Modélisation et apprentissage des préférences appliqués à la recommandation dans les systèmes d'impression

Vincent Labbé

THÈSE DE DOCTORAT DE L'UNIVERSITÉ PARIS 6

Spécialité : **Informatique**

(École Doctorale d'Informatique, Télécommunications et Électronique – EDITE)

Présentée pour obtenir le titre de

DOCTEUR DE L'UNIVERSITÉ PARIS 6

par

Vincent Labbé

Sujet de la thèse :

Modélisation et apprentissage des préférences appliqués à la recommandation dans les systèmes d'impression

Soutenue le 22 Septembre 2009
devant le jury composé de :

M. Thierry Artières	Examinateur
M. Stéphane Berche	Examinateur
Mme Bernadette Bouchon-Meunier	Directrice de thèse
M. Gilles Coppin	Rapporteur
M. Nicolas Labroche	Examinateur
M. Jérôme Lang	Rapporteur
Mme Brigitte Trousse	Examinateur

Table des matières

Remerciements

Table des figures

Liste des tableaux

Liste des notations

FS ensemble des formats standards

NS ensemble des noms de formats standards

$seuil$ seuil de tolérance

p position de l'éventuel cartouche

D ensemble des documents potentiellement imprimables

z^* zoom désiré par l'imprimeur

G^{sup} ensemble des marges supplémentaires

g^{sup} marges supplémentaires particulières

$g^{sup}_{\Uparrow}$ marge supplémentaire au-dessus de l'image

$g^{sup}_{\Leftarrow}$ marge supplémentaire à gauche de l'image

$g^{sup}_{\Downarrow}$ marge supplémentaire en-dessous de l'image

$g^{sup}_{\Rightarrow}$ marge supplémentaire à droite de l'image

G^{ext} ensemble des marges d'extrémités

g^{ext} marges d'extrémités particulières

$g^{ext}_{\Uparrow}$ marge d'extrémité au-dessus de l'image

$g^{ext}_{\Downarrow}$ marge d'extrémité en-dessous de l'image

G^{mat} ensemble des marges matérielles

g^{mat} marges matérielles particulières

$g^{mat}_{\Uparrow}$ marge matérielle au-dessus de l'image

$g^{mat}_{\Leftarrow}$ marge matérielle à gauche de l'image

$g^{mat}_{\Downarrow}$ marge matérielle en-dessous de l'image

$g^{mat}_{\Rightarrow}$ marge matérielle à droite de l'image

L largeur d'une source de média

H_{min} hauteur minimale d'un média

H_{max} hauteur maximale d'un média

$\mathcal{F}$ ensemble des possibilités de finitions

F finition particulière

S ensemble des sources de média

$\hat{S}$ ensemble des sources de médias disponibles

s source de média particulière

s_L largeur de la source de média s

$s_{H_{min}}$ hauteur minimale d'un média issu de la source de média s

$s_{H_{max}}$ hauteur maximale d'un média issu de la source de média s

s_F ensemble des finitions possibles, applicables à un média issu de la source de média s

$s_{G^{mat}}$ marges matérielles associées à la source de média s

M ensemble des médias potentiels

m média particulier

m_l largeur du média m

m_h hauteur du média m

m_f finition du média m

R ensemble des angles de rotations

r angle de rotation particulier

Z ensemble des valeurs de zoom

z zoom particulier

G ensemble des marges relatives à l'image

A_i domaine de définition de l'attribut i

A ensemble des domaines de définitions des attributs

$\mathbb{A}$ ensemble des alternatives

$\mathbb{H}_A$ ensemble des hypothèses relatives à A

$\mathbb{D}$ ensemble d'apprentissage (ou des décisions observées)

u fonction d'utilité

u_i fonction d'utilité marginale sur le i^{eme} attribut

P mesure de probabilité

Introduction

Les travaux de recherche relatés dans ce mémoire portent sur la modélisation et l'apprentissage automatique des préférences et sur leur application en vue d'améliorer l'utilisabilité de systèmes d'impression grand format. Après avoir présenté l'impression grand format, nous soulignons l'enjeu et les difficultés de l'automatisation de la configuration d'impression. Nous introduisons ensuite notre approche et les problématiques scientifiques qui s'y rattachent.

Méconnue du grand public, l'impression grand format n'en est pas moins omniprésente dans la vie quotidienne. L'impression de panneaux publicitaires, tels les panneaux promotionnels qu'on retrouve dans les magasins de la grande distribution, est un des domaines d'application les plus importants de l'impression grand format en couleurs. L'impression grand format en noir et blanc trouve plutôt ses applications dans les bureaux d'études : plans d'architecture, dessins techniques, etc. De l'image publicitaire à la documentation technique, en passant par l'imprimé artistique et le poster scientifique, le grand format est omniprésent. Dans les magasins et les entreprises, dans les galeries d'art et les colloques scientifiques, l'imprimé grand format est un outil de communication indispensable.

Si le grand format permet de bien communiquer son art, encore faut-il l'imprimer correctement, ce qui peut se révéler être un art en soi. Imprimer un document, c'est avant tout configurer les paramètres d'impression afin d'obtenir le résultat désiré. Ceux qui ont déjà imprimé, petit ou grand format, savent que cela n'est pas aussi simple que cela en a l'air, dés que l'on veut obtenir un résultat sortant de l'ordinaire.

Pour le professionnel de l'impression grand format, cette difficulté est multipliée par le nombre de documents à imprimer et par l'hétérogénéité des documents, qui peuvent avoir des dimensions, et d'autres caractéristiques, très diverses. C'est pourquoi l'*automatisation de la configuration des paramètres d'impression* est un enjeu majeur dans le monde des applications informatiques dédiées aux systèmes d'im-

pression professionnelle grand format.

Cette automatisation n'est pas triviale : elle doit répondre aux deux objectifs contradictoires habituels :

- proposer un comportement suffisamment riche pour répondre aux attentes de chacun.
- être suffisamment simple à utiliser pour tous.

Nos travaux ont été réalisés afin d'apporter des éléments de réponse à ce double objectif. Nous détaillons dans la suite notre approche sur ces deux points.

Premièrement, la richesse comportementale pose des problèmes aux concepteurs. D'une part, la complexité rend la spécification ardue. D'autre part, codée de façon classique, cette fonctionnalité est propice aux erreurs et rend la maintenance et l'évolutivité du code difficile. Nous nous sommes donc attaché à faciliter la tâche des développeurs en montrant comment la configuration automatique d'une impression grand format peut être modélisée comme un problème d'optimisation sous contraintes. En particulier comme un problème linéaire mixte en nombre entier. Cela permet de profiter des avantages bien établis de ce type d'approche :

- représentation déclarative de la logique métier, simplifiant la spécification, la maintenance et l'évolutivité.
- exploitation d'algorithmes de résolutions efficaces, diminuant l'effort de développement.

Deuxièmement, concernant l'utilisabilité, il a été constaté que cette fonctionnalité était sous-exploitée, dans le sens où elle fournit une palette de comportements dont seuls quelques uns sont effectivement utilisés. L'hypothèse, qui motive les travaux relatés dans ce mémoire, faite pour expliquer ce phénomène tient en ceci : le comportement du module est trop complexe à configurer. Ce problème nous a amené à proposer une solution sous la forme d'une interface adaptative permettant d'améliorer l'utilisabilité de l'application. Plus précisément, elle prend la forme d'un système de recommandation, ayant la faculté d'apprendre les préférences de l'utilisateur au fur et à mesure de l'interaction, afin d'améliorer la pertinence de ses recommandations.

La conception de ce système de recommandation nous a amené à proposer un nouveau modèle pour représenter les préférences de l'imprimeur, ainsi qu'à développer de nouveaux algorithmes d'apprentissage artificiel, adaptés aux contraintes de notre problématique. En particulier, nous nous sommes intéressés à adapter automatiquement la complexité de la struture du modèle, sous une contrainte forte de

linéarité.

Nos travaux constituent une approche globale de l'automatisation du paramétrage des impressions grand format, traitant des aspects de conception et d'utilisation. La présentation de ces différents aspects est répartie dans les chapitres de ce mémoire, qui est organisé de la façon suivante :

Le chapitre 1 introduit la problématique industrielle de l'automatisation de la configuration des paramètres d'impression de document grand format. L'approche générale de nos travaux est également présentée et positionnée par rapport au domaine des systèmes de recommandation.

Au chapitre 2, nous abordons le problème de l'évaluation de l'utilisabilité d'un logiciel. Nous présentons d'abord un état de l'art sur les méthodes d'évaluation, particulièrement focalisé sur les méthodes d'évaluation automatique à partir de logs d'interaction. Dans ce cadre, nous proposons des mesures statistiques permettant de tester objectivement le rendement d'une interface Homme-Machine (IHM), telle que celles utilisées par les imprimeurs professionnels. Nous appliquons notre approche dans une étude d'utilisabilité, réalisée sur une interface de configuration d'impression développée par Océ. Nos résultats soulignent la difficulté de la configuration d'impression.

Dans le chapitre 3, la configuration des paramètres d'impression est analysée et modélisée, afin de définir l'ensemble des configurations admissibles, connaissant les caractéristiques du document à imprimer, les contraintes et directives de l'imprimeur, ainsi que les ressources d'impression disponibles.

Le chapitre 4 traite du modèle mathématique permettant de décrire les préférences de l'imprimeur. Outre son pouvoir descriptif, ce modèle doit également répondre à un autre objectif : il doit pouvoir être utilisé efficacement, afin de déterminer la configuration préférée parmi les configurations admissibles. Après un état de l'art de l'existant, nous proposons un nouveau modèle de préférence de l'imprimeur répondant à nos objectifs.

Le chapitre 5 porte sur l'apprentissage automatique des préférences, à partir d'informations d'ordonnancement. Nous distinguons l'apprentissage de la structure de l'apprentissage des poids du modèle. Nous passons en revue différentes techniques

pour l'apprentissage des poids, à partir de comparaisons de paires d'items. Cet état de l'art nous amène à sélectionner l'algorithme rankSVM avec noyau linéaire pour réaliser cette tâche. Concernant l'apprentissage de la structure, nous proposons un algorithme original permettant d'adapter la complexité de l'espace de description des données, tout en conservant la linéarité du modèle appris.

Enfin nous présentons notre système de recommandation d'impression, basé sur l'approche développée, et discutons de ses performances.

Chapitre 1

Automatiser la configuration d'impression

Nous présentons dans ce chapitre la problématique industrielle à la base de nos travaux : l'automatisation de la configuration des paramètres d'impression d'un document.

Nous introduisons d'abord les différents aspects de la configuration d'impression grand format, ce qui nous permet de mettre en lumière l'intérêt industriel de cette problématique. Puis nous posons le cadre applicatif, à travers un logiciel de gestion d'impression développé par Océ ; et détaillons les objectifs industriels liés à cette problématique.

Dans la seconde partie de ce chapitre, nous présentons et justifions l'approche générale de nos travaux. Nous adoptons le paradigme des systèmes de recommandation afin d'améliorer l'utilisabilité de l'application. Un rapide état de l'art nous permet de positionner plus précisément notre démarche par rapport aux systèmes de recommandation existant.

1.1 Automatiser la configuration d'impression

Dans cette section, nous passons d'abord en revue les différents aspects de la configuration des paramètres d'impression d'un document, dans le cadre du grand format. Nous y montrons l'intérêt de son automatisation et posons une hypothèse quant à sa faisabilité. Ensuite nous faisons le point sur les deux cadres logiciels classiques, pilote d'imprimante et application autonome, au travers desquels l'imprimeur réalise cette configuration. Nous positionnons ainsi notre cadre applicatif, centré sur les applications autonomes dédiées à la gestion d'impression. Enfin nous présentons une solution logicielle actuelle, développée par Océ. Dotée d'une fonc-

tionnalité d'automatisation de la configuration d'impression, cette application fait partie de notre cadre applicatif. Nous en soulignons les limites et mettons en lumière les objectifs industriels relatifs à la problématique de l'automatisation de la configuration d'impression.

1.1.1 Configuration d'impression

Nous présentons ici les quatre aspects fondamentaux qui caractérisent la tâche de configuration d'impression grand format dans un cadre professionnel.

Configurer les paramètres d'impression d'un document est la tâche principale de l'imprimeur. Dans le monde du grand format, on distingue trois grandes classes de paramètres, correspondant aux différents aspects d'un imprimé :

- la mise en page
- la qualité d'impression
- la finition

Il existe également un quatrième aspect, propre à l'impression professionnelle grand format : la multiplicité et surtout l'*hétérogénéité* des documents à imprimer, constituant un *travail d'impression*. On doit considérer que la tâche de configuration de l'imprimeur ne concerne pas un document unique mais une série de documents hétérogènes. Ces documents, généralement des images (i.e. des pages uniques), présentent des caractéristiques différentes (dimensions, contenu, couleur, etc.) qui influent sur le paramétrage de l'impression. Cette dimension, orthogonale aux trois autres, donne tout son intérêt à l'automatisation de la configuration d'impression.

Seuls les aspects de mise en page, de finition et d'hétérogénéité font l'objet du travail de recherche relaté dans ce mémoire. Nous ne traitons pas la problématique de la qualité d'impression pour la raison suivante : les critères de qualité sont très dépendants du contenu de chaque document. Prendre en compte les critères de qualité lors de l'automatisation de la configuration d'impression demande une analyse du contenu de chaque document, ce qui est hors du champ de notre travail de recherche. Cela pourrait constituer une extension à nos travaux, c'est pourquoi nous revenons dessus dans les perspectives.

Afin de fournir au lecteur une vision d'ensemble de la tâche de configuration, nous introduisons cependant les quatre aspects dans les paragraphes suivants. Nous terminons cette section par un point sur l'hypothèse nécessaire à l'automatisation de la configuration d'impression : l'existence d'une politique d'impression.

Mise en page

L'impression sur du grand format répond à une double motivation : obtenir un support pour communiquer et sauvegarder un document sous forme papier. La première motivation est la communication. Bien communiquer à partir d'un imprimé grand format peut signifier différentes choses pour la mise en page, suivant le contexte. Par exemple, un photographe voulant présenter ses œuvres privilégiera des feuilles au moins aussi grandes que les images, permettant d'imprimer celles-ci à l'échelle 1 : 1, de façon à éviter l'apparition du grain et ce, en dépit d'éventuelles marges blanches. En revanche, un ingénieur qui présente une série de dessins techniques devant une assemblée privilégiera un agrandissement maximum des dessins, afin d'être visibles d'assez loin. De plus, contrairement aux images *bitmap* du photographe dont la précision diminue avec l'agrandissement, les images de documents techniques sont généralement *vectorielles* et donc ne souffrent pas d'un changement d'échelle. D'autres, lorsque les images présentent déjà des marges, permettront de rogner légèrement les bords des images si aucune feuille n'est assez grande pour les contenir, etc. On voit que selon le contexte, le besoin de mise en page se traduit par différentes *préférences* sur l'échelle, les marges, les coupes, etc.

Qualité

Il en est de même pour la qualité d'impression. Celle-ci se traduit en terme de résolution, qualité du papier et des encres, gestion de la couleur, des à-plats, de la netteté des lignes, etc. Par exemple, il existe différents algorithmes pour reconstituer "au mieux" les couleurs que l'on perçoit à l'écran sur le papier, c'est le problème du *gamut mapping*. De même, plusieurs méthodes sont possibles afin de distribuer les gouttes d'encre sur le papier (*halftoning*), avec chacune leurs avantages et inconvénients comme l'apparition d'effets de bandes ou de "vers". Certaines techniques favorisent ainsi l'impression des lignes, ce qui convient parfaitement à l'impression de dessins techniques ; d'autres rendent mieux avec l'impression de photos.

Finitions

Les clients d'impression grand format ont des besoins en termes de mise en page et de qualité mais pas seulement. En effet, la présentation, le transport ou la conservation des imprimés induisent des besoins en terme de finitions.

Par exemple, on peut envisager de conserver des imprimés grand format de différentes manières :

- empilés à plat. Cette solution est peu pérenne car elle implique des difficultés à retrouver un imprimé particulier dans la pile.

- rangés dans une armoire, accrochés verticalement. Cela peut être une bonne manière de faire, à la condition d'avoir prévu une bande supplémentaire de papier, éventuellement renforcée, servant de point d'attache pour les crochets, permettant ainsi de ne pas abîmer le contenu des imprimés.
- pliés et rangés dans un tiroir. Voilà sûrement la façon la plus simple de conserver ses A0. Cependant, le pliage manuel n'est pas aisé. De plus, pour retrouver le plan X plié et empilé parmi d'autres, on a tout intérêt à avoir son identifiant imprimé sur le côté visible du paquet plié.

Rajouter une bande permettant d'accrocher verticalement sans abîmer l'image, plier de façon à ce que le cartouche[1] d'un dessin technique se retrouve au-dessus... cela fait partie des finitions pouvant être souhaitées par un client.

Hétérogénéité

Une des grandes différences entre un travail d'impression petit format et grand format réside dans l'hétérogénéité des pages à imprimer. Pour imprimer un rapport de thèse, il suffit de paramétrer une fois la mise en page et la qualité pour toutes les pages. A l'inverse, imprimer une série de dessins techniques présente une difficulté supplémentaire : rien ne garantit que ces documents soient homogènes (en terme de dimensions, type de contenu, etc.). Cela signifie que, même si l'intention de l'ingénieur est simple (par exemple tout imprimer sur du A0, sans coupe et avec le moins de marge possible), les paramètres d'impression, valables pour un dessin technique, ne le sont pas forcément pour un autre.

On voit donc que, dans le monde du grand format, l'effort et le temps passé à configurer les paramètres d'impression sont proportionnels au nombre de documents (constitué généralement d'une page unique) qui constituent le travail d'impression. Cet aspect est fondamental pour comprendre l'importance de l'automatisation du processus de configuration d'impression.

Politique d'impression

Si un petit exemple suffit à en souligner l'intérêt, la question de la *faisabilité* de l'automatisation du processus de configuration mérite un premier éclaircissement. Automatiser une tâche suppose que celle-ci obéisse à des règles constantes. Autrement dit, si un client veut imprimer un ensemble de documents qui n'ont rien en commun, par exemple un mélange de photos, plans, dessins techniques, etc., correspondant à autant de contraintes d'impression différentes, il est illusoire de vouloir en

[1] encadré résumant toutes les informations nécessaires à l'identification d'un dessin technique, généralement accolé à un bord de la feuille

automatiser la configuration lorsque l'on n'a pas accès au contenu de ces documents. En revanche, lorsque ces documents appartiennent à une même catégorie (e.g. dessins techniques du projet Z), leur impression répond généralement à des contraintes communes. Dans l'exemple du paragraphe précédent : imprimer sur du A0, sans coupe et avec le moins de marge possible. Ces contraintes communes forment ce que l'on appelle une *politique d'impression*. L'automatisation de la configuration d'impression d'un ensemble de documents (i.e. d'un travail d'impression) suppose donc l'existence d'une telle politique d'impression, sous-jacente à l'intention du client.

Definition 1. Une politique d'impression est un ensemble de contraintes et de préférences permettant de configurer les paramètres d'impression de tout document, quel que soit le contexte, c'est-à-dire les ressources d'impression disponibles.

1.1.2 Cadre applicatif

Dans cette section, nous introduisons les deux cadres logiciels classiques, au travers desquels l'imprimeur réalise la configuration des paramètres d'impression des documents. Chaque cadre applicatif correspond à un type d'imprimeur : l'imprimeur occasionnel utilise plutôt un pilote d'imprimante, alors que l'imprimeur professionnel privilégiera une application autonome, dédiée à la gestion d'impression. Cette dernière catégorie constitue le cadre applicatif de nos travaux. Les applications autonomes de gestion d'impression, développées par Océ, fournissent une fonctionnalité particulière : l'automatisation de la configuration des paramètres d'impression d'un document. Très appréciée, cette fonctionnalité pose néanmoins certains problèmes, en terme d'utilisation comme de développement. La résolution de ces problèmes constitue l'objectif industriel de nos travaux de recherche.

Pilotes d'impression & imprimeurs occasionnels

Les pilotes d'impression sont le moyen le plus couramment utilisé pour imprimer sur grand comme petit format par les *imprimeurs occasionnels*. Architectes, communiquants, ingénieurs ou chercheurs, les imprimeurs occasionnels sont des professionnels ayant besoin d'imprimer sur grand format de façon ponctuelle. Ils utilisent les facilités d'impression comme outils dans leur travail. Ils disposent en général d'une seule imprimante grand format et impriment à travers une application (par exemple AutoCAD, Word ou AcrobatReader, pour ne citer que les plus utilisées) via un pilote d'impression. Ils ont des besoins divers en mise en page, en qualité ou en finitions qui correspondent à l'usage qu'ils vont faire des imprimés. Pour traduire ces besoins résultat papier, ils doivent paramétrer l'application et le pilote d'impression.

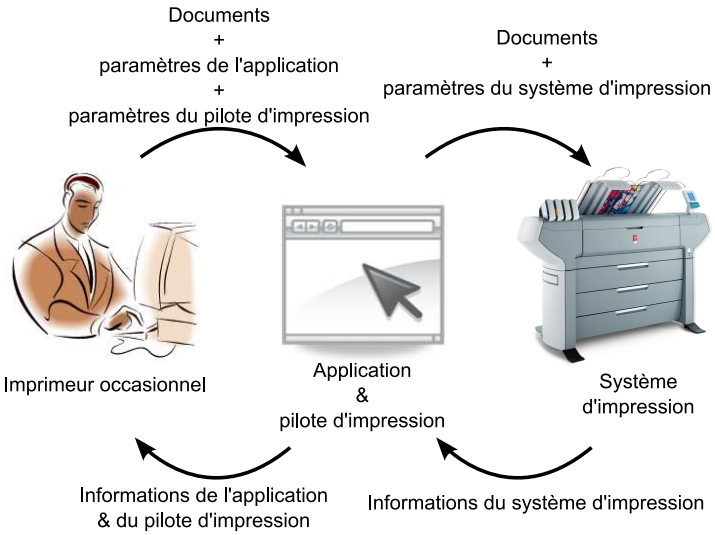


FIG. 1.1: Communication entre l'imprimeur occasionnel et le système d'impression

Partagées par les imprimeurs en petit format, les difficultés communes rencontrées au cours de cette tâches sont de deux sortes :

1. Difficulté à trouver les paramètres dans l'interface (de l'application ou du pilote), permettant de traduire son besoin. Cela arrive fréquemment lorsque des fonctionnalités trop complexes pour l'imprimeur sont proposées ou que les paramètres ne sont pas assez saillants (e.g. "enfouis" sous une cascade de fenêtres...).
2. Difficulté à prévoir le comportement de certains paramètres ou de certaines combinaisons de paramètres. Cela arrive lorsque le comportement d'un paramètre n'est pas suffisamment explicité par l'interface. Cela arrive également en cas de conflits entre paramètres. En particulier entre les paramètres de l'application et ceux du pilote d'impression. Par exemple, si l'utilisateur paramètre une rotation de l'image de 90° dans l'application et de même dans le pilote, la rotation finale peut être de 180°, 90° ou même 0°, suivant le comportement de l'application.

Logiciels dédiés & professionnels de l'impression

Lorsqu'ils n'ont pas les moyens matériels d'imprimer eux-mêmes, les imprimeurs occasionnels peuvent faire appel aux professionnels de l'impression. Reprographes ayant pignon sur rue ou personnel d'un centre de reprographie interne à une grande entreprise, l'impression est leur métier. Ils disposent généralement d'un parc d'imprimantes et de scanners grand format, qu'ils gèrent à travers des outils logiciels adap-

tés : des gestionnaires de travaux d'impression et de reprographie professionnels. Ils doivent répondre à des requêtes d'impression provenant de clients divers, avec des besoins d'impression correspondant à des usages différents. Pour chaque client, leur travail quotidien est d'abord de trouver la meilleure solution d'impression, en fonction des besoins, des ressources et des contraintes. Il faut ensuite traduire cette solution en paramètres d'impression pour chaque document à imprimer. Comme le nombre de documents peut être élevé (par exemple, plusieurs dizaines de plans représentant un projet architectural), la facilité de configuration des paramètres d'impression de chaque document est fondamentale. En tant que professionnels, les besoins d'optimisation des coûts sont pressants.

Les logiciels dédiés à la gestion d'impression grand format permettent de réduire ces coûts. Premièrement, en tant qu'applications autonomes, ils s'affranchissent des applications des clients. De plus, ils fournissent des fonctionnalités permettant de faciliter le travail de l'imprimeur. Parmi les plus appréciées, on trouve :

- la gestion transparente de différents formats de fichiers (Word, PostScript, PDF, GIF, JPG, etc.).
- la représentation graphique miniature du résultat attendu de l'impression.
- la configuration automatique des paramètres d'impression des documents, d'après un modèle des préférences de l'imprimeur.
- la manipulation aisée de nombreux documents, via le concept de *travail d'impression*. Un travail d'impression est un ensemble de documents à imprimer selon une même politique d'impression.
- la gestion aisée de nombreuses imprimantes, à travers la notion d'*imprimante abstraite*. Une imprimante abstraite est une description logique des caractéristiques d'une imprimante, pouvant correspondre à plusieurs machines réelles.

Les applications autonomes de gestion d'impression constituent notre cadre applicatif. Plus précisément, nos travaux sont focalisés sur la configuration automatique, qui est une des fonctionnalités les plus appréciées mais également une des plus problématiques, du point de vue des utilisateurs comme du point de vue des concepteurs. Nous abordons plus en détail les problèmes rencontrés dans les paragraphes suivants, à travers l'exemple d'une application autonome développée par Océ.

Publisher Engineering

Océ développe toute une gamme de logiciels, allant du pilote d'imprimante à l'application autonome dédiée à la gestion d'impression ou de reprographie. Parmi cette gamme d'applications, nous nous sommes intéressés particulièrement à une application autonome nommée *Publisher Engineering*. Celle-ci est dotée d'une fonctionnalité

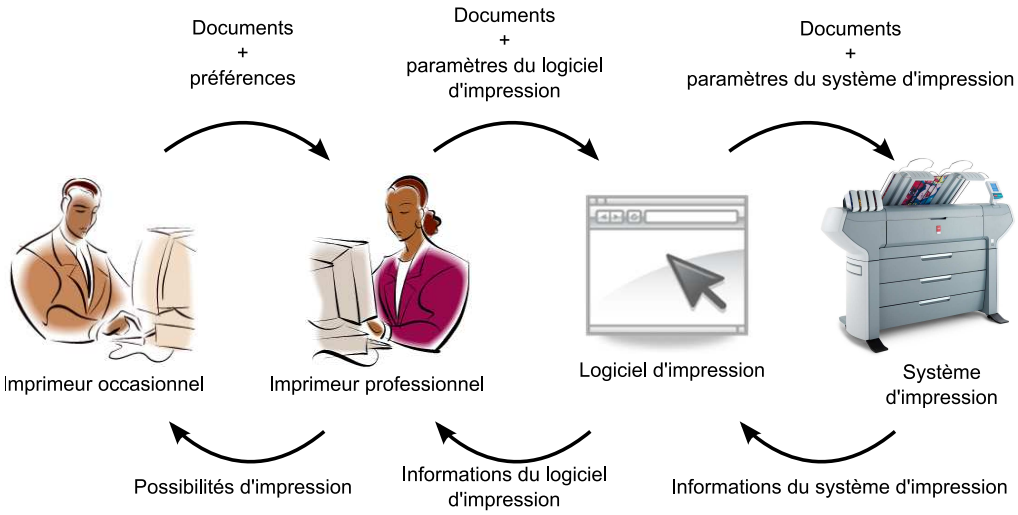


FIG. 1.2: Communication entre l'imprimeur occasionnel et le système d'impression via l'imprimeur professionnel

d'automatisation de la configuration d'impression. Plutôt dédiées aux professionnels de l'impression, elle permet de configurer les paramètres d'impression d'un lot de documents de façon "transparente" pour l'imprimeur. Cependant, afin de réaliser cette tâche, l'utilisateur doit au préalable déterminer sa politique d'impression. Celle-ci prend la forme d'un ensemble de paramètres "haut niveau", tel que la façon de choisir un média en fonction de la taille d'un document, avec des possibilités telles que : même taille exactement, au moins aussi grand, au moins aussi grand sauf s'il n'y en a pas, etc.

Une fois la politique d'impression configurée, il suffit à l'imprimeur de charger les documents qu'il désire imprimer dans l'application. Ceux-ci sont automatiquement et immédiatement paramétrés pour l'impression. Une représentation graphique miniature du résultat attendu est alors présentée pour chaque document. La figure 1.3 montre une capture d'écran de la fenêtre principale de l'application, après chargement et configuration automatique de trois documents.

Fer-de-lance des produits logiciels d'Océ, appréciée des imprimeurs, la fonctionnalité actuelle, qui automatise la configuration d'impression, n'est pourtant pas exempte de défauts, pour ses utilisateurs comme pour ses concepteurs.

Une plus-value sous-exploitée

Suite à plusieurs enquêtes, les ingénieurs d'Océ se sont aperçus que leur module d'automatisation était sous-exploité : la plupart des politiques disponibles sont très peu utilisées. A partir de ce constat, on pourrait conclure à l'inutilité d'une fonc-

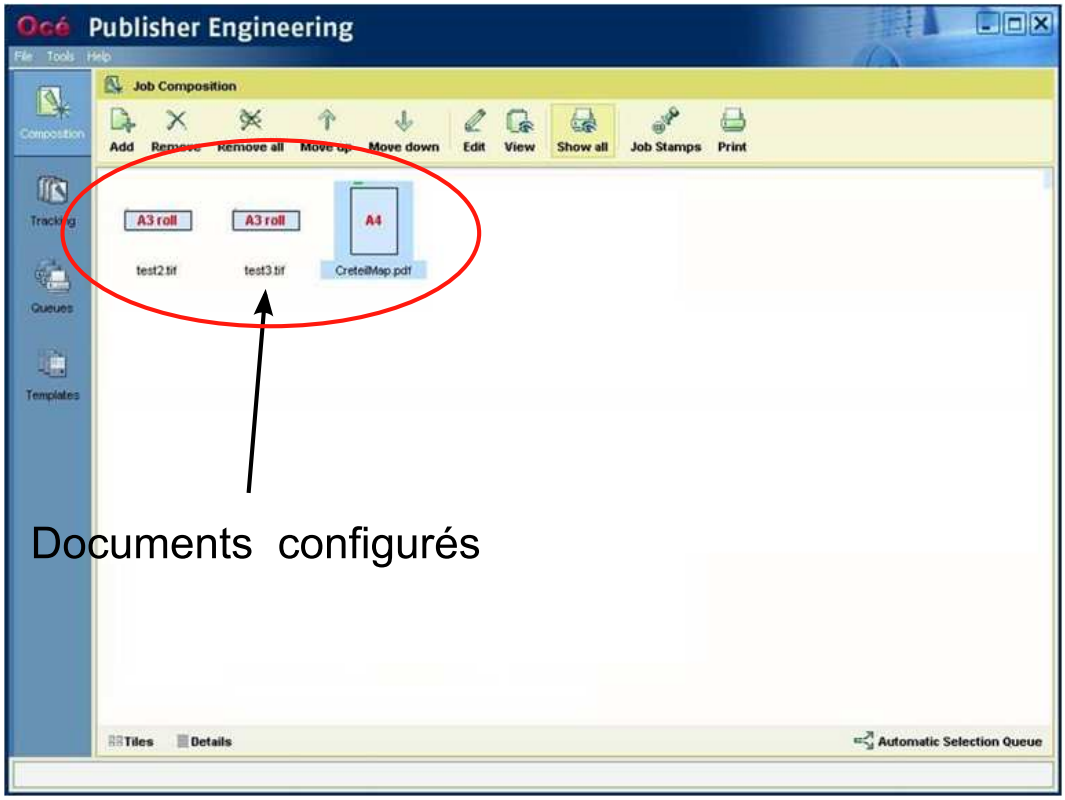


FIG. 1.3: Interface de l'application Océ Publisher Engineering

tionnalité trop complexe. Cependant, les enquêtes ont également fait remonter les difficultés ressenties lors du paramétrage des politiques d'impression. Le diagnostic devient alors un arbitrage entre complexité inutile et paramétrage trop difficile. La première option - une fonctionnalité inutilement riche - étant plus simple à corriger que la seconde, Océ a jusqu'à présent pris le parti de brider son module afin d'en simplifier à l'extrême l'utilisation, en mettant en avant les comportements les plus demandés. Les derniers logiciels développés ne permettent plus à l'imprimeur de paramétrer sa politique d'impression mais lui laisse le choix parmi quelques politiques usuelles pré-configurées.

Choisir une politique d'impression : un problème de décision multicritère.

Choisir une politique d'impression, c'est exprimer ses préférences sur le résultat, en terme de mise en page et de finition, que l'on veut obtenir, quel que soit le contexte. Le contexte ici désigne les propriétés du document à imprimer et les ressources du système d'impression. Nous avons vu que les préoccupations de l'imprimeur grand format sont diverses, du choix du média au type de pliage, en passant par les transformations et le positionnement de l'image. Suivant les contextes, différentes solutions s'offrent à l'imprimeur. La politique d'impression doit permettre de choisir le meilleur compromis. Configurer une politique d'impression revient à exprimer les priorités entre différentes caractéristiques comme la taille des marges, de l'image, des coupes éventuelles, etc. Autant de critères potentiellement conflictuels. Le rôle d'une politique d'impression est d'arbitrer ces conflits. Comme les cas d'application sont nombreux et variés suivant les contextes, il peut être difficile pour l'imprimeur de prévoir et de configurer une politique en conséquence. Nous pensons que c'est la première raison du constat de sous-utilisation des capacités proposées par le module d'Océ.

Limites de la palette des politiques d'impression

Malgré la richesse du module actuel, beaucoup de comportements, potentiellement pertinents, ne font pas partie de la palette des comportements configurables. Voici quelques exemples permettant de se faire une idée des limites actuelles du module.

Il est impossible à l'imprimeur d'exprimer ses préférences quant à un compromis entre plusieurs grandeurs. Par exemple, entre le zoom effectué sur l'image et les marges supplémentaires ou encore entre le zoom et les coupes. Ce type de possibilité fait partie des améliorations apportées dans notre approche.

Autre possibilité potentiellement intéressante : imprimer une longue image sur plusieurs feuilles si l'on n'a pas de rouleau disponible. De même, un imprimeur peut

vouloir imprimer une très grande image en plusieurs bandes, issues d'un rouleau A0 par exemple. Aujourd'hui ces fonctionnalités ne sont pas fournies. Un dernier exemple pour clore cette liste non exhaustive : un imprimeur peut avoir des règles particulières concernant la configuration d'impression. Par exemple, vouloir imprimer du A0 et A1 sur un média A2, du A2 sur du A3, du A3 sur du A4... Il n'est pas possible de spécifier ce genre de règle.

