to remain in the framework of our previous results [14] and increase confidence in the results, we use similar global configurations described in Table 1. For each configuration, we run the algorithm 30 times to reduce random effect impacts. The number of generations is not our stopping criterion, unlike usual practice. We decided to stop an execution either if we have a solution that satisfies all the preferences, or if execution lasts more than 10 seconds. These two stopping criteria are combined to many other measurements to compute comparison tables in Section 6. For this experiment, we defined three general preferences, which match practical problems for people with low vision.

- GP_1 : Have an uniform global background color brightness.
- GP_2 : Have a minimal contrast between the text and its direct background.
- GP_3 : Substitute an original color with a close one.

When we associate a general preference GP_i to a specific website, we obtain many preferences P_i corresponding to each element or element group on which the preference can be applied. For example, considering the last general preference GP_3 (*Substitute an original color with a close one*), if n colored elements on the page have a chance to be modified, we will obtain n preferences. When this preference is defined/selected by the user, each element included in the adaptation process is linked to its original color by a function comparing the distance with new color. The second preference GP_2 (requesting a minimal contrast between text and direct background) is defined to get a minimal brightness contrast for each text on a page regarding its background. The brightness used in the contrast computation is the relative perceived brightness defined in WCAG 2.0 [43]. This constraint is used to increase the readability of documents. The uniform background brightness general preference GP_1 is defined to set a global constraint between all background brightnesses on the page. It can be used to avoid dazzle due to the presence of light elements near dark elements.

In our context, users define their own preferences. This is not based on guidelines, like WCAG 2.0.

In this setup, we use four main different preference sets plus their aggregated version.

$$S_1 = P_2 \tag{9}$$

$$S_2 = P_2 \cup P_3 \tag{10}$$

$$S_3 = P_1 \cup P_2 \tag{11}$$

$$S_4 = P_1 \cup P_2 \cup P_3 \tag{12}$$

The first preference set S_1 contains only one general preference. It is defined by a user with difficulties in perceiving colors or lightness, and who wants to increase contrast. The S_2 preference set includes P_2 and P_3 preferences. This configuration is used when a user wants to increase brightness contrast to improve readability, while avoiding changing a lot of colors regarding the original page color context. The third preference set S_3 also includes P_2 preferences but here, together with P_1 preferences. The result of adaptation does not take care about the original color context but applies new background colors to get similar brightness on the entire page. The last preference set P_4 is the combination of all P_1, P_2 and P_3 preferences. We want at the same time a minimal contrast between text and its direct background, changing all backgrounds to get similar brightness, while choosing colors close to the original ones.

Genetic algorithms use objective functions to assess preferences. To implement the minimal contrast function, we compare the result of the brightness contrast function with a predefined threshold. The comparison result is a quality indicator (satisfaction). All objective functions representing preferences used in this experiment are based on this pattern (comparing a value with a threshold). In our experiment, we consider either one objective function for each preference, or aggregated objective functions that encapsulate several preferences (and reduce the number of objective functions). The number of aggregated functions is a parameter of our experiment.

Shared parameters are detailed in Tables 1 and 2. Due to specific parameters of each algorithm, configurations take into account the

NSGA version. Population size is a parameter of NSGA-II, while for NSGA-III, population is calibrated using the number of reference points. Tables 3 and 4 give a summary of the parameters used for each version.

Table 1: Global configuration summary.

Parameter	Value(s)
Crossover (resp. Mutation) prob.	0.94 (resp. 0.06)
Max exec time	10s
Max generations	unlimited
Repeat	30 times

Table 2: Preferences configuration summary.

Parameter	Value(s)
Preference sets	4
Maximal brightness difference	40%
Maximal color distance	40°
Minimal contrast	30%

Table 3: NSGA-II concerns configuration summary.

Parameter	Value(s)
Population Size	250

The next section presents and discusses the results obtained when applying the configurations on the chosen website panel.

