Algorithmes evolutionnaires pour l'adaptation personnalisée de pages Web

Yoann Bonavero, Marianne Huchard, Michel Meynard

Algorithmes évolutionnaires pour l'adaptation personnalisée de pages Web

Yoann Bonavero, Marianne Huchard, Michel Meynard
LIRMM, CNRS et Université de Montpellier
161 rue Ada - 34095 Montpellier Cedex 5 - France
Email : prénom.nom@lirmm.fr

***Résumé*—Le Web a permis le développement de nombreux services, rendant l'accessibilité des pages Web cruciale pour tous. Cependant de nombreuses pages Web restent difficiles d'accès pour les personnes en situation de déficience visuelle (moyenne et basse vision). Les solutions d'adaptation proposées par les navigateurs et les systèmes d'exploitation ne permettent pas de traiter finement de situations individuelles. Dans cet article, nous proposons une approche générale permettant d'adapter des pages Web en fonction de déficiences particulières (impliquant des besoins spécifiques à une personne) tout en tâchant de respecter autant que possible le design de la page initiale. L'approche cherche ainsi à offrir à l'utilisateur final d'un contenu Web la possibilité de profiter de toutes ses capacités visuelles lors de l'accès aux pages. Elle est testée dans une étude de cas dont les résultats sont encourageants.**

I. Introduction

Les technologies de l'information et de la communication, et en particulier le Web, ont permis le développement d'outils utilisés au quotidien pour de nombreux usages tels que l'accès à l'information, les services de commerce, les services administratifs, ou la vie sociale (forums, réseaux sociaux). Ces outils sont une chance pour réussir une société inclusive, apportant des solutions d'intégration pour les personnes selon leurs différences. Cependant, s'ils sont mal conçus ou mal employés, ils peuvent au contraire devenir une source d'exclusion supplémentaire.

Dans cet article, nous nous concentrons sur l'accessibilité des pages Web par les personnes atteintes de certaines déficiences visuelles (moyenne et basse vision). Nous désirons offrir à l'utilisateur final d'un contenu Web la possibilité de profiter de toutes ses capacités visuelles lors de l'accès aux pages. Nous tentons de trouver un compromis entre les besoins de cet utilisateur et le message transmis par le design initial. L'utilisateur peut exprimer des préférences plus complexes que la simple expression d'une couleur ou d'une taille de police. Par exemple, il peut demander un contraste minimum entre la couleur d'un texte et son arrière-plan (préférence utilisateur) et nous cherchons une adaptation qui propose des couleurs restant proches des couleurs initiales (préférence du concepteur).

Sous l'hypothèse que nous disposons des préférences, nous proposons une méthode d'adaptation qui calcule une solution approchée au problème dans un temps raisonnable sur des pages de complexité moyenne. Le problème est rendu complexe par la présence de préférences potentiellement opposées (notamment entre le besoin de l'utilisateur et le design initial) et le nombre d'éléments à adapter. Le calcul d'une solution, qui peut être une solution approchée, est effectué grâce à une méta-heuristique multi-objectifs (algorithme évolutionnaire) qui est étudiée dans deux de ses versions (NSGA-II et NSGA-III). Une étude de cas est menée sur dix pages Web variées pouvant présenter des problèmes pour des personnes déficientes visuelles. L'article approfondit l'étude présentée dans [1] avec les analyses ayant mené au choix des paramètres utilisés dans les algorithmes évolutionnaires et un nouvel éclairage sur les résultats des algorithmes d'adaptation.

Dans la section II nous présentons un aperçu général de l'approche. Puis nous détaillons la première étape de sélection des données et d'expression des contraintes d'adaptation provenant des préférences (section III). La section IV décrit le principe de fonctionnement des algorithmes évolutionnaires utilisés. Le choix des paramètres et les résultats obtenus sont présentés et analysés dans la section V. Enfin nous introduisons les principaux travaux connexes (section VI), puis nous concluons (section VII).

II. Aperçu du processus d'adaptation

Les figures 1 et 2 illustrent la méthode proposée sur deux éléments d'une page réelle [1]. Un extrait de la page du site Godaddy (https://uk.godaddy.com/), avec l'onglet `All Products` déplié, est présentée dans la figure 1. Sur le fond vert du menu, les textes blanc ne ressortent pas avec un contraste suffisant pour une lecture confortable en présence de certaines déficiences visuelles. Une personne ayant ces difficultés pourra exprimer qu'elle a besoin d'un contraste minimum entre les textes et les fonds. Par ailleurs, on cherchera à disposer, après l'adaptation, de nouvelles couleurs qui se rapprochent des couleurs d'origine, choisies par le concepteur de la page. Un exemple d'adaptation calculée par notre approche est présenté par la figure 2. La solution calculée consiste à éclaircir le vert du fond du menu et à écrire le texte en caractères marrons. Elle correspond à un compromis réalisé entre les différentes contraintes exprimées. Ici nous n'avons pas traité les tout petits caractères en jaune, qui restent peu lisibles dans les deux figures.