Spécification, maintenance et évolutivité difficiles

Enfin le dernier défaut souligné ici concerne les développeurs du module. La spécification du module est d'une difficulté proverbiale, ce qui entraîne différentes versions pouvant se comporter légèrement différemment sur différents produits. Ce qui par conséquent peut perturber les clients qui utilisent plusieurs logiciels, voire qui passent d'une version à une autre. Sa programmation n'est pas aisée non plus car, codés de façon classique, tous les cas doivent être prévus. La gestion des multiples priorités sur chaque cas est une source importante de bugs. Enfin, la maintenance du module pose de sérieux problèmes tant les parties de codes sont imbriquées les unes aux autres. De même pour l'évolutivité : ajouter une fonction au module demande ainsi de reconsidérer l'ensemble du code.

Objectifs industriels

Il est clair que brider une fonctionnalité afin de la rendre plus simple d'utilisation au plus grand nombre est une solution à court terme. Afin de développer un avantage concurrentiel, les fonctionnalités sont appelées partout à être de plus en plus riches et donc complexes. Complexes à spécifier, à développer, à maintenir et bien sûr à utiliser. La configuration d'impression ne fait pas exception. Les possibilités des systèmes d'impression évoluent, proposant davantage de comportements. Pour exploiter au mieux ces possibilités, les imprimeurs ont des besoins de fonctionnalités toujours plus puissantes. Les déconvenues d'Océ concernant ce problème illustrent bien deux difficultés inévitables dans la course à la complexité :

1. Transformer la richesse d'une fonctionnalité en valeur ajoutée pour le client.
2. Maîtriser les coûts de conception, de développement, de maintenance et d'évolution du code.

Les travaux de recherche présentés dans ce mémoire apportent des éléments de réponse à ces deux problèmes, appliqués à la configuration d'impression. Ils répondent aux deux objectifs industriels suivants :

1. Faciliter le développement et la maintenance de la gestion du choix automatique d'une configuration d'impression, pour les concepteurs.
2. Faciliter le paramétrage d'une politique d'impression, pour l'utilisateur.

avec la contrainte de réaliser ces objectifs à fonctionnalité au moins équivalente.

1.2 Interface adaptative pour système d'impression

Notre approche en vue d'améliorer l'utilisabilité de l'automatisation de la configuration d'impression repose sur la conception d'un système possédant une interface adaptative. Une interface adaptative permet à une application de s'autopersonnaliser, en fonction de l'historique des interactions et en vue de faciliter d'une façon ou d'une autre son utilisation. L'interface peut s'adapter à ses utilisateurs de différentes manières, par exemple en mettant en avant certaines de ses fonctionnalités servant de raccourcis aux tâches les plus souvent exécutées par l'utilisateur, ou en recommandant des alternatives susceptibles d'intéresser l'utilisateur.

Pour améliorer l'utilisabilité des applications d'impression, on peut envisager deux possibilités :

1. aider l'imprimeur à paramétrer son modèle de préférences, c'est-à-dire sa politique d'impression, de manière explicite.
2. utiliser une interface adaptative en vue de personnaliser la politique d'impression de façon automatique.

C'est cette seconde approche que nous avons choisie. Le principe est de s'affranchir des paramètres techniques en privilégiant l'expression de la préférence sur le résultat attendu, c'est-à-dire les configuration d'impression elle-mêmes. Cette perspective nous amène à considérer les systèmes de recommandation comme solution.

1.2.1 Recommandation d'impression

Le problème de notre imprimeur peut être vu comme un problème de décision multicritère, appliqué au choix d'une politique d'impression. Dès lors, on peut envisager des méthodes d'aide à la décision afin de faciliter ce choix. D'une certaine manière (non académique) ceci a été le point de vue adopté par Océ jusqu'à présent. En effet, toutes les simplifications apportées à l'interface de l'utilisateur vont dans ce sens : faciliter les choix. Cependant, on a constaté que, même simplifiée à l'extrême, la phase de paramétrage de la politique d'impression reste rébarbative aux yeux des clients.

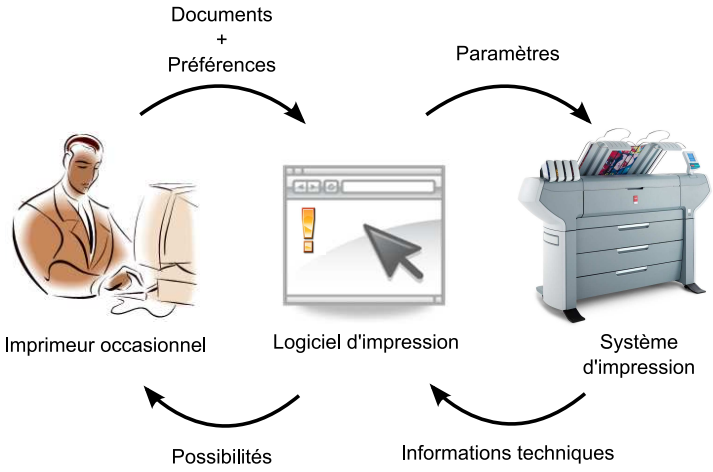


FIG. 1.4: Impression à travers une interface adaptative

Nous avons donc adopté un autre point de vue, basé sur l'interprétation suivante : cette phase est finalement considérée comme "du travail en plus" ; le travail qui importe vraiment est la configuration des paramètres d'impression des documents que l'imprimeur veut concrètement imprimer. Partant de cette hypothèse, nous devons idéalement exclure toute phase d'explicitation des préférences de l'imprimeur quant à la politique d'impression qu'il désire mener pour configurer ses documents. Cela concerne tout processus de paramétrage direct mais aussi tout processus de type *élicitation* des préférences, c'est-à-dire tout procédé visant à obtenir des informations sur les préférences d'un décideur de façon explicite. Par exemple, on peut éliciter des préférences à travers un questionnaire interactif. Le lecteur intéressé trouvera des éléments théoriques ainsi que des exemples de techniques et de réalisations, en matière d'élicitation des préférences, dans [CP04, Den03, GW05, LHL97a, HH97, Mou03].

Pour résumer, nous pensons que le concept même de politique d'impression doit être invisible pour l'imprimeur. La question de la configuration se pose alors différemment.

Un système de recommandation d'impression

Nous introduisons notre réponse à cette question par un parallèle entre le problème qui se pose à l'imprimeur qui veut configurer une impression et celui de l'acheteur qui doit choisir un produit dans un catalogue. En effet, ils doivent tous les deux chercher une solution, produit ou configuration, qui corresponde à leurs attentes, dans un ensemble d'alternatives. Il y a certes quelques différences notables. Le "catalogue des configurations" dépend des possibilités du système d'impression et des propriétés des documents à imprimer ; autrement dit, à chaque nouveau do-

cument est associé un nouveau catalogue. De plus, le "catalogue des configurations" n'est pas défini dans une base de données (en extension) mais par un ensemble de contraintes sur un ensemble infini (en intention). Néanmoins ils se trouvent confrontés à des problèmes similaires, c'est-à-dire faire un choix parmi une multitude trop vaste pour être exhaustivement examinée.

Cette remarque nous a guidé vers la recherche d'une solution inspirée du monde du commerce électronique. Depuis quelques années, de plus en plus de sites web marchands se dotent de systèmes de recommandation. Ceux-ci doivent faciliter la tâche de l'acheteur en lui proposant des produits qui sont sensés l'intéresser particulièrement. Plusieurs types de profilages utilisateur ont été proposés afin de réaliser ces recommandations. On peut trouver des exemples de sites web proposant des services de recommandation, ainsi qu'une taxonomie des systèmes de recommandation utilisés dans [SKR99]. La plupart des recommandeurs nécessitent que l'utilisateur donne des informations sur ses préférences de manière explicite. Dans tous les cas, les profils sont raffinés à travers les choix des utilisateurs. C'est l'approche que nous avons suivie pour résoudre le problème de l'automatisation de la configuration d'impression. Dans notre optique, la politique d'impression tient lieu de profil utilisateur et les configurations - éventuellement manuelles - des paramètres d'impression des différents documents représentent les choix de l'imprimeur. La configuration de la politique d'impression est la conséquence d'un apprentissage automatique des préférences de l'imprimeur, au fur et à mesure de ses choix.

Remarquons bien que l'on ne se place pas, ici, dans une problématique d'aide à la décision "classique". L'imprimeur qui veut configurer les paramètres d'impression d'un ensemble de documents est supposé avoir les idées claires sur le résultat qu'il veut obtenir. Son problème n'est donc pas de cerner ses préférences par rapport à la configuration de tel ou tel document mais bien de les configurer tous. Il est en situation de décisions multicritères répétées. Celles-ci n'étant déjà pas triviales prises individuellement, la répétition du processus de décision, c'est-à-dire de l'expression de ses préférences à travers l'interface est particulièrement coûteuse en temps, en effort et en risque d'erreur.

1.2.2 Systèmes de recommandation

Dans cette section nous introduisons les deux grandes familles de systèmes de recommandation et justifions notre intérêt pour les systèmes basés sur le contenu. Nous présentons certains de ces systèmes, appliqués à des tâches diverses. Nous discutons de leur adéquation à notre problème, plus particulièrement du point de vue de l'interaction.

Recommandation basée sur le contenu

Dans le domaine des systèmes de recommandation, deux tendances principales et complémentaires ont émergé :

- Les systèmes à base de *filtrage collaboratif*. Ces systèmes agrègent des informations de préférence de plusieurs utilisateurs. Ils basent leurs recommandations sur la similarité entre profils utilisateur. Sont recommandés des objets que d'autres utilisateurs, qui ont des goûts en commun[2], ont choisis. A l'extrême, aucune caractéristique des objets n'est considérée, à part leur identifiant. Tout est basé sur les similarités de goûts entre utilisateurs. La pertinence de cette approche est conditionnée par un nombre d'utilisateurs conséquents. Or dans notre problématique, l'imprimeur est généralement le seul interlocuteur du système, ce qui limite l'utilisation du filtrage collaboratif.
- A l'inverse, les systèmes de recommandation dits "basés sur le contenu", englobant les systèmes dits basés "sur la connaissance", "sur l'utilité" et "sur les cas", considèrent uniquement les choix précédents de l'utilisateur à conseiller. De fait, il peut n'y avoir qu'un seul utilisateur, ce qui correspond à notre contrainte. Cette fois, les objets sont décrits par diverses caractéristiques, permettant de cibler les préférences. C'est l'approche que nous avons suivie.

Le lecteur intéressé par une classification plus fine des types de systèmes de recommandation, ainsi que d'une présentation des recommandeurs hybrides, trouvera matière dans [Bur02]. Dans notre contexte de recommandation, le défi est de se mettre "à la place de l'imprimeur" afin de lui recommander la configuration la plus pertinente pour chaque nouveau document, de manière à lui faire gagner du temps et des efforts. Cela nécessite d'apprendre ses préférences, au fur et à mesure de son interaction avec le système, et d'évaluer chaque solution potentielle pour recommander celle jugée la meilleure.

Les systèmes "FindMe"

Robin Burke et al. proposent, dans [BHY97, Bur00, Bur99], un système de recommandation générique nommé Recommender Personal Shopper (RPS), basé sur la connaissance des utilisateurs et des produits. Ce système a été appliqué à différentes problématiques concrètes comme la recommandation de voitures avec Car Navigator, de films avec PickAFlick, d'appartement avec RentMe, ou de restaurants avec Entree.

Le principe de l'interaction de ce système de recommandation est le suivant : l'utilisateur choisit d'abord un objet, appelé "source", dans le catalogue et demande

[2] au sens d'une mesure de similarité

à voir d'autres objets "similaires" à l'objet source. Le calcul de cette "similarité globale" entre deux objets s'appuie sur une liste ordonnée de mesures de similarité. Ces mesures dépendent du domaine d'application. Les objets du catalogue sont alors ordonnés par similarité avec la source et les plus ressemblants lui sont proposés. L'utilisateur peut également spécifier des contraintes comme "moins cher que X" ou "plus grand que Y", permettant de filtrer l'ensemble des objets du catalogue susceptibles d'être ordonnés.

Les connaissances qui fondent la recommandation sont donc de deux sortes :

- connaissances des préférences de l'utilisateur, qui se divisent également en deux :
  - un objet "source" qui sert de modèle dans la recherche d'objet similaires. Les objets sont typiquement décrits par un vecteur de nombres entiers.
  - un ensemble de contraintes, servant de filtres, établit explicitement par l'utilisateur. Ces contraintes sont du type "Prix $<$ 100" et permettent de réduire la taille du catalogue que le système va ordonner.
- connaissance des produits. La connaissance des produits se traduit par l'adaptation d'une ou de plusieurs *stratégie d'ordonnancement*, c'est-à-dire des listes des mesures de similarité qui vont être utilisées pour ordonner les objets. Chaque liste est ordonnée : de la similarité jugée la plus importante à la moins. La réalisation de cette liste est l'adaptation principale du système générique de recommandation à un domaine particulier.

Les deux difficultés principales liées à l'application de cette méthode tiennent dans la détermination des mesures de similarité, ainsi que dans leur ordonnancement. Ces deux tâches sont dévolues au concepteur, qui doit donc ordonner les "objectifs" de sorte que l'ordonnancement reflète les préoccupations du plus grand nombre. Une fois paramétrées, les listes des métriques de similarité sont les mêmes pour tous les utilisateurs.

Dans notre cadre applicatif, il semble impossible de souscrire à l'hypothèse que tous les imprimeurs - ou même un seul - ont en permanence une même hiérarchie d'objectifs ; par exemple qu'ils considèrent tous et tout le temps le zoom d'abord, puis les marges, etc. De plus, la détermination manuelle de l'ordre des préoccupations est possible pour un petit nombre d'objectifs mais devient difficilement gérable lorsque leur nombre augmente.

L'agent décisionnel APT

L'agent décisionnel APT [SL01] est un système de recommandation dédié à la recherche d'appartement. L'utilisateur donne un certain nombre d'indications pour initialiser son profil, comme le prix maximum et le lieu. Le système lui propose alors

un échantillon d'appartements remplissant ces conditions. L'utilisateur peut détailler les caractéristiques de chaque appartement de l'échantillon et spécifier lesquelles sont importantes pour lui. Il possède à cet effet douze cases, six positives et six négatives, rangées par ordre d'importance. Il peut donc indiquer six caractéristiques (i.e. attribut et valeur d'attribut) particulièrement pertinentes pour qualifier un "bon" appartement et six autres pour qualifier un "mauvais" appartement.

Initialement, tous les attributs possèdent un poids, paramétré par les concepteurs. Certains attributs sont considérés comme "cruciaux", ce qui augmente leur poids. En spécifiant la pertinence, positive ou négative, d'une caractéristique, l'utilisateur modifie le poids de l'attribut, plus ou moins fortement suivant la case où il range cette caractéristique. Il peut également spécifier qu'un attribut, non initialement considéré comme crucial, l'est pour lui.

Enfin, il existe un autre mode pour renseigner le profil utilisateur, sur la base d'une comparaison de deux appartement. Lorsque l'utilisateur choisit l'appartement qu'il préfère, le système infère jusqu'à trois caractéristiques pertinentes de la façon suivante : sont retenues les caractéristiques qui sont uniques à l'appartement préféré et qui ne figurent pas déjà dans la liste des caractéristiques pertinentes. S'il y en a plusieurs, elles sont classées par ordre de poids, avec les attributs considérés "cruciaux" en priorité.

Il semble donc que le modèle des préférences de l'utilisateur tienne en trois parts :

1. l'ensemble des contraintes initiales
2. la liste des attributs pondérés par les spécifications des caractéristiques pertinentes
3. l'historique de l'interaction

On voit que par rapport au système FindMe décrit au paragraphe précédent, l'agent décisionnel APT offre une souplesse supplémentaire en permettant à l'utilisateur de modifier les priorités sur les caractéristiques des solutions. Cependant l'interaction implique celui-ci dans le paramétrage de son propre modèle de préférence. Cela contredit notre hypothèse selon laquelle le modèle de préférence doit être totalement invisible pour l'imprimeur.

L'assistant de voyage automatisé

L'assistant de voyage automatisé (Automated Travel Assistant) est un système de recommandation exclusivement orienté sur les problèmes de sélection de vols [LHL97b]. Également basé sur une interaction de type "proposition/critique", le système ATA est cependant mieux expliqué par ses auteurs. Les préférences de l'utilisateur y sont représentées par une fonction d'utilité additive (voire 4.3.2). Cette

fonction d'utilité est à géométrie variable puisque l'utilisateur peut y ajouter ou enlever des critères, de même qu'il peut, et *doit*, ajuster lui-même les fonction d'utilité marginales, c'est-à-dire les fonctions donnant la préférence pour un attribut particulier. Ces modifications sont réalisées via une interface graphique dédiée.

A partir du modèle de préférence courant, le système recommande cinq alternatives dont trois représentent des "meilleurs compromis", au sens où ce sont des solutions non dominées et suffisamment différentes les unes des autres de manière à conserver une diversité permettant une exploration rapide des solutions ; et les deux dernières représentent des choix extrêmes, c'est-à-dire les meilleurs solutions sur les critères particuliers du prix (le vol le moins cher) et du nombre d'arrêt. Pour motiver cette politique de proposition, les auteurs avancent qu'elle permettrait de faire converger l'utilisateur rapidement en évitant qu'il ne tombe dans un "optimum local". Autrement dit, cela éviterait qu'un utilisateur ne se satisfasse d'une solution sous-optimale au vue de ses préférences, par manque d'exploration de l'espace des solutions. Les deux solutions extrêmes présentées sont supposées fournir des repères à l'utilisateur, de sorte qu'il connaisse les bornes inférieures atteignables pour ces deux attributs.

Le problème majeur de ce système réside dans l'ajustement du modèle de préférence (i.e. la fonction d'utilité), complètement dévolu à l'utilisateur et donc inenvisageable dans notre cas de figure. L'intérêt de l'approche nous semble plutôt tenir dans la stratégie de proposition qui cherche à maintenir une certaine diversité dans les recommandations.

1.3 Conclusion

Dans ce chapitre, nous avons présenté la problématique industrielle à la base de nos travaux : l'automatisation de la configuration des paramètres d'impression d'un document. Nous avons introduit les différents aspects de la configuration d'impression grand format (mise en page, qualité, finition et hétérogénéité) et montré l'intérêt industriel de son automatisation. Nous avons également décrit notre cadre applicatif : les logiciels autonomes de gestion d'impression, ainsi que les limites de l'approche actuelle d'automatisation, en terme de facilité d'utilisation et de maintenance.

Dans la seconde partie, nous avons présenté l'approche générale de nos travaux - améliorer l'utilisabilité grâce à un système de recommandation capable d'apprendre uniquement sur la base des choix précédents de l'imprimeur - et justifié notre point de vue quant à rendre invisible le modèle des préférences de l'utilisateur, c'est-à-dire la politique d'impression. Nous avons positionné notre approche par rapport par un

rapide état de l'art sur les systèmes de recommandations.

Chapitre 2

Étude d'utilisabilité

Le défi principal de nos travaux est l'amélioration de l'utilisabilité des logiciels permettant la configuration automatique d'impressions grand format. Dans ce chapitre, nous éclairons le concept d'utilisabilité et les méthodes permettant de l'évaluer. Après un état de l'art sur ces méthodes, nous proposons une approche adaptée à notre problème. Nous illustrons notre approche par une étude réalisée sur l'application d'Océ Publisher Engineering, présentée au chapitre précédent (voir p. 25).

2.1 Évaluer l'utilisabilité

2.1.1 Définition et méthodes

Définition de l'utilisabilité

Pour évaluer une production logicielle, que ce soit un site web ou une application, il est aujourd'hui commun de parler d'utilisabilité. Cette notion, décrite dans la norme ISO9241-11, décrit le potentiel d'un produit à être utilisé.

L'utilisabilité indique dans quelle mesure une interface permet à ses utilisateurs, dans un contexte donné d'utilisation, d'atteindre des objectifs spécifiques avec efficacité et rendement tout en instaurant un état de satisfaction. On observe trois critères principaux qui sont :

- L'efficacité (effectiveness) : jusqu'à quel point l'utilisateur atteint ses objectifs.
- Le rendement (efficiency) : les ressources nécessaires pour les atteindre.
- La satisfaction : déterminer si le système est agréable lorsque l'on réalise une tâche.

On trouve également dans la littérature [Nie93] deux autres paramètres de cette définition de l'utilisabilité :

- La sécurité : c'est le nombre d'erreurs commises par l'utilisateur, leur importance, et la rapidité de correction.
- La facilité d'apprentissage et de mémorisation : une compréhension et assimilation rapide et durable du mode de fonctionnement du système.

Parmi ces critères, la satisfaction est difficilement quantifiable car elle relève de l'appréciation personnelle de l'utilisateur mais aussi de l'importance qu'accorde l'expert en utilisabilité aux remarques de ce dernier.

L'étude de l'utilisabilité est donc l'ensemble des méthodes mises en œuvre pour mesurer l'utilisabilité d'une interface utilisateur et pour déterminer quels sont les problèmes spécifiques qu'elle comporte. Une étude est constituée de trois phases, la phase de recueil qui est la période où l'on amasse des données, par exemple l'enregistrement des réactions de l'utilisateur, le temps de réalisation d'une tâche, les erreurs, etc. Puis une phase d'analyse où s'effectue l'interprétation des données d'utilisabilité pour identifier les différents problèmes de l'interface. Et enfin la phase de critique qui se traduit par la suggestion de solutions ou d'améliorations pour amoindrir les problèmes.

Méthodes d'évaluation

L'évaluation de l'utilisabilité regroupe plusieurs méthodologies permettant de mesurer les divers aspects de l'utilisabilité d'une interface utilisateur et d'en identifier les problèmes spécifiques [Nie93]. L'évaluation de l'utilisabilité est une part importante du processus de conception d'une interface utilisateur, qui consiste en des cycles itératifs de conception, prototypage et évaluation [IH01]. Évaluer l'utilisabilité entraîne différentes activités, dépendant de la méthode employée. Ces activités sont couramment regroupées en trois classes.

- la *capture* : collecter des données d'utilisabilité, telles que le temps de complétion d'une tâche, les erreurs ou les estimations subjectives
- l'*analyse* : interpréter des données précédentes pour identifier les problèmes d'utilisabilité de l'interface
- la *critique* : suggestion de solutions ou d'améliorations pour soulager les problèmes.

Une grande gamme de techniques d'évaluation de l'utilisabilité a été proposée, et un sous-ensemble de celles-ci est toujours d'usage courant.

L'enquête Parmi les divers procédés utilisés dans les études d'utilisabilité, on trouve l'enquête d'utilisateur qui est le moyen le plus naturel de récolter l'information chez les clients. Par l'intermédiaire d'un entretien ou simplement par l'envoi d'un questionnaire, on récolte diverses informations, comme la description de l'em-

ployé, celle de l'entreprise où il exerce, le produit utilisé mais surtout son contexte d'utilisation. On emmagasine aussi les remarques et critiques du produit en plus des améliorations que l'utilisateur aimerait voir apparaître dans de futures versions. Cette méthode comporte deux inconvénients majeurs : un gros travail de filtrage et la subjectivité des propos recueillis. En effet, on se retrouve avec une quantité parfois importante d'informations inutiles ou anodines qu'il faut retirer pour ne conserver que les données exploitables pour l'analyse. De plus, que cela soit lors d'un entretien ou pour un questionnaire, les réponses sont souvent moins spontanées que celles obtenues lors d'une séance enregistrée, leurs valeurs relevant surtout de l'interprétation que leur donne l'ergonome qui les étudie.

L'inspection L'inspection désigne une famille de méthodes d'évaluation basées sur l'expérience de quelques experts en utilisabilité. Diverses listes d'heuristiques ont été mises au point pour faciliter la tâche aux experts [Nie94]. Il est reconnu qu'un petit groupe d'experts, de l'ordre de 4 ou 5, découvre environ 95% des problèmes d'utilisabilité d'une interface. Ce type de méthode requiert la présence d'expert en utilisabilité mais non d'utilisateur. Cette approche est efficace pour déterminer les défauts ponctuels d'une interface et les corriger. En revanche, elle n'est pas adaptée pour l'estimation quantitative de l'utilisabilité globale.

La modélisation analytique Certains systèmes permettent la modélisation analytique des utilisateurs et des interfaces, générant des prédictions quant aux temps requis pour effectuer telle tâche, ou au nombre de fois que la main de l'utilisateur devra passer du clavier à la souris, par exemple [JK94]. Ce type d'analyse a l'avantage de ne demander ni expert en utilisabilité, ni utilisateur cobaye mais il donne des résultats plutôt « bas niveau », pas toujours adaptés à notre besoin d'évaluation de l'utilisabilité. De plus, cette approche demande une modélisation manuelle parfois complexe.

La simulation Le pendant de la modélisation analytique est la simulation. Des modèles d'utilisateurs et d'interfaces sont utilisés pour simuler l'utilisation de l'application et donner des prédictions tout comme la modélisation analytique. La difficulté avec cette technique est qu'un modèle fiable de l'utilisateur n'est pas facile à mettre au point. Aucune approche n'a rencontré un véritable succès à ce jour et à notre connaissance.

Les tests utilisateurs Enfin, la dernière famille regroupe les méthodes sous l'appellation de test. Ces méthodes sont toutes basées sur l'expérimentation avec de

vrais utilisateurs, dans des conditions imposées, permettant de tester les performances et réactions des utilisateurs face à des problèmes particuliers. L'enregistrement des informations pendant une session de test peut prendre des formes diverses, qui conditionnent la phase d'analyse. Le moyen le moins coûteux d'enregistrer des informations pendant une session de test est sans doute la prise de notes manuelle. Cela implique la présence d'un observateur à côté des utilisateurs mais cette présence est de toute façon souvent requise afin d'expliquer les différentes tâches que devront accomplir les utilisateurs pendant le test. Un inconvénient plus subtil est la subjectivité et l'incomplétude des notes prises par l'observateur. Un autre support utilisé est l'enregistrement vidéo. Cette méthode nécessite éventuellement un expert supervisant la séance mais implique surtout un visionnage des bandes assez coûteux en terme temps.

Une troisième méthode permet de s'affranchir de ce type d'inconvénient : l 'enregistrement des évènements de l'interface. Les évènements de l'interface utilisateur (évènements UI) sont des "productions naturelles" des opérations effectuées sur un système muni d'une interface utilisateur basée sur les fenêtres graphiques. Ces évènements indiquent le comportement de l'utilisateur sur les composants qui constituent l'interface utilisateur d'une application (i.e. les clics de souris sur les boutons, menus ou listes, etc...). Puisque ces évènements peuvent être automatiquement capturés et qu'ils indiquent le comportement de l'utilisateur face à l'interface, ils ont longtemps été regardés comme une source d'informations potentiellement fertile, concernant les usages et l'utilisabilité. Cependant comme les évènements d'une interface utilisateur sont typiquement extrêmement volumineux et riches en détails, un support automatisé est généralement requis. Sans automatisation, l'extraction de l'information à un niveau d'abstraction qui est utile aux investigateurs devient rapidement impossible.

Jusqu'à présent, chez Océ, la plupart des tests d'utilisabilité repose soit sur des enquêtes auprès d'utilisateurs, soit sur des tests enregistrés. Le principe mis en œuvre est d'enregistrer fidèlement tout ce que l'utilisateur réalise durant la phase de test, grâce à un logiciel commercial. Cette technique est combinée à un enregistrement vidéo, par le biais d'une Webcam, qui permet d'enregistrer les réactions de l'utilisateur en direct. Une fois enregistrée, la séance de test peut alors être « rejouée » précisément. Cette méthode est coûteuse à la fois en logiciel et en temps de visionnement des enregistrements. De plus, elle ne fait que reproduire les actions de l'utilisateur et donc ne permet pas un traitement automatique des données récupérées.

Discussion Chaque technique a ses propres exigences et généralement, différentes techniques découvrent différents problèmes d'utilisabilité.

Comment peut-on atteindre la systématicité des résultats et la couverture entière

de l'estimation de l'utilisabilité ? Une solution est d'augmenter le nombre d'équipes d'utilisabilité évaluant le système et d'augmenter le nombre de participants aux études. Une alternative est d'automatiser certains aspects de l'évaluation : capture, analyse ou critique.

Les fichiers d'interactions et les données qu'ils contiennent sont reconnus depuis longtemps comme une riche source d'information permettant d'identifier les problèmes d'utilisabilité et d'automatiser l'évaluation des interfaces utilisateurs.

De plus, avec l'enregistrement d'évènements, les saisies existent dans un format qui est facilement transportable vers d'autres, et par la suite, analysable par un ergonome après la session de tests. Il n'est pas nécessaire pour un observateur d'être présent durant la phase de test pour savoir ce que le sujet est en train de faire et comment il le réalise, cela pouvant être déduit des fichiers. Ceci ouvre de nouvelles possibilités d'évaluation d'utilisabilité à moindre coût. Cependant, il est clair que s'en tenir aux seuls enregistrements des actions de l'utilisateur ne suffit pas dans certains cas pour évaluer l'utilisabilité d'une interface, et d'autres formes de suivi et d'analyse devront être employées.

Malgré cela, nous avons choisi d'explorer l'approche reposant sur l'*enregistrement de logs d'interaction* car elle est la mieux adaptée à une analyse quantitative et est complémentaire des méthodes aujourd'hui employées par Océ. De plus, elle ne nécessite aucun expert en utilisabilité. Enfin, elle se base sur des données fournies par des utilisateurs réels, ce qui constitue une certaine garantie de l'intérêt des résultats.

Dans la section suivante, nous présentons les différents aspects de l'analyse de données d'interaction en vue de l'évaluation de l'utilisabilité.

Nous traitons de la récupération de ces données en annexe, où nous présentons divers moyen de récupérer les évènements de l'interface utilisateur et nous détaillons les différents types d'évènements dans une courte étude permettant de sélectionner un sous-ensemble à enregistrer.

2.1.2 Problématiques du traitement des données d'interaction

Beaucoup de techniques ont été développées pour exploiter les fichiers de logs d'évènements en vue d'évaluer l'utilisabilité. [HR00] classe ces techniques en sept catégories, correspondant à autant de problématiques. Nous les présentons ici brièvement.

1. *Synchroniser différentes sources.* Cette problématique concerne les moyens de lier des évènements temporellement précis d'un fichier de logs avec d'autres

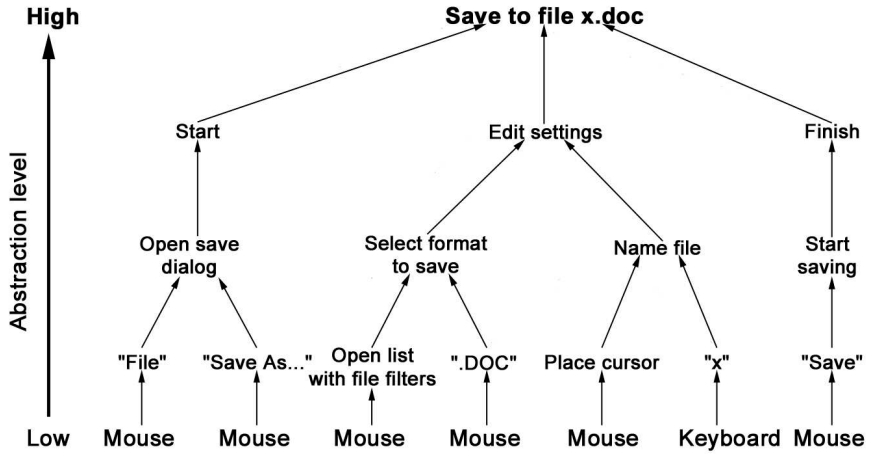


FIG. 2.1: Décomposition d'une interaction

sources : vidéo, bande son ou notes d'un observateur. Le but est de constituer une représentation la plus complète possible de la séance de tests.

2. *Transformer les logs en actions utilisateur.* Dans cette problématique, on veut transformer un flux d'évènements "bas niveau" vers en un flux "haut niveau", permettant de faciliter l'analyse et la détection des problèmes d'utilisabilité. Ceci est réalisé par la sélection de sous-ensemble d'évènements particulièrement pertinent, par un étiquetage sémantique et par le regroupement des évènements filtrés en actions utilisateur. La figure 2.1 illustre les différents niveaux d'abstraction pour une action donnée.

3. *Calculer les statistiques globales.* Calculer l'histogramme de l'utilisation des fonctionnalités, compter le nombre moyen des affichages d'erreurs par session, ou des consultations de l'aide, etc... Ces statistiques sont de bons indices d'utilisabilité qui peuvent, d'une part révéler la présence de problèmes et, d'autre part constituer les éléments d'une mesure globale objective de l'utilisabilité.

4. *Détecter les séquences*, ou comment identifier des séquences d'évènements "bas niveaux", potentiellement intéressantes pour l'expert en utilisabilité. En général couplée avec la comparaison de séquence (voir paragraphe suivant) elle peut également être employée pour transformer automatiquement le flux d'évènements en actions d'utilisateur (problématique n°2).

5. *Comparer deux séquences*, ou comment mettre en parallèle des traces d'évènements ou d'actions et détecter les différences avec une trace de référence. La comparaison entre un relevé défini comme « expert » et l'enregistrement d'un « débutant » permet de voir où se situent les difficultés majeures dans l'exécution d'une tâche.

6. *Classer les séquences.* Cette problématique concerne les moyens de caractériser automatiquement les séquences d'évènements. Utile pour détecter des modes d'utilisation (de type productif ou exploratoire par exemple), et pour transformer des données sur des évènements de bas niveau en actions de haut niveau.

7. *Visualiser.* Cet aspect concerne tous les problèmes de présentation des fichiers de logs ou des résultats d'une analyse. L'objectif est de faciliter l'interprétation des résultats par l'expert, par l'affichage de statistiques, des comparaisons des traces, de la localisation des clics souris, etc.

2.1.3 Systèmes d'analyse automatique

Les systèmes d'analyse des fichiers de logs automatisent l'extraction d'information pertinente à partir des données récupérées de façon implicite ou explicite. D'après [IH01], on peut distinguer trois approches générales pour l'analyse des logs :

- basée sur les statistiques
- basée sur les séquences
- basée sur les tâches

Nous présentons dans la suite des travaux portant sur l'analyse de fichiers de logs, uniquement dans le cadre d'études d'utilisabilité d'applications "de bureau". Bien que présentant des points communs avec notre problématique, nous ne parlons ici pas des systèmes d'analyse de fichiers de logs web. Le lecteur intéressé par le sujet peut se reporter par exemple aux récents travaux [MTT04, LLY08, DSLDCT06].

Basée sur les statistiques

Les méthodes basées sur les statistiques génèrent des mesures de performances quantitatives. Nous présentons dans la suite trois systèmes d'analyse de logs basés sur cette approche.

Le premier, DRUM [MR93], permet à l'évaluateur de revoir une vidéocassette du test d'utilisabilité et d'enregistrer manuellement les points de départ et de fin des tâches. DRUM traite alors les logs et en déduits plusieurs mesures, incluant : le temps de complétion d'une tâche, l'efficacité de l'utilisateur (i.e. la réussite divisée par le

temps de complétion de tâche), et les périodes productives (i.e. les portions de temps pendant lesquelles l'utilisateur ne rencontre pas de problème). DRUM synchronise également l'occurrence des évènements dans le log avec la vidéocassette, permettant d'accélérer l'analyse de la vidéo.