6. EXPERIMENT RESULTS

The presented results extend previous results obtained for NSGA-II [14]. We apply NSGA-II to more websites, with a version of the algorithm which fixes some limitations of the color distance function. Besides, we compare NSGA-III results on the same websites, in order to evaluate the assumption of superiority of NSGA-III. These websites have been chosen by one of the co-authors who has low vision. Other widely visited websites that we observed have no serious problems and were not really interesting to be included in our experiment.

6.1 Godaddy with NSGA-II

We first analyze into details the Godaddy website adaptation, which is an intermediary problem in terms of number of HTML elements used, structural parts and number of colors.

Table 5 gives execution times, number of generations and information about non terminated executions for all non-aggregated preference sets with NSGA-II.

The preference sets S_1 and S_3 have at least one good solution returned by NSGA-II.

Nearly all executions lead to a good solution with S_1. Only 4% of executions cannot find a good solution and reach 10s. Finding new colors for having the required minimal contrast is not too difficult in this case. There are 22 color variables: 14 text colors and 8 background colors. Several text color variables can be linked by a contrast constraint to a same background color variable, when these texts share the same background. S_1 contains 14 constraints. Because we are not working with aggregated objective functions, we also have 14 objective functions.

Table 4: NSGA-III concerns configuration summary.

Parameter	Value(s)
Minimum population size	256
Divisions number	from 2 to p

Table 5: NSGA-II "Godaddy" with **non** aggregated preference sets

Prefs. set	S_1	S_3	S_2	S_4
Obj. Funcs.	14	15	36	37
Exec. time (s)				
Average	5.87	9.89	10.00	10.00
Standard deviation	1.64	0.53	0.00	0.00
Min	3.67	6.64	10.00	10.00
Max	10.00	10.00	10.00	10.00
Generations				
Average	244	376	200	193
Standard deviation	64.8	27.6	7.9	6.4
Min	154	264	192	183
Max	397	415	228	224
Non cpltd. exec.				
Percentage	4	94	100	100
Satisfied prefs.	9.5	10.4	7.2	7.9
Standard deviation	0.5	3.8	1.3	1.4
Min	9	3	4	5
Max	10	14	10	12

The second preference set S_3 is a little more complex. In addition to the constraints of S_1, we add a global constraint linking all background color brightnesses. This constraint is satisfied when all backgrounds have a similar brightness, or in other words, when the difference between the lightest and the darkest background is low. This new constraint complicates the problem. As a result, only 6% of executions give a good solution before 10s. When executions reach 10s, the number of satisfied objective functions is a good quality indicator. Here, on average, about 68% of objective functions are satisfied (9.5 out of 14), for S_1, and about 69% (10.4 out of 15) for S_3.

The two last columns of Table 5 refer to S_2 and S_4 preference sets. These two sets are much more complex and do not give good solutions in the given time. S_2 and S_4 have both all contrast constraints, such as S_1 and S_3, but S_2 integrates the constraint of proximity with original colors, and finally S_4 has all constraints. Results are slightly different. There are 20% of satisfied objective functions for S_2 versus about 21.4% for S_4. In best cases, solutions satisfy about 30% of objective functions. Worst cases are about 12%. For NSGA-II, all executions have been made with a fixed population of 250 individuals.

The aggregation of objective functions allows us to address the known problem of NSGA-II, that drastically looses efficiency when the objective function number grows. Table 6 gives results for aggregated objective functions. Aggregation is realized by creating two (aggregated) objective functions for each general preference GP_2 and GP_3. The global preference GP_1 regarding the uniformity of background color brightness cannot be aggregated because it only generates one preference linking all background color variables.