1. Une version couleur de l'article est disponible à l'adresse http://www.lirmm.fr/~huchard/Documents/Papiers/NsgaHandicap2016.pdf.

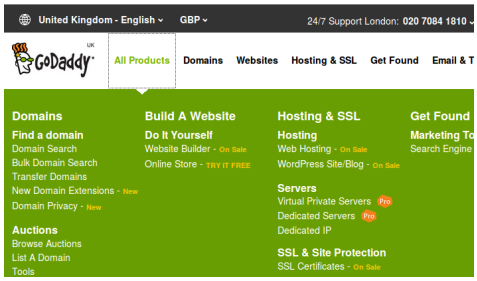

FIGURE 1. Extrait de la page Godaddy d'origine

FIGURE 2. Extrait de la page Godaddy transformée

De manière plus générale, et parce que nous recherchons un processus générique, les trois composantes principales de l'approche sont :

— La détermination de *variables*, associant un élément d'une page à une de ses caractéristiques. Par exemple, la couleur d'un texte, la couleur d'un fond, la taille de police d'un texte, etc.

— L'énoncé d'objectifs de personnalisation, qui se traduisent par des fonctions (*fonctions objectif*) calculant la manière dont un ensemble d'associations (variable, valeur) satisfait l'objectif de personnalisation. Par exemple, une fonction objectif consiste à calculer à quel point un objectif général de contraste minimum de 30% entre tout texte et son fond est satisfait.

— L'utilisation d'un *algorithme d'adaptation générique*, indépendant du type de variable et de la nature des objectifs de personnalisation.

Nous avons choisi d'utiliser des algorithmes évolutionnaires qui travaillent sur l'ensemble de variables décrivant une page Web. Un *individu* (encore appelé *solution*) est une affectation de valeurs à toutes les variables. Les algorithmes évolutionnaires évaluent un individu à l'aide des fonctions objectif et font évoluer une population d'individus par le biais d'opérateurs de croisement, de mutation et de sélection, jusqu'à trouver un ou plusieurs individus satisfaisant les objectifs de personnalisation. La méthode proposée peut donc s'appliquer à tout ensemble de variables et tout ensemble de fonctions objectif pertinentes sur ces variables.

III. SÉLECTION DES VARIABLES ET OBJECTIFS DE PERSONNALISATION

Nous reprenons l'exemple du site Godaddy pour illustrer le processus de sélection de variables et la manière dont des objectifs de personnalisation généraux sont instanciés sur une page Web réelle.