Le second système, MIKE [OH88], permet à un évaluateur d'estimer l'utilisabilité d'un modèle d'interface utilisateur, qui peut être rapidement modifié et compilé en une interface fonctionnelle. MIKE capture des données d'utilisation et génère un nombre de statistiques, incluant le temps des performances, la fréquence des commandes, le nombre d'opérations physiques requises pour compléter une tâche ainsi que les changement requis du focus de l'attention de l'utilisateur par l'écran. De plus, MIKE calcule séparément les statistiques concernant la sélection de commandes (e.g. parcourir un menu, taper le nom d'une commande ou cliquer sur un bouton) de celles concernant la spécification de commandes (e.g. entrer des arguments à une commande) afin d'aider l'évaluateur à localiser les problèmes spécifiques.

Le troisième système, AMME [Rau93], utilise les réseaux de Petri pour reconstruire et analyser le processus de résolution de problème de l'utilisateur. Cela nécessite un fichier log spécialement formaté et un fichier de description du système créé à la main (i.e. une liste des états de l'interface et une matrice de transition). Il mesure alors la complexité comportementale (i.e. étapes pour accomplir une tâche), la « routinisation » (i.e. l'utilisation répétitive de séquences d'actions), et le ratio entre la réflexion et le temps d'attente. Des études utilisateurs avec novices et experts ont validé ces mesures quantitatives et montré que la complexité comportementale est négativement corrélée avec l'apprentissage (i.e. plus l'utilisateur apprend, plus le nombre d'étapes pour réaliser une tâche diminue).

Basée sur les séquences

Les méthodes orientées extraction de séquences (pattern matching) telles que MRP (Maximum Repeating Pattern) [SH91], analysent le comportement de l'utilisateur capturé dans les logs. MRP détecte et rapporte les actions répétées de l'utilisateur (e.g. invocations consécutives des mêmes commandes et erreurs) qui pourraient indiquer des problèmes d'utilisabilité. Des études avec MRP ont montrées que cette technique peut être utile pour détecter des problèmes avec des utilisateurs experts, mais un filtrage additionnel des données est nécessaire pour la détection des problèmes avec des utilisateurs novices. L'aspect manuel ou automatique de ce filtrage n'est pas clair dans la littérature. Trois méthodes d'évaluation emploient l'extraction de séquences en conjonction avec des modèles de tâche. Nous discutons de ces méthodes dans la suite.

Basée sur les tâches

Les méthodes basées sur les tâches analysent les anomalies entre l'anticipation que font les concepteurs sur le modèle des tâches de l'utilisateur et ce que fait réellement un utilisateur in situ.

Le système IBOT [ZAD98] analyse automatiquement les fichiers de logs pour détecter les évènements liés à la complétion d'une tâche. Il interagit avec le système d'exploitation Windows afin de capturer des évènements bas niveau (e.g. action sur le clavier et la souris) ainsi que l'information des tampons d'écran (i.e. une image de l'écran qui peut être traitée pour identifier automatiquement les composants graphiques ou widgets). Le système combine alors ces données dans des abstractions de l'interface (e.g. sélection de menu et opération de recherche dans la barre de menu). L'évaluateur peut utiliser le système pour comparer le comportement prévu par les concepteurs avec celui de l'utilisateur réel sur ces tâches, afin de reconnaître des comportements inefficaces ou incorrects durant la complétion de tâche.

Les systèmes QUIP (Quantitative User Interface Profiling) [HL99] et KALDI [AQM99] fournissent une approche plus avancée dans l'analyse orientée tâche de fichiers de log issus d'interfaces utilisateur JAVA. Contrairement aux autres méthodes, QUIP agrège les traces de multiples interactions d'utilisateurs et compare les séquences d'actions répondant à une tâche particulière de ces utilisateurs avec la séquence prévue par les concepteurs. QUIP encode l'information quantitative relative au temps et aux traces dans des graphes orientés. La durée moyenne entre deux actions est indiquée par la couleur de chaque flèche, et la proportion des utilisateurs ayant réalisé une séquence particulière d'actions est indiquée par la largeur des flèches. La séquence d'actions prévue par les concepteurs est indiquée par une ombre hachurée (voir 2.2).

Dans les systèmes précédents, l'évaluateur doit programmer l'interface utilisateur pour collecter les données d'utilisation nécessaires, et doit analyser manuellement les graphes pour identifier les problèmes d'utilisabilité. KALDI récupère des données d'utilisation et des captures d'écran dans des applications JAVA. Il permet également à l'utilisateur de classer des tâche (soit manuellement soit via un filtrage automatique), de comparer les performances des utilisateurs sur ces tâches et de rejouer les captures d'écran synchronisés. Il décrit les logs graphiquement pour en faciliter l'analyse.

Basée sur les tâches et les séquences

Plusieurs systèmes basés sur les tâches incorporent l'extraction de séquences dans leur analyse. De cette combinaison résulte les analyses les plus avancées. Ces

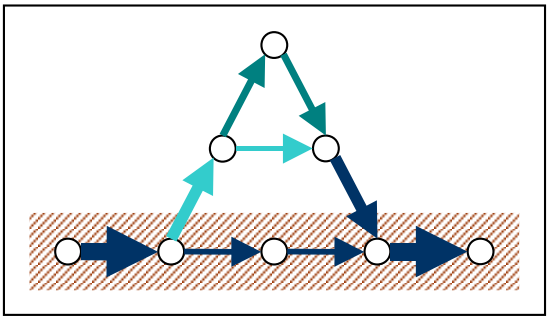


FIG. 2.2: chaque nœud représente une action utilisateur ; plus une flèche est grosse, plus nombreux sont les utilisateurs à effectuer cette séquence ; plus une flèche est sombre, plus le temps entre les deux actions est court.

systèmes incluent EMA et USINE.

EMA [Bal96] utilise un modèle de tâche créé manuellement et des heuristiques comportementales standard pour extraire les séquences d'utilisation qui peuvent indiquer un problème d'utilisabilité. EMA étend l'approche MRP par la détection de séquences additionnelles, tels que l'annulation immédiate d'une tâche, le changement de direction pendant la réalisation d'une tâche, ou les anomalies entre la complétion réelle et le modèle de tâche. EMA fournit des résultats sous la forme d'un fichier de logs annotés, que l'évaluateur doit inspecter manuellement pour identifier les problèmes d'utilisabilité.

USINE (user interface evaluator) [LPCC98] emploie la notation ConcurTaskTrees pour exprimer les relations temporelles entre tâches. A partir de cette information, USINE cherche les erreurs de préconditions (i.e. séquences de tâches qui violent les relations temporelles) et rapporte également des statistiques numériques (e.g. le temps de complétion de tâche), des informations sur les séquences de tâches, les tâches manquantes et les préférences de l'utilisateur reflétées par les données d'utilisation. Pour utiliser ce système, l'évaluateur doit créer des modèles de tâches en utilisant l'éditeur de ConcurTaskTrees, de même qu'une table spécifiant les relations entre les logs et les modèles de tâches. USINE traite les fichiers de logs et fournit des rapports détaillés accompagnés de graphes pour souligner les problèmes d'utilisabilité.

D'autres systèmes plus récents, comme [DR08], permettent de visualiser les parcours d'un ou plusieurs utilisateurs dans une interface. A partir d'un ensemble de tâches à réaliser, l'étude du comportement d'un groupe d'utilisateur peut ainsi mettre en évidence des séquences d'actions communes. Cette technique permet par exemple de caractériser le niveau d'expertise des utilisateurs.

Discussion

Bien que les systèmes présentés varient largement par rapport aux informations qu'ils fournissent, toutes ces approches offrent des avantages significatifs en regard de l'analyse manuelle des données d'utilisation, potentiellement très nombreuses. Les techniques hybrides basées sur les tâches et l'extraction des séquences, comme USINE, paraissent être les plus efficaces (i.e. fournissent les résultats les plus complets pour l'amélioration de l'utilisabilité). Elles requièrent néanmoins un effort supplémentaire et du temps d'apprentissage par rapport aux approches plus simples d'extraction de séquences. Cet effort supplémentaire est principalement lié au développement des modèles de tâche. Cependant, bien que les méthodes d'extraction de séquences soient plus simples à utiliser et à apprendre, elles permettent seulement la détection de problèmes liés à des séquences d'utilisation pré-spécifiés. Les méthodes basées sur les statistiques sont efficaces dans l'association des mesures avec des aspects particuliers de l'interface. AMME aide également l'évaluateur à comprendre le processus de résolution de problème de l'utilisateur et à conduire des études par simulation. Cependant, ces approches demandent à l'évaluateur d'effectuer plus d'analyses que les techniques basées sur les tâches, afin d'identifier la source des problèmes d'utilisabilité.

2.2 Étude préliminaire

Nous avons profité de la campagne de bêta-tests, organisée par Océ, sur l'application Publisher Engineering, présentée dans la section 1.1.2, pour récupérer des données d'utilisation grâce à une méthode inspirée de [Dam08]. Ces données sont une base pour mettre au point notre outil d'évaluation de l'utilisabilité basé sur les logs.

2.2.1 Sélection des évènements

Dans la sélection des évènements à prendre en compte, nous avons été confronté à deux critères contradictoires :

- maximiser l'information recueillie
- minimiser l'intrusion du processus d'enregistrement
  - pour le client, en terme d'impact sur la vitesse d'exécution de l'application, sur la taille du fichier de log généré, sur les informations recueillies et leur lisibilité
  - pour l'équipe de développement, en terme d'effort d'intégration et de risque de régression.

Conditions d'utilisation

Pour fixer les idées, le second critère, imposé par Océ, implique les conditions suivantes :

- pour le client
  - Transparence totale du processus d'enregistrement à l'exécution. Cette condition est peu contraignante, au prix toutefois d'un système supplémentaire de tampons, permettant de différer des écritures trop nombreuses dans le fichier de logs.
  - Fichier de logs ne dépassant pas 2 Mo. Cette condition est beaucoup plus contraignante. Le volume des données enregistrées peut croître exponentiellement avec les types d'évènements récupérés. Les évènements les plus gourmands, tels que ceux liés aux mouvements de la souris, ont été d'emblée écartés, car de trop bas niveau et trop gourmands en mémoire, donc négligeables et nuisibles. De plus, pour être certain de ne pas dépasser le volume convenu, nous avons dû introduire un mécanisme de blocage.
  - Aucune donnée personnelle enregistrée. Cette condition exige l'élimination de tous les évènements relatifs aux entrées clavier, aux noms de fichiers, etc.
  - Logs enregistrés dans un format directement lisible par le client. Cela impose le " non codage " des logs. Dans un sens, cela facilite le travail du programmeur mais l'inconvénient évident est que l'on ne peut pas compresser les données récupérées.
- pour l'équipe de développement
  - Interdiction de modifier le code source existant. L'instrumentalisation est donc interdite, ce qui ne pose pas de problème à notre solution.
  - Intégration au niveau "installation du logiciel"

Ces conditions ont entraîné une sélection a priori des évènements à récupérer. Nous avons fixé la capacité d'enregistrement de sessions d'utilisation à une centaine, soit 20Ko par session en moyenne. Partant de cette contrainte, nous avons testé plusieurs jeux d'évènements afin d'estimer le volume moyen généré par une utilisation "classique" pendant une session. Ces tests ont permis d'éliminer d'office certains évènements générant des volumes trop exigeants.

Une autre piste est d'éviter tout gaspillage d'espace, c'est-à-dire toute redondance d'information. Or beaucoup d'évènements sont redondants (pour le détail des évènements, voir l'annexe 177). Enfin, il a fallu garder un maximum d'information de sorte à être capable de "rejouer" une session le plus complètement possible.

Évènements récupérés

Nous avons choisi de récupérer trois types d'évènements : *ActionEvent*, *WindowEvent* et *ItemEvent*. Nous détaillons dans la suite ce qui a justifié ces choix et les premières limites qu'ils impliquent.

WindowEvent Dans un premier temps, il est intéressant de pouvoir quantifier le temps d'exécution de l'application. De ce fait, on est obligé de tenir compte des évènements du type *WindowEvent* qui offrent des informations sur l'affichage des différentes fenêtres. Cet évènement permet également d'observer la navigation entre les différentes fenêtres de l'application. Cela permet par exemple de détecter que l'utilisateur bascule sur une autre application (intra ou inter JVM). Cela permet également de lever l'ambiguïté sur certains évènements, par exemple produits par des boutons ayant le même label.

ActionEvent L'*ActionEvent* constitue aussi un évènement indispensable car de nombreux composants en génèrent lorsqu'ils sont sollicités. Avec ces évènements, on a déjà une quantité d'information qui permet de retracer une large partie des actions de l'utilisateur. Cependant leurs valeurs dépendent très fortement de l'instanciation des différents attributs qui a été faite lors du développement. On voit aussi apparaître la notion de bruit c'est-à-dire qu'une même action peut générer plusieurs fois un même type d'évènement ou des évènements très similaires. Par exemple, la confirmation de l'ouverture d'un fichier à travers une fenêtre dédiée entraîne l'envoi de deux évènements de type *ActionEvent* mais l'un est produit par le bouton "ok" et l'autre par la fenêtre. Ce phénomène est explicable par le fait qu'un composant peut être lui-même constitué d'autres composants. Cependant la quantité d'informations apportée par cet évènement est telle qu'il est indispensable à toute analyse approfondie.

ItemEvent Les *ItemEvent* sont une classe d'évènements utilisée par les menus, onglets, cases à cocher, listes déroulantes, boutons radio, etc. Ils viennent en complément des *ActionEvent* et permettent de voir la navigation au sain de l'interface. Les items couvrent aussi les actions relatives aux choix, donnant une information sur le comportement de l'utilisateur et sa façon d'utiliser l'application. Dans certains cas, on observe cependant un phénomène de redondance avec des évènements de type *ActionEvent*.

Évènements filtrés

Avec ces trois évènements, on dispose d'une source d'information importante, tant sur la plan qualitatif que quantitatif. Il est cependant nécessaire de justifier l'éviction de certains autres, qui constituent pourtant la majeure partie des évènements produits par une interface, comme ceux relatifs à la souris ou au clavier. Ces évènements, bien qu'importants, sont de trop bas niveau pour une analyse d'utilisabilité d'une application comme celle étudiée. Ils n'apportent pas de données très pertinentes. De plus, le moniteur d'évènements ne doit en aucun cas être trop intrusif pour l'utilisateur, en particulier concernant les informations privées. Relever les évènements clavier se limiterait alors à l'utilisation des raccourcis.

Une des lacunes est la non gestion des évènements relatifs aux arbres, aux tables et aux listes. Pour justifier cette décision, on peut noter que l'application étudiée utilise une structure arborescente non pas pour représenter des connaissances mais juste pour afficher le contenu des dossiers sur le disque. Il n'y a donc aucun intérêt à se focaliser sur ce type d'évènements.

On peut noter ici un point important dans la sélection a priori des évènements : l'adaptation aux particularités de l'interface considérée. De même ici, l'utilisation des *Popup* ou *ToolTip* est très réduite, rendant leur enregistrement peu intéressant. Mais dans une interface où leur emploi est nécessaire, une partie de l'analyse des fonctionnalités aurait été faussée par ce choix. Ceci nous amène à parler des autres évènements (voir l'annexe A pour une description plus complète des évènements de l'IHM fournis par l'API SWING) qui n'ont pas été utilisés :

- Le *ComponentEvent* nous donne des informations sur les modifications d'un composant, ceci peut être utilisé pour déceler des problèmes de mise en page dans l'application (en repérant, par exemple, les séquences où l'utilisateur passe du temps à redimensionner l'interface). Cependant il sera difficile de distinguer une séquence d'actions faite en réponse à un problème de ce type et une engendrée par la modification " ludique " de la fenêtre.
- L'*AdjustementEvent* souligne le même type de problèmes mais cette fois-ci concernant l'utilisation abusive des barres de défilement.
- Les évènements *AncestorEvent* et *ContainerEvent* n'offrent pas d'informations réelles sur le comportement de l'utilisateur puisqu'ils n'ont quasiment d'intérêt que pour le programmeur.
- *CaretEvent*, *DocumentEvent* et *TextEvent* ont des propriétés comparables puisqu'ils concernent des *TextComponent* (et plus particulièrement des objets de la classe *Document* pour *DocumentEvent*). Il est utile de les retenir pour une application orientée traitement de texte et dont la confidentialité n'est pas une contrainte.

- L'enregistrement des *InternalFrameEvent* est particulièrement indiqué pour les applications qui ont un espace de travail les utilisant. Dans le même esprit que les *WindowEvent*, ces évènements permettent de voir l'utilisation des différentes frames et d'en extraire des statistiques globales intéressantes.
- *MenuEvent* est un évènement qui devient vite redondant lorsqu'on utilise les *ItemEvent* (pour la navigation dans les menus) et les *ActionEvent* (pour la sélection d'un élément du menu).
- *UndoableEditEvent* est un évènement très intéressant pouvant révéler un problème de feedback dans l'application. Mais les outils java disponibles pour implanter le Undo/Redo sont encore trop peu utilisés dans les interfaces.
- *PropertyChangeEvent* est un évènement qui est généré à chaque fois qu'un des composants écoutés est modifié, survolé (bouton), activé, etc. Il implique beaucoup de redondance.

Après ce petit aperçu des avantages et inconvénients qu'apporterait l'enregistrement de chaque type d'évènements, on voit bien que le problème du filtrage a priori est situé sur la sélection des évènements non seulement utiles pour le type d'interface étudiée mais aussi du niveau d'abstraction sur lequel ils reposent.

Format d'une ligne de log

A chaque évènement enregistré, on associe les informations suivantes (voir la figure 2.3) :

- un numéro de ligne, qui est auto-incrémenté.
- un identifiant du poste sur lequel est utilisé l'application.
- un identifiant de session, également auto-incrémenté.
- un champ regroupant la date et l'heure (précision à la seconde).
- le type d'évènement, parmi les neufs suivants :
  - WINDOW_ACTIVATED
  - WINDOW_DEACTIVATED
  - WINDOW_OPENED
  - WINDOW_CLOSING
  - WINDOW_CLOSED
  - WINDOW_ICONIFIED
  - WINDOW_DEICONIFIED
  - ACTION
  - ITEM_STATE_CHANGED
- la description de l'évènement. Cette information est optionnelle et dépend de l'implémentation. Elle correspond par exemple au nom interne du bouton référencé par l'évènement.

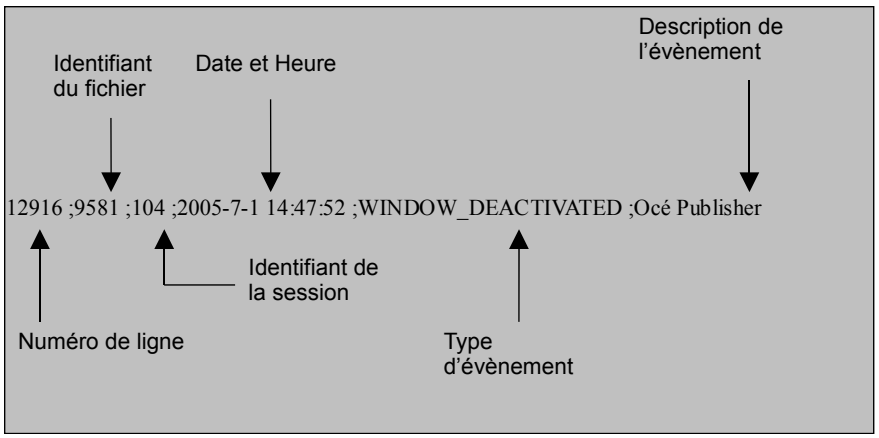


FIG. 2.3: Format d'une ligne de log

2.2.2 Premiers résultats

Nous avons récupéré trois fichiers de logs à l'issue des bêta-tests. Chaque fichier correspond à plusieurs mois d'utilisation du logiciel installé sur un poste d'un client. Sur les trois clients démarchés, deux ont une utilisation "corporate" du logiciel, c'est-à-dire qu'ils s'en servent comme un outil de support à leur travail, pour imprimer leurs plans de conceptions par exemple. Le dernier client concerné est d'un autre type : l'impression est son corps de métier ; l'utilisation du logiciel est donc plus intensive (fig. 2.4).

Comme nous l'avons mentionné plus haut, une première analyse a été réalisée sur les fichiers obtenus lors du bêta-test. Nous ignorons si plusieurs utilisateurs se sont partagés le même poste ou si toutes les traces obtenues dans un fichier sont le fruit d'un unique utilisateur. Les trois fichiers obtenus forment un tout de 26 400 lignes toutes exprimées dans le même formalisme (fig. 2.3).

Les premières analyses ont été réalisées grâce à un outil logiciel spécialement développé (voir l'annexe B). Elles ont pour objectif d'établir des statistiques globales d'utilisabilité, à partir de séquences d'évènements. L'étude se base sur la tâche de plus haut niveau : l'impression de documents. Le but est de détecter les séquences d'actions liées à une impression et d'en extraire l'information d'utilisabilité. Cette information tient en trois points :

1. le temps de travail effectivement passé à résoudre la tâche
2. le nombre de manipulations nécessaires
3. le nombre d'erreurs commises par impression réussie

Nous détaillons dans la suite l'analyse de chacun de ces trois critères.

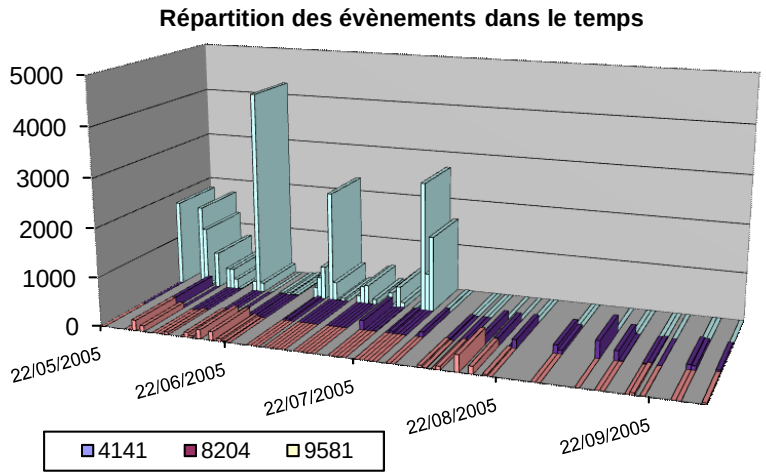


FIG. 2.4: Répartition des évènements dans le temps

Le temps de travail effectif

Du fait que l'enregistrement ne résulte pas d'une séance de tests encadrés, nous devons diviser le temps d'utilisation en trois parties :

- le temps d'activité où l'utilisation de l'application est effective
- le temps d'absence où l'utilisateur est passé sur une autre application, repérable par une désactivation immédiatement suivie d'une activation de la fenêtre de travail courante
- le temps d'inactivité où l'utilisateur reste sur l'application mais ne produit aucune action depuis un moment suffisamment long.

Afin de définir une période d'inactivité, nous avons choisi de fixer un seuil de temps de 5 minutes. Au-delà de cette durée, le temps entre deux évènements consécutifs est considéré comme une période d'inactivité (i.e. on considère que l'utilisateur fait autre chose). Nous distinguons ainsi les temps de " réflexion " (pause pour lire le texte à l'écran, etc.), des périodes où l'utilisateur est, par exemple, parti manger en laissant l'application ouverte.

Sur le diagramme (2.5), on peut voir l'importance des durées d'absence et d'inactivité. De telles durées peuvent être facilement expliquées par le fait que l'utilisateur se sert du logiciel in situ sur son lieu de travail, qu'il peut être dérangé à tout moment, qu'il peut aller manger ou qu'il peut " swapper " avec d'autres applications.

Le diagramme 2.6 montre le temps effectif moyen par impression réussie, par session. Nous définissons plus précisément ce qu'est une “impression réussie” dans notre contexte dans la section 2.2.2. Pour l'exemple donné, on peut calculer le temps

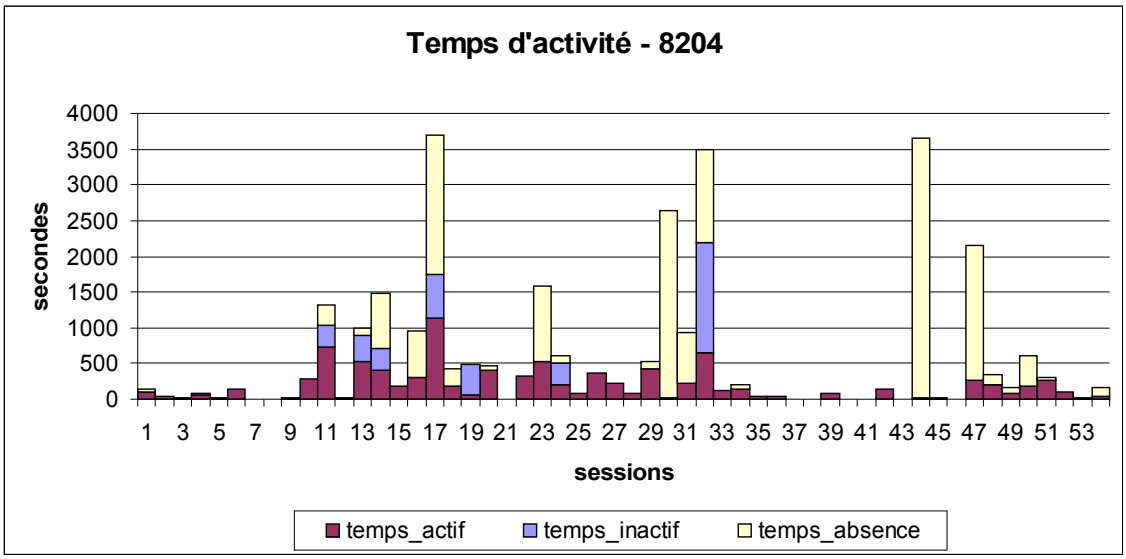


FIG. 2.5: Temps d'activité

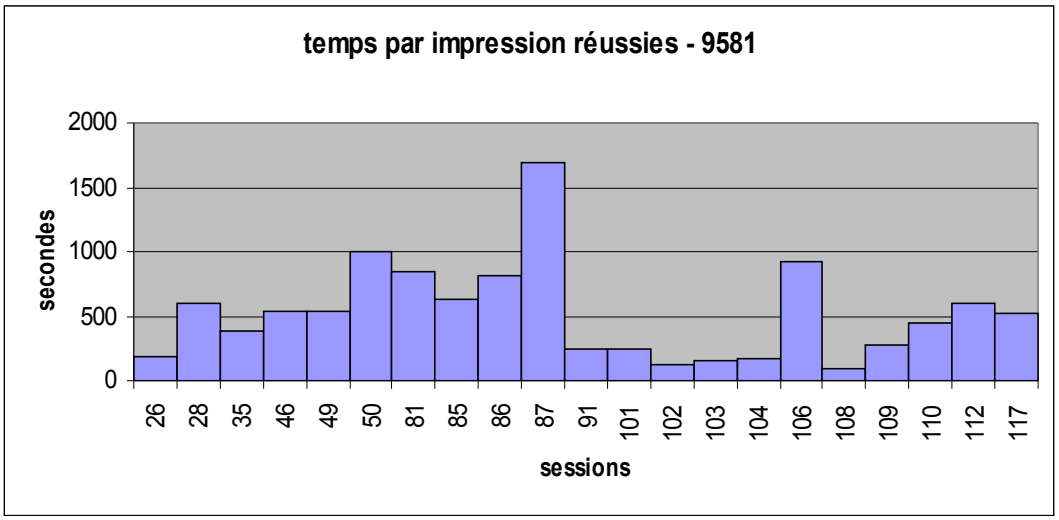


FIG. 2.6: Temps par impression réussie

effectif moyen d'une impression réussie, sur l'ensemble des sessions, qui est de 526 secondes, soit presque 10 minutes. Cette information est précieuse dans notre évaluation de l'utilisabilité.

Pour mémoire, on rappelle qu'une impression représente ici l'impression effective d'un groupe de documents et non d'un seul.

Le nombre de manipulations

Le nombre de manipulations est calculé après un filtrage des évènements enregistrés. En effet, la plupart d'entre eux ne désigne pas une nouvelle action de l'utilisateur.

On remarque sur la figure 2.7 des pics de manipulations. Ils correspondent en réalité à des évènements redondants qui n'ont pas été filtrés. Typiquement à la session 87, certaines actions, telles que l'appui prolongé sur un "compteur", peuvent générer de très nombreux évènements alors qu'en réalité, cela ne correspond qu'à une action de l'utilisateur. Cela fait partie des pièges à éviter pendant l'analyse des logs. En moyennant sur l'ensemble des sessions, on obtient un nombre de manipulations moyen par impression réussie de 186.

Le nombre d'erreurs

Lorsque l'on considère la tâche haut niveau "imprimer des documents", une erreur est synonyme d'impression ratée. Nous définissons une impression ratée par le

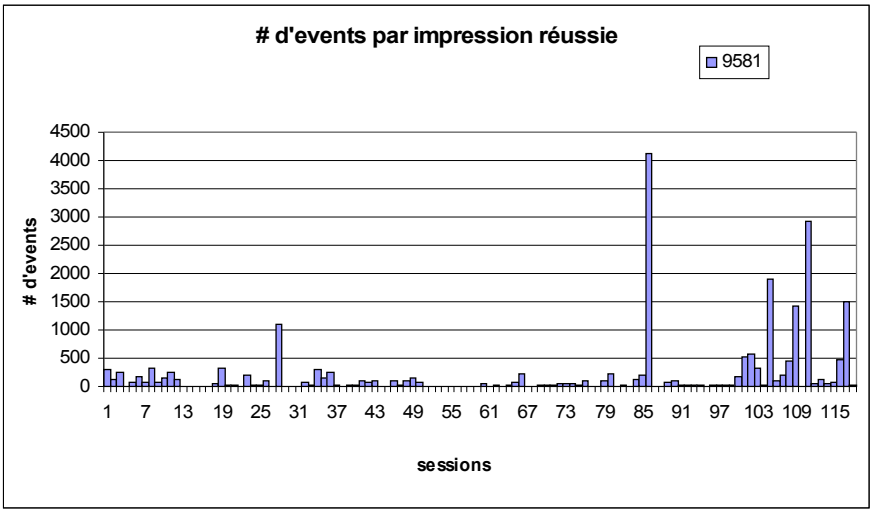


FIG. 2.7: Nombre d'évènements par impression réussie

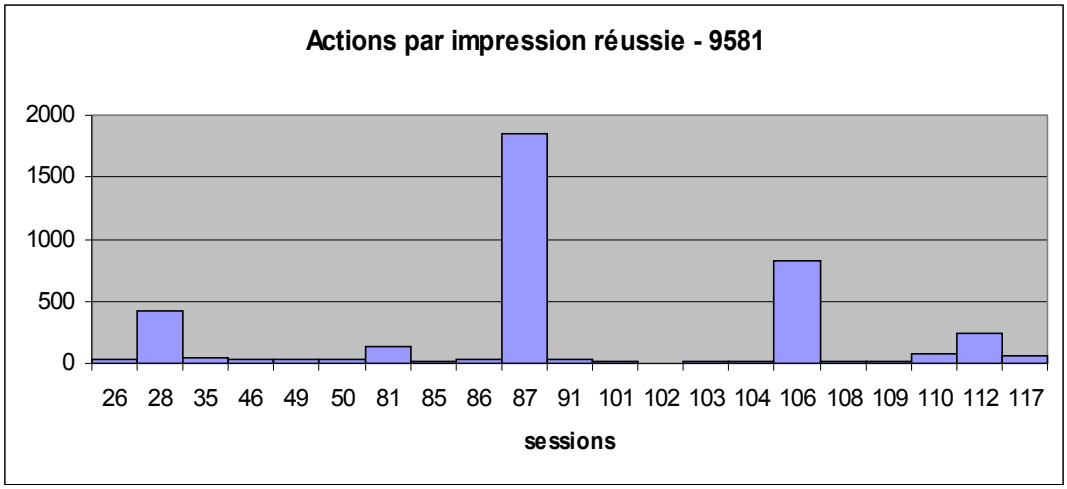


FIG. 2.8: Nombre d'actions par impression réussie (utilisateur 9581)

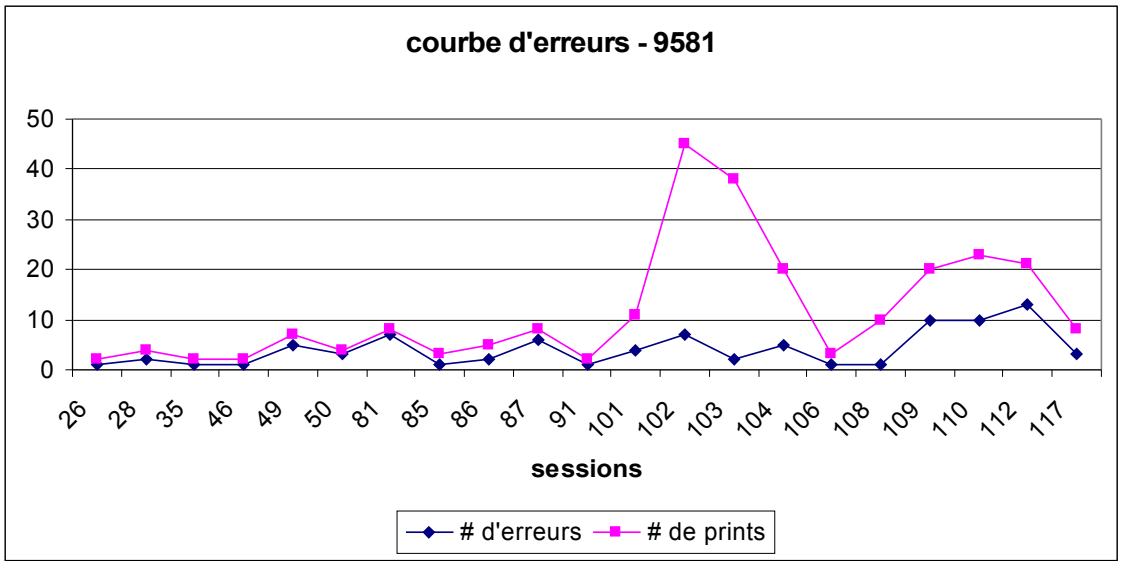


FIG. 2.9: Courbes comparées du nombre d'impressions et d'erreurs (utilisateur 9581)

critère suivant : lorsque l'utilisateur imprime deux fois de suite exactement le même ensemble de document, en modifiant les paramètres d'impression entre les deux.

Nous sommes cependant conscients que l'interprétation en terme d'erreur ne se justifie pas toujours. Par exemple, une telle séquence d'évènements peut transcrire un test de l'utilisateur qui va imprimer d'abord en qualité brouillon, avant d'imprimer en qualité supérieure. On peut interpréter ce cas d'au moins deux manières : soit l'utilisateur n'est pas sûr des réglages d'impression, soit il n'est pas sûr du rendu de l'image elle-même. C'est cette dernière option qui peut fausser notre interprétation en terme d'erreur, la première en faisant partie en tant que test. Nous considérons donc un bruit inhérent à ce genre d'observation.