We observe that this aggregation has a positive impact on the execution time and on finding a good solution. Whereas without aggregation, for the first preference set S_1, 4% of executions could not terminated before 10s, now, with the aggregated preference set S_{1aggr} (first column of Table 6) all executions return a good solution in 0.4s on average. There is also a significant improvement for

Table 6: NSGA-II "Godaddy" with aggregated preference sets

Prefs. set	S_{1aggr}	S_{3aggr}	S_{2aggr}	S_{4aggr}
Obj. Funcs.	2	3	4	5
Exec. time (s)				
Average	0.41	0.46	10.00	10.00
Standard deviation	0.05	0.10	0.00	0.00
Min	0.32	0.32	10.00	10.00
Max	0.49	0.84	10.00	10.00
Generations				
Average	21	22	414	386
Standard deviation	2.5	5.1	2.4	1.9
Min	17	16	409	383
Max	26	41	418	390
Non cpltd. exec.				
Percentage	0	0	100	100
Satisfied prefs.	–	–	0.6	1.3
Standard deviation	–	–	0.8	0.4
Min	–	–	0	1
Max	–	–	2	2

S_{3aggr} preference set, with all executions returning a good solution, while for S_3, we had 94% of non-completed executions. The average execution time is about 0.46s with aggregation, when it is about 10s without. Executing NSGA-II on Godaddy website with S_{2aggr} and S_{4aggr} does not produce any good solution before10 seconds (as in the non-aggregated case). For these two last cases, there is no significant improvement, but we can notice that there is no particular performance loss, results are similar. Aggregation somewhat simplifies the problem by reducing the number of objective functions to be processed by the algorithm. In NSGA-II the number of objective functions is a main complexity component. We observe a significant improvement for the simplest configurations.

6.2 Godaddy with NSGA-III

The third version of NSGA is presented as a "many-objective" evolutionary algorithm, able to deal with much more objective functions while keeping good efficiency. We apply the same set of preferences on the same problem (without any aggregation), now with NSGA-III (Table 7). In this case, population is sized regarding the number of objective functions and the number of divisions (Sect. 4).

Table 7: NSGA-III "Godaddy" with **non** aggregated preference sets

Prefs. set	S_1	S_3	S_2	S_4
Pop. size	256	256	668	704
Obj. Funcs.	14	15	36	37
Exec. time (s)				
Average	2.71	3.53	10.00	10.00
Standard deviation	0.48	0.70	0.00	0.00
Min	1.76	2.43	10.00	10.00
Max	3.63	5.78	10.00	10.00
Generations				
Average	56	65	12	10
Standard deviation	9.8	12.7	0.0	0.2
Min	37	45	12	10
Max	75	106	12	11
Non cpltd. exec.				
Percentage	0	0	100	100
Satisfied prefs.	–	–	15.5	15.3
Standard deviation	–	–	1.2	1.2
Min	–	–	13	13
Max	–	–	18	18

Table 8: NSGA-III "Godaddy" with aggregated preference sets

Prefs. set	S_{1aggr}	S_{3aggr}	S_{2aggr}	S_{4aggr}
Pop. size	256	256	256	256
Obj. Funcs.	2	3	4	5
Exec. time (s)				
Average	0.38	0.40	9.11	10.00
Standard deviation	0.04	0.04	2.68	0.00
Min	0.27	0.31	0.94	10.00
Max	0.46	0.48	10.00	10.00
Generations				
Average	21	21	382	392
Standard deviation	2.3	1.9	111.6	3.4
Min	16	17	43	385
Max	26	25	432	400
Non cpltd. exec.				
Percentage	0	0	90	100
Satisfied prefs.	–	–	2.5	3.0
Standard deviation	–	–	0.6	0.7
Min	–	–	1	2
Max	–	–	3	4

The NSGA-III algorithm brings an overall improvement. In the two first cases (S_1 and S_3), the execution time is really reduced. S_1 execution time is reduced by twofold and S_3 execution time by threefold. Moreover, in both cases all executions give a good solution. For the last two cases, even if executions do not return good solutions, the quality of solutions returned when the algorithm reaches 10s is really improved. With NSGA-III, we have about 43% of satisfied objective functions in S_2 and S_4 versus about 21% with NSGA-II. The maximal and the minimal number of satisfied objective functions are also increased.