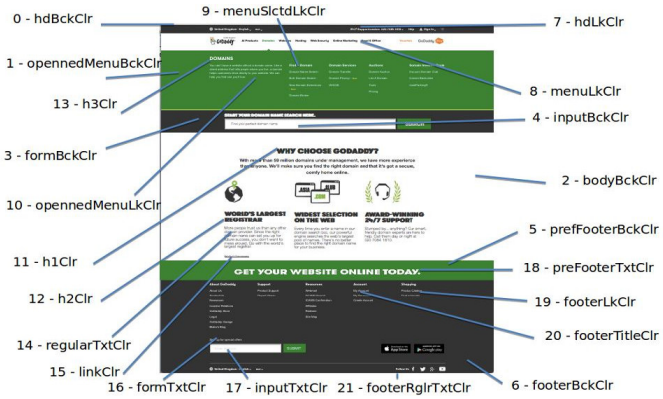

FIGURE 3. Sélection des éléments sur la page Godaddy

a) Sélection des variables: La figure 3 montre le travail à réaliser sur une page pour déterminer les variables. Les éléments perçus par l'utilisateur et représentant un intérêt pour l'adaptation sont étiquetés par un numéro et un identificateur. Par exemple, la variable `1-openedMenuBckClr` représente la couleur de fond (*background*) du menu ouvert (dans la représentation sRGB) ; la variable `10-openedMenuLkClr` représente la couleur du texte des liens (*link*) du menu ouvert. Cette tâche peut être effectuée automatiquement pour des cas simples, mais dans le cas général, dès que l'on veut des variables d'un plus haut niveau d'abstraction, se pose la question de déterminer ces objets dans le code source HTML ou CSS. Par exemple si on veut une variable exprimant la couleur d'un *menu*, ce type d'élément n'existe pas en tant que tel dans le code source HTML 4 (le menu peut être représenté par une liste à puces ou une suite de liens), et il est donc moins aisé de l'extraire automatiquement et de lui associer une couleur ou une taille de police. Suivant la qualité du code source, il se peut qu'il soit également difficile de déterminer la valeur d'origine si dans la page plusieurs tailles de polices ont été mélangées dans les textes de ce menu. D'autres questions se posent, comme de savoir par exemple, si on doit distinguer des paragraphes qui se suivent dans l'adaptation, ou bien s'il s'agit d'un même type d'élément qui doit être traité de manière uniforme lors de l'adaptation. Cette tâche cruciale pour le bon déroulé de l'approche est donc délicate à réaliser.

b) Objectifs de personnalisation et leur instanciation sur une page: Les objectifs de personnalisation comprennent des besoins de l'utilisateur en situation de déficience visuelle et les aspects du design initial de la page que l'on veut préserver. Dans les études de cas que nous avons menées, nous avons défini trois objectifs généraux : (P_1) la différence des brillances de couleurs des arrière-plans doit rester inférieure à une certaine valeur (40%), (P_2) le contraste de brillance entre un texte et son arrière-plan doit être supérieur à une certaine valeur (30%), (P_3) les couleurs d'origine doivent être préservées (la nouvelle couleur doit former un angle de moins de 40° avec la couleur d'origine). Pour calculer ces objectifs

sur un ensemble de variables (données avec une valeur), nous utilisons les définitions du *Web Content Accessibility Guidelines 2.0 [2]*. Ci-dessous sont présentés quelques exemples de ces objectifs appliqués à la page Godaddy (les variables sont représentées seulement par leur numéro). Le choix des seuils de référence s'est fait en se basant sur l'expérience personnelle de déficients visuels.

Exemple d'objectif sur le contraste de brillance

- Contraste entre la couleur de fond du menu ouvert et le texte du menu : `contrast(1, 10)` $\geq$ 30%

Objectif de proximité de brillance d'arrière-plans

- `distBrightness(0,1,2,3,4,5,6)` $\leq$ 40%

Exemple d'objectif sur la distance de couleurs

- `distOrigClr(0, [51, 51, 51])` $\leq 40^{\circ}$

  Avec [51, 51, 51] couleur initiale du fond de l'entête.

La satisfaction de chacun de ces objectifs est mesurée par une fonction objectif dans la suite.

IV. Algorithmes d'adaptation

Pour trouver des solutions d'adaptation, nous nous sommes tout d'abord tournés vers la théorie des préférences [3], mais les algorithmes pour calculer des solutions se sont avérés être des algorithmes exacts (calculant toutes les solutions pour choisir une des meilleures) et n'ont pas pu donner de résultats dans des temps raisonnables sur des ensembles de variables même très petits. Nos ensembles de valeurs possibles, notamment pour représenter les couleurs, sont très grands et l'espace de solutions possibles est par conséquent très important également [4]. Pour limiter le nombre de couleurs tout en gardant assez de solutions, nous considérons 32768 couleurs dans la représentation sRGB. Si on a 10 variables de couleurs, le nombre de solutions est déjà de $32768^{10} \sim 1.4\ 10^{45}$.

Pour résoudre ce problème d'optimisation, et sachant que plusieurs objectifs de personnalisation sont impliqués, nous avons choisi d'utiliser l'algorithme évolutionnaire multi-objectifs NSGA [5], [6], qui s'inspire de la théorie de l'évolution et s'attache à faire évoluer une population d'individus (ou solutions) à l'aide d'opérateurs génétiques (sélection, croisement, mutation). La version NSGA-II est très populaire [7] et la version NSGA-III est apparue récemment pour traiter les cas dans lesquels le nombre de fonctions objectif est élevé.

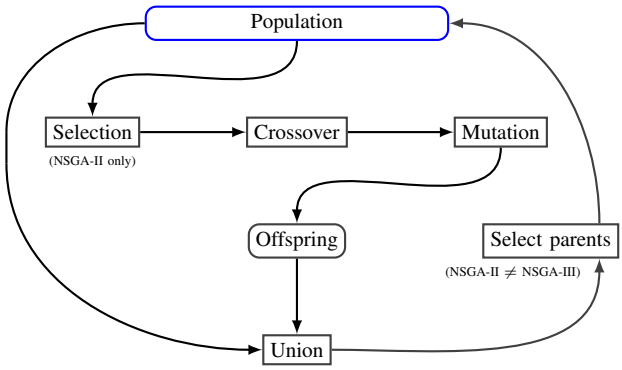

FIGURE 4. Schéma d'une itération des algorithmes NSGA-II et NSGA-III

L'algorithme procède par la construction de populations successives (une génération à chaque itération) jusqu'à un critère d'arrêt (durée de temps écoulée ou obtention d'une solution avec une qualité suffisante). La population initiale est initialisée aléatoirement. Pour une taille de population N, on sélectionne de manière équiprobable N individus dans l'espace de recherche.