On voit sur la figure 2.9 deux types d'utilisations. De la session 26 à 91, l'application a été testée, c'est une utilisation de type "exploratoire". Au contraire, entre les sessions 91 et 106, on assiste a une phase particulièrement "productive", caractérisée par beaucoup d'impressions et un taux "d'erreurs" faible par session.

Conclusions de l'analyse

Avec les données de l'exemple, pour une impression réussie on a :
- Taux d'erreur = 1,3
- Temps de travail = 10 minutes
- Nombre de manipulations = 180
- Nombre d'impressions par jour = 3,5

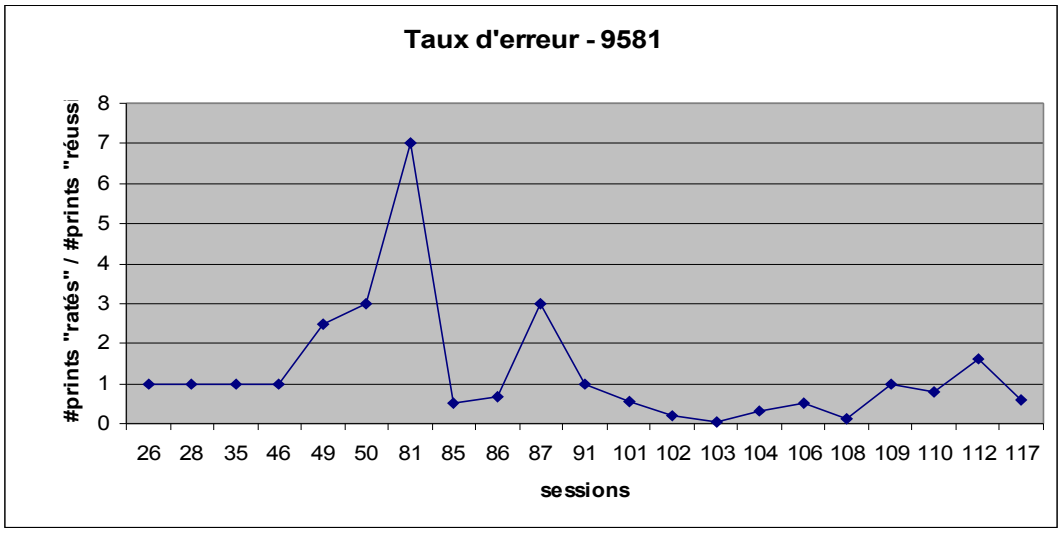


FIG. 2.10: Taux d'erreur (utilisateur 9581)

Ces premières statistiques - temps, manipulations, erreurs - nous permettent d'avoir une idée des ressources nécessaires pour réussir la configuration d'une impression.

Afin d'évaluer plus complètement l'utilisabilité de cette application, il nous manque les informations concernant le degré de réalisation de chaque impression. Cette données n'est malheureusement pas accessible sans test adéquat.

2.3 Conclusion

Nous présentons dans ce chapitre la notion d'utilisabilité d'une application informatique et la palette de méthodes permettant de l'évaluer. Nous traitons particulièrement des techniques d'évaluation automatique, basées sur la récupération et l'analyse automatique de données d'interaction entre l'utilisateur et l'application.

Pour illustrer notre propos, nous avons mené une étude d'utilisabilité d'une application en phase bêta, développé par Océ. Nous en présentons les résultats, obtenus à l'aide d'un outil d'analyse automatique de logs, que nous avons développé. Notre approche s'inscrit dans une perspective complémentaire aux méthodes actuellement en usage chez Océ pour l'étude d'utilisabilité des futurs produits.

Ces travaux préliminaires illustrent la difficulté de la tâche de configuration d'impression et renforce la conviction du besoin de méthodes automatiques pour aider à la résolution de ce problème.

Chapitre 3

Configuration d'impression

Dans ce chapitre, nous analysons et modélisons l'espace des paramètres d'impression, afin de définir l'ensemble des configurations admissibles, connaissant les caractéristiques du document à imprimer, les contraintes et directives de l'imprimeur, ainsi que les ressources d'impression disponibles.

3.1 Les données du problème

Dans cette section, nous proposons une modélisation des données d'entrée du problème de configuration d'impression grand format. Nous regroupons ces données en trois catégories. La première regroupe les notions liées au contexte applicatif : la définition des formats standards et du seuil de tolérance d'une part, les caractéristiques du document à imprimer de l'autre. Les deux premières données restent généralement constantes, après une première initialisation, contrairement au document à imprimer, qui change à chaque fois. Les données reflétant les desiderata de l'imprimeur sont classées dans la seconde catégorie. Elles regroupent le zoom désiré et les marges dites "utilisateur". Enfin la dernière catégorie abrite les données représentant les ressources de l'imprimeur, c'est-à-dire les sources de média disponibles et les contraintes associées.

3.1.1 Contexte applicatif

Formats

Les principaux formats d'impression standards européens, moyens et grands, sont les suivants : A4, A3, A2, A1, A0. On désigne communément comme "petit format" tout média pouvant être contenu dans une feuille A4, la feuille de bureautique standard en Europe. Au-delà, une impression est considérée "moyen ou grand format".

Le très grand format commence dès que la plus petite dimension du média dépasse la plus petite dimension du standard A0. Notre problématique concerne l'impression en moyen et grand format, c'est-à-dire les médias dont au moins l'une des dimension est plus grande que le plus petit côté du A4 et dont la plus petite dimension est inférieure au petit côté du A0. Dans la suite de ce document, nous désignons l'ensemble de ces médias sous l'appellation "grand format" par souci de concision. Cela englobe les médias aux formats standards, du A4 au A0, mais aussi des médias de formats non standards. Les dimensions des formats standards sont récapitulées dans la figure 3.1. Elle illustre le rapport entre les formats : le A4 est un demi A3, le A3 est un demi A2, etc.

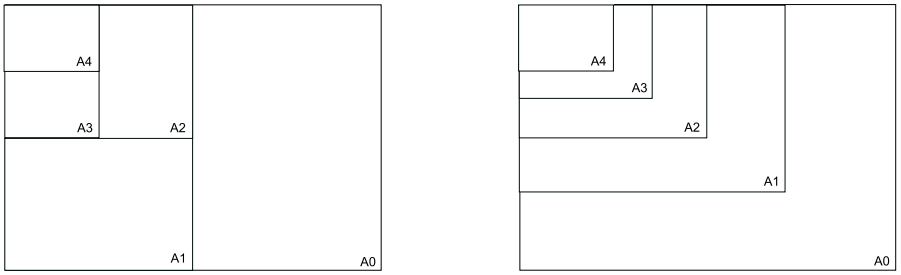
Nous définissons l'ensemble des formats standards FS, comme un ensemble de triplets $(nom, grand\,cote, petit\,cote) \in NS \times \mathbb{N} \times \mathbb{N}$, où NS désigne l'ensemble des noms de formats standards, *grand cote* et *petit cote* désignent respectivement la taille du plus grand et du plus petit côté du format. En général, cet ensemble reste constant pour un imprimeur donné. Nous considérons dans la suite l'ensemble des formats standards suivants (tailles données en mm) :

$$FS = \{(A0, 1189, 841), (A1, 841, 594), (A2, 594, 420), (A3, 420, 297), (A4, 297, 210)\}$$

Au contraire d'une impression petit format, une impression grand format peut être réalisée sur des médias de formats non standards. En effet, les grands médias sont souvent stockés sous forme de *bobine* ou *rouleau*. Dans ce cas, la dimension du média imprimé correspondant à la largeur du rouleau est standard, alors que l'autre dimension est variable. On peut donc imprimer des médias de formats standards ou non, selon les besoins. Notons qu'un rouleau permet d'obtenir deux formats standards différents. Par exemple la largeur d'un rouleau A3 correspond à la plus grande dimension du A4 et à la plus petite dimension du A3, on peut donc imprimer des médias aux formats standards A4 et A3 avec un rouleau A3. Et de même pour toutes les tailles de rouleaux standardisés.

	Hauteur	Largeur
A4	210	297
A3	297	420
A2	420	594
A1	594	841
A0	841	1189

TAB. 3.1: Dimensions des standards européens en orientation paysage (en mm)



(a) Orientation portrait / paysage alternés

(b) Orientation paysage

FIG. 3.1: Formats standards européens

Seuil de tolérance

Comme son nom l'indique, ce paramètre, noté $s \in \mathbb{R}_*^+$, définit un seuil de tolérance sur la taille des coupes et marges éventuelles, pour les stratégies qui les interdisent normalement. Il est particulièrement utile pour la gestion des marges matérielles. En effet, lorsque celles-ci sont traduites en marges supplémentaires, les dimensions des médias s'en trouvent réduites. Le seuil permet de tolérer les petites coupes correspondantes lorsque l'on veut imprimer une image au format standard.

Le seuil concerne chaque côté de l'image. Par exemple, un seuil de 3mm signifie que l'on tolère jusqu'à 3mm de marge (ou de coupe), de chaque côté de l'image.

Fixé par l'administrateur du système à une valeur bornant la taille des marges matérielles, il reste généralement constant, de même que l'ensemble des formats standards supportés.

Document à imprimer

Nous supposons dans le reste de ce manuscrit qu'un document n'est composé que d'une page unique. C'est pourquoi nous emploierons indistinctement les mots "document" et "image". Dans notre problématique de mise en page et de finition, nous négligeons complètement le contenu des documents, mise à part la position d'un éventuel cartouche (présent sur les dessins techniques, par exemple). Nous définissons donc un document par ses dimensions : hauteur h et largeur l, ainsi que par la position p de l'éventuel cartouche. Nous considérons les dimensions comme des valeurs réelles strictement positives. Le cartouche, s'il existe, est obligatoirement accolé à un coin de l'image. Par convention, la variable de position prend ses valeurs dans l'ensemble $\{0...4\}$ avec les significations suivantes :

$p = 0$, pour une image sans cartouche

$p = 1$, pour un cartouche en haut à gauche

$p = 2$, pour un cartouche en bas à gauche
$p = 3$, pour un cartouche en bas à droite
$p = 4$, pour un cartouche en haut à droite
Soit D l'ensemble des documents potentiellement imprimables, on a :

$$D = \left\{ (l, h, p) \in \left(\mathbb{R}_*^+\right)^2 \times \{0, \ldots, 4\} \right\}$$

Dans la suite, on supposera toujours avoir accès à ces informations, quelle que soit la façon de les récupérer[1]. Le triplet représentant le document à imprimer forme la première entrée de notre problème de configuration, la seconde étant l'ensemble des médias disponibles.

3.1.2 Directives de l'imprimeur

Zoom désiré

Par défaut, le zoom désiré par l'imprimeur, noté $z^* \in Z^* = [0.01, 100]$, est fixé à 1. Mais il peut également prendre une autre valeur, généralement prise entre 1% et 10000%. Le zoom final obtenu dépendra cependant de la politique d'impression utilisée. Ainsi une stratégie d'ajustement de l'image au média paramétrée sur <Exact> garantira que le zoom obtenu sera bien le zoom désiré ; alors que le paramétrage <Ajuster l'image> fournira éventuellement une autre valeur de zoom au final. Toutefois, même avec ce dernier paramétrage, la spécification d'une valeur de zoom particulière n'est pas sans conséquence. En effet, elle influence par exemple le choix du média. Le lecteur trouvera plus de détails sur l'influence du zoom désiré à la section 4.1.1.

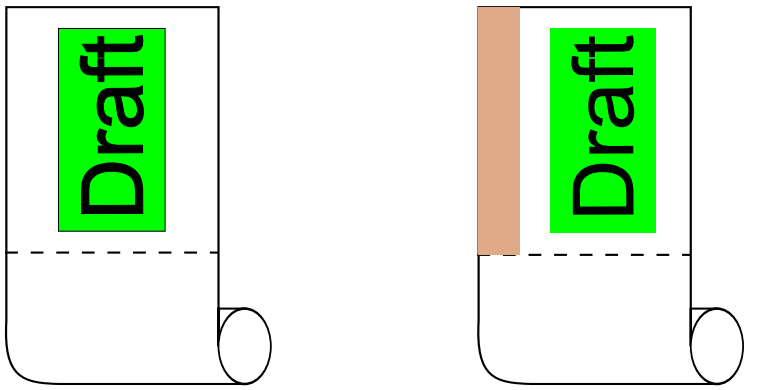
Marges supplémentaires

Les marges supplémentaires, notées

$$g^{sup} = \left(g_{\Uparrow}^{sup}, g_{\Leftarrow}^{sup}, g_{\Downarrow}^{sup}, g_{\Rightarrow}^{sup}\right) \in G^{sup} = \left(\mathbb{R}^+ \cup \{auto\}\right)^4$$

sont fournies par l'utilisateur. Elles sont exprimées par rapport à l'orientation de l'image. La valeur par défaut est <auto>, qui permet de faire coïncider les marges supplémentaires avec les marges matérielles d'une source de média, en fonction de la rotation qu'a subie l'image.

[1] Il existe deux manières standard de procéder : soit on déduit la position du cartouche d'après le format et l'orientation du document ainsi que d'après la norme qu'il est sensé suivre (AFNOR, STANDARD ou ERICSSON), soit on effectue une reconnaissance du contenu de l'image.



(a) Impression sans marge supplémentaire

(b) Impression avec marge supplémentaire

FIG. 3.2: Comportement lors d'un ajout de marge supplémentaire

Lorsque l'imprimeur veut ajouter des marges supplémentaires autour de l'image, tout se passe comme s'il réduisait les dimensions du média sélectionné. Par exemple, si l'imprimeur veut ajouter 10cm de marge supplémentaire au-dessus de l'image et que celle-ci est tournée de 90°, le comportement attendu revient à réduire la largeur du média sélectionné de 10cm, à résoudre le problème avec cette nouvelle dimension et à reporter le résultat sur la véritable dimension du média, ce qui aura pour effet de décaler l'image de 10cm vers la droite du média. La figure 3.2 illustre cet exemple avec une image en position centrée.

Marges d'extrémités

Les marges d'extrémités, notées

$$g^{ext} = \left(g^{ext}_{\Uparrow}, g^{ext}_{\Downarrow}\right) \in G^{ext} = \left(\mathbb{R}^{+}\right)^{2}$$

prévues à des fins d'archivage, se gèrent sur le même principe que les marges supplémentaires, à ceci près qu'elles se placent relativement au média et non à l'image. Appliquées historiquement aux médias issus de rouleaux, les marges d'extrémités ne concernent que le bord haut ou le bord bas d'un média, en aucun cas ses bords gauche ou droit. La figure 3.3 montre deux configurations d'impression d'une même image, présentant la même marge d'extrémité à la fin du média, avec des angles de rotation différents.

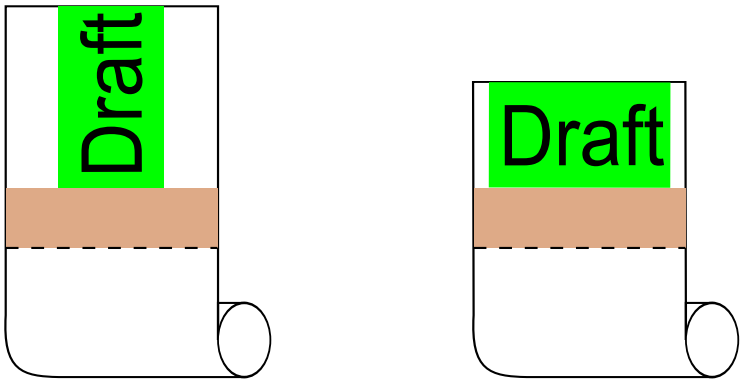


FIG. 3.3: Une même marge d'extrémité pour deux angles de rotation

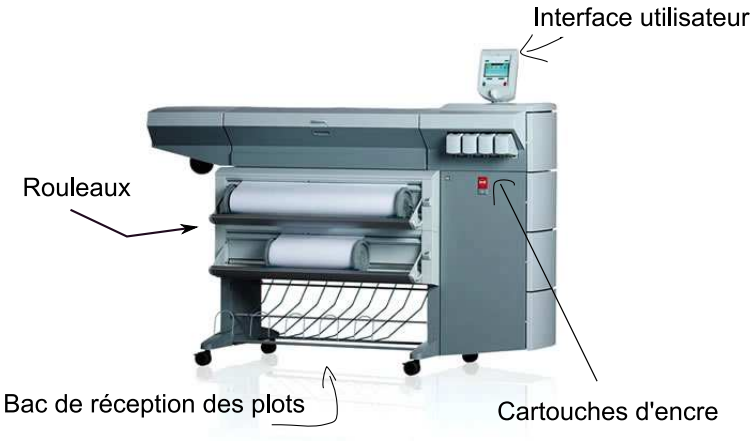


FIG. 3.4: Imprimante Océ TCS-300

3.1.3 Ressources du système d'impression

Un professionnel de l'impression a en général à sa disposition un parc d'imprimantes et de scanners. Chaque imprimante a des capacités propres telles que les formats supportés, la possibilité d'imprimer en couleur ou pas, la résolution maximale, la qualité des rendus, la taille de la découpe minimale et maximale en cas de rouleaux, la capacité de pliage, etc.

Une imprimante grand format peut stocker ses médias dans des tiroirs, sous forme de feuilles de formats standards, ou dans des rouleaux, de largeurs également standardisées. Elle peut également disposer de plusieurs sources de papier ou média (nous employons dans la suite les termes indistinctement), tiroirs ou rouleaux 3.4. Bien que l'utilisation de rouleaux ajoute beaucoup de souplesse dans le choix du format d'un média, cela induit également des contraintes qu'il faut connaître pour imprimer correctement. Ces contraintes sont détaillées dans les paragraphes suivants.

L'ensemble des sources de média disponibles représente le contexte de travail de l'imprimeur. Une source de média est représentée par les caractéristiques des médias qu'elle peut fournir. On considère les attributs suivants : hauteur, largeur, qualité, couleur et poids. Bien que faisant partie des critères de choix du média, nous avons décidé de ne pas prendre en compte les attributs de qualité, de couleur et de poids des médias. Ces attributs ne concernent directement ni la mise en page, ni la finition de l'impression. De plus, ils ne sont pas toujours renseignés, c'est-à-dire que le système ne peut pas systématiquement avoir connaissance de ces informations, suivant les possibilités de communication des imprimantes. Parmi ces caractéristiques, nous focaliserons donc uniquement sur les dimensions des médias.

Les désignations "largeur" et "hauteur" pour les dimensions du média sont purement conventionnelles. Dans la suite de ce document, nous considérons la largeur du média comme la dimension parallèle à l'imprimante. Dans le cas d'un média issu d'un rouleau, la largeur du média correspond à la largeur du rouleau ; elle est donc constante, contrairement à la hauteur qui varie de la découpe minimale à la découpe maximale prévue par l'imprimante.

Certaines imprimantes peuvent se voir ajouter une fonctionnalité intéressante : un plieur. C'est généralement un module mécanique optionnel et attenant à l'imprimante elle-même. Néanmoins, une fois reliées, ces deux machines forment un système d'impression unique et nous parlerons de capacité de pliage du système d'impression ou, par abus de langage, de capacité de pliage d'une imprimante. Plier un imprimé s'avère particulièrement pertinent dans le grand format. Il est en effet plus pratique de transporter et de stocker un imprimé au format A4 qu'au format A0. Dans le grand format, il existe deux types principaux de pliage : le "pli simple" et le "pli complet". Le pli simple est un pliage de la feuille en accordéon. Le pli complet plie d'abord en accordéon puis replie dans l'autre sens.

Nous ajoutons aux caractéristiques définissant une source de média, les possibilités de finition. Par finition nous entendons le pliage et nous emploierons un terme ou l'autre indifféremment dans la suite de ce mémoire. Sur la plupart des plieurs, ces possibilités sont au nombre de quatre, en plus de la finition sans pliage, suivant que l'on considère un pliage simple ou double, et un cartouche placé en haut ou en bas du média. Dans tous les cas, le cartouche est supposé être "collé" au côté gauche du média. Par souci de simplification et sans perte de généralité, nous ne distinguerons pas dans la suite le pliage simple du pliage double. En effet, cette distinction n'apporte rien quant à la recherche d'une configuration préférée. Par convention, nous notons les différentes finitions de la manière suivante :

1. sans pliage
2. pliage en considérant le cartouche en bas à gauche du média

5.2 Folding conforming to type C for filing without retention, e.g. in boxes or wallets

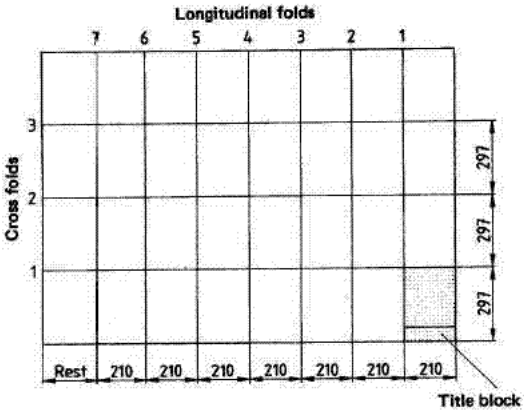
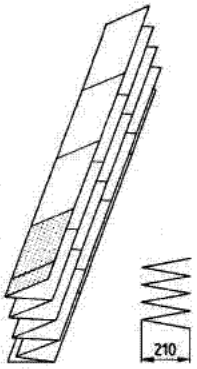
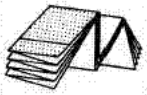
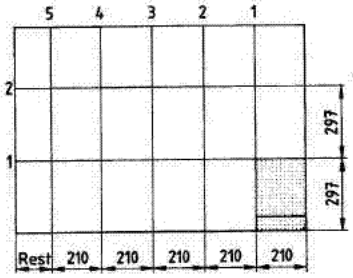
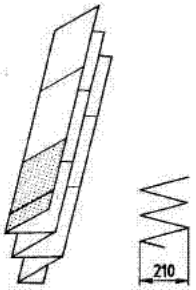
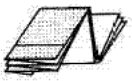
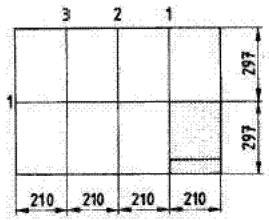
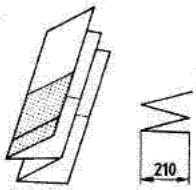
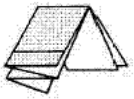
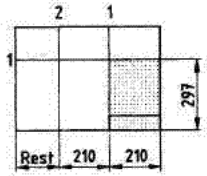
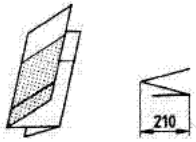
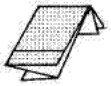
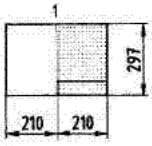
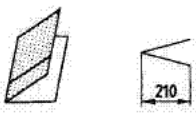
Format	Folding diagram	Longitudinal folding	Cross folding
2 A0 1189 × 1682			
A0 841 × 1189			
A1 594 × 841			
A2 420 × 594			
A3 297 × 420			

FIG. 3.5: Méthode de pliage à la norme STANDARD

3. pliage en considérant le cartouche en haut à gauche du média

Toutes les imprimantes ne possèdent pas de plieur et tous les plieurs ne proposent pas ces trois possibilités. Certains nécessitent un cartouche en bas du média par exemple. Soit $\mathcal{F}$ l'ensemble des possibilités de finitions possibles, associé à une source de média (sachant que la possibilité "sans pliage" est toujours présente), en se reportant à la numérotation ci-dessus, on a $\mathcal{F} = \{\{1\}, \{1,2\}\{1,3\}, \{1,2,3\}\}$.

Découpe minimale et maximale

En cas de média stocké sur rouleau, l'imprimante doit découper la feuille à la fin de l'impression. Cette découpe peut être réalisée arbitrairement, avec cependant deux contraintes à respecter :

- découpe minimale : c'est la longueur minimale en-dessous de laquelle l'imprimante ne peut pas découper. Cette contrainte est due au processus mécanique de guidage des feuilles dans les imprimantes. C'est donc une contrainte induite par le matériel et cette longueur de découpe minimale varie avec les modèles d'imprimantes. Généralement, la machine est réalisée de sorte que la découpe minimale corresponde au plus petit standard imprimable, du A4 par exemple.
- découpe maximale : longueur maximale avant découpe. Cette limitation peut être due à deux facteurs :
  - contrainte matérielle : lorsque l'on atteint la fin du rouleau.
  - contrainte logicielle : pour des raisons d'efficacité de stockage et de calcul, le nombre de bits alloués au codage des coordonnées des points de l'image peut être minimisé. En contrepartie, cela impose une longueur maximale correspondant à la plus grande valeur représentable par le codage, divisée par la résolution de l'imprimante. Par exemple, si les coordonnées sont codées sur 16 bits, la plus grande valeur entière positive représentable est 65535, ce qui, avec une résolution de 600dpi[2], nous amène à une longueur maximale : $L_{max} = \frac{65535}{600} \approx 110\,pouces \approx 280\,cm$. Les imprimantes Océ peuvent généralement imprimer jusqu'à environ 15 mètres de rouleau en une passe.

Marges matérielles

Une autre contrainte que l'imprimeur doit prendre en compte est l'ajout inévitable des marges matérielles. Les marges matérielles sont de fines bandes blanches (de quelques mm d'épaisseur) bordant obligatoirement tout imprimé. Elles permettent notamment de palier les imperfections inévitables du processus de guidage du papier lors de l'impression. En effet, si l'on imprime jusqu'aux extrêmes bords de la feuille,

[2]"Dot Per Inch" équivalent de "point par pouce" (PPP) où un pouce = 2,54cm

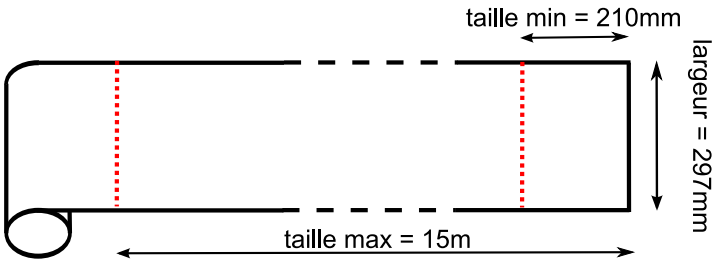


FIG. 3.6: Découpe min et max sur un rouleau A3

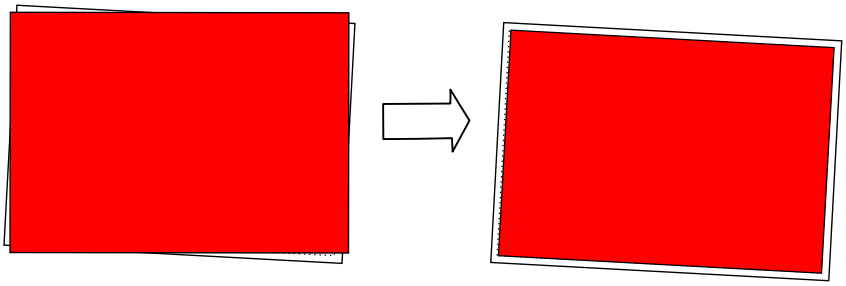


FIG. 3.7: Illustration de la nécessité des marges matérielles : en pointillé les marges, en rouge le document imprimé exagérément biaisé pour l'exemple.

le moindre écart de parallélisme entre la feuille et l'imprimante peut faire apparaître une marge blanche d'un côté pendant que de l'autre, de l'encre est crachée en dehors de la feuille, ce qui peut "salir" les impressions suivantes. Ces marges matérielles ont donc un rôle de "tampon". Elles s'avèrent particulièrement nécessaires pour les longs imprimés car les effets d'un écart de parallélisme sont proportionnels aux dimensions des feuilles.

Enfin nous complétons les caractéristiques d'une source de média par la donnée des marges matérielles

$$g^{mat} = \left(g^{mat}_{\Uparrow}, g^{mat}_{\Leftarrow}, g^{mat}_{\Downarrow}, g^{mat}_{\Rightarrow}\right) \in G^{mat} = \left(\mathbb{R}^{+}\right)^{4}$$

Celles-ci dépendent directement de l'imprimante. Elles sont exprimées en fonction du média : $g^{mat}_{\Uparrow}$ représente la marge matérielle située en haut du média, $g^{mat}_{\Leftarrow}$ représente celle à gauche du média, etc.

Source de média

Definition 2. On définit une *source de média* par un quintuplet $(L, H_{min}, H_{max}, F, g^{mat})$ tel que $L, H_{min}, H_{max} \in \mathbb{R}^{+}_{*}$, $H_{min} \leq H_{max}$, $F \in \mathcal{F}$ et $g^{mat} \in G^{mat}$.

Note. Pour une source de média $(L, H_{min}, H_{max}, F, g^{mat})$ correspondant à un tiroir

de feuilles, on a $H_{min} = H_{max}$. Dans le cas d'un rouleau, L désigne la largeur du rouleau, H_{min} correspond à la découpe minimale de l'imprimante et H_{max} à sa découpe maximale. Dans tous les cas, F désigne les possibilités de pliage associées à la source de média, et g^{mat} les marges matérielles associées à l'imprimante.

Soit S l'ensemble des sources de média, dans la suite on notera l'ensemble (fini) des sources de médias disponibles :

$$\hat{S} \in \boldsymbol{P}(S)$$

où $\boldsymbol{P}(S)$ désigne l'ensemble des parties de S.

Note. L'ensemble des sources de médias disponibles $\hat{S}$ représente les possibilités du système d'impression, en terme de mise en page et de finition. C'est le contexte de travail de l'imprimeur.

3.2 L'espace des configurations admissibles

Dans notre problématique, l'ensemble des alternatives possibles, c'est-à-dire l'ensemble des décisions ou solutions admissibles, est l'ensemble des impressions que l'on peut obtenir à partir d'un document dans un contexte particulier. Plus précisément, nous désignons ici sous le terme "impression" la mise en page ainsi que la finition. Nous omettons donc de cette définition tout ce qui touche au contenu imprimé, à la qualité du rendu, etc. Le contexte représente ici les possibilités du système d'impression : les imprimantes et médias disponibles avec leurs caractéristiques. A partir du contexte, nous pouvons définir la première caractéristique d'une impression : le média c'est-à-dire la feuille imprimée. Ensuite, vient la manière d'imprimer l'image sur le média : le positionnement et les transformations de l'image, rotation et zoom. Enfin le type de finition, c'est-à-dire le type de pliage.

Nous présentons ici la tâche de configuration d'une impression grand format en décrivant les différents paramètres que l'imprimeur doit configurer.

3.2.1 Support et finition

Source de média

Un professionnel de l'impression a en général à sa disposition un parc d'imprimantes et de scanners. Chaque imprimante a des capacités propres telles que les formats supportés, la possibilité d'imprimer en couleur ou pas, la résolution maximale, la qualité des rendus, la taille de la découpe minimale et maximale en cas de rouleaux, la capacité de pliage, etc. Le choix d'une imprimante est donc primordial.

Cependant les applications proposent parfois une abstraction d'imprimante nommée queue d'impression. Une queue d'impression représente une "imprimante virtuelle" vérifiant un ensemble de caractéristiques. Par exemple, l'administrateur du logiciel peut avoir configuré une queue d'impression en A0 qui simule une imprimante munie d'un rouleau A0 et qui, en réalité, permet d'imprimer sur plusieurs imprimantes possédant cette caractéristique.

Média

La seconde tâche à effectuer est de choisir le média sur lequel on veut imprimer. Cela ne pose en général aucun souci pour une impression en petit format car on imprime sur des feuilles A4 blanches et identiques quel que soit le tiroir d'où elles sont tirées. Dans le cas du grand format, cela peut se compliquer : si plusieurs supports sont disponibles sur l'imprimante considérée, on peut avoir le choix entre différents formats, du A3 au A0 par exemple. De plus, si le support est un rouleau, il se pose également la question de savoir quand couper la feuille. Par exemple, on peut la couper juste après l'image ou préférer une coupe correspondant à un format standard[3].

Nous définissons le média sélectionné m par sa largeur m_l, sa hauteur m_h et la finition effectuée dessus m_f.

$$m = (m_l, m_h, m_f) \in M = \mathbb{N}^2 \times \{1, 2, 3\}$$

Les définitions de la largeur, de la hauteur et de la finition suivent les conventions adoptées pour les sources de médias (voir 3.1.3). Autrement dit, si un média $m = (m_l, m_h, m_f)$ provient d'une source $s = (s_L, s_{H_{min}}, s_{H_{max}}, s_F, s_{g^{mat}})$ alors les contraintes suivantes doivent être vérifiées :

$$s_{H_{min}} \leq m_h \leq s_{H_{max}}$$

$$m_l = s_L$$

$$m_f \in s_F$$

On note $media\,(s, m) \in S \times M$ le prédicat qui vérifie cet ensemble de contraintes.

La sélection d'un média englobe donc la détermination de la découpe, dans le cas d'un média issu d'un rouleau, ainsi que du type de finition.

[3] même si l'image n'a pas une taille standard

Pour illustrer ce propos, supposons que l'on dispose de deux imprimantes. La première est munie d'un rouleau A0 et permet de plier les imprimés. La seconde est munie d'un rouleau A1 et ne permet pas le pliage. Nous avons donc les deux sources de médias s_1 et s_2 suivantes :

$$\begin{aligned} s_1 &= \left(2384, 515, 20000, \{1,2,3\}, g_1^{mat}\right) \\ s_2 &= \left(1684, 515, 20000, \{1\}, g_2^{mat}\right) \end{aligned}$$

Notre ensemble des sources de médias disponibles est $\hat{S} = \{s_1, s_2\}$. Dans la suite de l'exemple, considérons les médias de taille standard. Nous pouvons ainsi imprimer sur des médias aux formats standards A0, A1 et A2. Le format A1 peut être obtenu de deux manières différentes : soit dans la longueur du rouleau A1, soit dans la largeur du rouleau A0. En revanche, le format A0 ne peut être obtenu qu'à partir du rouleau A0, de même que le format A2, découpé dans la largeur du rouleau A1. On voit que l'on peut obtenir des imprimés pliés de formats A0 ou A1 mais que le A2 ne pourra pas être plié automatiquement. Plus formellement, soit $M_{\hat{S}}$ l'ensemble des médias disponible à partir des sources $\hat{S}$, on a :

$$\begin{aligned} (3370, 2384, i) &\in M_{\hat{S}}, \quad \forall i \in \{1 \ldots 3\} \\ (2384, 1684, i) &\in M_{\hat{S}}, \quad \forall i \in \{1 \ldots 3\} \\ (1684, 1191, 1) &\in M_{\hat{S}} \\ mais\ (1684, 1191, i) &\notin M_{\hat{S}}, \quad \forall i \in \{2,3\} \end{aligned}$$

Finition

Au-delà du choix sommaire du pli simple ou complet, il existe une subtilité qui rend la tâche de pliage parfois délicate : le cartouche. En effet, dans un dessin grand format - un dessin technique ou un plan d'architecture par exemple - il existe en général ce qu'on appelle un cartouche : un rectangle accolé à un coin du dessin et qui contient toutes les informations nécessaire à l'identification de l'image. Lorsque l'on plie ce genre de dessin, il est donc très intéressant de retrouver le cartouche sur la face visible après pliage. Cette simple remarque peut amener à des considérations ardues. En effet, il faut orienter et positionner l'image sur le média en conséquence.