Table 8 gives results of the application of NSGA-III on the four preference sets with objective function aggregation. We saw in Table 6 that aggregation of objective functions had a positive impact on the computation time and on the quality of solutions returned in case of non-completed executions. With NSGA-III, aggregation also has a positive impact. While execution times are reduced, the quality of solutions returned when the algorithm reaches 10s (S_2 or S_4) is improved. This quality is about 60%. Thus we observe that, even if NSGA-III was designed to deal with many objectives, the aggregation may be useful in our context.

We also notice that NSGA-III takes more time to complete a generation. But results show that it often needs less generations to converge to a good solution than NSGA-II. These results give us an intuition about trends regarding the two algorithm versions. Next section compares the two algorithms on several very different websites.

6.3 A wider comparison

Tables 9 and 10 show more results on execution of NSGA-II and NSGA-III for all preference sets. These tables allow us to compare results on 9 real web pages. The first table gives figures for executions without aggregation, while the second one gives figures with objective function aggregation.

When each preference is implemented as one objective function (no aggregation), the first preference set S_1 gives good results on a large part of the studied websites (Table 9). The two last websites have many HTML elements. The computation of a minimal brightness contrast for the text regarding its background is not too difficult for several web pages. NSGA-III gives better results for this preference set. In particular, we have a real improvement on the two last cases: The UBOFormation example returns at least one good solution for all executions and we can get some good

Table 9: (Time / non-completed executions / satisfaction for non-completed executions) Global comparison on several real websites

Pref. set	S_1		S_3		S_2		S_4	
Algorithm	NSGA-II	NSGA-III	NSGA-II	NSGA-III	NSGA-II	NSGA-III	NSGA-II	NSGA-III
units / Websites	s / % / %	s / % / %	s / % / %	s / % / %	s / % / %	s / % / %	s / % / %	s / % / %
Godaddy	5.87 / 4 / 67.9	2.71 / 0 / –	9.89 / 94 / 69.3	3.53 / 0 / –	10 / 100 / 20	10 / 100 / 43.1	10 / 100 / 21.4	10 / 100 / 41.4
Parempuyre	0.69 / 0 / –	0.29 / 0 / –	0.78 / 0 / –	0.41 / 0 / –	10 / 100 / 31.4	9.95 / 97 / 82.9	10 / 100 / 33.2	10 / 100 / 79.5
Legibase	0.34 / 0 / –	0.12 / 0 / –	0.71 / 4 / 83.3	0.18 / 0 / –	10 / 100 / 48.5	5.80 / 20 / 90.8	10 / 100 / 40.7	7.16 / 44 / 87.1
NFB	0.39 / 0 / –	0.15 / 0 / –	0.46 / 0 / –	0.21 / 0 / –	10 / 100 / 58.6	3.81 / 4 / 92.9	10 / 100 / 56	6.49 / 34 / 92.7
BLFamilyPortal	3.77 / 0 / –	2.53 / 0 / –	9.92 / 97 / 92.3	5.67 / 17 / 92.3	10 / 100 / 21.7	10 / 100 / 43.1	10 / 100 / 21.6	10 / 100 / 41.9
BLMyAccount	2.24 / 0 / –	0.94 / 0 / –	9.40 / 90 / 90	1.74 / 0 / –	10 / 100 / 22.3	10 / 100 / 61.9	10 / 100 / 23.3	10 / 100 / 58.5
GDArchitect	0.70 / 0 / –	0.25 / 0 / –	0.72 / 0 / –	0.30 / 0 / –	10 / 100 / 35	9.20 / 80 / 88.3	10 / 100 / 34.7	9.47 / 87 / 85.8
UBOLEA	10 / 100 / 32.4	9.95 / 87 / 86.7	10 / 100 / 16.4	10 / 100 / 75.5	10 / 100 / 17.7	10 / 100 / 34.8	10 / 100 / 18.2	10 / 100 / 34.1
UBOFormation	8.97 / 54 / 61.8	5.88 / 0 / –	10 / 100 / 34.4	7.33 / 17 / 94.4	10 / 100 / 19.3	10 / 100 / 37.8	10 / 100 / 19.5	10 / 100 / 37.9