La figure 4 décrit une itération. Dans NSGA-II on commence par une étape de sélection (absente dans NSGA-III). Puis les opérateurs génétiques de croisement et de mutation sont appliqués pour engendrer une descendance depuis la population courante. Le croisement consiste à produire un ou deux individus à partir de deux individus parents. Pour chaque variable du (des) nouvel (nouveaux) individu(s), on choisit aléatoirement de prendre la valeur de cette variable dans le premier ou dans le second parent. La mutation consiste à sélectionner aléatoirement certaines variables d'un individu et à leur affecter de nouvelles valeurs, choisies elles aussi aléatoirement. La population courante pop et la descendance sont ensuite unifiées en une seule population $unifiedPop$ (pop est vidée). À ce stade on dispose d'une population de $2N$ individus. Les fonctions objectif sont évaluées sur chaque individu de $unifiedPop$. Les individus sont ensuite triés par fronts de non-domination. Ce tri se déroule comme suit : tous les individus non dominés (ils sont meilleurs que les autres sur au moins un objectif) de la population se voient affectés au rang 1. Les individus qui, en ignorant les individus du rang 1, ne sont plus dominés se voient affectés au rang 2, etc. Les individus de rang 1 font partie du premier front $\mathcal{F}_1$, les individus de rang 2 font partie du second front $\mathcal{F}_2$ et ainsi de suite. On ajoute un par un à pop les fronts qui y tiennent intégralement, du meilleur (plus petit rang) au moins bon, tant que l'on ne dépasse pas N individus. Dès qu'on arrive à un front qui ne tient pas intégralement dans pop, s'il reste de la place, on sélectionne dans ce front une partie des individus. Cette sélection se fait par le choix d'individus isolés. Dans NSGA-II, l'isolement d'un individu est calculé par rapport aux autres individus de ce front. Dans NSGA-III, l'isolement est calculé par rapport aux individus de ce dernier front mais également par rapport aux individus des fronts précédemment insérés. Dès que la nouvelle population pop est complète, on passe à l'itération suivante. Les algorithmes NSGA-II, NSGA-III ainsi que les algorithmes de calcul des fonctions objectif sont détaillés dans [8].

V. Etude de cas

a) Paramètres des algorithmes et leur choix: L'une des difficultés lors de l'utilisation des algorithmes évolutionnaires est le choix des valeurs pour leurs différents paramètres. Pour chaque ensemble de paramètres, les algorithmes sont exécutés 60 fois. L'algorithme est stoppé quoi qu'il arrive à 10 secondes puisque nous recherchons des solutions dans un temps raisonnable. Le nombre de générations n'est pas limité.

Pour NSGA-II et NSGA-III, la taille de la population intervient de manière significative dans la complexité de l'algorithme car cette taille est élevée au carré ($\mathcal{O}(MN^2)$

avec M le nombre de fonctions objectifs et N la taille de la population). Les courbes de la figure 5 montrent le résultat de l'exécution de NSGA-II avec des tailles de population diverses sur un exemple particulier. On peut observer que le temps d'exécution (pour atteindre une bonne solution) et le nombre de générations sont élevés quand la population est de trop petite taille (en-dessous de 20). Au-delà d'une taille de population de 250 individus, le temps d'exécution augmente sans exagérer et le nombre de générations reste relativement stable. Le temps d'exécution est très faible pour les tailles de population de 60 à 140 sur cet exemple. Au-delà de cet exemple, étudier les populations de 100 à 300 (par pas de 50) s'est trouvé être un bon compromis sur l'ensemble des expérimentations pour NSGA-II. Pour NSGA-III, la taille de population minimale choisie est de 256.