3.2.2 Transformations

Une fois le média choisi, il faut positionner l'image dessus, choisir une valeur de zoom et de rotation. En général, le positionnement s'effectue sur l'une des neuf

positions, fruits du produit des trois positions sur l'axe vertical (haut, milieu, bas) et horizontal (gauche, milieu, droite). Le zoom peut varier de 1% à 10000% et la rotation s'effectue par tranche de 90°.

Rotation de l'image

L'image peut être tournée de quatre façons différentes sur le média. L'ensemble des rotations possibles R possède donc quatre valeurs, exprimées en degrés :

$$R = \{0, 90, 180, 270\}$$

Dans la suite, on notera $r \in R$ la rotation de l'image d'une configuration d'impression.

L'angle de rotation est exprimé par rapport à l'orientation du média. Par convention, le haut de l'image correspond au côté haut du média pour un angle de rotation de 0°, au côté gauche du média pour un angle de 90°, etc.

Nous allons voir qu'il est inutile de considérer toutes les valeurs de rotation pour définir l'ensemble des configurations possibles. En effet, en fonction de la position de l'éventuel cartouche, on distingue différentes symétries qui permettent d'éliminer deux cas sur quatre. Considérons les cas un par un.

cas sans cartouche : il est clair que les ensembles des solutions d'impression obtenus avec les angles de rotation de 0° et de 180° sont en tout point identiques. De même pour les ensembles engendrés par les rotation de 90° et 270°. Il suffit donc de considérer un seul des couples suivants, au choix : $(0, 90)$, $(0, 270)$, $(180, 90)$ ou $(180, 270)$. Dans la suite, nous considérons le couple $(0, 90)$.

cas avec un cartouche : pour tous les cas de pliage, il y a uniquement deux positions de l'image sur le média à considérer, correspondant au positionnement du cartouche, soit en bas à gauche du média, soit en haut à gauche. Suivant l'emplacement du cartouche sur l'image, la rotation est donc conditionnée.

- position 1 (sans cartouche) : on considère par convention les angles 0° et 90°.
- position 2 (cartouche en haut à gauche) : on doit considérer les angles 0° et 90°.
- position 3 (cartouche en bas à gauche) : on doit considérer les angles 0° et 270°.
- position 4 (cartouche en bas à droite) : on doit considérer les angles 180° et 270°.
- position 5 (cartouche en haut à droite) : on doit considérer les angles 90° et 180°.

Quels que soient les cas, il est suffisant de considérer deux angles de rotation sur

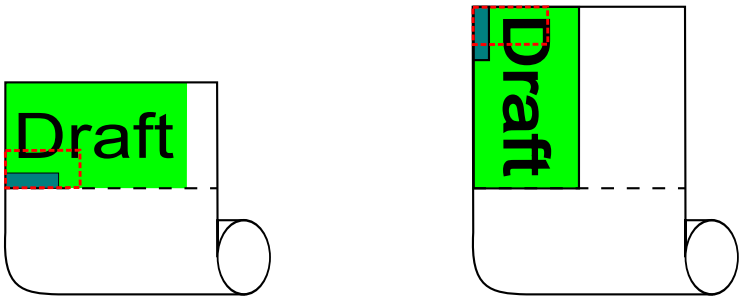


FIG. 3.8: Angles à considérer pour une image avec un cartouche en bas à gauche (position 3). En pointillés rouges la zone visible une fois le média plié.

quatre.

Zoom

Nous considérons le zoom comme une valeur continue bornée sur l'intervalle [0.01 , 100] ou, en pourcentage de 1% à 10000%. Ces bornes correspondent aux valeurs actuellement utilisées dans les applications d'Océ. On note Z l'ensemble des valeurs de zoom possibles :

$$Z = [0.01\,,\ 100]$$

Dans la suite, le zoom d'une configuration d'impression sera noté $z \in Z$.

Position

Afin d'exprimer la position de l'image sur le média, de façon invariante par rapport aux transformations de l'image, c'est-à-dire à la rotation et au zoom, nous définissons la position de l'image par les coordonnées de son centre, exprimées dans un repère orthonormé, ancré au centre du média. On note P l'ensemble des positions du centre de l'image par rapport au centre du média :

$$P = \left\{(x, y) \in \mathbb{R}^2\right\}$$

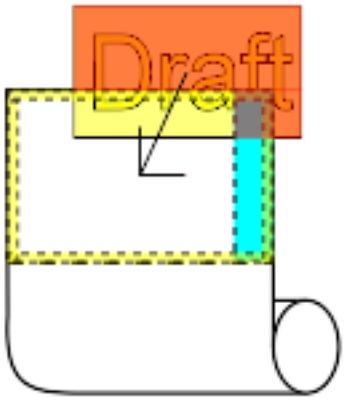


FIG. 3.9: Position

Une position $(x, y) \in P$ peut se déduire en fonction des autres caractéristiques de sorte que, pour un angle de rotation de 0° par exemple, on ait

$$x = \frac{1}{2}z \cdot d_l - \frac{1}{2}m_l + g_{\Leftarrow} - c_{\Leftarrow} \quad (3.1)$$

$$x = -\frac{1}{2}z \cdot d_l + \frac{1}{2}m_l - g_{\Rightarrow} + c_{\Rightarrow} \quad (3.2)$$

$$y = \frac{1}{2}z \cdot d_h - \frac{1}{2}m_h + g_{\Downarrow} - c_{\Downarrow} \quad (3.3)$$

$$y = -\frac{1}{2}z \cdot d_h + \frac{1}{2}m_h - g_{\Uparrow} + c_{\Uparrow} \quad (3.4)$$

où d_l et d_h désignent respectivement la largeur et hauteur du document à imprimer ; m_l et m_h celles du média et z le zoom appliqué.

De plus, en combinant les équations 3.1 et 3.2, ainsi que 3.3 et 3.4, on obtient le système d'équation suivant :

$$(3.1) - (3.2) : \quad zd_l - m_l + g_{\Leftarrow} + g_{\Rightarrow} - c_{\Leftarrow} - c_{\Rightarrow} = 0 \quad (3.5)$$

$$(3.3) - (3.4) : \quad zd_h - m_h + g_{\Uparrow} + g_{\Downarrow} - c_{\Uparrow} - c_{\Downarrow} = 0 \quad (3.6)$$

Ce système est identique pour des angles de rotation de 0° ou de 180° et il suffit d'inverser m_l et m_h pour retrouver un système valide sur les deux autres angles. On

a donc :

$$r = 0 \; ou \; r = 180 \implies \begin{cases} zd_l - m_l + g_{\Leftarrow} + g_{\Rightarrow} - c_{\Leftarrow} - c_{\Rightarrow} &= 0 \\ zd_h - m_h + g_{\Uparrow} + g_{\Downarrow} - c_{\Uparrow} - c_{\Downarrow} &= 0 \end{cases}$$

$$r = 90 \; ou \; r = 270 \implies \begin{cases} zd_l - m_h + g_{\Leftarrow} + g_{\Rightarrow} - c_{\Leftarrow} - c_{\Rightarrow} &= 0 \\ zd_h - m_l + g_{\Uparrow} + g_{\Downarrow} - c_{\Uparrow} - c_{\Downarrow} &= 0 \end{cases}$$

Ces équations sont suffisantes pour contraindre le problème et on peut se passer de la variable de position dans notre modèle.

On note

$$position\,(d, m, g, c, r, z) \subseteq D \times M \times G \times C \times R \times Z$$

le prédicat qui vérifie cet ensemble de contraintes.

3.2.3 Marges et coupes

Du choix du média et de la découpe (si rouleau), du positionnement, de la rotation et du zoom, il découle parfois des marges blanches, ou plus simplement des *marges*, et/ou des coupes de l'image. Marges et coupes peuvent être volontaires ou non. Par exemple, l'ingénieur désirant imprimer un dessin technique afin de le présenter en réunion, peut demander à ajouter une marge supplémentaire au-dessus de l'image. Cette marge sert à relier le média à un support ou à d'autres médias. Dans notre exemple, le support peut être un tableau muni de crochets. Elle permet de faire les trous nécessaires au passage des crochets, hors de l'image. Une autre utilisation possible des marges est de permettre l'ajout d'estampille extérieure au document. Quant aux coupes, elles peuvent être volontaires si l'image originale admet elle-même de grandes marges blanches par exemple. Marges et coupes peuvent aussi être involontaires mais tolérées dans un compromis avec les autres critères.

Marges minimales

On introduit ici, afin de simplifier les écritures, les marges minimales que doit respecter la configuration d'impression. Ces marges sont la synthèse des marges supplémentaires et d'extrémité, et dépendent de la rotation effectuée sur l'image.

Plus formellement, soit r la rotation effectuée sur l'image,

$$g^{sup} = \left(g^{sup}_{\Uparrow}, g^{sup}_{\Leftarrow}, g^{sup}_{\Downarrow}, g^{sup}_{\Rightarrow}\right) \text{ les marges supplémentaires,}$$

$$g^{mat} = \left(g^{mat}_{\Uparrow}, g^{mat}_{\Leftarrow}, g^{mat}_{\Downarrow}, g^{mat}_{\Rightarrow}\right) \text{ les marges matérielles,}$$

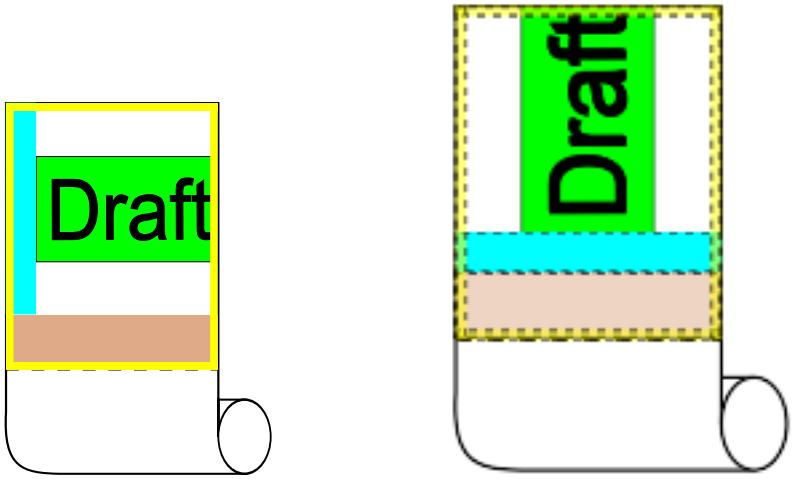


FIG. 3.10: Cas d'une image avec marge gauche supplémentaire (bleu), marge d'extrémité finale (rose) et marges matérielles (jaune)

et $g^{ext} = \left(g^{ext}_{\Uparrow}, g^{ext}_{\Downarrow}\right)$ les marges d'extrémité, on appelle marges minimales

$$g^{min} = \left(g^{min}_{\Uparrow}, g^{min}_{\Leftarrow}, g^{min}_{\Downarrow}, g^{min}_{\Rightarrow}\right) \in G^{min} = \left(\mathbb{R}^{+}\right)^{4}$$

le quadruplet tel que :

$$g^{sup}_{\Uparrow} = auto \implies \begin{cases} r = 0 & \implies & g^{min}_{\Uparrow} & = & g^{mat}_{\Uparrow} & + & g^{ext}_{\Uparrow} \\ r = 90 & \implies & g^{min}_{\Uparrow} & = & g^{mat}_{\Leftarrow} & & \\ r = 180 & \implies & g^{min}_{\Uparrow} & = & g^{mat}_{\Downarrow} & + & g^{ext}_{\Downarrow} \\ r = 270 & \implies & g^{min}_{\Uparrow} & = & g^{mat}_{\Rightarrow} & & \end{cases}$$

$$g^{sup}_{\Uparrow} \neq auto \implies \begin{cases} r = 0 & \implies & g^{min}_{\Uparrow} & = & g^{sup}_{\Uparrow} & + & g^{ext}_{\Uparrow} \\ r = 90 & \implies & g^{min}_{\Uparrow} & = & g^{sup}_{\Uparrow} & & \\ r = 180 & \implies & g^{min}_{\Uparrow} & = & g^{sup}_{\Uparrow} & + & g^{ext}_{\Downarrow} \\ r = 270 & \implies & g^{min}_{\Uparrow} & = & g^{sup}_{\Uparrow} & & \end{cases}$$

$$
g_{\Leftarrow}^{sup} = auto \implies \begin{cases} r = 0 & \implies & g_{\Leftarrow}^{min} & = & g_{\Leftarrow}^{mat} & & \\ r = 90 & \implies & g_{\Leftarrow}^{min} & = & g_{\Downarrow}^{mat} & + & g_{\Downarrow}^{ext} \\ r = 180 & \implies & g_{\Leftarrow}^{min} & = & g_{\Rightarrow}^{mat} & & \\ r = 270 & \implies & g_{\Leftarrow}^{min} & = & g_{\Uparrow}^{mat} & + & g_{\Uparrow}^{ext} \end{cases}
$$

$$
g_{\Leftarrow}^{sup} \neq auto \implies \begin{cases} r = 0 & \implies & g_{\Leftarrow}^{min} & = & g_{\Leftarrow}^{sup} & & \\ r = 90 & \implies & g_{\Leftarrow}^{min} & = & g_{\Leftarrow}^{sup} & + & g_{\Downarrow}^{ext} \\ r = 180 & \implies & g_{\Leftarrow}^{min} & = & g_{\Leftarrow}^{sup} & & \\ r = 270 & \implies & g_{\Leftarrow}^{min} & = & g_{\Leftarrow}^{sup} & + & g_{\Uparrow}^{ext} \end{cases}
$$

$$
g_{\Downarrow}^{sup} = auto \implies \begin{cases} r = 0 & \implies & g_{\Downarrow}^{min} & = & g_{\Downarrow}^{mat} & + & g_{\Downarrow}^{ext} \\ r = 90 & \implies & g_{\Downarrow}^{min} & = & g_{\Rightarrow}^{mat} & & \\ r = 180 & \implies & g_{\Downarrow}^{min} & = & g_{\Uparrow}^{mat} & + & g_{\Uparrow}^{ext} \\ r = 270 & \implies & g_{\Downarrow}^{min} & = & g_{\Leftarrow}^{mat} & & \end{cases}
$$

$$
g_{\Downarrow}^{sup} \neq auto \implies \begin{cases} r = 0 & \implies & g_{\Downarrow}^{min} & = & g_{\Downarrow}^{sup} & + & g_{\Downarrow}^{ext} \\ r = 90 & \implies & g_{\Downarrow}^{min} & = & g_{\Downarrow}^{sup} & & \\ r = 180 & \implies & g_{\Downarrow}^{min} & = & g_{\Downarrow}^{sup} & + & g_{\Uparrow}^{ext} \\ r = 270 & \implies & g_{\Downarrow}^{min} & = & g_{\Downarrow}^{sup} & & \end{cases}
$$

$$
g_{\Rightarrow}^{sup} = auto \implies \begin{cases} r = 0 & \implies & g_{\Rightarrow}^{min} & = & g_{\Rightarrow}^{mat} & & \\ r = 90 & \implies & g_{\Rightarrow}^{min} & = & g_{\Uparrow}^{mat} & + & g_{\Uparrow}^{ext} \\ r = 180 & \implies & g_{\Rightarrow}^{min} & = & g_{\Leftarrow}^{mat} & & \\ r = 270 & \implies & g_{\Rightarrow}^{min} & = & g_{\Downarrow}^{mat} & + & g_{\Downarrow}^{ext} \end{cases}
$$

$$
g_{\Rightarrow}^{sup} \neq auto \implies \begin{cases} r = 0 & \implies & g_{\Rightarrow}^{min} & = & g_{\Rightarrow}^{sup} & & \\ r = 90 & \implies & g_{\Rightarrow}^{min} & = & g_{\Rightarrow}^{sup} & + & g_{\Uparrow}^{ext} \\ r = 180 & \implies & g_{\Rightarrow}^{min} & = & g_{\Rightarrow}^{sup} & & \\ r = 270 & \implies & g_{\Rightarrow}^{min} & = & g_{\Rightarrow}^{sup} & + & g_{\Downarrow}^{ext} \end{cases}
$$

On note

$$
margesmin\left(g^{min}, g^{sup}, g^{mat}, g^{ext}, r\right) \subseteq G^{min} \times G^{sup} \times G^{mat} \times G^{est} \times R
$$

le prédicat qui vérifie cet ensemble de contraintes.

Marges

Nous considérons ici les marges, de façon générale, *relativement à l'orientation de l'image*. On distingue quatre marges différentes : au-dessus, à gauche , au-dessous, et à droite de l'image. Elles sont exprimées de façon absolue, c'est-à-dire dans une unité de longueur quelconque, par exemple le millimètre. Nous considérons dans la suite les marges comme des valeurs réelles positives. On note $G \in (\mathbb{R}^+)^4$ l'ensemble des marges possibles. Dans la suite, le quadruplet des marges totales d'une configuration d'impression sera noté :

$$g = (g_{\Uparrow}, g_{\Leftarrow}, g_{\Downarrow}, g_{\Rightarrow}) \in G$$

où

$g_{\Uparrow}$ représente la marge au-dessus de l'image

$g_{\Downarrow}$ représente la marge au-dessous de l'image

$g_{\Leftarrow}$ représente la marge à gauche de l'image

$g_{\Rightarrow}$ représente la marge à droite de l'image

On a vu dans le paragraphe précédent que ces marges devaient respecter les *marges minimales*, ce qui fournit des bornes inférieures. De plus, il est clair que ces valeurs sont bornées par les dimensions du média sélectionné. On ne peut en effet pas obtenir des marges plus grandes que la taille du média sur lequel l'image est imprimée. La dimension à considérer pour borner chaque marge dépend de la rotation réalisée sur l'image. Par exemple, pour une rotation de 90°, la somme des marges du dessus et du dessous est forcément inférieure à la largeur du média.

Plus généralement, soit r la rotation effectuée sur l'image,

$$g^{min} = \left(g_{\Uparrow}^{min}, g_{\Leftarrow}^{min}, g_{\Downarrow}^{min}, g_{\Rightarrow}^{min}\right)$$

les marges minimales, et $m = (m_h, m_l, m_f)$ le média sélectionné, les contraintes suivantes doivent être vérifiées :

$$\begin{aligned} g_{\Uparrow} &\geq g_{\Uparrow}^{min} \\ g_{\Leftarrow} &\geq g_{\Leftarrow}^{min} \\ g_{\Downarrow} &\geq g_{\Downarrow}^{min} \\ g_{\Rightarrow} &\geq g_{\Rightarrow}^{min} \end{aligned}$$

$$r = 0 \; ou \; r = 180 \implies \begin{cases} g_{\Uparrow} + g_{\Downarrow} & < & m_h \\ g_{\Leftarrow} + g_{\Rightarrow} & < & m_l \end{cases}$$

$$r = 90 \; ou \; r = 270 \implies \begin{cases} g_{\Uparrow} + g_{\Downarrow} & < & m_l \\ g_{\Leftarrow} + g_{\Rightarrow} & < & m_h \end{cases}$$

On note $marges\,(g, g^{min}, m) \subseteq G \times G^{min} \times M$ le prédicat qui vérifie cet ensemble de contraintes.

Coupes

De la même façon que pour les marges, nous considérons les coupes de l'image, relativement à l'orientation de l'image. On distingue donc quatre coupes différentes : du haut, de la gauche , du bas et de la droite de l'image. Elles sont exprimées de façon absolue, c'est-à-dire dans une unité de longueur quelconque. Nous considérons dans la suite les coupes comme des valeurs réelles positives.

L'ensemble des coupes possibles est noté $C \in (\mathbb{R}^+)^4$.

On note :

$$c = (c_{\Uparrow}, c_{\Leftarrow}, c_{\Downarrow}, c_{\Rightarrow}) \in C$$

un quadruplet particulier de coupes, où

$c_{\Uparrow}$ représente la coupe du haut de l'image

$c_{\Downarrow}$ représente la coupe du bas de l'image

$c_{\Leftarrow}$ représente la coupe de la partie gauche de l'image

$c_{\Rightarrow}$ représente la coupe de la partie droite de l'image

Il est clair que ces valeurs sont bornées par les dimensions de l'image à imprimer. Par exemple, la somme des coupes haute et basse est forcément inférieure à la hauteur de l'image. De même pour la direction horizontale. Notons que contrairement aux marges, les bornes sur les coupes ne dépendent pas de la rotation. En revanche, le zoom doit être pris en compte dans le calcul des dimensions de l'image à imprimer.

Soit l'image $i = (i_h, i_l, p)$ ayant subie la transformation d'échelle z, les contraintes suivantes doivent être vérifiées :

$$\begin{aligned} c_{\Uparrow} + c_{\Downarrow} &< z \cdot i_h \\ c_{\Leftarrow} + c_{\Rightarrow} &< z \cdot i_l \end{aligned}$$

On note $coupes\,(C, z, i)$ le prédicat qui vérifie cet ensemble de contraintes.

3.2.4 Admissibilité

Conditions d'admissibilité

Rappel des notations :

M l'ensemble des médias disponibles

R l'ensemble des angles de rotation

Z l'ensemble des valeurs de zoom

G l'ensemble produit des valeurs de marges, exprimées par rapport à l'orientation de l'image

C l'ensemble produit des coupes, exprimées par rapport à l'orientation de l'image

On appelle espace des solutions E l'espace produit suivant :

$$E = M \times R \times Z \times G \times C$$

L'espace des solutions admissibles est inclus dans E. Nous résumons ici l'ensemble des contraintes nécessaire à l'admissibilité d'une configuration d'impression.

Soit une image à imprimer :

$$i = (i_l, i_h, i_p) \in D$$

, une source de média :

$$s = (s_L, s_{H_{min}}, s_{H_{max}}, s_F, s_{g^{mat}}) \in \hat{S}$$

avec $s_{g^{mat}} = \left(g^{mat}_{\Uparrow}, g^{mat}_{\Leftarrow}, g^{mat}_{\Downarrow}, g^{mat}_{\Rightarrow}\right) \in G^{mat}$, les marges supplémentaires :

$$g^{sup} = \left(g^{sup}_{\Uparrow}, g^{sup}_{\Leftarrow}, g^{sup}_{\Downarrow}, g^{sup}_{\Rightarrow}\right) \in G^{sup}$$

, les marges d'extrémité :

$$g^{ext} = \left(g^{ext}_{\Uparrow}, g^{ext}_{\Downarrow}\right) \in G^{ext}$$

, et les marges globales :

$$g = (g_{\Uparrow}, g_{\Leftarrow}, g_{\Downarrow}, g_{\Rightarrow}) \in G$$

On dit que la configuration d'impression :

$$e = (m, r, z, g, c) \in E$$

est admissible pour le quadruplet (i, s, g^{sup}, g^{ext}) si et seulement si la conjonction des cinq prédicats suivants est vérifiée :

1. $media\,(s, m)$
2. $position\,(i, m, g, c, r, z)$
3. $margesmin\,(g^{min}, g^{sup}, s_{g^{mat}}, g^{ext}, r)$
4. $marges\,(g, g^{min}, m)$
5. $coupes\,(c, z, i)$

Nous notons $A\,(i, s, g^{sup}, g^{ext})$ l'ensemble des solutions admissibles, étant donné une image i , une source de média s, les marges supplémentaires g^{sup} et les marges d'extrémité g^{ext}.

Espace des solutions admissibles

Soit $\overline{A}\left(i, \hat{S}, g^{sup}, g^{ext}\right) \subset E$ l'espace des solutions admissibles, sachant le document à imprimer i , l'ensemble des sources de médias disponibles $\hat{S}$, les marges supplémentaires g^{sup} et les marges d'extrémité g^{ext}. On a :

$$\overline{A}\left(i, \hat{S}, g^{sup}, g^{ext}\right) = \bigcup_{s \in \hat{S}} A\left(i, s, g^{sup}, g^{ext}\right)$$

3.3 Conclusion

Dans ce chapitre nous avons énuméré, expliqué et formalisé l'ensemble des données formant l'entrée de notre problème de configuration d'impression grand format.

Nous avons modélisé l'espace des paramètres d'impression, afin de définir l'ensemble des configurations admissibles, connaissant les caractéristiques du document à imprimer, les contraintes et directives de l'imprimeur, ainsi que les ressources d'impression disponibles.

Chapitre 4

Modélisation des préférences

Dans la première partie de ce chapitre, nous présentons et critiquons le modèle actuel des préférences de l'imprimeur, sa *politique d'impression*.

Dans la seconde partie, nous proposons une formalisation de ce modèle qui, conjuguée au modèle de la configuration d'impression, permet de considérer le choix de la configuration préférée comme un problème d'optimisation linéaire mixte en nombre entier. Nous discutons des avantages de cette approche par rapport à l'approche actuelle.

Dans la dernière partie, nous examinons divers modèles de préférence afin d'améliorer le modèle actuel, sur les trois points suivants :

- le pouvoir d'expression
- la complexité des algorithmes permettant de trouver la meilleure solution
- l'apprentissage automatique

Nous concluons sur le choix d'un nouveau modèle : une fonction d'utilité additive dont les utilités marginales sont des fonctions linéaires par morceaux. Ce choix implique des hypothèses fortes sur les préférences de l'imprimeur. Afin de respecter ces hypothèses, nous proposons une adaptation de la famille d'attributs représentant le problème.

4.1 Modèle de préférence actuel

Dans la suite, nous employons indistinctement les expressions "modèle de préférence de l'imprimeur" et "politique d'impression".

4.1.1 Paramètres

Dans cette section, nous décrivons les paramètres des politiques d'impression, en vigueur dans les applications actuelles d'Océ. Ces paramètres forment le modèle de préférence de l'imprimeur. Nous montrons, sur trois images exemples, les configurations obtenues avec quatre politiques d'impression usuelles. Cela permet de mettre en lumière certaines difficultés rencontrées par les imprimeurs, quant au choix d'une politique.

Paramètres des politiques d'impression

Cinq paramètres composent les politiques d'impression, actuellement en vigueur dans les applications Océ. Nous les décrivons succinctement dans la liste suivante, à travers les valeurs qu'ils peuvent prendre. Les valeurs symboliques sont notées <XXX>.

1. Sélection du média

   (a) un des formats standards disponibles, par exemple A0 ou A1. Le média sélectionné doit respecter ce format ; il peut être issu d'un tiroir ou d'un rouleau.

   (b) un des rouleaux disponibles, ce qui force le média à avoir la largeur du rouleau.

   (c) <exact> : les dimensions du média doivent correspondre aux dimensions de l'image à l'échelle demandée (i.e. "pré-zoomée"), au seuil de tolérance s près (la rotation étant fixée). Ni marges ni coupes ne sont tolérées au-delà du seuil de tolérance.

   (d) <plus_grand> : les dimensions du média doivent être au moins aussi grandes que celles de l'image à l'échelle demandée, permettant d'imprimer celle-ci sans en couper un bout. En même temps, le média doit être le plus petit possible. Le média choisi sera donc le plus petit des plus grands, s'il existe.

   (e) <plus_grand_sinon_plus_petit> : idem que le paramétrage précédent sauf dans le cas où il n'existe pas de média plus grand que l'image ; il permet alors de choisir un média plus petit, en particulier le plus grand des plus petits. On voit que l'on se ramène dans un premier temps au cas précédent et, s'il n'existe aucune solution, on adopte la stratégie expliquée dans le point suivant.

(f) <aussi_proche_que_possible> suppose de choisir le média dont les dimensions sont aussi proches que possible des dimensions de l'image "pré-zoomée".

2. Ajustement de l'image au média, en terme de zoom

   (a) <exact> spécifie que l'échelle de l'image doit être absolument respectée et que l'image doit apparaître complètement sur le média. En d'autres termes, cela signifie que les coupes sont interdites (au-delà du seuil) et que le zoom (100% par défaut mais cela peut être n'importe quelle valeur exacte fournie par l'imprimeur) est constant.

   (b) <réduire_pour_ajuster> : même chose que précédemment sauf que la réduction de l'image est permise. Une réduction intervient pour éviter toute coupe à l'image. Elle est toujours la plus restreinte possible.

   (c) <zoomer_pour_ajuster> spécifie que les marges doivent être minimisées, et que les coupes sont interdites. Ce paramétrage suppose que l'imprimeur préfère un zoom aussi proche du zoom demandé que possible mais que l'importance de la différence de zoom est négligeable par rapport à celle de la différence de marges (on ne parle pas des coupes qui sont simplement interdites) lorsque le média est plus grand que l'image.

   (d) <couper_pour_ajuster> : spécifie que l'image doit être imprimée à l'échelle demandée, quitte à être coupée, le moins possible évidemment.

3. Positionnement de l'image sur le média. Ce paramètre admet neuf valeurs symboliques, correspondant aux

   (a) quatre coins du média

   (b) quatre bords du média (position centrée sur un bord)

   (c) centre du média

4. Gestion d'éventuelles erreurs. Ce paramètre est utilisé dans le cas où l'imprimeur veut un format particulier qui n'est pas disponible. Il peut prendre deux valeurs :

   (a) <Alerte> indique juste une erreur et redonne la main à l'imprimeur.

   (b) <Sinon_plus_grand> permet de basculer sur la stratégie de sélection du média <Plus_grand>.

5. Ajustement de la découpe du média. Ce paramètre règle la stratégie de découpe, dans le cas d'un média issu d'un rouleau. Il peut prendre deux valeurs :

(a) <Standard> indique que le média doit être découpé de manière à correspondre à un format standard.

(b) <Synchrone> permet une découpe d'une longueur quelconque (entre la taille de découpe minimale et maximale).

Relations entre paramètres

On peut souligner que certaines valeurs pour tel paramètre de la politique d'impression peuvent "désactiver" d'autres paramètres. Par exemple, lorsque la sélection du média correspond à un média standard particulier, le paramètre d'ajustement de découpe n'a plus de sens car la valeur "standard" est alors obligatoire. A contrario, c'est le seul cas où le paramètre d'ajustement d'erreur devient actif.

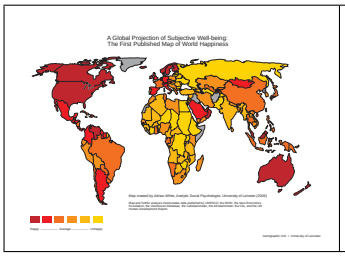
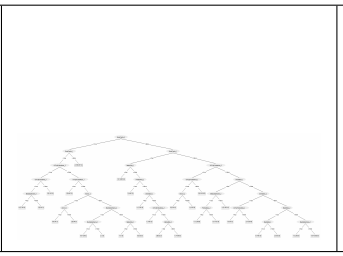
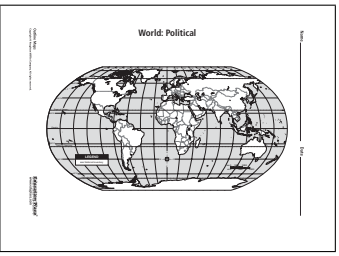
L'ajustement de l'image <exact> n'a pas de sens avec la sélection de média <aussi_proche_que_possible>. Pas plus que l'ajustement <Réduire_pour_ajuster> n'a de sens avec les stratégies <Exact> et <Plus_grand>, etc...

Certaines combinaisons de valeurs n'ont donc aucun sens et sont interdites dans l'interface du système.

4.1.2 Exemples de politiques

Dans les paragraphes suivants, nous présentons quelques exemples de politiques d'impression très utilisées. Dans leur description, nous omettons la position de l'image sur le média, qui est centrée pour les quatre politiques présentées. Nous illustrons leur comportement sur trois images de dimensions différentes (4.1). Ces trois images sont en orientation paysage. La première est au format A3. La seconde est un format non standard et la troisième au format standard américain Letter.

Dans nos exemples, l'imprimeur a pour sources de média disponibles *deux rouleaux aux formats A3 et A1*. Nous détaillons dans les paragraphes suivants le résultat de la configuration automatique pour chaque image.

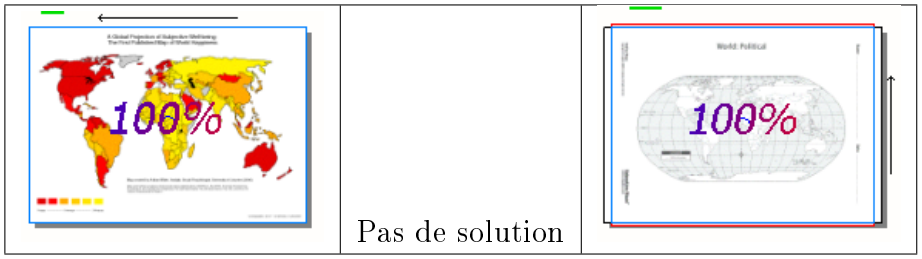
Image 1 : 420x297	Image 2 : 1026 x 360	Image 3 : 279 x 216
		

TAB. 4.1: Trois images illustratives (dimensions en mm)

Politique n°1

- sélection du média : “plus grand”
- ajustement de l’image : “ajustement exact”
- gestion d’erreur : ne s’applique pas ici car l’imprimeur n’a pas sélectionné un média particulier
- ajustement de la découpe : “standard”

On voit sur la figure 4.2 les résultats obtenus avec cette politique. La première image ayant un format standard correspondant à un média disponible, ne pose pas de problème et sera imprimée sans marge ni coupe (à part les coupes relatives aux marges matérielles[1]). En revanche, la seconde image pose un problème de configuration car le système ne dispose pas de média de format standard assez grand pour la contenir entièrement. Il préfère alors alerter l’imprimeur pour que celui-ci avise. Si l’imprimeur décide d’imprimer tout de même l’image, il lui faut alors spécifier manuellement les paramètres d’impression. Enfin, la troisième image sera imprimée sur le rouleau A3, dans sa largeur, c’est-à-dire que l’image est tournée de 90° par rapport au média. L’imprimeur obtient donc une feuille A4. On note les marges et les fines bandes coupées. Les marges sont tolérées car la stratégie de sélection du média est “plus grand” et non “exacte”. Quant aux coupes, elles sont également tolérées car en-deçà du seuil de tolérance.



TAB. 4.2: Résultats de la politique n°1

Politique n°2

- échelle demandée : 50%
- sélection du média : “plus grand”
- ajustement de l’image : “exact”
- gestion d’erreur : ne s’applique pas ici car l’imprimeur n’a pas sélectionné un média particulier
- ajustement de la découpe : “synchrone”

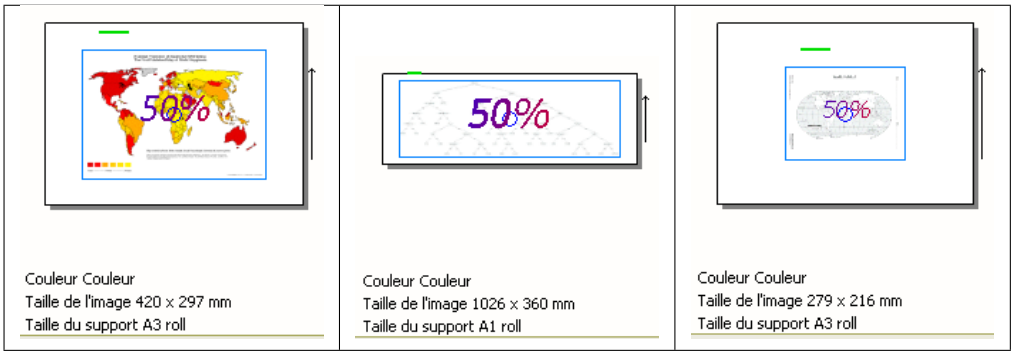
[1] les marges matérielles sont trop minces pour être représentées sur les schémas

On voit sur 4.3 les résultats obtenus avec cette politique. Les trois images ont bien été imprimées à 50% de leur taille d'origine. Cependant, malgré le paramètre d'ajustement de la découpe "synchrone" on peut noter qu'aucun média n'a été découpé juste après l'image. Ce phénomène est dû à la limite de la découpe minimale.