Table 10: (Time / non-completed executions / satisfaction for non-completed executions) Global comparison on several real websites with objective functions aggregation

Pref. set	S_{1aggr}		S_{3aggr}		S_{2aggr}		S_{4aggr}	
Algorithm	NSGA-II	NSGA-III	NSGA-II	NSGA-III	NSGA-II	NSGA-III	NSGA-II	NSGA-III
units / Websites	s / % / %	s / % / %	s / % / %	s / % / %	s / % / %	s / % / %	s / % / %	s / % / %
Godaddy	0.41 / 0 / –	0.38 / 0 / –	0.46 / 0 / –	0.40 / 0 / –	10 / 100 / 15	9.11 / 90 / 62.5	10 / 100 / 26	10 / 100 / 60
Parempuyre	0.24 / 0 / –	0.18 / 0 / –	0.24 / 0 / –	0.19 / 0 / –	7.52 / 74 / 47.5	1.97 / 14 / 75	8.16 / 80 / 60	4.34 / 37 / 80
Legibase	0.27 / 0 / –	0.10 / 0 / –	0.21 / 0 / –	0.13 / 0 / –	7.17 / 70 / 52.5	0.96 / 4 / 75	8.13 / 80 / 54	1.89 / 4 / 80
NFB	0.22 / 0 / –	0.13 / 0 / –	0.24 / 0 / –	0.17 / 0 / –	4.25 / 37 / 55	0.58 / 0 / –	7.23 / 70 / 54	2.41 / 10 / 80
BLFamilyPortal	0.48 / 0 / –	0.44 / 0 / –	0.83 / 4 / 66.7	0.52 / 0 / –	10 / 100 / 30	8.19 / 80 / 65	10 / 100 / 22	10 / 100 / 60
BLMyAccount	0.52 / 0 / –	0.32 / 0 / –	0.90 / 4 / 66.7	0.38 / 0 / –	8.50 / 84 / 47.5	4.83 / 44 / 72.5	10 / 100 / 34	9.72 / 97 / 72
GDArchitect	0.22 / 0 / –	0.15 / 0 / –	0.31 / 0 / –	0.17 / 0 / –	6.87 / 67 / 47.5	1.36 / 7 / 75	6.59 / 64 / 64	3.00 / 20 / 80
UBOLEA	2.78 / 24 / 45	1.82 / 14 / 50	10 / 100 / 20	10 / 100 / 60	10 / 100 / 20	9.70 / 97 / 55	10 / 100 / 26	10 / 100 / 46
UBOFormation	0.47 / 0 / –	0.45 / 0 / –	1.46 / 10 / 66.7	0.45 / 0 / –	10 / 100 / 22.5	9.40 / 94 / 60	10 / 100 / 40	9.41 / 94 / 68

solutions for the UBOLEA example. Results with the second preference set S_3 confirm these trends. The preference sets S_2 and S_4 give another important information. In addition to finding some good solutions when it is not the case with NSGA-II, NSGA-III highly increases the level of satisfaction of returned approximate solutions. Indeed, when an execution does not return a good solution, either because it does not find one existing good solution, or because such solution does not exist (conflict between preferences), the solution quality has a huge importance. When we have 34.7% of satisfaction (for non-completed executions) on GDArchitect example with NSGA-II and the S_4 preference set, we reach 85.8% with NSGA-III. With the same configuration on Parempuyre example, we have now about 80% with NSGA-III, when it is about 33% with NSGA-II.

We have seen that the objective function aggregation also has a positive impact with NSGA-III on the Godaddy example. Table 10 shows this behavior for other websites. The UBOLEA example website, with S_3 preference set, shows a significant improvement in the returned solution quality, even if both NSGA-II and NSGA-III do not return any good solution. The S_4 preference set case, actually shows a significant improvement in the number of executions leading to a good solution, in execution time and in the quality of solutions.