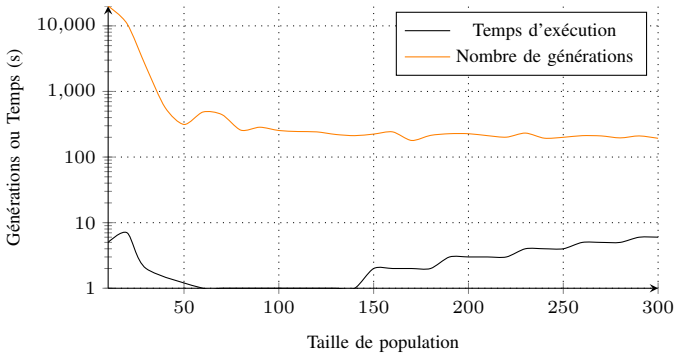

FIGURE 5. Temps d'exécution / Génération en fonction de la taille de population pour NSGA-II (Exemple : Godaddy S_1)

Quatre ensembles d'objectifs combinent les 3 objectifs généraux :

$$S_1 = P_2 \qquad S_3 = P_1 \cup P_2$$
$$S_2 = P_2 \cup P_3 \qquad S_4 = P_1 \cup P_2 \cup P_3$$

Les objectifs de personnalisation peuvent être ou non regroupés. S_{iagr} représentera l'ensemble S_i d'objectifs avec regroupement, les détails en sont expliqués dans [8].

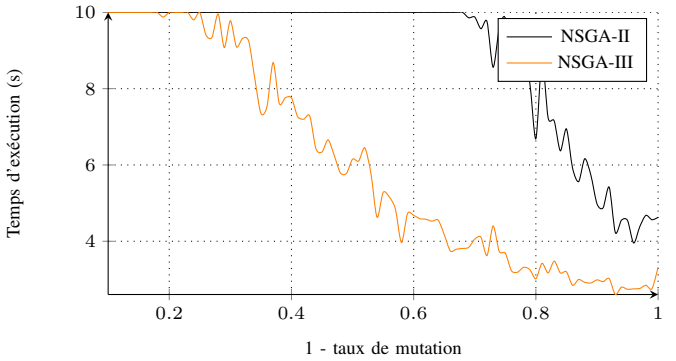

FIGURE 6. Temps d'exécution en fonction de (1 - taux de mutation) (Exemple : Godaddy S_1)

Le taux de croisement et de mutation détermine la quantité d'individus qui vont potentiellement être affectés par une mutation. Avec un taux de 94%, seuls 6% (100% - 94%) des croisements seront suivis d'une mutation. La figure 6 montre l'impact de différentes valeurs du taux de croisement sur le temps nécessaire pour trouver au moins une bonne solution (sur le site Godaddy, avec les objectifs de S_1, pris comme exemple représentatif). Les deux courbes suivent la même tendance, lorsque la part d'aléatoire diminue, le temps mis en moyenne pour trouver au moins une bonne solution diminue aussi. On peut constater un léger rebond lorsque l'on s'approche de la valeur 1 (pas de mutation). Selon le problème étudié, l'approche de la valeur 1 peut se matérialiser comme une stabilisation ou un rebond (voire même une amélioration dans certains cas). Sur la courbe correspondant à NSGA-III, la stabilisation puis la remontée en 1 est assez bien marquée. On observe une différence nette entre les deux courbes, celle qui correspond à NSGA-III entame sa descente plus rapidement que celle qui correspond à NSGA-II. De plus pour toute valeur supérieure à environ 0.3, le temps moyen pour NSGA-III est plus faible que celui de NSGA-II. Une étude sur différentes configurations a d'autre part permis de mettre en évidence qu'une valeur moyenne pour ce taux de 94% permettait d'obtenir le temps le plus faible tout en donnant de bons résultats.

b) Pages Web étudiées: Dix pages Web variées ont été sélectionnées à la fois pour leur diversité en terme de nombre d'éléments, par rapport au nombre de couleurs utilisées mais aussi parce que leur apparence est propice à des problèmes d'accessibilité. Des sites Web de contenu, des sites de formation ou bien des plate-formes de e-commerce ont donc été utilisés. Une partie de ces pages Web (Legibase, BLMyAccount et BLFamilyPortal) ont été étudiées dans le cadre d'un projet avec l'entreprise Berger Levrault qui travaille sur l'accessibilité de ses applications. Entre une dizaine et une soixantaine de variables ont été déterminées suivant les pages.

c) Résultats: Les deux algorithmes ont été implémentés en C++. L'ensemble des résultats a été obtenu sur un ordinateur portable grand public disposant d'un processeur quatre cœurs (2.6Ghz) et muni de 8GB de RAM DDR3.

Pour les cas simples (sans agrégation) avec une dizaine de fonctions objectifs, NSGA-II permet généralement de trouver des solutions optimales dans un temps compris entre 0,3 et 0,7 secondes. Au-delà de ces cas simples, les temps d'exécution de NSGA-II se situent entre quelques secondes et 10s (arrêt du programme). Dans les cas allant d'une dizaine à une soixantaine d'objectifs (toujours sans agrégation), le taux de satisfaction pour NSGA-II varie d'environ 17% à un peu moins de 60%. En utilisant l'agrégation des fonctions objectifs, NSGA-II trouve en moins de 10s une solution optimale sur plus de la moitié des sites Web, même avec les ensembles les plus complexes, et quand les 10s sont atteintes, les taux de satisfaction sont plus élevés. Cela indique que le procédé d'agrégation est bénéfique pour NSGA-II.