La première image est imprimée dans la largeur du rouleau A3, ce qui implique une découpe synchrone inférieure à la découpe minimale. Évidemment, cela entraîne des marges supplémentaires. Considérons un instant la question suivante : pourquoi l'image a-t-elle été tournée ainsi ? En effet il existe une autre solution, équivalente d'un point de vue politique d'impression : tourner l'image de 90°. Dans ce cas, la découpe minimale correspondrait exactement à la largeur de l'image[2] car celle-ci, zoomée à 50%, est devenu un A4. Alors pourquoi avoir choisi une solution plutôt que l'autre ? Quelle que soit la réponse, ce comportement n'est pas paramétrable dans une politique d'impression.

On peut faire un constat tout à fait similaire pour la troisième image. En revanche, il n'y avait pas d'alternative pour la seconde. En effet, imprimée dans la largeur du rouleau A1, la découpe est toujours minimale mais tournée de 90°, l'image dépasserait de beaucoup cette limite et la taille du média serait alors beaucoup plus grande. Or le plus petit média (pour une image donnée) est toujours préféré, à zoom et coupes équivalents, ce qui interdit cette solution.

On voit donc sur cet exemple que certains cas sont indiscernables à travers le prisme d'une politique d'impression comme définie par le module d'Océ. Cela peut entraîner des difficultés à prévoir le comportement.



TAB. 4.3: Résultats de la politique n°2

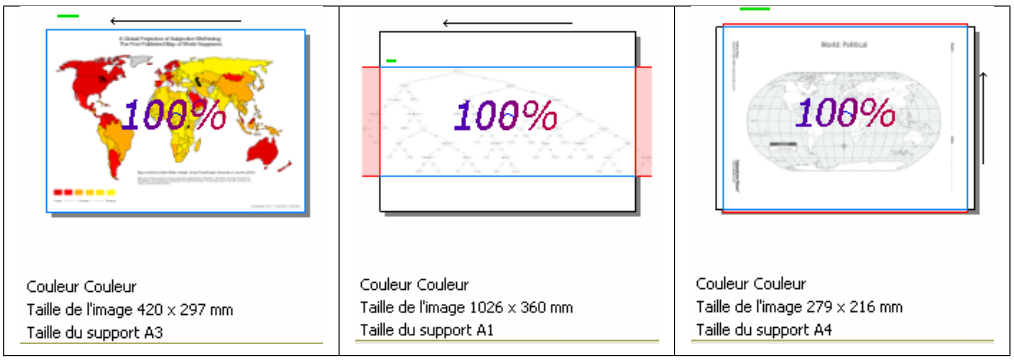
Politique n°3

– sélection du média : "plus grand et sinon plus petit"

[2]généralement la découpe minimale permet de réaliser le plus petit format, ici un A4

- ajustement de l'image : "couper pour ajuster"
- gestion d'erreur : pas d'effet ici
- ajustement de la découpe : "standard"

On voit sur la figure 4.4 les résultats obtenus avec cette politique. Pour les images 1 et 3, on retrouve les même résultats qu'avec la politique 1. L'image 1 correspond exactement au média disponible donc pas besoin de coupe. L'image 3 est imprimée avec une coupe inférieure au seuil de coupe acceptable, ce qui n'a donc pas d'incidence. En revanche, la seconde image présente un nouveau comportement. Imprimée à partir du rouleau A1, elle est tronquée sur ces bords gauche et droit à cause du paramètre de découpe "standard". L'imprimeur obtient donc un média au format A1 avec une image coupée sur les côtés.



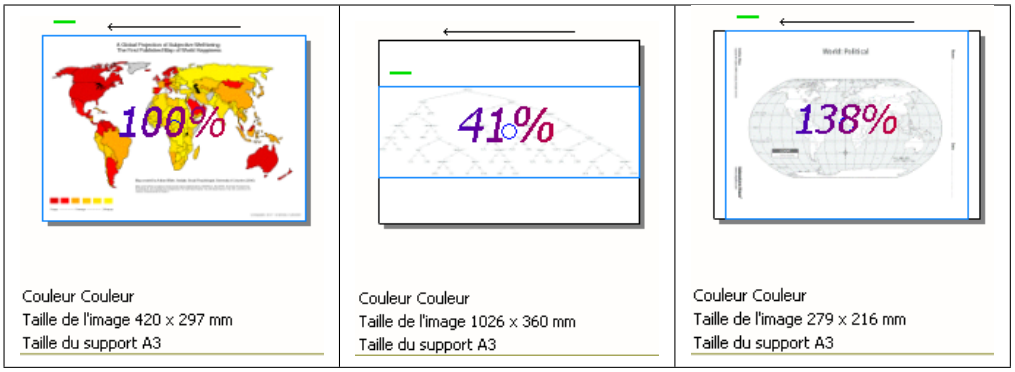
TAB. 4.4: Résultats de la politique n°3

Politique n°4

- sélection du média : A3
- ajustement de l'image : "zoomer pour ajuster"
- ajustement de l'erreur : exact
- ajustement de la découpe : "standard" (obligatoire)

On voit sur la figure 4.5 les résultats obtenus avec cette politique. On retrouve encore le même résultat pour l'image 1. En revanche, on voit que l'image 2 a été réduite de sorte à tenir dans la longueur d'un A3. De même, l'image 3 a été agrandie de sorte à minimiser les marges.

On a remarqué que certains cas ne peuvent pas être distingués dans certaines politiques d'impression. Or les choix sont tout de même réalisés par le système selon des règles qui ne sont pas explicites. Cela contribue à rendre le comportement du module de configuration automatique moins prévisible et peut poser aux imprimeurs des problèmes dans la configuration de leurs politiques d'impression.



TAB. 4.5: Résultat de la politique n°4

4.2 Exploitation du modèle de préférence

Un des principaux problèmes relatifs à l'automatisation de la configuration d'impression est de trouver la meilleure configuration d'impression, étant donné la politique d'impression, le contexte, les desiderata et les ressources de l'imprimeur. Jusqu'ici codé en dur de manière procédurale, le module qui s'occupe de cette tâche pose problème :

- aux concepteurs, de par sa spécification délicate,
- aux développeurs, à cause de sa richesse comportementale.

Nous proposons dans cette section d'attaquer ce problème dans le cadre de l'optimisation sous contraintes. Nous montrons qu'une fois formalisé de cette façon, il est possible d'exploiter des moteurs de résolution performants.

4.2.1 Un problème d'optimisation sous contraintes

Dans notre problématique, on veut trouver l'alternative préférée, au sens de l'imprimeur.

On se place dans le cadre de la décision solitaire dans le certain, c'est-à-dire que l'imprimeur sélectionne seul une solution "préférée" parmi un ensemble d'alternatives, dont les effets sont connus de façon certaine.

Le modèle de préférence étant fixé, on peut considérer ce problème, de prime abord, comme un problème de satisfaction de contraintes flexibles (SCSP pour Soft Constraint Satisfaction Problem). Nous faisons ici un bref rappel sur les CSP et CSP flexibles.

Un problème d'optimisation...

Choisir la configuration préférée peut être vu comme un problème d'optimisation sous contraintes. En effet, quelle que soit la politique d'impression sélectionnée, l'imprimeur cherche toujours à minimiser :

- les marges (hors marges matérielles, supplémentaires et d'extrémités qui sont considérées constantes)
- les coupes quand elles sont autorisées (et sinon à l'intérieur du seuil de tolérance)
- l'écart - si autorisé - entre le zoom désiré (par défaut 100%) et le zoom effectué

On peut donc modéliser notre fonction objectif comme une fonction des marges, des coupes et du zoom, à minimiser. De plus, on ne peut pas supposer a priori une préférence quelconque quant à l'orientation des marges ou des coupes, si celles-ci sont autorisées. Par exemple, ça n'a pas de sens de préférer une marge gauche de 10cm à une marge droite, haute ou basse de 10cm. On peut donc considérer une symétrie sur les marges. De même pour les coupes. Nous supposerons également une symétrie autour du zoom désiré, de sorte qu'une réduction de 50% soit perçue de la même manière qu'une augmentation de 50%.

Nous considérons la fonction objectif f_{obj} suivante :

$$minimiser \qquad f_{obj} = \alpha\,(g_{\Uparrow} + g_{\Leftarrow} + g_{\Downarrow} + g_{\Rightarrow}) + \beta\,(c_{\Uparrow} + c_{\Leftarrow} + c_{\Downarrow} + c_{\Rightarrow}) + \gamma\,\|z^* - z\|$$

où α, $\beta\,et\,\gamma$ sont des réels strictement positifs.

Rappelons que z^* représente le zoom désiré.

...sous contraintes

Nous modélisons les paramètres des politiques d'impression dans les paragraphes suivants.

Sélection du média

- Le cas d'un format spécifique se traduit naturellement par les contraintes de taille. Soit $formatX \in FS$ le format standard choisi par l'utilisateur, on a, suivant l'orientation du média :

$$\begin{cases} m_h &= formatX_{grand\,cote} \\ m_l &= formatX_{petit\,cote} \end{cases} \quad ou \quad \begin{cases} m_h &= formatX_{petit\,cote} \\ m_l &= formatX_{grand\,cote} \end{cases}$$

– Pour une largeur de rouleau spécifique, on a simplement la contrainte suivante :

$$m_l = largeur$$

– <Exact> les dimensions du média doivent correspondre aux dimensions de l'image i zoomée à l'échelle désirée z^*, au seuil de tolérance s près (la rotation étant fixée). Ni marges ni coupes ne sont tolérées au-delà du seuil de tolérance. Ce paramétrage se traduit directement par un jeu de contraintes tel que :

$$r = 0 \; ou \; r = 180 \implies \begin{cases} \|z^* \cdot i_h - m_h\| \leq 2s \\ \|z^* \cdot i_l - m_l\| \leq 2s \end{cases}$$

$$r = 90 \; ou \; r = 270 \implies \begin{cases} \|z^* \cdot i_h - m_l\| \leq 2s \\ \|z^* \cdot i_l - m_h\| \leq 2s \end{cases}$$

Ce paramétrage n'a de sens que conjugué avec l'ajustement exact de l'image au média. On peut alors remarquer qu'en conjuguant les deux, en exploitant les marges et les coupes, on "s'affranchit" de la rotation :

$$\begin{aligned} z &= z^* \\ g_{\Uparrow} \leq s \;\; et \;\; c_{\Uparrow} &\leq s \\ g_{\Leftarrow} \leq s \;\; et \;\; c_{\Leftarrow} &\leq s \\ g_{\Downarrow} \leq s \;\; et \;\; c_{\Downarrow} &\leq s \\ g_{\Rightarrow} \leq s \;\; et \;\; c_{\Rightarrow} &\leq s \end{aligned}$$

– <Plus_grand> les dimensions du média doivent être au moins aussi grandes que celles de l'image, permettant d'imprimer celle-ci sans en couper un bout. En même temps, le média doit être le plus petit possible, ce qui est déjà traduit dans la fonction objectif. Le média choisi sera donc le plus petit des plus grands, s'il existe. On peut traduire ce paramétrage en ajoutant les contraintes suivantes :

$$r = 0 \; ou \; r = 180 \implies \begin{cases} z^* \cdot i_h < m_h \\ z^* \cdot i_l < m_l \end{cases}$$

$$r = 90 \; ou \; r = 270 \implies \begin{cases} z^* \cdot i_h < m_l \\ z^* \cdot i_l < m_h \end{cases}$$

– <Plus_grand_sinon_plus_petit> idem que le paramétrage précédent sauf dans le cas où il n'existe pas de média plus grand que l'image ; il permet alors de choisir un média plus petit, en particulier le plus grand des plus petits. Il

y a ici un effet de seuil. On voit que l'on se ramène dans un premier temps au cas précédent et, s'il n'existe aucune solution, on adopte la stratégie expliquée dans le point suivant. Nous considérons cette succession conditionnelle de stratégies comme deux modèles distincts. Ce processus de succession éventuelle est intégrée à un méta-niveau, c'est-à-dire hors du modèle courant. Dans la suite, nous désignons ces différents modèles sous le terme de "phases".

– <Aussi_proche_que_possible> cette stratégie suppose de choisir le média dont les dimensions sont aussi proches que possible des dimensions de l'image à l'échelle demandée. La fonction objectif permet déjà de minimiser les marges, les coupes, et l'écart entre zoom obtenu et zoom désiré. Aucune contrainte supplémentaire n'est donc nécessaire, la fonction objectif seule suffit à discriminer les différents médias. Notons que les coefficients de la fonction objectif α et β doivent être égaux afin de ne favoriser ni les médias plus grands, ni les plus petits. Remarquons aussi que le terme concernant la minimisation de l'écart de zoom permet de discriminer entre deux médias ayant les mêmes proportions. Par exemple entre un A3 et un A2, si l'image est ajustée à la taille du média, les marges et coupes résultantes peuvent être égales (nulles en particulier). Il ne reste alors que ce dernier terme pour sélectionner le média le plus proche de la taille souhaitée.

Ajustement de l'image

– <Exact> spécifie que l'échelle de l'image doit être absolument respectée et que l'image doit apparaître complètement sur le média. En d'autres termes, cela signifie que les coupes sont interdites (au-delà du seuil) et que le zoom (100% par défaut mais cela peut être n'importe quelle valeur exacte fournie par l'imprimeur) est constant. On traduit cela par les contraintes suivantes :

$$\begin{aligned} z &= z^* \\ c_{\Uparrow} &\leq s \\ c_{\Leftarrow} &\leq s \\ c_{\Downarrow} &\leq s \\ c_{\Rightarrow} &\leq s \end{aligned}$$

– <Réduire_pour_ajuster> même chose que précédemment sauf que la réduc-

tion de l'image est permise.

$$\begin{aligned} z &\leq z^* \\ c_{\Uparrow} &\leq s \\ c_{\Leftarrow} &\leq s \\ c_{\Downarrow} &\leq s \\ c_{\Rightarrow} &\leq s \end{aligned}$$

– <Zoomer_pour_ajuster> spécifie que les marges doivent être minimisées, et que les coupes sont interdites. Ce paramétrage suppose que l'imprimeur préfère un zoom aussi proche du zoom demandé que possible mais que l'importance de la différence de zoom est négligeable par rapport à celle de la différence de marges (on ne parle pas des coupes qui sont simplement interdites) lorsque le média est plus grand que l'image. Cette préférence s'exprime dans la fonction objectif par un ratio $\frac{\gamma}{\alpha}$ proche de zéro.

$$\begin{aligned} c_{\Uparrow} &\leq s \\ c_{\Leftarrow} &\leq s \\ c_{\Downarrow} &\leq s \\ c_{\Rightarrow} &\leq s \end{aligned}$$

<Couper_pour_ajuster> spécifie que le zoom doit être constant (échelle 1 :1 par défaut) et permet les coupes, le moins possible évidemment. La seule contrainte à ajouter est donc celle du zoom constant :

$$z = z^*$$

Positionnement de l'image Les contraintes sur le positionnement décrites ici supposent une image sans cartouche. En effet, ces contraintes sont déjà présentes dans le cas inverse. Nous avons vu que le positionnement concerne les neufs positions canoniques. Dans le cas des positions du bord, on traduit le positionnement par les contraintes suivantes :

<En_haut_à_gauche> se traduit par la volonté d'obtenir des marges et des coupes nulles au-dessus et à gauche de l'image, quel que soit le média. Il faut cependant tenir compte des marges minimales définies dans la section

3.2.3. On peut donc spécifier cette volonté par l'ajout des contraintes :

$$\begin{aligned} g_{\Uparrow} &= g_{\Uparrow}^{min} \\ g_{\Leftarrow} &= g_{\Leftarrow}^{min} \\ c_{\Uparrow} &= 0 \\ c_{\Leftarrow} &= 0 \end{aligned}$$

Même principe pour toutes les positions des coins.

<En_haut_au_milieu> se traduit de la sorte :

$$\begin{aligned} g_{\Uparrow} &= g_{\Uparrow}^{min} \\ g_{\Leftarrow} - g_{\Leftarrow}^{min} &= g_{\Rightarrow} - g_{\Rightarrow}^{min} \\ c_{\Uparrow} &= 0 \\ c_{\Leftarrow} &= c_{\Rightarrow} \end{aligned}$$

Même principe pour toutes les positions de bordures centrales.

<Position_centrale> peut se traduire par ce qui suit

$$\begin{aligned} g_{\Uparrow} - g_{\Uparrow}^{min} &= g_{\Downarrow} - g_{\Downarrow}^{min} \\ g_{\Leftarrow} - g_{\Leftarrow}^{min} &= g_{\Rightarrow} - g_{\Rightarrow}^{min} \\ c_{\Uparrow} &= c_{\Downarrow} \\ c_{\Leftarrow} &= c_{\Rightarrow} \end{aligned}$$

Gestion d'erreur

<Alerte> indique juste une erreur et redonne la main à l'imprimeur. Pour traduire ce paramètre, il n'y a rien à ajouter au modèle.

<Sinon_plus_grand> permet de basculer sur la stratégie de sélection du média <Plus_grand>. Dans ce cas, il suffit de passer d'un modèle à l'autre, de la même façon qu'avec la stratégie de sélection automatique de média <Plus_grand_sinon_plus_petit>.

Ajustement de la découpe

<Standard> indique que la découpe doit correspondre à un format standard. Il suffit donc d'obliger les dimensions du média sélectionné à être standard, par un filtrage des médias disponibles. On place cette manipulation dans les pré-traitements.

<Synchrone> permet une découpe d'une longueur quelconque. C'est le cas général et il n'y a pas de contraintes particulière à ajouter.

Un exemple concret reprenant l'ensemble de ces contraintes est présenté dans la section C.1 de l'annexe C. Dans la section suivante, nous montrons comment transformer le problème d'optimisation obtenu, afin de le résoudre de manière efficace.

4.2.2 Méthodes de résolution

Nous avons vu que configurer une impression d'après une politique d'impression standard peut s'écrire comme un problème d'optimisation sous contraintes. Bien qu'élégante, cette formulation n'a que peu de valeur ajoutée si elle ne permet pas une résolution efficace du problème. En effet, la configuration automatique d'impression étant une fonctionnalité de l'interface utilisateur, elle nécessite un traitement rapide, de façon à ne pas alourdir l'interaction. Cette contrainte forte nous amène à considérer la résolution de ce problème d'optimisation sous un angle particulier.

Dans cette section nous examinons deux types de méthodes de résolution. Nous présentons d'abord un paradigme très général qui semble naturellement correspondre à notre problème : les réseaux de contraintes flexibles. Nous mettons cependant en lumière les limites de cette approche. Nous nous intéressons ensuite à un autre paradigme : la programmation mathématique linéaire mixte, et montrons comment passer de notre problème initial d'optimisation, contenant notamment des contraintes d'implication, à une suite d'optimisation linéaire mixte en nombre entier. Cette reformulation permet d'exploiter des algorithmes de résolution existants, de façon à conserver au maximum une simplicité déclarative combinée avec une résolution efficace et précise.

Réseau de contraintes flexibles

Nous introduisons d'abord succinctement le paradigme des réseaux de contraintes classiques (en anglais "Constraint Solving Problem", CSP) avant d'examiner leur extension aux contraintes flexibles. Le lecteur intéressé par une introduction plus complète des CSP peut se référer par exemple à [Tsa93].

CSP

Definition 3. Un CSP est définit par le triplet (V, D, C) où

- V est un ensemble de variables.
- D un ensemble de domaines (pour ces variables).

– C un ensemble de contraintes (sur ces variables) ; constitué dans notre cadre par les choix de l'utilisateur.

La solution d'un tel problème est l'assignement d'une valeur de son domaine à chaque variable, tel que toutes les contraintes soient satisfaites. De nombreux algorithmes ont été développés afin d'accélérer la recherche exacte d'une solution. Le principe de base est de réduire les domaines des variables pendant l'exploration de l'espace des solutions, en maintenant une certaine *consistance* par rapport aux contraintes. Initialement prévus pour manipuler des variables discrètes, certains algorithmes ont été adapté à la manipulation de variables continues [Fal94, ShF96, Lho93], ce qui est nécessaire pour résoudre notre problème.

Mais résoudre le problème de satisfaction n'est pas suffisant. Dans notre contexte, cette approche revient à trouver une configuration d'impression admissible mais en aucun cas la "préférée". De nombreuses recherches ont été réalisées afin d'étendre les CSP à la prise en compte de coûts ou de préférences. On regroupe aujourd'hui les fruits de ces travaux sous le nom de CSP flexibles.

CSP flexible Nous présentons ici le principe des CSP flexibles. Le lecteur intéressé peut se reporter à [Sch00] pour une introduction plus complète. Dans ce paradigme, on ne cherche plus à résoudre un problème de satisfaction mais d'optimisation. Le critère à optimiser (habituellement à minimiser), appelé indifféremment *fonction objectif* ou *fonction de coût*, réalise une agrégation d'une information de satisfaction des contraintes. Selon l'opérateur d'agrégation utilisé, différents CSP flexibles ont été proposés (voir le tableau 4.6).

Deux cadres algébriques permettant de regrouper ces différentes approches dans une représentation commune ont été développé : les CSP valuées ou "VCSP" (pour Valued CSP) et les CSP à base de semi-anneaux (ou SCSP pour Semiring-based CSP). La différence essentielle entre ces deux formalismes se situe dans l'ordre pouvant être exprimé sur les solutions. Les VCSP expriment un ordre total alors que les SCSP peuvent exprimer un ordre partiel.

Nous présentons ici uniquement le cadre algébrique des VCSP, introduit dans [SFV95], qui nous semble suffisant pour donner une vue d'ensemble des CSP flexibles. Le lecteur intéressé trouvera une comparaison détaillée de ces deux cadres dans [BMR+99].

Definition 4. Un VCSP est défini par le quintuplet (V, D, C, S, ρ) où

– *(V,D,C)* est un CSP classique.
– $S = (E, \oplus, \preceq, \top, \bot)$ est une structure de valuation (appelée monoïde commutatif) telle que

CSP	E	$\preceq$	$\top$	$\bot$	$\oplus$	Références
classique	$\{vrai, faux\}$	$vrai \preceq faux$	$faux$	$vrai$	$\bigwedge$	
pondéré	$\mathbb{N} \bigcup +\infty$	$\leq$	$+\infty$	0	$+$	[FW92]
flou	$[0,1]$	$\geq$	0	1	min	[Rut94]
possibiliste	$[0,1]$	$\leq$	1	0	max	[Sch92]
lexicographique	$[0,1]^* \bigcup \{\top\}$	Lex	$\top$	$\emptyset$	$\bigcup$	
probabiliste	$[0,1]$	$\leq$	1	0	$a+b-ab$	[?]

TAB. 4.6: Différents types de CSP flexibles

- E est un ensemble totalement ordonné par $\preceq$, muni d'un élément minimal $\bot$ et maximal $\top$.
- E est muni d'une loi de composition interne commutative et associative $\oplus$, appelé opérateur d'agrégation (ou de combinaison), qui vérifie les propriétés suivantes :
  - monotonie : $\forall x, y, z \in E \; tels \; que \; x \preceq z \; on \; a \; (x \oplus y) \preceq (z \oplus y)$
  - élément neutre : $\forall x \in E, \; x \oplus \bot = x$
  - élément absorbant[3] :$\forall x \in E, \; x \oplus \top = \top$
- $\rho : C \rightarrow E$, une application associant une valuation à chaque contrainte du réseau[4]

Résoudre une VCSP revient à trouver le tuple t, instance de V qui minimise la fonction de coût l suivante :

Soit $C' \subseteq C$ l'ensemble des contraintes violées pat t, on a

$$l = \bigoplus_{c \in C'} \varphi(c)$$

Les CSP flexibles ont été classés et nommés différemment selon l'opérateur d'agrégation utilisé. Le tableau 4.6 résume les principaux types de CSP flexibles. Parmi ces différents formalismes, le CSP pondéré semble être le meilleur candidat pour modéliser notre problème. La structure de valuation d'un CSP pondéré est la suivante : $S = (\mathbb{N} \bigcup +\infty, +, \leq, +\infty, 0)$. La résolution d'un CSP pondéré revient donc à minimiser la somme des poids des contraintes violées. La fonction objectif de l'imprimeur peut ainsi être représentée. Les contraintes d'admissibilité des configurations d'impression doivent être associées à l'élément maximal (et absorbant), ici la valeur $+\infty$.

Malheureusement, il semble qu'il soit particulièrement difficile d'adapter les algo-

[3] cette propriété peut être déduite des autres axiomes

[4] la version consistant à associer une valuation à chaque instanciation d'une contrainte n'est pas fondamentalement différente, voir [BMR+99].

rithmes de résolution des CSP flexibles aux domaines continus, de façon efficace. A notre connaissance, malgré des travaux récents mené dans ce sens ([ACZ03, CNT05, ACZ04, NGCB08]), aucun algorithme performant et exact de résolution de CSP pondéré, sur les domaines continus, n'est encore disponible.

Programmation linéaire mixte

Notre problème présente une particularité très importante : beaucoup de contraintes sont linéaires. Cela met à notre disposition une palette d'algorithmes de résolution, plus spécifiques et plus efficaces.

Notre objectif est de transformer un problème d'optimisation non linéaire en une série de problèmes linéaires, de manière à pouvoir exploiter au maximum les algorithmes de résolution existants. Rappelons qu'un problème d'optimisation linéaire, ou programme linéaire, est un problème d'optimisation mathématique pouvant s'écrire sous la forme :

$$\begin{array}{rl} \textit{minimiser} & f(x) = \sum_{i=1}^{n} c_i x_i \\ \textit{sous les conditions} & \left\{ \begin{array}{l} \sum_{i=1}^{n} a_{1i} x_i \left\{ \begin{array}{l} \leq \\ = \\ \geq \end{array} \right\} b_1 \\ \vdots \\ \sum_{i=1}^{n} a_{pi} x_i \left\{ \begin{array}{l} \leq \\ = \\ \geq \end{array} \right\} b_p \end{array} \right. \end{array}$$

Il s'agit d'optimiser (minimiser dans notre exemple) une fonction objectif linéaire, sous un ensemble de contraintes également linéaires. Les variables x_i sont réelles. Les coefficients c_i, a_{ij} et b_i sont appelés paramètres du modèle. Ils doivent être tous spécifiés pour que le programme linéaire soit complètement déterminé. La résolution efficace de ce type de problème d'optimisation est étudiée depuis plus de 60 ans. L'algorithme du simplex, inventé dans les années 40 constitue la méthode centrale de résolution.

Lorsque l'on impose à certaines variables d'être discrètes, on parle alors de programme linéaire mixte en nombre entier. On utilise classiquement un algorithme de type "branch and bound" couplé avec un simplex pour résoudre le problème.

Linéarisation du problème D'après la formalisation de notre problème, une fois qu'un média est sélectionné et que la rotation est fixée, on peut noter que, mis à

part un calcul de valeur absolue dans la fonction objectif sur lequel nous revenons plus loin, les relations entre les différentes variables sont linéaires. Nous proposons, dans un premier temps, d'inclure la rotation dans les données du problème. Trouver la meilleure configuration d'impression revient alors à résoudre non plus un mais plusieurs problèmes, correspondant d'une part aux différentes valeurs de rotation envisagées et, d'autre part, aux éventuelles successions de stratégies définies précédemment. Concernant la rotation, on a vu que, quelles que soient les données d'entrée, les symétries du problème permettent de ne considérer que deux valeurs d'angle qui dépendent uniquement de la position d'un éventuel cartouche sur l'image. Quant à la succession de stratégie, elle a été envisagée dans deux cas de figure seulement :

- avec la politique <Plus_grand_sinon_plus_petit> lorsqu'il n'existe pas de média plus grand que l'image.
- lorsque l'imprimeur spécifie un format exact de média qui n'est pas disponible et que la stratégie de gestion d'erreur est <Sinon_plus_grand>.

Dans ces deux cas, on envisage uniquement une seconde politique. Donc, au plus, il y a quatre politiques à considérer, c'est-à-dire quatre problèmes d'optimisation à résoudre.

Dans chacun de ces problèmes, il reste deux points non linéaires : le calcul de la valeur absolue de la fonction objectif et la disjonction des différentes sources de média. Nous modélisons ces deux difficultés en étendant notre programme linéaire avec des variables entières.

Disjonction des sources de média La disjonction des sources de média est directement modélisable par un ensemble de variables binaires, une par source, dont on impose que la somme soit égale à un. De la sorte, une seule source peut être sélectionnée à la fois. Par ailleurs, l'ensemble représentant les possibilités de finition d'une source est codée par trois variables binaires F_1, F_2 et F_3. Formellement, soit

n sources de média $s^1, \ldots, s^n$ on obtient le système de contraintes suivant :

$$
\begin{aligned}
s^1, \ldots, s^n &\in \{0,1\} \\
\sum_{i=1}^{n} s^i &= 1 \\
s_L &= \sum_{i=1}^{n} s^i_L \cdot s^i \\
s_{H_{min}} &= \sum_{i=1}^{n} s^i_{H_{min}} \cdot s^i \\
s_{H_{max}} &= \sum_{i=1}^{n} s^i_{H_{max}} \cdot s^i \\
s_{g^{mat}_{\Uparrow}} &= \sum_{i=1}^{n} s^i_{g^{mat}_{\Uparrow}} \cdot s^i \\
s_{g^{mat}_{\Leftarrow}} &= \sum_{i=1}^{n} s^i_{g^{mat}_{\Leftarrow}} \cdot s^i \\
s_{g^{mat}_{\Downarrow}} &= \sum_{i=1}^{n} s^i_{g^{mat}_{\Downarrow}} \cdot s^i \\
s_{g^{mat}_{\Rightarrow}} &= \sum_{i=1}^{n} s^i_{g^{mat}_{\Rightarrow}} \cdot s^i \\
s_{F_1} &= \sum_{i=1}^{n} s^i_{F_1} \cdot s^i \\
s_{F_2} &= \sum_{i=1}^{n} s^i_{F_2} \cdot s^i \\
s_{F_3} &= \sum_{i=1}^{n} s^i_{F_3} \cdot s^i
\end{aligned}
$$

Fonction linéaire par morceaux La fonction "valeur absolue" est une fonction continue linéaire par morceaux, ce qui nous permet de la traiter à l'intérieur de notre modèle. En effet, la manipulation de variables discrètes permet de modéliser de telles fonctions.

Soit $(p_1, \ldots, p_n) \in (\mathbb{R}^+)^n$ un n-uplet de réels positifs, tel que au plus deux variables *adjacentes*[5] soient non nulles. Soit $(x_1, \ldots, x_n) \in \mathbb{R}^n$ le n-uplet de réels représentant les abscisses des points de rupture de la fonction : $\forall i \in \{1, \ldots, n-1\}\,,\ x_i <$

[5]suivant l'indexation

x_{i+1}. Soit $(y_1, \ldots, y_n) \in \mathbb{R}^n$ le n-uplet représentant les ordonnées des points de rupture de la fonction et $f : [x_1, x_n] \to \mathbb{R}$ linéaire par morceaux tel que $\forall i \in \{1, \ldots, n-1\}$,f est strictement linéaire sur $[x_i, x_{i+1}]$. Pour modéliser f on introduit les variables de couples suivantes :

$$\begin{aligned} \sum_{i=1}^{n} p_i &= 1 \\ \forall i \in \{1 \ldots n\},\ p_i &\geq 0 \\ \sum_{i=1}^{n} p_i \cdot x_i &= x \\ \sum_{i=1}^{n} p_i \cdot y_i &= f(x) \end{aligned}$$

Reste à modéliser la contrainte d'adjacence. Pour y arriver de manière linéaire, nous introduisons un nouveau n-uplet $(c_1, \ldots, c_{n-1}) \in \{0,1\}^{n-1}$ représentant les *n-1* différents couples de variables adjacentes. Seul un couple peut être sélectionné à la fois, nous ajoutons donc une contrainte sur la somme des variables couples :

$$\begin{aligned} c_1, \ldots, c_{n-1} &\in \{0,1\} \\ \sum_{i=1}^{n-1} c_i &= 1 \\ p_1 &\leq 1 \cdot c_1 + 0 \cdot c_2 + 0 \cdot c_3 + 0 \cdot c_4 + \ldots + 0 \cdot c_{n-1} \\ p_2 &\leq 1 \cdot c_1 + 1 \cdot c_2 + 0 \cdot c_3 + 0 \cdot c_4 + \ldots + 0 \cdot c_{n-1} \\ p_3 &\leq 0 \cdot c_1 + 1 \cdot c_2 + 1 \cdot c_3 + 0 \cdot c_4 + \ldots + 0 \cdot c_{n-1} \\ &\vdots \\ p_n &\leq 0 \cdot c_1 + 0 \cdot c_2 + 0 \cdot c_3 + 0 \cdot c_4 + \ldots + 1 \cdot c_{n-1} \end{aligned}$$

La figure 4.1 illustre cette modélisation sur un exemple.

Ensembles Ordonnés spéciaux Ce type de contraintes étant fréquent dans le domaine de l'optimisation discrète, une notion facilitant leur manipulation a été introduite dans les années 70 [BT70] : les ensembles ordonnés spéciaux (Special Ordered Sets, SOS).

Il y a deux sortes de SOS.

Un SOS de type 1 (noté SOS1 ou S1 dans la littérature) est un ensemble de variables, telles qu'au plus une variable puisse prendre une valeur différente de zéro. Leur application la plus fréquente concerne les ensembles de variables binaires, ce

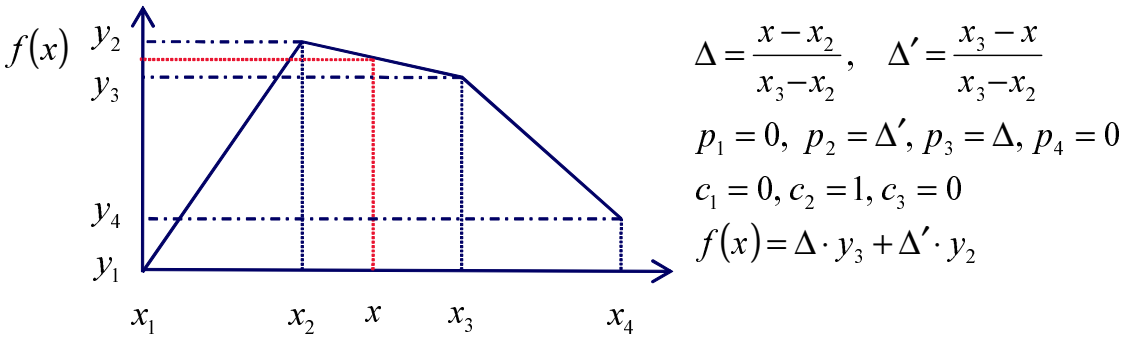


FIG. 4.1: Exemple de fonction linéaire par morceaux

qui permet d'exprimer le choix d'une possibilité parmi un ensemble. Dans notre cas, le choix d'une source de média peut être exprimé grâce à un SOS1.