To summarize, NSGA-III is a real enhancement of its prior version NSGA-II. NSGA-III allows us to obtain good solutions for more difficult problems. Moreover, the quality of the returned solution, when the solution does not satisfy all preferences, is really better.

6.4 Threats to validity

6.4.1 Construct validity

Experiment has been made with several distinct configurations for the algorithms. Two algorithms have been used with common and specific parameters. The preference sets and the configurations have been chosen for analyzing benefit of the new version of NSGA and the effect of aggregation on it. Aggregation that was positive for NSGA-II as it is shown in our previous paper [14] keeps similar impact for NSGA-III. The experiment allows us to gather many information including execution time, number of generations and number of satisfied objective functions. From these figures, we compute many other indicators, to get for example the number of executions reaching 10 seconds. Experiment results allow us to know if the approach can be used in a real case study. When an execution returns at least one good solution in less than 10 seconds, we focus on the elapsed time and on the number of generations regarding this time. When an execution does not return any good solution in less than 10 seconds, we give information about the number of satisfied objective functions that give a good estimate of the quality of the returned solution. Here the population size for NSGA-II has the same value as in our previous paper. However, this value could be changed to measure its effects. In the two algorithms, we use a same and fixed crossover and mutation probability. They could be changed to evaluate the impact of this probability.

6.4.2 Internal validity

Information on the studied websites has been collected by three persons. Colors of HTML elements have been obtained using a software picking screen pixel color: "gcolor2" on Linux system and the digital colorimeter included in the MAC OS platform. We simplify web pages by defining a new model of variables and constraints used by resolution algorithms.

6.4.3 External validity

The experiment is based on a selection of 9 very different websites. These websites have been chosen from noticed accessibility problems for people with low vision and also regarding their diversity in terms of number of HTML elements or number of colors used. We focus our study on brightness, contrast and colors because they represent widely encountered difficulties for people with low

vision. Moreover they are issues on which assistive technologies do not bring relevant adaptation. The diversity of input data (website, preferences, resolution algorithms used and their parameters) is significant.

6.4.4 *Conclusion validity*

For each configuration (website, algorithm, preference set, etc.), we made 30 executions. We also tested our algorithms with 60 executions, and we obtained similar results. In our results, we considered many measures including average, min, max and standard deviation for execution time, number of generations and number of satisfied objective functions. Previous papers [14, 15] give other comparisons, for example with an exact algorithm from Preference Theory.

7. CONCLUSION

Assistive technologies may provide a relevant help for people with low vision. However, their general purpose can lead to irrelevant adaptation for people with specific needs. The main drawback of these solutions is the important alteration of the page appearance, which corresponds to the designer preferences. Indeed, colors may be completely changed, page layout can be modified and it is quite impossible to target a structural part of a page like the menu. Our approach works at HTML element level, and with properties that describe the elements. Moreover, it computes an adaptation for specific user preferences, while taking into account the designer preferences. These preferences may be inconsistent. We compare the efficiency of two evolutionary algorithms to find a relevant adaptation in the huge search space of possible adaptations. The comparison is done on real websites that are very different. We obtain results that show general trends. We also study the behavior of aggregating objective functions. The aggregation allows us to reduce running time, and to strongly increase the number of executions that return good solutions. NSGA-III, used with objective function aggregation, gives the best results. As future work, we plan to investigate and compare results obtained with a CSP (Constraint Satisfaction Problem) approach and with a deep learning approach. Currently, gathering information from web pages is manually done. The structural object recognition in HTML pages is tricky and we are working on an approach to automatize this acquisition. We are also planning experiments conducted with a panel of people with low vision to reinforce our findings.

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 experiment.

8. REFERENCES

[1] Accessibility in Android. `https://developer.android.com/design/patterns/accessibility.html`. Accessed: 2015-01-10.