Les figures 7 (a) à (d) et les figures 8 (a) et (b) mettent en évidence l'amélioration apportée par NSGA-III. Un rapport de

2 est souvent constaté sur le niveau de satisfaction (figure 7 (a) et (b)) ou bien sur le temps d'exécution (figure 7 (c) et (d)). Nous le montrons respectivement sur les ensembles d'objectifs S_4 et S_{4agr} en ce qui concerne les pourcentages d'objectifs satisfaits et sur les ensembles d'objectifs S_1 et S_{1agr} en ce qui concerne les temps d'exécution. Les figures 8 (a) et (b) montrent que le pourcentage d'exécutions atteignant 10s est significativement diminué pour S_4 lorsque le nombre d'objectifs est inférieur à 20 et pour S_{4agr} lorsqu'il est inférieur à 30. Dans ces graphiques, l'abscisse représente le nombre d'objectifs associés aux différentes pages Web de notre corpus. Dans les graphiques 7 (a) et (b) (haut), l'ordonnée représente le pourcentage d'objectifs satisfaits pour NSGA-III (le plus élevé) divisé par le pourcentage d'objectifs satisfaits pour NSGA-II (le moins élevé). Dans les graphiques 7 (c) et (d) (bas), l'ordonnée représente le temps moyen d'exécution pour NSGA-II (le plus élevé) divisé par le temps moyen d'exécution pour NSGA-III (le moins élevé). Dans les graphiques 8 (a) et (b), l'ordonnée représente le pourcentage d'exécutions atteignant 10s. Le procédé d'agrégation est également bénéfique pour NSGA-III, ce qui est inattendu puisque cet algorithme a été créé pour prendre en charge de manière plus performante de nombreuses fonctions objectif.

VI. Travaux connexes

L'accessibilité numérique est actuellement traitée par divers dispositifs présents dans les systèmes d'exploitation, dans les navigateurs (notamment sous forme d'extensions) et par des outils d'assistance, commerciaux pour la plupart. Ces solutions ont plusieurs inconvénients : elles appliquent des modifications globales, qui ne prennent pas en compte les problèmes spécifiques d'un utilisateur final et qui ne résolvent pas certains problèmes pourtant courants (par ex. un filtre d'inversion de couleur ne peut pas augmenter un contraste trop faible) ; elles peuvent changer complètement le design d'une page Web, l'utilisateur final perdant ainsi l'information véhiculée par ce design. L'organisme W3C (World Wide Web Consortium), qui est à l'origine des langages HTML et CSS a également fait des propositions sur l'accessibilité, comme l'initiative WAI-ARIA (Web Accessibility Initiative - Accessible Rich Internet Application [9]). Cependant, les développeurs de page voient souvent les travaux relatifs à l'accessibilité comme une perte de temps par rapport à la cible principale. De plus, même le respect des guides et des normes ne permet pas une prise en compte fine des différentes déficiences et de leurs conséquences sur la vision.

Parallèlement, des travaux de recherche se préoccupent de mieux prendre en compte les préférences et les besoins d'un utilisateur. Ces travaux traitent majoritairement les éléments de base présents dans les documents HTML tels que la taille d'un texte, la couleur d'un texte et de son arrière-plan [10], [11], et plus rarement des éléments de plus haut niveau comme les listes, les menus ou les titres. Les profils utilisateurs sont représentés de manière assez fruste sous la forme d'un ensemble de valeurs attendues pour certaines caractéristiques de base comme les polices de caractères ou les couleurs [10], [11]. L'acquisition de ces profils se fait par l'intermédiaire d'une interface de saisie (qui pose elle-même un problème d'accessibilité), à partir de tests visuels [12], ou par apprentissage d'après des actions réalisées par l'utilisateur pour adapter les pages à son besoin [13], [14]. Une approche basée sur des algorithmes d'optimisation pour adapter les couleurs d'une page pour la dichromasie est proposée dans [15]. Dans les travaux présentés ici nous prenons en compte les besoins de l'utilisateur, mais nous tâchons également de préserver le message transmis par le design initial. Nous cherchons à prendre en compte des préférences plus complexes que la simple expression d'une couleur ou d'une taille de police et ces préférences ne sont pas forcément orientées vers une déficience particulière.