Un SOS de type 2 (noté SOS2 ou S2) est un ensemble de variables ordonnées, telles qu'au plus deux variables successives puissent prendre une valeur différente de zéro. On voit que ce type d'ensemble permet de faciliter la modélisation d'une fonction linéaire par morceaux.

Les SOS sont largement utilisés dans le domaine de l'optimisation discrète et les moteurs de résolution comme LP_SOLVE savent les manipuler de façon optimisée. Le lecteur intéressé peut se référer à [BF76].

Un exemple complet reprenant cette modélisation est présenté dans la section C.2 de l'annexe C.

Comparaison avec la méthode classique La méthode de programmation classique procédurale n'est pas la mieux adaptée pour gérer ce type de problème de résolution. Bien que la plus rapide en temps de calcul, elle est beaucoup plus difficile à coder et surtout à maintenir et à faire évoluer.

4.3 Vers un modèle de préférence général

La politique d'impression est le modèle de préférence de l'imprimeur. Qu'il soit visible ou non, il est nécessaire à l'automatisation de la configuration des paramètres d'impression des documents.

Le modèle de préférence actuel présente deux défauts :

1. Il ne permet pas de distinguer tous les cas de figures, si bien que le comportement attendu est parfois incertain.

2. Il présente des limites d'expressivité importantes, notamment en matière de compromis.

Nous avons vu dans la section précédente comment traduire les politiques d'impression actuelles dans une fonction objectif associée à des contraintes. Ce formalisme nous a permis d'aborder le problème de la sélection de la configuration préférée comme un problème d'optimisation linéaire mixte en nombre entier, permettant l'utilisation de solveurs performants. Non seulement cette approche permet une résolution efficace mais son côté déclaratif facilite également la maintenance et l'évolutivité du code.

Cependant, elle soulève deux problèmes :

1. L'absence de cadre théorique pour la constitution de la fonction objectif et des contraintes associées ne nous permet pas de dire quelles sont les limites d'expression de cette modélisation.
2. Dans notre contexte de système de recommandation adaptatif, le modèle de préférence de l'imprimeur doit être appris automatiquement d'après ses choix précédemment réalisés. Or l'inférence de contraintes est particulièrement ardu (voir différentes approches sur le sujet : [CBQ, RS04, LL05, RVW09, BCOP07]).

Dans cette section, nous répondons à ces deux problèmes en proposant un nouveau modèle de préférence de l'imprimeur, plus général et basé sur la théorie. Nous présentons d'abord les bases théoriques des modèles de préférences. Puis nous explorons différents modèles proposés par la littérature et discutons de leur adéquation pour représenter les préférences de l'imprimeur. Nous concluons par la constitution d'une nouvelle famille d'attributs, permettant de modéliser les préférences de l'imprimeur par une fonction d'utilité additive, sans plus de contraintes associées.

4.3.1 Relation de préférence

Soit A l'ensemble des configurations d'impression possibles, étant donné un document à imprimer et un système d'impression. Nous voulons modéliser les préférences que peut avoir un imprimeur sur les éléments de A.

Pré-ordre

Le modèle le plus général pour exprimer une préférence sur les éléments de A est une relation binaire notée $\succeq$, sous-ensemble de A^2, qui représente la relation "au moins aussi bien que". Les propriétés généralement souhaitées pour une relation de préférences sont les suivantes :

– réflexivité : $x \succeq x$, pour tout x

- complétude : pour tout $x \neq y$, $non\,(x \succeq y) \Rightarrow y \succeq x$
- transitivité : $x \succeq y$ et $y \succeq z \Rightarrow x \succeq z$

Ces propriétés font de $\succeq$ un pré-ordre complet. Elles correspondent à une hypothèse de forte rationalité du décideur. Elles permettent de choisir facilement un élément (ou l'élément, s'il est unique) préféré de A.

Afin de mieux cerner les propriétés de la structure de pré-ordre complet, nous présentons la décomposition classique de cette relation en trois relations incompatibles entre elles : *indifférence*, *incomparabilité* et *préférence stricte*. Considérons une paire quelconque d'objets $\{x, y\}$, on est confronté à une et une seule des quatre situations suivantes :

1. L'indifférence : $x \succeq y$ et $y \succeq x$, notée $x \sim y$
2. l'incomparabilité : $non\,(x \succeq y)$ et $non\,(y \succeq x)$, notée $x \lesseqgtr y$
3. La préférence stricte : $x \succeq y$ et $non\,(y \succeq x)$, notée $x \succ y$
4. La préférence stricte : $y \succeq x$ et $non\,(x \succeq y)$, notée $y \succ x$

On peut noter quelques propriétés remarquables du pré-ordre complet :

- l'absence d'incomparabilité ($\lesseqgtr$ vide)
- la transitivité de l'indifférence $\sim$
- la transitivité de la préférence stricte $\succ$
- la combinaison simple de l'indifférence et de la préférence stricte ($[x \sim y \; et \; y \succ z \Rightarrow x \succ z]$ et $[x \succ y \; et \; y \sim z \Rightarrow x \succ z]$)

Assumption 5. *Nous supposons que l'imprimeur voulant configurer l'impression d'un document dans un contexte particulier a une telle relation dans la tête, et que celle-ci est stable. C'est-à-dire qu'il est capable d'ordonner les éléments de A, de sorte à en choisir un meilleur.*

Quasi-ordre

Si les propriétés du pré-ordre complet paraissent toutes assez naturelles, elles ne sont pas toujours respectées dans la réalité. Pour s'en persuader, on peut examiner l'exemple bien connu suivant, dû à Luce (1956).

Example 6. Supposons que l'on préfère boire le café sans sucre. Un serveur aligne devant nous une centaine de tasses de café. Dans la première il dépose un grain de sucre, dans la seconde deux grains, etc. L'expérience imaginaire consiste à dire que l'on ne trouve aucune différence entre le goût de deux tasses consécutives, elles nous sont donc indifférentes. Par contre, entre la première tasse et la dernière, la différence sera sensible : la transitivité de la relation d'indifférence ne sera donc pas vérifiée.

Dans cet exemple, la transitivité de $\sim$ étant violée, $\succsim$ perd également cette propriété et n'est donc plus un pré-ordre. On voit que cet exemple s'appuie sur une notion de seuil de sensibilité.

Considérer que les préférences du décideur sont représentables par un pré-ordre complet revient à faire des hypothèses fortes sur la rationalité du décideur. On peut affaiblir ces hypothèses de manière à permettre la non transitivité de l'indifférence. La structure engendrée s'appelle alors un quasi-ordre.

Doit-on prendre en compte un seuil de sensibilité dans notre modèle des préférences de l'imprimeur ? Certes ce seuil existe dans la pratique. On peut ainsi aisément remplacer les tasses de café par des imprimés et les grains de sucre par un incrément minuscule appliqué au zoom de l'image par exemple.

Example 7. Supposons qu'un client veuille imprimer une image à l'échelle 1 :1. L'imprimeur aligne devant lui 100 solutions d'impressions. La première respecte scrupuleusement l'échelle demandée. Sur la seconde, l'image a été agrandie avec un zoom de 100,1%. Sur la troisième, le zoom appliqué est de 100,2%, et ainsi de suite. Le client ne percevra pas la différence entre deux impressions successives. Elles lui seront donc indifférentes. En revanche entre la première et la dernière impression, la différence de 10% sera tout à fait perceptible, ce qui occasionnera une préférence stricte du client pour la première.

Cette expérience imaginaire met en lumière la difficulté pour un imprimeur de choisir entre deux configuration d'impression très proches, au sens de la perception de l'imprimeur. Cela peut d'autant plus facilement arriver du fait de l'approximation de la représentation numérique (arrondi sur la valeur du zoom) et/ou graphique (miniature) du résultat attendu, dans l'interface utilisateur.

Cela étant, on peut considérer que l'imprimeur ne va pas rencontrer souvent de telles situations dans sa démarche de configuration. Comme nous nous plaçons dans une démarche d'apprentissage automatique, on peut considérer ce type de cas comme du "bruit" ; ce qui nous permet de choisir un pré-ordre, plus facile à manipuler.

Dans la suite, nous considérons que les préférences de l'imprimeur sur les configuration d'impression peuvent être modélisées par un pré-ordre complet.

4.3.2 Fonction d'utilité

Pré-ordre complet et fonction d'utilité.

Le pré-ordre complet est la structure la plus communément utilisée dans le cadre de la représentation des préférences. Elle possède des propriétés intéressantes, en

particulier :

Theorem 8. *(Fishburn, 1970). Lorsque A est un ensemble fini ou dénombrable, une relation $\succeq$ sur A est un pré-ordre complet si et seulement s'il existe une fonction $u : A \to \mathbb{R}$ telle que $\forall a, b \in A$ on a*

$$a \succeq b \Leftrightarrow u(a) \geq u(b)$$

On appelle cette fonction u une *fonction d'utilité*. Elle est unique à une transformation strictement croissante près. Dans la terminologie de la théorie de la mesure, on dit que u définit une échelle ordinale. Cela signifie que les valeurs de u n'ont pas de signification dans l'absolu, pas plus que les distances entre elles. Seul leur ordre est significatif.

Note. La structure de quasi-ordre est également représentable par une fonction d'utilité, associée à un seuil d'indifférence $\eta \in \mathbb{R}^+$, tel que :

$$\forall a, b \in A \qquad \begin{cases} a \succ b & \Leftrightarrow \quad u(a) > u(b) + \eta \\ a \sim b & \Leftrightarrow \quad |u(a) - u(b)| \leq \eta \end{cases}$$

D'autres contraintes sur la représentation numérique sont nécessaires pour obtenir une structure plus riche, par exemple pour définir une échelle (cardinale) d'intervalle. Sur ce type d'échelle, les distances entre les valeurs de u sont significatives les unes par rapport aux autres. La fonction u doit alors également vérifier :

$$\forall a, b, c, d \in A \qquad (a, b) \gtrapprox (c, d) \Leftrightarrow u(a) - u(b) \geq u(c) - u(d)$$

où $(a, b) \gtrapprox (c, d)$ désigne la relation, telle que la différence de préférence entre a et b est au moins aussi forte que la différence de préférence entre c et d. Dans ce cas, la fonction u est unique, à une transformation affine strictement positive près.

Dans notre problématique, on veut pouvoir recommander la meilleure alternative, c'est-à-dire une configuration d'impression. Pour réaliser cela, seul l'ordre de préférence des différentes alternatives est nécessaire, ce qui fait de l'échelle ordinale un candidat suffisant pour le représenter. De plus, dans notre modèle d'interaction, rien ne nous permet a priori de comparer les différences de préférences entre deux couples d'alternatives.

Dans la suite, nous considérons donc que le modèle de préférence de l'imprimeur est une fonction d'utilité exprimée sur une échelle ordinale.

Espace de solutions approchées

Avant d'examiner plus en profondeur la structure de notre fonction d'utilité, il nous faut considérer un dernier point : notre ensemble d'alternatives n'est pas dénombrable. En effet, l'ensemble des configurations d'impression, comme défini au chapitre précédent, est un ensemble produit dont certains facteurs sont non dénombrables (zoom, marges et coupes) :

$$\begin{aligned} E &= M \times R \times Z \times G \times C \\ &= \mathbb{N}^2 \times \{1,2,3\} \times \{0,90,180,270\} \times \mathbb{R}_*^+ \times \left(\mathbb{R}^+\right)^4 \times \left(\mathbb{R}^+\right)^4 \end{aligned}$$

D'après le théorème de Cantor, un pré-ordre complet $\gtrsim$ sur un ensemble non dénombrable X est représentable par une fonction d'utilité, si et seulement s'il existe un sous-ensemble $\widehat{X} \subset X$ dénombrable et dense par rapport à la relation $\gtrsim$. Autrement dit :

$$\forall a, b \in \widehat{X},\ a \gtrsim b \ \Rightarrow\ \exists c \in \widehat{X}\ a \gtrsim c \gtrsim b$$

Notons que la manière la plus simple de contourner cette difficulté est certainement de considérer l'approximation "dénombrable" $\widehat{E}$ de E, tel que :

$$\begin{aligned} \widehat{E} &= M \times R \times \widehat{Z} \times \widehat{G} \times \widehat{C} \\ &= \mathbb{N}^2 \times \{1,2,3\} \times \{0,90,180,270\} \times \mathbb{N}_* \times \mathbb{N}^4 \times \mathbb{N}^4 \end{aligned}$$

où le zoom, les marges et les coupes sont approchés par des nombres entiers. $\widehat{E}$ est un produit d'ensembles dénombrables, donc l'est également. Notons que cette approximation n'est pas très restrictive. En effet, on peut choisir une échelle aussi fine que nécessaire pour le zoom comme pour la taille des marges et des coupes. Les équations d'admissibilité restent inchangées, mis à part un coefficient pour le zoom. Par exemple $\frac{1}{1000}$ si l'on se base sur une granularité au millième.

Nous considérons le modèle de préférence de l'imprimeur comme un pré-ordre complet sur l'ensemble approché des configurations d'impression admissibles $\widehat{E}$, représenté par une fonction d'utilité, exprimée sur une échelle ordinale.

4.3.3 Décomposition

Dans notre problématique, il est impossible d'exprimer, et encore moins de manipuler, la relation de préférence en extension, c'est-à-dire de représenter notre fonction d'utilité par un tableau de valeurs. Même en supposant que les dimensions du média, la taille des marges et des coupes, ainsi que le zoom soient exprimés sur des ensembles finis, le cardinal de leur produit est bien trop grand, si l'on veut conserver un minimum de précision[6].

Nous devons donc définir notre fonction d'utilité de manière plus compacte. Nous allons voir qu'il est possible de "décomposer" une fonction d'utilité de différentes manières, sous certaines conditions sur la structure de la relation de préférence associée.

Nous commençons par introduire la notion d'indépendance préférentielle et ses dérivées, qui sont au centre de toute décomposition.

Indépendance préférentielle

Notation. Soit $V = \{V_1, \ldots, V_n\}$ l'ensemble des variables décrivant une alternative, aussi appelé ensemble des attributs ou ensemble des critères. On note D_i le domaine des valeurs de V_i et D le domaine de V. On a $D = \prod_{i=1}^{n} D_i$.

Si X désigne un sous-ensemble de V alors $inst(X)$ désigne l'ensemble des instanciations possibles des variables de ce sous-ensemble. On a $inst(X) = \prod_{i \in \{1 \ldots n\}/V_i \in X} D_i$.

Si X et Y forme une partition de V (i.e. $X \cap Y = \emptyset$ et $X \cup Y = V$) et $x \in X$, $y \in Y$ alors on note $xy \in inst(V)$ l'instanciation correspondante de l'ensemble des variables.

On note $\succeq$ une relation de préférence correspondant à un pré-ordre complet. On désigne par $u_{\succeq}$, ou plus simplement u la fonction d'utilité ordinale $u : inst(V) \to \mathbb{R}$, représentant $\succeq$, c'est-à-dire $\forall a, b \in inst(V),\ a \succeq b \Leftrightarrow u(a) \geq u(b)$.

Definition 9. (Préférence ceteris paribus) Soit X et Y une partition de V et $x_1, x_2 \in inst(X)$. On dit que x_1 est préféré à x_2 *ceteris paribus*[7] si et seulement si

$$\forall y \in inst(Y) \quad x_1 y \succeq x_2 y$$

Autrement dit, la préférence de x_1 sur x_2 est constante, quelle que soit la valeur de Y.

[6] de l'ordre de 10^{45} pour une définition des dimensions du média, de la taille des marges et coupes et du zoom sur 10000 valeurs

[7] diminutif de l'expression latine "ceteris paribus sic stantibus" qui signifie "toutes choses étant égales par ailleurs"

Definition 10. (Indépendance préférentielle) Soit X et Y une partition de V. X est préférentiellement indépendant de Y pour $\succeq$ si et seulement si

$$\forall x_1, x_2 \in inst(X)\ \forall y_1, y_2 \in inst(Y) \quad x_1y_1 \succeq x_2y_1 \Leftrightarrow x_1y_2 \succeq x_2y_2$$

Autrement dit, les préférences sur les éléments de X sont indépendantes des valeurs de Y.

Definition 11. (Indépendance préférentielle mutuelle) Soit $Z = \{Z_1, \ldots, Z_k\}$ une partition de V. On dit que les $Z_1, \ldots, Z_k$ sont mutuellement préférentiellement indépendant pour $\succeq$ si et seulement si quels que soient X, Y formant une partition de Z, X est préférentiellement indépendant de Y pour $\succeq$.

Cette définition nous permet de présenter le modèle de préférence central dans la théorie de l'utilité.

Utilité additive

La décomposition la plus utilisée est certainement la décomposition additive de la fonction d'utilité.

Definition 12. (Utilité additive) Une fonction d'utilité $u : V \rightarrow \mathbb{R}$ est dite additive si et seulement si elle peut s'écrire sous la forme

$$u(v_1, \ldots v_n) = \sum_{i=1}^{n} u_i(v_i)$$

avec $\forall i \in \{1 \ldots n\}$ $u_i : V_i \rightarrow \mathbb{R}$. Les fonction u_i sont appelées les utilités marginales.

Theorem 13. *(Existence d'une fonction d'utilité additive) Soit $V = \{V_1, \ldots, V_n\}$. Les deux propositions suivantes sont équivalentes :*

- Les attributs $V_1, \ldots, V_n$ sont mutuellement préférentiellement indépendant pour $\succeq$.
- $\forall (v_1, \ldots v_n) \in inst(V)$ on peut écrire u sous la forme

$$u(v_1, \ldots v_n) = \sum_{i=1}^{n} u_i(v_i)$$

On voit que l'existence d'une fonction d'utilité additive suppose une forte indépendance entre attributs. Nous examinons dans la suite si cette hypothèse est valide pour les préférences de l'imprimeur.

Dépendances dans les préférences de l'imprimeur

Rappelons que l'espace des configurations d'impression admissibles est défini comme un sous-ensemble de l'ensemble suivant :

$$E = M \times R \times Z \times G \times C$$

où M désigne le média choisi (dimensions et pliage éventuel), R la rotation de l'image par rapport au média, Z le zoom effectué sur l'image, G les marges exprimées par rapport à l'orientation de l'image et C les coupes également relatives à l'orientation de l'image.

La fonction d'utilité de l'imprimeur doit donc permettre d'ordonner totalement les éléments de E. Examinons les préférences sur chaque attribut.

- La préférence sur le choix d'un média traduit la volonté d'obtenir un format standard (ou largeur standard, pour la sélection d'un rouleau particulier), ainsi qu'un pliage éventuel. Il est clair qu'elle ne dépend pas des autres attributs.
- Concernant la rotation, une préférence pour un angle particulier n'a pas vraiment de sens pour l'imprimeur. On peut supposer sans risque[8] qu'aucun angle n'est préféré à un autre *a priori*, quelles que soient les valeurs des autres attributs.
- La préférence sur le zoom doit permettre de traduire la volonté d'obtenir une échelle particulière, ainsi que les préférences vis-à-vis des échelles plus grandes et plus petites que celle demandée. Cela ne dépend pas non plus de la valeur des autres attributs.
- En revanche, les préférences sur les marges sont intrinsèquement dépendantes de la valeur de la rotation effectuée sur l'image, par rapport au média. En effet, si nos marges sont définies par rapport à l'image, alors l'expression des marges d'extrémités est dépendante de la rotation. Inversement, si les marges sont définies par rapport au média, les marges utilisateur sont alors dépendantes de la rotation.
- Enfin pour les coupes, on doit pouvoir traduire le seuil de tolérance et si elles sont autorisées au-delà du seuil. On suppose sans risque[9] que la préférence est identique pour tous les côtés, ce qui implique qu'elle est indépendante de la rotation.

Cette analyse informelle nous permet de constater deux choses :

1. Les préférences de l'imprimeur ne peuvent pas être représentées par une fonction d'utilité additive sur la famille d'attributs $\{M, R, Z, G, C\}$.

[8] de ne pas pouvoir représenter les préférences réelles de certains imprimeurs.
[9] Idem.

2. Mise à part la dépendance des préférences sur les marges par rapport à la rotation, considérer que les attributs sont préférentiellement indépendants les uns des autres ne limite pas, a priori, l'expressivité du modèle pour les imprimeurs.

4.3.4 Nouveau modèle de l'imprimeur

Modèles graphiques

Des modèles graphiques ont été développés, permettant de représenter de manière compacte des relations de préférence sur des espaces multi-attributs, présentant certaines particularités. Dans cette section, nous passons les quatre principaux en revue et discutons de leur aptitude à modéliser les préférences de l'imprimeur, ainsi qu'à être efficacement utilisés dans notre contexte.

CP-Net Les CP-Nets (pour Conditional Preference Networks) ont été introduits dans [BBHP99]. Un CP-Net permet de représenter des préférences ceteris paribus de type qualitatif. Il s'appuie sur la notion d'indépendance préférentielle conditionnelle.

Definition 14. (Indépendance préférentielle conditionnelle) Soit X, Y et Z une partition de V. X est préférentiellement indépendant de Y , conditionnellement à Z, pour $\succeq$, si et seulement si

$$\forall x_1, x_2 \in inst\,(X)\ \ \forall y_1, y_2 \in inst\,(Y)\ \ \exists z \in inst\,(Z) \quad tel\,que$$

$$x_1 y_1 z \succeq x_2 y_1 z \Leftrightarrow x_1 y_2 z \succeq x_2 y_2 z$$

Concrètement, un CP-Net est un graphe orienté tel que :
- chaque nœud correspond à un attribut
- un arc entre deux noeud $\overrightarrow{A_i A_j}$ signifie que les préférences du décideur sur les valeurs de l'attribut A_j sont dépendantes de la valeur de l'attribut A_i
- à chaque noeud sont associées autant de relations d'ordre (ou de pré-ordre) sur les éléments de l'attribut, que de combinaisons possibles des valeurs des parents du noeud.

Exemple 1. Pour illustrer le comportement des CP-Nets, prenons l'exemple simplifié d'un imprimeur. On s'intéresse au sous-ensemble d'attributs suivant :
- Média $\in$ {A1, A0}
- Rotation $\in$ {0°, 90°}
- Zoom $\in$ {50%, 100%}

- Marge au dessus de l'image $\in \{0, 10\}$
- Marge à droite de l'image $\in \{0, 10\}$
- Coupe $\in \{0, 5\}$

On sait que les préférences de l'imprimeur s'organisent comme suit :

- Le format A1 est préféré au A0, ceteris paribus.
- L'angle de rotation de 0° est préféré par rapport à 90°, ceteris paribus.
- Un zoom de 100% est préféré à 50%, ceteris paribus.
- Une marge d'extrémité "avant" est souhaitée, donc
  - pour une rotation de 0° : la marge haute est préférée à 10 et la marge droite à 0
  - pour une rotation de 90° : la marge haute est préférée à 0 et la marge droite à 10
- La coupe est préférée nulle, ceteris paribus

Ces préférences sont représentées par le CP-Net de la figure 4.2. On y voit la dépendance des préférences sur les marges par rapport à la rotation et, en creux les indépendances des préférences sur les autres attributs.

Que peut-on en déduire sur la relation de préférence de l'imprimeur ? On voit trivialement que la configuration d'impression (A1, 0°, 100%, 10, 0, 0) est la préférée ; de même la configuration (A0, 90°, 50%, 10, 0, 5) est la pire. Entre les deux, on peut déduire par exemple que (A1, 0°, 100%, 10, 0, 0) est préféré à (A1, 0°, 100%, 0, 10, 0). En revanche le modèle ne permet pas de déterminer la préférence entre (A1, 0°, 100%, 0, 10, 5) et (A1, 90°, 50%, 0, 10, 0).

La relation de préférence obtenue est un ordre partiel.

Notons que l'on déduit la préférence "(A1, 0°, 100%, 0, 10, 0) $>$ (A1, 90°, 100%, 0, 10, 0)" de la propriété de préférence ceteris paribus sur l'attribut *Rotation*. Il en résulte que l'attribut *Rotation* est plus important que les attributs sur les *Marges*. Le respect de la propriété ceteris paribus implique implicitement que les attributs "parents" sont plus importants pour le décideur que les attributs "enfants". En revanche, l'importance relative entre deux attributs sans lien de descendance n'est pas exprimable, ce qui engendre l'ordre partiel.

Un CP-Net peut-il représenter les préférences d'un imprimeur ? On voit qu'il permet de représenter la dépendance préférentielle des marges par rapport à la rotation. Cependant, un CP-Net permet de manipuler uniquement des attributs symboliques, ce qui est incompatible avec nos attributs.

L'impossibilité d'exprimer les importances relatives de deux attributs sans lien de parenté est un autre inconvénient.

Du point de vue de l'efficacité d'utilisation, les CP-Nets montrent également

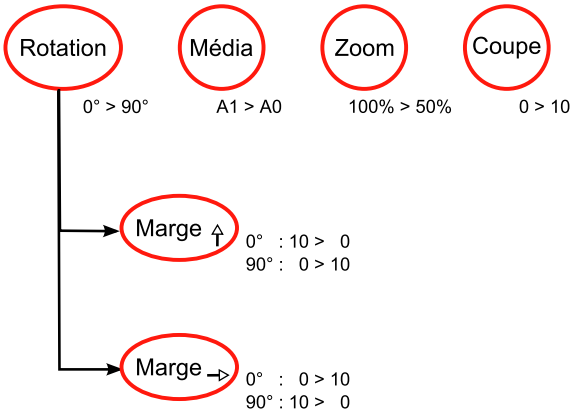


FIG. 4.2: Exemple de CP-Net

leurs limites pour notre problématique. En effet, résoudre le problème d'optimisation, c'est-à-dire trouver la meilleure configuration parmi l'ensemble des configurations admissibles, est un problème NP-complet (du fait que l'ensemble des solutions admissibles est strictement inclus dans l'ensemble produit des domaines de valeurs des attributs).

TCP-Net Introduit par [BD02], les TCP-Nets sont une extension des CP-Nets. Ils permettent d'exprimer l'importance relative entre attributs conditionnellement indépendants. Sont distinguées l'importance relative inconditionnelle et conditionnelle.

Definition 15 (Importance relative inconditionnelle)**.** Étant donnés deux attributs A_i et A_j conditionnellement indépendants sachant $Z \subseteq A - \{A_i, A_j\}$, A_i est inconditionnellement plus important que A_j, noté $A_i \rhd A_j$, si et seulement si

$$\forall z \in Z,\ \forall a_i, a_i' \in A_i,\ \forall a_j, a_j' \in A_j \textit{ tel que } a_i \succeq a_i' \textit{ sachant } z,$$

$$on\,a\ \ (a_i, a_j, z) \succeq \left(a_i', a_j', z\right) \tag{4.1}$$

Definition 16. Étant donnés deux attributs $A_i$et A_j conditionnellement indépendants sachant $Z = A - \{A_i, A_j\}$ et Z_1, Z_2 tel que $Z_1 \cap Z_2 = \emptyset$ et $Z_1 \cup Z_2 = Z$, A_i est plus important que A_j, étant donné $z_1 \in Z_1$, noté $A_i \rhd_{Z_1} A_j$, si et seulement si

$$\forall z_2 \in Z_2,\ \forall a_i, a_i' \in A_i,\ \forall a_j, a_j' \in A_j \textit{ tel que } a_i \succeq a_i' \textit{ sachant } z,\ on\,a\ :$$

$$(a_i, a_j, z_1, z_2) \succeq \left(a'_i, a'_j, z_1, z_2\right) \tag{4.2}$$

Graphiquement, un TCP-Net est un CP-Net étendu avec

- un ensemble d'arcs $\left\{\overrightarrow{A_iA_j}\right\}$, reliant des attributs inconditionnellement indépendant, exprimant la plus grande importance inconditionnelle de A_i par rapport à A_j.
- un ensemble d'arêtes $\left\{\overline{A_iA_j}\right\}$, reliant des attributs conditionnellement indépendants, exprimant la plus grande importance de A_i par rapport à A_j, conditionnellement à la valeur d'un ensemble de variables, appelé ensemble de sélection.
- un ensemble de fonctions, une fonction par arête, qui détermine l'importance relative entre les deux attributs concernés, conditionnellement aux valeurs de l'ensemble de sélection.

Exemple 2. Reprenons l'exemple précédent en ajoutant les informations suivantes au modèle de préférence.

- Le plus important pour l'imprimeur est de ne pas couper l'image.
- De plus, le zoom est plus important que le choix du média.

Ces nouvelles préférences sont représentées par le TCP-Net de la figure 4.3. On y voit que les importances inconditionnelles de la coupe par rapport à la rotation et au zoom ; et du zoom par rapport au média. Par transitivité, le choix de la coupe est donc plus important que tout autre critère. N'étant pas pertinente, aucune importance conditionnelle n'est représenté ici.

Ce nouveau modèle permet de représenter une relation d'ordre plus riche. Il permet par exemple de choisir la configuration (A1, 90°, 50%, 0, 10, 0) par rapport à (A1, 0°, 100%, 0, 10, 5), du simple fait de la coupe.

Bien que résolvant partiellement le problème de l'importance inter-attribut, les TCP-Nets ne comblent pas les autres limites des CP-Nets. Les attributs manipulés sont toujours symboliques. De plus, les TCP-Nets ne permettent pas d'exprimer des intensités de préférences inter-attribut suivant les valeurs que prennent ceux-ci.

Concernant la complexité de résolution du problème d'optimisation, elle a été moins étudiée que pour les CP-Nets. Mais les TCP-Nets étant une généralisation des CP-Nets, la résolution du problème d'optimisation est au moins aussi difficile.

GAI-Net Introduit par [BG95, GP04, GP05], les GAI-Nets s'appuient sur une décomposition basée sur l'indépendance additive généralisée.

Definition 17 (Décomposition additive généralisée)**.** Soit $A = \{A_1, \ldots, A_n\}$ et $Z_1, \ldots, Z_k \subseteq A$ tel que $\bigcup_{i=1}^{k} Z_i = A$. On dit que les $Z_1, \ldots, Z_k$ sont GAI-indépendant

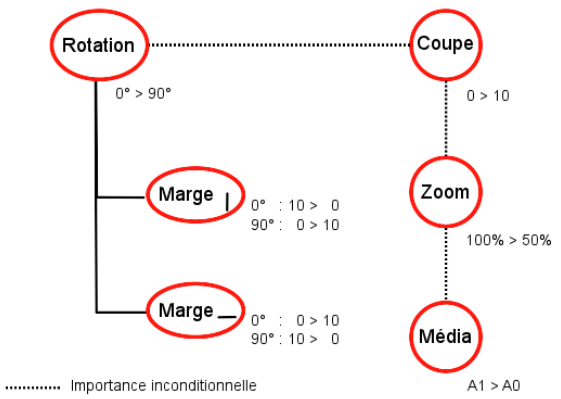


FIG. 4.3: Exemple de TCP-Net

(pour Generalised Additive Independance) pour une relation de préférence $\succeq$ si et seulement si il existe des fonctions $u_i : \prod_{A_j \in Z_i} A_j \to \mathbb{R}$ telles que

$$\forall a = (a_1, \ldots, a_n) \in A \quad u(a) = \sum_{i=1}^{k} u_i \left(a_{\Downarrow Z_i}\right)$$

où $a_{\Downarrow Z_i}$ est la projection de a sur le sous-ensemble Z_i.
On dit que u est GAI-décomposable.

Note. Les Z_i ne sont pas forcément disjoints.

Un GAI-Net est représenté par un graphe non orienté où
- chaque noeud représente une clique (i.e. un sous-ensemble Z_i de A).
- une arête entre deux noeuds implique que l'intersection de leurs cliques est non nulle.
- il existe au moins un chemin entre deux noeuds dont les cliques ont des éléments en commun ; les noeuds de ce chemin contiennent tous les éléments communs des deux cliques.

Exemple 3.

Les GAI-Nets peuvent potentiellement représenter toute fonction d'utilité (en posant trivialement tous les attributs dans une seule clique). Cependant, moins les cliques composant le graphe contiennent d'attributs, plus le graphe est intéressant. En effet, le problème d'optimisation a une complexité d'ordre exponentiel, en fonction de la taille de la plus grande clique.

De plus, pour pouvoir utiliser des algorithmes de résolution efficaces, le réseau GAI doit être sous forme d'arbre. Il est possible de transformer tout GAI-Net sous forme d'arbre, au prix d'un éventuel grossissement de la plus grande clique. Le

problème de trouver la transformation optimale, en terme de taille de clique, est NP-complet.

On voit que si ce modèle est théoriquement intéressant, il reste difficile à mettre en oeuvre de façon performante.

UCP-Net Les UCP-Nets, présentés dans [BBB01], constituent une sorte de compromis entre les CP-Nets (dont ils sont une extension) et les GAI-Nets (dont ils sont un sous-ensemble). Les UCP-Nets sont des CP-Nets dont les relations d'ordres sur les valeurs d'un attribut (associé à un noeud) ont été remplacés par des fonctions d'utilité conditionnelle. On peut noter qu'un UCP-Net représente une relation de préférence complète, contrairement aux CP-Nets.

Exemple 4.

Les UCP-Nets permettent une certaine souplesse dans l'expression des préférences. Cependant, ils sont sujets à diverses limites. D'une part le graphe doit être acyclique, d'autre part, pour respecter la propriété de préférence *ceteris paribus*, les préférences sur les valeurs des attributs parents doivent être plus intenses que celles des attributs enfants (par une limitation des valeurs des fonctions d'utilité), ce qui induit une priorité *a priori* sur les attributs. Ces deux conditions ne sont a priori pas limitatives pour notre problème. En effet, notre hypothèse d'indépendance conditionnelle sur les attributs définissant une configuration d'impression se représente par un graphe acyclique, le seul noeud parent étant celui de la rotation. De plus, l'imprimeur n'a a priori aucune préférence sur l'angle de rotation pour lui-même, ce qui permet de donner une même valeur de préférence pour chaque angle de rotation. De cette manière, la rotation est "prioritaire" mais le choix d'un angle particulier ne permet pas de discriminer deux configuration d'impression.

Du point de vue de l'efficacité à déterminer l'alternative optimale, les UCP-Nets présentent de bon résultats (temps linéaire en nombre d'attribut) dans le cas où l'espace des solutions admissibles est équivalent à l'espace produit des domaines des attributs, il en est différemment lorsque les ces deux espaces ne sont pas identiques.

Modèle additif

Nous avons vu que le modèle le mieux adapté pour représenter une politique d'impression est l'UCP-Net. Cependant l'UCP-Net permettant de représenter les préférences de l'imprimeur présente une particularité : son seul noeud parent, de l'attribut rotation, n'est pas discriminant dans le choix d'une configuration d'impression.