[2] Accessibility in Windows 8. `http://www.microsoft.com/enable/products/windows8/`. Accessed: 2015-01-10.

[3] Apple Accessibility OS X. `https://www.apple.com/accessibility/osx/`. Accessed: 2015-01-10.

[4] Chrome accessibility support. `http://www.chromium.org/user-experience/low-vision-support`. Accessed: 2015-01-10.

[5] Cloud4all. `http://www.cloud4all.info/`. Accessed: 2015-03-25.

[6] Freedom Scientific tools. `http://www.freedomscientific.com/`. Accessed: 2015-01-10.

[7] Gnome screen magnifier gnome-mag. `https://wiki.ubuntu.com/Accessibility/Reviews/gnome-mag`. Accessed: 2015-01-10.

[8] KDE screen magnifier Kmag. `http://kmag.sourceforge.net/`. Accessed: 2015-01-10.

[9] Mozilla accessibility support. `http://www.chromium.org/user-experience/low-vision-support`. Accessed: 2015-01-10.

[10] VoiceOver. `http://www.apple.com/accessibility/osx/voiceover/`. Accessed: 2015-01-10.

[11] Window-Eyes. `http://www.gwmicro.com/`. Accessed: 2015-01-10.

[12] J. P. Bigham, R. S. Kaminsky, R. E. Ladner, O. M. Danielsson, and G. L. Hempton. Webinsight:: Making web images accessible. In *Proceedings of the 8th International ACM SIGACCESS Conference on Computers and Accessibility*, Assets '06, pages 181–188, New York, NY, USA, 2006. ACM.

[13] J. P. Bigham and R. E. Ladner. Accessmonkey: a collaborative scripting framework for web users and developers. In *Proceedings of the 2007 international cross-disciplinary conference on Web accessibility (W4A)*, pages 25–34. ACM, 2007.

[14] Y. Bonavero, M. Huchard, and M. Meynard. Improving Web Accessibility: Computing New Web Page Design with NSGA-II for People with Low Vision. *International Journal on Advances in Internet Technology, issn 1942-2652*, 7(3-4):243–261, 2014.

[15] 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)*, pages 40–45, 2014.

[16] D. Cai, S. Yu, J.-R. Wen, and W.-Y. Ma. Extracting content structure for web pages based on visual representation. In *Web Technologies and Applications*, pages 406–417. Springer, 2003.

[17] J. Chen, B. Zhou, J. Shi, H. Zhang, and Q. Fengwu. Function-based object model towards website adaptation. In *Proceedings of the 10th international conference on World Wide Web*, pages 587–596. ACM, 2001.

[18] K. Deb, S. Agrawal, A. Pratap, and T. Meyarivan. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Trans. Evolutionary Computation*, 6(2):182–197, 2002.

[19] K. Deb and H. Jain. An evolutionary many-objective optimization algorithm using reference-point-based nondominated sorting approach, part I: solving problems with box constraints. *IEEE Trans. Evolutionary Computation*, 18(4):577–601, 2014.

[20] S. Ferretti, S. Mirri, C. Prandi, and P. Salomoni. Exploiting reinforcement learning to profile users and personalize web pages. In *IEEE 38th Annual Computer Software and Applications Conference, COMPSAC Workshops 2014, Vasteras, Sweden, July 21-25, 2014*, pages 252–257. IEEE, 2014.

[21] S. Ferretti, S. Mirri, C. Prandi, and P. Salomoni. User centered and context dependent personalization through experiential transcoding. In *Proc. IEEE Consumer Communications and Networking (CCNC 2014), Workshop on Networking Issues in Multimedia Entertainment (NIME'14)*, 2014.

[22] A. Foti and G. Santucci. Increasing web accessibility through an assisted color specification interface for colorblind people. pages 41–48, 2009.