VII. Conclusion

Nous avons présenté une approche générale permettant d'adapter certains éléments d'une page Web en fonction de déficiences visuelles et préservant autant que possible le design initial de la page. L'approche s'appuie sur un algorithme évolutionnaire, NSGA, que nous étudions dans deux de ses versions. Nous l'avons mise en œuvre sur des pages Web réelles et dans un cas particulier portant sur des brillances et des couleurs de textes et d'arrière-plans. Les résultats sont prometteurs mais nécessitent des améliorations sur plusieurs points et en particulier sur les temps de calcul en affinant les algorithmes d'optimisation. Un travail futur se consacrera à l'automatisation du recueil des variables et des préférences.

Remerciements

Les auteurs remercient le labex NUMEV (convention ANR-10-LABX-20) et Berger Levrault pour le soutien qu'ils apportent à ce travail.

Références

[1] Y. Bonavero, M. Huchard, and M. Meynard, "Reconciling user and designer preferences in adapting web pages for people with low vision," in *Proceedings of the 12th Web for All Conference, W4A '15*, L. Carriço, S. Mirri, T. J. Guerreiro, and P. Thiessen, Eds. ACM, 2015, pp. 10 :1–10 :10. [Online]. Available : http://doi.acm.org/10.1145/2745555.2746647

[2] *Web Content Accessibility Guidelines*, 2008, http://www.w3.org/TR/WCAG/.

[3] S. Kaci, *Working with Preferences : Less Is More*, ser. Cognitive Technologies. Springer, 2011.

[4] 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.

[5] 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, pp. 182–197, 2002. [Online]. Available : http://dx.doi.org/10.1109/4235.996017

[6] 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," *Evolutionary Computation, IEEE Transactions on*, vol. 18, no. 4, pp. 602–622, Aug 2014.

[7] A. Liefooghe, "Metaheuristics for multiobjective optimisation : Cooperative approaches, uncertainty handling and application in logistics," Theses, Université des Sciences et Technologie de Lille - Lille I, Dec. 2009. [Online]. Available : https://tel.archives-ouvertes.fr/tel-00464166