[23] M. Harman, S. A. Mansouri, and Y. Zhang. Search-based software engineering: trends, techniques and applications. *ACM Comput. Surv.*, 45(1):11:1–11:61, Dec. 2012.

[24] 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*, pages 82–87. IEEE, 1994.

[25] M. A. Islam, Y. Borodin, and I. V. Ramakrishnan. Mixture model based label association techniques for web accessibility. In K. Perlin, M. Czerwinski, and R. Miller, editors, *UIST*, pages 67–76. ACM, 2010.

[26] H. Jain and K. Deb. An evolutionary many-objective optimization algorithm using reference-point based nondominated sorting approach, part II: handling constraints and extending to an adaptive approach. *IEEE Trans. Evolutionary Computation*, 18(4):602–622, 2014.

[27] S. Kaci. *Working with Preferences: Less Is More*. Cognitive Technologies. Springer, 2011. ISBN:978-3-642-17279-3.

[28] 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*, volume 1, pages 98–105. IEEE, 1999.

[29] M. Laumanns, L. Thiele, K. Deb, and E. Zitzler. Combining convergence and diversity in evolutionary multiobjective optimization. *Evolutionary computation*, 10(3):263–282, 2002.

[30] D. Lunn, S. Bechhofer, and S. Harper. A user evaluation of the sadie transcoder. In S. Harper and A. Barreto, editors, *Proceedings of the 10th International ACM SIGACCESS Conference on Computers and Accessibility, ASSETS 2008, Halifax, Nova Scotia, Canada, October 13-15, 2008*, pages 137–144. ACM, 2008.

[31] D. Lunn, S. Harper, and S. Bechhofer. Combining sadie and axsjax to improve the accessibility of web content. In D. Sloan, C. Asakawa, and H. Takagi, editors, *Proceedings of the International Cross-Disciplinary Conference on Web Accessibility, W4A 2009, Madrid, Spain, April 20-21, 2009*, ACM International Conference Proceeding Series, pages 75–78. ACM, 2009.

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

[33] S. Mirri, C. Prandi, and P. Salomoni. Experiential adaptation to provide user-centered web content personalization. In *Proc. IARIA Conference on Advances in Human oriented and Personalized Mechanisms, Technologies, and Services (CENTRIC2013)*, pages 31–36, 2013.

[34] S. Mirri, P. Salomoni, C. Prandi, and L. A. Muratori. Gapforape: an augmented browsing system to improve web 2.0 accessibility. *New Review of Hypermedia and Multimedia*, 18(3):205–229, 2012.

[35] 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*, pages 72–79. ACM Press, 2004.

[36] G. Santucci. Vis-a-wis: Improving visual accessibility through automatic web content adaptation. In C. Stephanidis, editor, *Universal Access in Human-Computer Interaction. Applications and Services, 5th International Conference, UAHCI 2009, Held as Part of HCI International 2009, San Diego, CA, USA, July 19-24, 2009. Proceedings, Part III*, volume 5616 of *Lecture Notes in Computer Science*, pages 787–796. Springer, 2009.

[37] B. Tibbitts, S. Crayne, V. Hanson, J. Brezin, C. Swart, and J. Richards. HTML parsing in Java for accessibility transformation. In *Proceedings of XML 2002 – XML Conference and Exposition*, 2002.

[38] L. Troiano, C. Birtolo, and M. Miranda. Adapting palettes to color vision deficiencies by genetic algorithm. In *Proceedings of the 10th Annual Conference on Genetic and Evolutionary Computation*, GECCO '08, pages 1065–1072, New York, NY, USA, 2008. ACM.

[39] 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.

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

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

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

[43] World Wide Web Consortium, `http://www.w3.org/TR/WCAG20/`. *Web Content Accessibility Guidelines*. Accessed: 2014-11-09.

[44] E. Zitzler and L. Thiele. Multiobjective evolutionary algorithms: a comparative case study and the strength Pareto approach. *IEEE Trans. Evolutionary Computation*, 3(4):257–271, 1999.