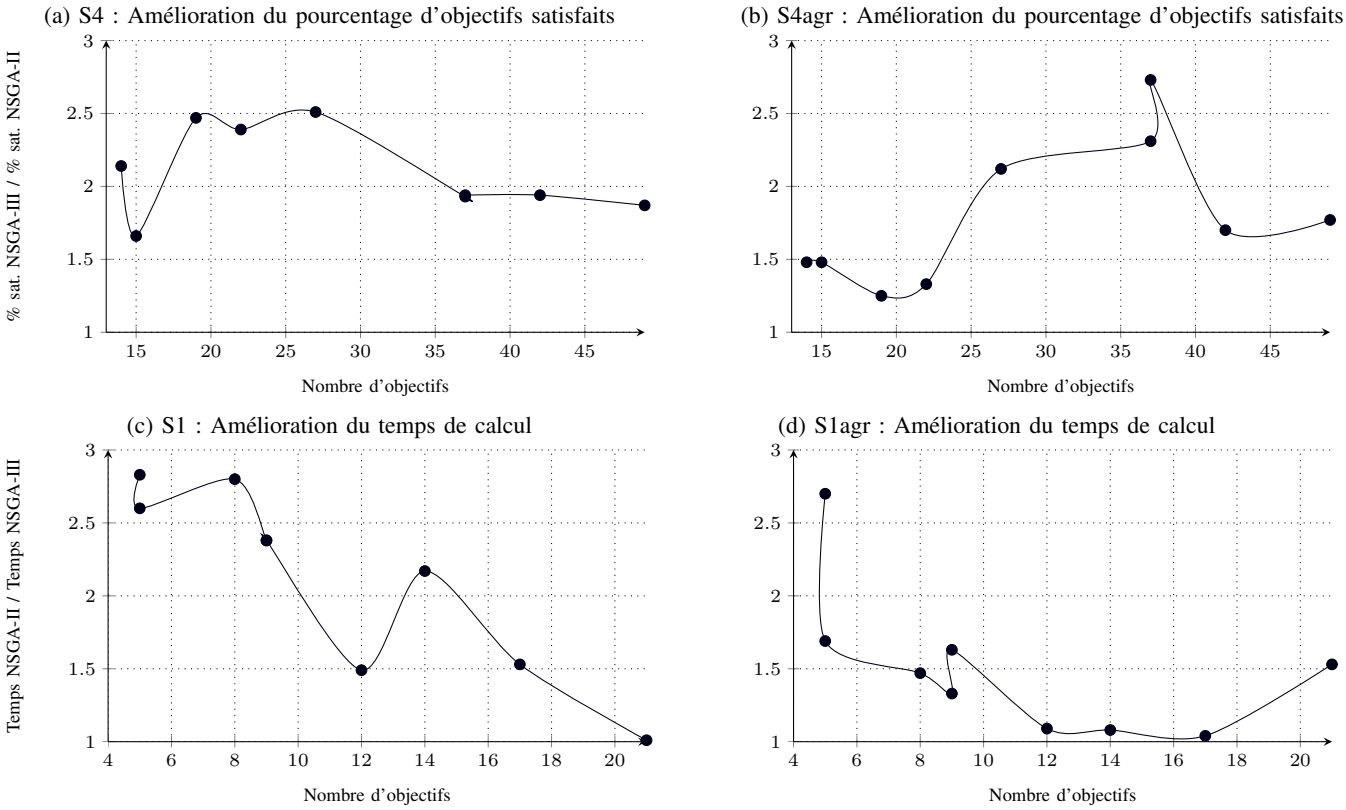

FIGURE 7. Amélioration apportée par NSGA-III sur les pourcentages d'objectifs satisfaits et le temps de calcul

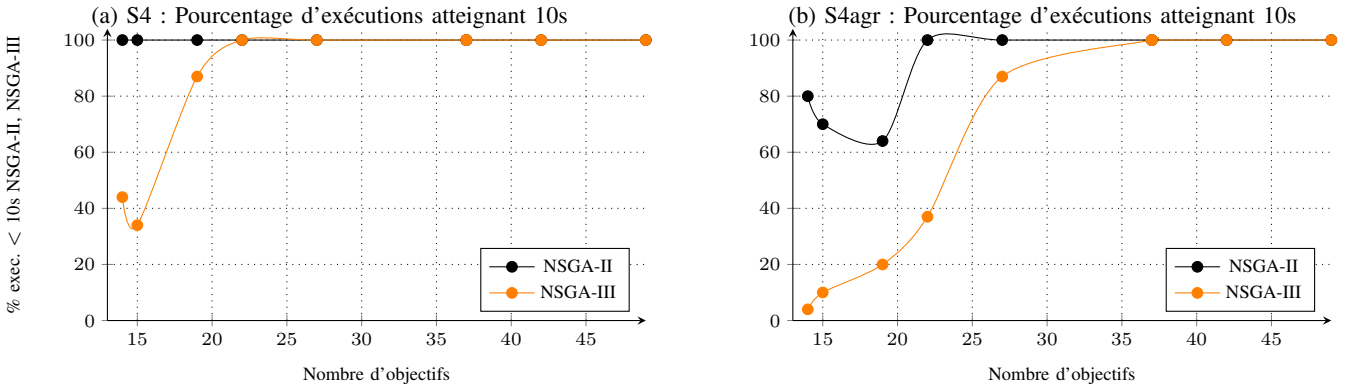

FIGURE 8. Amélioration apportée par NSGA-III : diminution du nombre d'exécutions atteignant les 10s

[8] Y. Bonavero, "Une approche basée sur les préférences et les méta-heuristiques pour améliorer l'accessibilité des pages web pour les personnes déficientes visuelles," Ph.D. dissertation, Université de Montpellier, novembre 2015. [Online]. Available : http://hal-lirmm.ccsd.cnrs.fr/tel-01312294

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

[10] G. Santucci, "Vis-a-wis : Improving visual accessibility through automatic web content adaptation," in *Universal Access in Human-Computer Interaction. Applications and Services, 5th International Conference, UAHCI 2009, Held as Part of HCI International 2009. Proceedings, Part III*, 2009, pp. 787–796.

[11] 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.

[12] A. Foti and G. Santucci, "Increasing web accessibility through an assisted color specification interface for colorblind people," *IxD&A*, vol. 5-6, pp. 41–48, 2009.

[13] 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, 2014*. IEEE, 2014, pp. 252–257. [Online]. Available : http://dx.doi.org/10.1109/COMPSACW.2014.45

[14] ——, "User centered and context dependent personalization through experiential transcoding," in *Proc. IEEE Consumer Communications and Networking (CCNC 2014), Workshop NIME'14*, 2014, pp. 486 – 491.

[15] A. Mereuta, S. Aupetit, N. Monmarché, and M. Slimane, "Web page textual color contrast compensation for CVD users using optimization methods," *J. Math. Model. Algorithms in OR*, vol. 13, no. 4, pp. 447–470, 2014. [Online]. Available : http://dx.doi.org/10.1007/s10852-013-9239-3