Cette caractéristique permet "d'aplanir" le modèle afin de retrouver une fonction d'utilité additive. En effet, nous avons vu que l'imprimeur peut avoir des préférences

sur les marges, soit par rapport à l'image, soit par rapport au média. Ces préférences, elles, ne changent pas quel que soit l'angle de rotation. Nous considérons donc deux jeux d'attributs pour représenter les marges. L'une par rapport à l'image, comme définit dans 3.2.3, que nous noterons maintenant G^i pour bien la différencier, et l'autre par rapport au média, noté G^m.

$$G^i = \left\{ \left(g^i_{\Uparrow}, g^i_{\Leftarrow}, g^i_{\Downarrow}, g^i_{\Rightarrow} \right) \in \left(\mathbb{R}^+ \right)^4 \right\}$$

$$G^m = \left\{ \left(g^m_{\Uparrow}, g^m_{\Leftarrow}, g^m_{\Downarrow}, g^m_{\Rightarrow} \right) \in \left(\mathbb{R}^+ \right)^4 \right\}$$

Cela implique une redondance puisque les valeurs de ces deux jeux d'attributs sont identiques, à une rotation près. Mais cela permet d'exprimer les préférences de l'imprimeur de façon additive. Les contraintes suivantes doivent être ajoutées :

$$r = 0 \quad \Rightarrow \quad \begin{cases} g^i_{\Uparrow} = g^m_{\Uparrow} \\ g^i_{\Leftarrow} = g^m_{\Leftarrow} \\ g^i_{\Downarrow} = g^m_{\Downarrow} \\ g^i_{\Rightarrow} = g^m_{\Rightarrow} \end{cases}$$

$$r = 90 \quad \Rightarrow \quad \begin{cases} g^i_{\Uparrow} = g^m_{\Leftarrow} \\ g^i_{\Leftarrow} = g^m_{\Downarrow} \\ g^i_{\Downarrow} = g^m_{\Rightarrow} \\ g^i_{\Rightarrow} = g^m_{\Uparrow} \end{cases}$$

$$etc...$$

Nous proposons d'utiliser une fonction d'utilité additive pour représenter les préférences de l'imprimeur à travers une famille d'attributs adaptée.

Nous pouvons également réaliser d'autres adaptations, de manière à simplifier le modèle. D'abord, nous pouvons à présent supprimer la rotation du modèle de préférence, sans perte d'expressivité. Les préférences sur les coupes étant supposées symétriques, nous pouvons ne conserver qu'une dimension de coupe. Enfin, les préférences sur les médias peuvent s'exprimer, non pas sur l'ensemble M, mais sur un ensemble M', restreint aux formats standard, rouleaux compris, tel que $M' = FS \cup LS$ où LS est l'ensemble des largeurs des rouleaux standards, par exemple :

$$LS = \{(A0, 841), (A1, 594), (A2, 420), (A3, 297), (A4, 210)\}$$

Plus formellement, on définit l'ensemble des configurations d'impression sur le nouvel ensemble d'attributs suivant :

$$\begin{aligned} E' &= M' \times Z \times G^i \times G^m \times C' \\ &= (FS \cup LS) \times \mathbb{N}_* \times \mathbb{N}^4 \times \mathbb{N}^4 \times \mathbb{N} \end{aligned}$$

4.4 Conclusion

Dans ce chapitre, nous avons présenté le modèle des préférences de l'imprimeur, actuellement en vigueur dans les produits d'Océ. Nous en avons proposé une formalisation, mis en lumière les relations entre les différents paramètres et pointé certaines limites comportementales.

Nous répondons ensuite à un des objectifs industriels de nos travaux de recherche : faciliter le développement et la maintenance de la gestion des politiques d'impression, pour les développeurs. Notre démarche générale est de considérer le problème sous un angle permettant au maximum l'utilisation d'algorithmes efficaces connus.

Nous proposons d'abord une modélisation mathématique du problème du choix automatique d'une configuration d'impression, d'après une politique d'impression. Ce modèle permet de considérer le choix d'une configuration comme un problème d'optimisation sous contraintes. De plus, il met en lumière les symétries qui réduisent la taille de l'espace de recherche. Cependant, le problème d'optimisation modélisé est non linéaire et présente notamment des contraintes d'implication, ce qui en fait un mauvais candidat à une résolution efficace. Nous montrons alors comment passer du modèle initial, qui correspond à un problème d'optimisation non linéaire, à une série minimale de problèmes linéaires mixtes en nombre entier. Cette reformulation permet d'attaquer la résolution avec des algorithmes efficaces existants. De plus, elle reste très déclarative, ce qui rend plus facile aux développeurs la manipulation de l'aspect logique de la fonctionnalité, tout en laissant le contrôle de côté.

Enfin, dans la troisième partie, nous examinons diverses modélisations de préférence, dans le but d'améliorer le modèle actuel. Ces modèles sont analysés suivant deux axes :

- le pouvoir d'expression
- la complexité des algorithmes permettant de trouver la meilleure solution

Après avoir donné les hypothèses sous lesquelles on peut formuler la relation de préférence de l'imprimeur comme une fonction d'utilité, nous passons en revue :

- le modèle classique : l'utilité additive
- quatre modèles graphiques : les CP-Nets, TCP-Nets, GAI-Nets et UCP-Nets.

Nous concluons sur le choix du modèle additif, sous réserve d'adapter les attributs sur

lesquels s'expriment les préférences. Cela permet de nous affranchir des dépendances préférentielles présentes initialement.

Chapitre 5

Apprentissage des préférences

Ce chapitre traite de l'apprentissage automatique du modèle de préférence établi au chapitre précédent, c'est-à-dire une fonction d'utilité additive, dont les utilités marginales sont des fonctions continues linéaires par morceaux. Le processus d'apprentissage est basé sur les décisions observées de l'utilisateur. Dans notre cadre applicatif, cela correspond aux configurations d'impression, paramétrées manuellement par l'imprimeur. Nous divisons cette tâche en deux problématiques distinctes.

- L'apprentissage du nombre de morceaux composant chaque fonction d'utilité marginale, ainsi que leur disposition. Dans la suite, nous faisons référence à cette tâche sous la dénomination d'apprentissage de la *structure* des utilités marginales.
- L'apprentissage des utilités marginales proprement dites, que nous désignons sous le terme de *pondération* des utilités marginales.

Après une introduction théorique sur l'apprentissage d'une fonction d'utilité à partir de décisions observées, nous proposons un état de l'art sur les techniques relatives à ces deux problématiques, développées conjointement dans les domaines de l'analyse multicritère et de l'apprentissage artificiel. Nous positionnons notre propre approche, qui s'inspire des travaux des deux communautés, et mettons en lumière son originalité pour l'apprentissage de la structure des utilités marginales.

5.1 Formalisation

Nous introduisons ici une formalisation du problème d'apprentissage, dans le cadre de l'apprentissage artificiel supervisé, et les notations employés dans le reste du chapitre.

5.1.1 Ensemble d'apprentissage

Soit n attributs numérotés de 1 à n et $A_i \subseteq \mathbb{R},\ i = 1 \dots n$, le domaine de définition[1] associé à l'attribut i.

On note $A = \{A_i,\ i = 1, \dots, n\}$ l'ensemble des domaines de définition des attributs. On suppose dans la suite que $\forall i,\ A_i$ est soit un intervalle fermé de $\mathbb{R}$ soit un sous-ensemble fini de $\mathbb{R}$.

Soit un ensemble d'alternatives $\mathbb{A} \subseteq \prod_{i=1}^{n} A_i$, nous notons $\succeq$ la relation de préférence sur $\mathbb{A}^2$, à apprendre.

L'ensemble d'apprentissage, également nommé dans la suite *ensemble des décisions observées*, noté $\mathbb{D}$, est un sous-ensemble de la relation de préférence, $\mathbb{D} \subseteq \succeq$.

5.1.2 Espace des hypothèses

On cherche à apprendre une fonction d'utilité additive u telle que :

$$u \quad : \quad \begin{cases} \mathbb{A} & \rightarrow & [0,1] \\ a & \rightarrow & \frac{1}{n}\sum_{i=1}^{n} u_i\left(a_i\right) \end{cases}$$

$$\text{et telle que} \quad \forall a, b \in \mathbb{A}$$

$$a \succeq b \quad \Leftrightarrow \quad u\left(a\right) \geq u\left(b\right)$$

Où $u_i : A_i \rightarrow [0,1]$ désigne l'utilité marginale correspondant au $i^{ème}$ attribut, supposée continue et linéaire par morceaux, si A_i est un intervalle de $\mathbb{R}$.

Paramètres du modèle

Soit u_i une fonction continue et linéaire par morceaux, définie sur l'intervalle $A_i = \left[\underline{A_i}, \overline{A_i}\right]$ de $\mathbb{R}$. Soit p_i le nombre de morceaux de u_i, $p_i \geq 1$. La fonction u_i est complètement définie par les données suivantes :

– le vecteur des points de rupture : X_i tel que

$$X_i = (x_1, \dots, x_{p_i+1}) \quad \in \quad \mathbb{R}^{p_i+1} \tag{5.1}$$

$$x_1 \quad = \quad \underline{A_i} \tag{5.2}$$

$$x_{p_i+1} \quad = \quad \overline{A_i} \tag{5.3}$$

$$\forall j \in \{1, \dots, p_i\}, \quad x_j \in A_i \quad et\ x_j < x_{j+1} \tag{5.4}$$

[1] numérique ou symbolique

– le vecteur Y_i des valeurs de la fonction u_i aux points de rupture, tel que

$$Y_i = (y_1, \ldots, y_{p_i+1}) \in [0,1]^{p_i+1} \tag{5.5}$$

Ces données permettent de calculer les valeurs de u_i par interpolation linéaire. Pour tout $a_i \in A_i$ il existe $j \in \{1, \ldots, p_i\}$ tel que $x_j \leq a_i \leq x_{j+1}$. On peut alors calculer la valeur de $u_i(a_i)$ tel que :

$$u_i(a_i) = \frac{x_{j+1} - a_i}{x_{j+1} - x_j} \cdot y_j + \frac{a_i - x_j}{x_{j+1} - x_j} \cdot y_{j+1}$$

Note. Les attributs symboliques (i.e. dont le domaine de définition est un ensemble *fini*) sont également représentables dans ce formalisme, tel que, si l'attribut j est symbolique, on a :

– p_j représente le cardinal de l'ensemble de définition $p_j = |A_j|$
– X_j est un vecteur d'index des éléments de A_j (ordonnés de manière quelconque).
– Y_j est le vecteur des valeurs de u_j

Pour des alternatives définies par leur ensemble de domaines de définition A, notre *espace d'hypothèse*s, c'est-à-dire l'ensemble des paramètres de notre modèle est donc :

$$\mathbb{H}_A = \left\{ \left((p_1, X_1, Y_1), \ldots, \left(p_{|A|}, X_{|A|}, Y_{|A|}\right)\right) / \forall i = 1 \ldots |A|,\ p_i \geq 1 \textit{ et } \textit{struct}\,(A_i, p_i, X_i, Y_i) \right\}$$

Où *struct* est le prédicat qui vérifie les conditions 5.1 à 5.5.

Nous nous plaçons dans le cadre d'une *incertitude stricte* sur l'espace des hypothèses, c'est-à-dire que l'on ne possède pas d'information a priori sur leur pertinence[2].

Le but de l'apprentissage est de trouver une hypothèse $h \in \mathbb{H}_A$, étant donné l'espace des attributs A, qui optimise une fonction de coût sur notre ensemble d'apprentissage.

5.2 Pondération des utilités marginales

Cette section concerne l'apprentissage des poids associés aux points de ruptures des utilités marginales, c'est-à-dire des vecteurs $Y_i,\ i = 1 \ldots n$. Il est supposé que les structures, p_i *et* X_i, de celles-ci sont fixées.

[2]Par exemple, lorsque l'information a priori est sous forme d'une distribution de probabilité, on parle d'*incertitude bayésienne*

Nous commençons par analyser le choix de la fonction de coût, utilisée dans le processus d'apprentissage, en partant de considérations statistiques. L'analyse concluant sur un manque de robustesse dans la détermination de l'utilité "optimale", nous considérons dans les deux sections suivantes deux approches permettant de palier ce problème : l'une par agrégation et l'autre par sélection.

5.2.1 Fonction de coût

Minimiser la fonction de coût doit permettre de trouver l'utilité dont la relation de préférence induite est aussi proche que possible de la relation de préférence réelle.

Afin de nous guider dans le choix d'une fonction de coût, nous nous appuyons sur une mesure statistique de corrélation entre variables ordinales. Nous faisons un bref rappel de deux de ces mesures dans le paragraphe suivant.

Mesures de corrélation entre deux variables ordinales

On peut considérer la relation de préférence du décideur comme une variable ordinale sur l'ensemble des alternatives. De même pour la relation induite par la fonction d'utilité que l'on veut apprendre. D'un point de vue statistique, le but de l'apprentissage est de corréler autant que possible ces deux variables ordinales. Pour évaluer cette corrélation, il existe deux mesures statistiques standard : le *coefficient de Spearman*, habituellement noté ρ et le *coefficient de Kendall*, noté τ. Ces deux mesures varient sur l'intervalle $[-1, +1]$ avec la signification suivante :

-1 pour deux classement exactement inverses

0 pour des classements indépendants

+1 pour des classements parfaitement identiques

Nous présentons ici les deux coefficients et expliquons pourquoi le τ de Kendall est mieux adapté à notre problème. Dans la suite, on note $\mathbb{A}_n = \{a_1, \ldots, a_n\}$ un ensemble de n alternatives et $C_1 : \mathbb{A}_n \to \{1 \ldots n\}$ et $C_2 : \mathbb{A}_n \to \{1 \ldots n\}$ deux ordonnancements (i.e. permutations) sur cet ensemble.

Coefficient de Spearman Le coefficient de corrélation de rang de Spearman, noté ρ, se calcule de la façon suivante :

$$\rho = 1 - \frac{6 \sum_{i=1}^{n} \left(C_1(a_i) - C_2(a_i)\right)^2}{n\left(n^2 - 1\right)}$$

On voit que le coefficient de Spearman s'intéresse à la différence des rangs. Or dans notre problématique, nous disposons uniquement d'informations d'ordre sur

des paires d'alternatives. On ne dispose donc pas de l'information nécessaire pour utiliser cette mesure comme fonction de coût.

Coefficient de Kendall Le coefficient de corrélation de rang de Kendall, noté τ, s'intéresse aux couples d'alternatives différemment ordonnées. Il se calcule de la façon suivante :

Soit la fonction $\Phi : \mathbb{A}_n \times \mathbb{A}_n \rightarrow \{0, 1\}$ tel que

$$\forall a, b \in \mathbb{A}_n \quad \Phi(a,b) \begin{array}{ll} = 0 & si \begin{cases} C_1(a) \leq C_1(b) \quad et \quad C_2(a) \leq C_2(b) \\ \qquad\qquad\qquad\quad ou \\ C_1(a) \geq C_1(b) \quad et \quad C_2(a) \geq C_2(b) \end{cases} \\ = 1 & sinon \end{array}$$

on peut alors écrire le coefficient de Kendall sous la forme :

$$\tau = 1 - 2 \cdot \frac{\sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \Phi(a_i, a_j)}{n(n-1)}$$

Ce qui revient à calculer la différence entre le nombre de couples bien ordonnés et le nombre de couples mal ordonnés, normalisée par le nombre total de couples. Ne considérant que l'information sur les couples, on voit que ce calcul se prête très bien à notre problématique. Le fait de ne posséder qu'une information partielle des relations de préférence n'est pas un problème.

C'est la mesure de référence pour construire une fonction de coût dans un problème d'apprentissage d'ordonnancement basé sur des comparaisons de paires.

Nous montrons dans la suite comment cette mesure peut se traduire concrètement, en examinant deux fonctions de coût s'y référant.

Minimiser le nombre de contraintes violées

La manière la plus naturelle de traduire le coefficient de Kendall dans une fonction de coût est de chercher à minimiser le nombre de couples mal ordonnés [JLS82]. Cela peut se traduire par la résolution du problème d'optimisation linéaire mixte suivant, grâce à l'introduction d'une variable d'ajustement booléenne ξ_{ab} par décision

$(a, b) \in \mathbb{D}$.

$$\begin{aligned} \text{minimiser} \quad & : \quad \sum_{(a,b)\in\mathbb{D}} \xi_{ab} \\ \text{sous les contraintes :} & \\ u(a) - u(b) + m \cdot \xi_{ab} \quad & \geq \delta \qquad \forall (a, b) \in \mathbb{D} \\ \xi_{ab} \quad & \in \{0, 1\} \qquad \forall (a, b) \in \mathbb{D} \end{aligned}$$

Où δ est un petit réel strictement positif, dont l'introduction est nécessaire pour éviter la solution triviale de tous les poids égaux à zéro [JLS82].

La constante m doit vérifier $m \geq \delta + 1$, de sorte que, la fonction d'utilité u étant normalisée sur $[0, 1]$, une contrainte $u(a) - u(b) + m \cdot \xi_{ab} \geq 0,\ (a, b) \in \mathbb{D}$ est vérifiée si $u(a) \geq u(b) + \delta$ ou $u(a) < u(b) + \delta$ et $\xi_{ab} = 1$. La fonction de coût à minimiser représente donc le nombre de contraintes violées.

Utiliser cette fonction de coût pose plusieurs problèmes. D'une part, le problème d'optimisation est linéaire mixte en nombres entiers. Sa résolution est donc assez coûteuse en temps de calcul. D'autre part, la fonction d'utilité obtenue n'est pas forcément robuste, c'est-à-dire que la garantie qu'elle reflète les véritables préférences du décideur sont faibles. En terme d'apprentissage, cela signifie qu'elle ne minimise pas forcément le risque réel.

Plus précisément, après résolution du problème, on se trouve dans l'une des trois situations suivantes :

1. Le cas trivial : une fonction d'utilité unique vérifie toutes les contraintes. C'est la situation idéale qui, en pratique, a peu de chance d'être rencontrée.
2. Cas du problème sous-contraint : lorsqu'il existe un ensemble (généralement infini) de fonctions d'utilité, vérifiant toutes les contraintes induites par les décisions observées[3]. Dans ce cas, rien ne garantit que l'utilité sélectionnée est meilleure que les autres utilités admissibles. On parle ici du problème de la *robustesse* de la solution obtenue.
3. Cas du problème sur-contraint : lorsqu'il n'existe aucune fonction d'utilité permettant de satisfaire toutes les contraintes en même temps. Que ce soit à cause du bruit dans les observations, d'un espace d'hypothèses trop pauvre, ou du fait d'une approximation heuristique, cette situation peut survenir assez rapidement. Dans ce cas, on distingue deux possibilités :
   (a) l'utilité minimisant le nombre de contraintes est unique, ce qui ne signifie pas que c'est la meilleure en terme de *généralisation*.

[3]Dans la pratique, on retrouve par exemple cette situation au départ de l'interaction du décideur avec le système de recommandation, lorsque le nombre de décisions observées est trop faible.

(b) il existe d'autres fonctions d'utilité violant exactement le même nombre de contraintes (identiques ou non), ce qui nous ramène au cas du problème sous-contraint.

Ces raisons nous amènent à examiner, dans le paragraphe suivant, une autre fonction de coût.

Maximiser les écarts

Nous introduisons ici une fonction de coût dont la minimisation conduit à la résolution d'un problème d'optimisation linéaire. Le critère d'optimalité maximise les écarts des scores d'utilité des alternatives de chaque décision observée. Pour cela, on considère à présent les variables d'ajustement réelles positives $\xi_{ab} \in \mathbb{R}^+,\ \forall\,(a,b) \in \mathbb{D}$. Le problème d'optimisation devient :

$$\begin{aligned} \text{minimiser} \quad &: \quad \sum_{(a,b)\in\mathbb{D}} \xi_{ab} \\ \text{sous les contraintes :} & \\ u\,(a) - u\,(b) + \xi_{ab} \quad &\geq \delta \qquad \forall\,(a,b) \in \mathbb{D} \\ \xi_{ab} \quad &\in \mathbb{R}^+ \qquad \forall\,(a,b) \in \mathbb{D} \end{aligned}$$

Comme $u\,(a) - u\,(b) \in [-1,1]$ on pose généralement $\delta = 1$, qui est la borne inférieure des valeurs de δ permettant de prendre en compte l'ajustement de toutes les contraintes du problème, quelle que soit la valeur du terme $u\,(a) - u\,(b)$.

Utiliser cette fonction de coût permet d'améliorer grandement le temps de calcul de la résolution. Cependant, le problème de la robustesse de l'utilité obtenue est toujours posé. D'une part, on peut se retrouver dans le cas du problème sous-contraint où plusieurs fonctions d'utilité ont un même score de zéro, avec la même absence de garantie sur la minimisation du risque réel de l'utilité sélectionnée. D'autre part, le cas où l'utilité optimale est unique est également à considérer. En effet, la résolution de ce problème d'optimisation ne donne qu'une solution approchée au sens de la minimisation du coefficient de Kendall. Par conséquent une solution présentant un score un peu moins bon peut s'avérer finalement mieux corrélée aux préférences réelles du décideur.

Ce problème a été abordé de différentes manières par les communautés de l'analyse multicritère et de l'apprentissage artificiel ; mais les deux s'accordent pour souligner son importance fondamentale.

On peut regrouper les différentes réponses apportées en deux classes principales :

1. l'agrégation de plusieurs utilités potentiellement intéressantes
2. l'ajout d'un critère supplémentaire de sélection de l'utilité optimale

Ces deux types d'approches sont examinés dans les sections suivantes.

5.2.2 Agrégation d'utilités

L'approche présentée dans cette section cherche à augmenter la robustesse de la fonction d'utilité *finale*, c'est-à-dire à minimiser le risque réel de l'ordonnanceur, en se basant sur un ensemble de fonctions d'utilité que nous appellerons *potentielles* ou *faibles*. On se trouve alors face à un nouveau problème de décision multicritère, où les différents critères sont les évaluations des alternatives sur les utilités faibles.

Cette méthode pose essentiellement deux problèmes :

1. déterminer un ensemble d'utilités potentielles (ou faibles)
2. agréger les résultats des utilités potentielles pour donner le résultat de l'*utilité finale* (ou *forte*)

Nous développons ces deux points dans les sections suivantes.

Déterminer un ensemble d'utilités potentielles

Nous présentons ici brièvement deux approches : la première est issue du domaine de l'analyse multicritère et la seconde du domaine de l'apprentissage artificiel.

Analyse post-optimale le terme d'analyse "post-optimale" [JLS82, JLS01] fait référence à l'utilité optimale, au sens de la fonction de coût précédemment définie. Sous cette appellation ont été développées plusieurs méthodes permettant d'agréger un ensemble d'utilités potentielles, générées à partir de l'utilité optimale obtenue. Nous donnons dans cette section le principe de la génération des utilités potentielles. Leur agrégation fait l'objet d'une partie de la section suivante.

Soit c^* le coût de l'utilité optimale et soit $k \in [0, 1]$, l'analyse post-optimale consiste dans l'exploration du polytope obtenu en ajoutant au système linéaire la contrainte :

$$\sum_{(a,b)\in\mathbb{D}} \xi_{ab} \leq kc^*$$

Une fois les coordonnées des sommets du polytope déterminées, on peut calculer les coordonnées d'un vecteur quelconque du polytope par simple combinaison linéaire des vecteurs sommets. Différentes méthodes peuvent être utilisées afin de calculer les coordonnées de ces sommets. Ces méthodes n'ayant pas fait l'objet de notre

étude, nous nous contentons ici de les évoquer. Le lecteur intéressé peut se reporter à [CC57, MN68].

RankBoost Issu du domaine de l'apprentissage artificiel, RankBoost [FISS98] est un algorithme d'apprentissage d'ordonnancement basé sur une méthode appelée "boosting", introduite dans [FS97]. Appliqué initialement au problème de classification, le principe du boosting est de combiner plusieurs classifieurs "faibles", de manière pondérée, afin d'obtenir un classifieur "fort", présentant un meilleur taux de classification. L'idée maîtresse est d'entraîner les classifieurs faibles, les uns après les autres sur l'ensemble d'apprentissage entier, mais en modifiant l'importance des différents exemples, de sorte à équilibrer les lacunes du classifieur précédent. Aucune hypothèse sur la nature des classifieurs faibles n'est posée a priori, ce qui laisse une totale liberté de choix. Avec RankBoost, cette idée est adaptée au problème de l'ordonnancement.

RankBoost opère par tour. Au tour $t \in \{1 \dots T\}$ correspond l'entraînement d'un "ordonnanceur" sur l'ensemble d'apprentissage. L'apprentissage de cette fonction se fait en minimisant une fonction de coût, pondérée par une distribution d'importance sur les exemples. Dans notre contexte, cela revient à apprendre une fonction d'utilité additive $u^t : \mathbb{A} \to [0, 1]$, optimale pour la fonction de coût pondérée.

La distribution d'importance est mise à jour à chaque tour afin que les exemples les moins bien modélisés par l'ordonnanceur précédent deviennent prioritaires pour l'apprentissage du prochain ordonnanceur. L'algorithme s'arrête après un nombre prédéfini de tours. L'ensemble des utilités potentielles est donc construit incrémentalement, guidé par les résultats des fonctions de coût successives.

Agréger les résultats

Différentes procédures d'agrégation peuvent être envisagées afin de combiner les "notes" des différentes fonctions d'utilité. Il semble néanmoins que seul un petit nombre d'entre elles l'ont été effectivement dans le domaine de l'aide à la décision multicritère, sous le nom de *critères de décision*. Nous en présentons ici une liste représentative. La procédure d'agrégation de l'algorithme RankBoost est présentée à la fin.

Soit $\widehat{U}$ l'ensemble des utilités potentielles obtenues, supposé fini, nous notons $u^* = f\left(\widehat{U}\right)$, l'utilité finale où f est la fonction d'agrégation.

Critères basés sur les extrêmes Le critère *Maximin*, proposé par [Wal50] et cité par [SH04] pour le cas des utilités incertaines, sélectionne une alternative dont

la plus mauvaise note (évaluée sur l'ensemble des hypothèses admissibles) est la plus haute.

$$\forall a \in \mathbb{A} \quad u^*(a) = \min_{u \in \widehat{U}} u(a)$$

Maximin est un critère qualifié de "pessimiste" . Son pendant optimiste est le critère *Maximax* :

$$\forall a \in \mathbb{A} \quad u^*(a) = \max_{u \in \widehat{U}} u(a)$$

Le critère de la *valeur centrale* supporté par [SH01], est un compromis entre les deux, tel que :

$$\forall a \in \mathbb{A} \quad u^*(a) = \frac{1}{2}\min_{u \in \widehat{U}} u(a) + \frac{1}{2}\max_{u \in \widehat{U}} u(a)$$

L'*index de pessimisme-optimisme,* introduit dans [Fre86], généralise les trois approches précédentes, dans le sens où il propose de choisir l'alternative optimisant une somme pondérée de la pire et de la meilleure utilité.

$$\forall a \in A \quad u^*(a) = \alpha \min_{u \in \widehat{U}} u(a) + (1-\alpha) \max_{u \in \widehat{U}} u(a)$$

où le paramètre α représente l'inclinaison du décideur entre pessimisme et optimisme.

Le regret minimum Le critère de *regret minimax*, introduit par [Sav51] dans le contexte de la décision dans l'incertain (sur les conséquences des choix), a été ensuite proposé par [BBB01, SH01] pour faire des décisions robustes dans le cadre d'incertitudes sur les fonctions d'utilité. Le regret maximal est défini comme la plus grande différence entre le score de l'alternative choisie et le score d'une autre alternative, sur l'ensemble des utilités. Ce critère choisit l'alternative dont le regret maximal est minimal. La fonction d'utilité finale est donc :

$$\forall a \in \mathbb{A} \quad u^*(a) = 1 - \max_{u \in \widehat{U}} \max_{a' \in \mathbb{A}} [u(a') - u(a)]$$

Le défaut principal de cette approche est son incapacité à vérifier le principe d'indépendance aux alternatives non pertinentes. Cela signifie que pour un même ensemble d'utilités potentielles, le regret minimax ne donnera pas forcément les mêmes scores aux éléments communs de deux ensembles d'alternatives différents.

Maximisation de l'espérance Le dernier critère présenté est celui de la *maximisation de l'espérance*. L'absence d'une quelconque information de probabilité sur

les hypothèses admissibles, peut être considérée équivalente à une probabilité equirépartie sur l'ensemble des hypothèses. Dans ce cadre, une décision rationnelle optimale maximise l'espérance :

$$\begin{aligned} \forall a \in A \quad u^*(a) &= \underset{u \in U}{E}\, u(a) \\ &= \frac{1}{n} \sum_{u \in \widehat{U}} u(a) \end{aligned}$$

On peut remarquer que les critères réellement envisagés dans la problématique de l'aide à la décision multicritère, pour décider sous utilité strictement incertaine, présentent peu ou pas de paramètres. En effet, le choix d'un critère particulier peut déjà s'avérer délicat et le réglage de paramètres, tels que les poids dans une somme pondérée, complique encore le problème. En effet, dans le contexte de l'aide à la décision, cela implique davantage d'interactions entre le décideur et l'expert, ce qui est contraignant. En revanche, du point de vue de l'apprentissage artificiel, cela l'est moins. Cet "avantage" est exploité par l'algorithme RankBoost.

RankBoost Nous avons vu au paragraphe 5.2.2, que RankBoost opère incrémentalement. A chaque nouvelle itération, l'algorithme apprend un nouvel ordonnanceur, en optimisant une fonction de coût pondérée, de sorte à apprendre en priorité les exemples les moins bien ordonnés par l'ordonnanceur précédent. A chaque nouvel ordonnanceur u est associé un coefficient α_u. Celui-ci est calculé de sorte à minimiser une borne supérieure sur le risque empirique de l'ordonnanceur fort.

La fonction d'utilité globale est construite comme la somme des scores des ordonnanceurs faibles, pondérés par leurs coefficients.

$$\forall a \in A \quad u^*(a) = \sum_{u \in \widehat{U}} \alpha_u u(a)$$

Pour résumer, RankBoost construit incrémentalement, à la fois l'ensemble des utilités potentielles $\widehat{U}$ (dont le nombre est un paramètre T) et la fonction d'agrégation, en déterminant les poids $\alpha_u,\ u \in \widehat{U}$ de la somme.

5.2.3 Sélection d'une utilité

Nous présentons dans cette section des méthodes permettant de sélectionner une fonction d'utilité parmi l'ensemble des fonctions admissibles. Ces méthodes s'appuient sur un terme de régularisation, permettant de discriminer les différentes

fonctions admissibles.

Terme de régularisation Dans [Bly02], le système apprend les préférences de l'utilisateur en fonction des comparaisons qu'il fait, selon un processus itératif alternant choix de l'utilisateur et recommandation du système. Le modèle de préférences est l'utilité additive pondérée :

$$\forall a \in \mathbb{A} \quad u(a) = \sum_{i=1}^{n} u_i(a_i)$$

Les alternatives sont définies par un vecteur de *critères* et non d'attributs. Cela signifie que la valeur d'un critère est directement proportionnelle au degré de préférence du décideur sur cet axe particulier. En général, les critères sont bornés entre 0 et 1, où 0 indique la valeur rédibitoire et 1 la valeur préféré. Les utilités marginales s'écrivent telles que

$$\forall i = 1..n,\ \forall x \in [0,1], \quad u_i(x) = w_i x$$

où $w_i \in [-1,1]$. L'apprentissage se fait par la résolution du programme linéaire suivant :

$$\begin{aligned} \text{maximiser} \quad & : \quad c^\top \cdot w \\ \text{sous les contraintes :} & \\ u(a) - u(b) \quad & \geq 0 \qquad \forall (a,b) \in \mathbb{D} \\ -1 \leq \quad w_i \quad & \leq 1 \end{aligned}$$

On peut noter que cette méthode ne donne pas de solution dans le cas d'un système sur-contraint. Dans le cas inverse, la fonction objectif permet de discriminer parmi les hypothèses concurrentes satisfaisant toutes les contraintes. L'optimisation ne porte donc plus sur un critère de satisfaction des contraintes (qui doivent par ailleurs être toutes vérifiées) mais sur un terme de *régularisation* de l'espace des hypothèses [GJP95].

Le vecteur c est un paramètre, appelé vecteur "objectif". Il est intéressant de noter que le programme linéaire est résolu avec plusieurs vecteurs c différents, générant des comportements divers et permettant ainsi d'obtenir une palette de recommandations[4]. En particulier, l'auteur précise deux d'entre eux :

[4] C'est également une façon d'obtenir un ensemble d'utilités faibles à agréger par la suite, comme présenté à la section 5.2.2

- $c = w^{t-1}$, où w^{t-1} est le vecteur résultat calculé à l'étape précédente (t-1) du processus de recommandation[5]. Cela amène à une nouvelle utilité aussi proche que possible de l'ancienne, donnant à l'algorithme un comportement "conservateur".
- $c = \sum_{(a,b)\in\mathbb{D}} a - b$, la somme des vecteurs de décision. La nouvelle utilité sera aussi proche que possible de la moyenne des décisions observées.

Remarquons ici que l'auteur ne propose pas de critère "optimal" d'optimisation mais plusieurs afin de balayer un spectre plus ou moins large de comportements.

Minimisation du risque structurel Les récents développements théoriques de l'apprentissage artificiel supervisé, autour de la théorie de la régularisation ([GJP95]) et de la minimisation du risque structurel ([Vap95]), offrent un cadre permettant de définir ce "critère optimal". Le principe inductif introduit par Vapnik, sous le nom de minimisation du risque structurel, étend celui de la minimisation du risque empirique par la considération du potentiel de l'espace des hypothèses. Développé dans le cadre de la classification, il peut également être appliqué aux problèmes de régression et d'ordonnancement. Basée sur ce principe, une méthode, désignée sous l'acronyme SVM[6] ([Vap95]), a été mise au point.

Séparateur à vaste marge Lorsqu'il existe une infinité d'hyperplans séparateurs entre deux classes, celui qui maximise la distance minimale (appelée "marge"), entre l'hyperplan et les exemples d'apprentissage, est optimal au sens du *risque structurel*, c'est-à-dire qu'il présente la meilleure garantie en généralisation. Si l'hyperplan est défini par le vecteur w, alors maximiser la marge revient à minimiser $\|w\|$. On notera une propriété remarquable : seuls les points situés "à la frontière" de chaque classe jouent un rôle dans le calcul de l'hyperplan optimal. Ces points sont appelés *vecteurs support*. Les autres points, situés "à l'intérieur" des frontières, ne sont pas pris en compte. Soit $\{(x_i, y_i)\,,\ i = 1 \ldots n\}$ l'ensemble d'apprentissage tel que $\forall i\ y_i \in \{-1, 1\}$ La recherche de l'hyperplan optimal est un problème d'optimisation sous contraintes qui peut s'écrire sous la forme vectorielle suivante :

$$\begin{array}{ll} \mathit{Minimiser} & \|w\|^2 \\ \mathit{Sous\ les\ contraintes} & y_i \left(w^\top x_i + w_0\right) \geq 1 \quad i = 1, \ldots, n \end{array}$$

[5] i.e. avec au minimum une décision de moins dans l'ensemble d'apprentissage

[6] pour Suppot Vector Machine ou, une traduction française : Séparateur à Vaste Marge

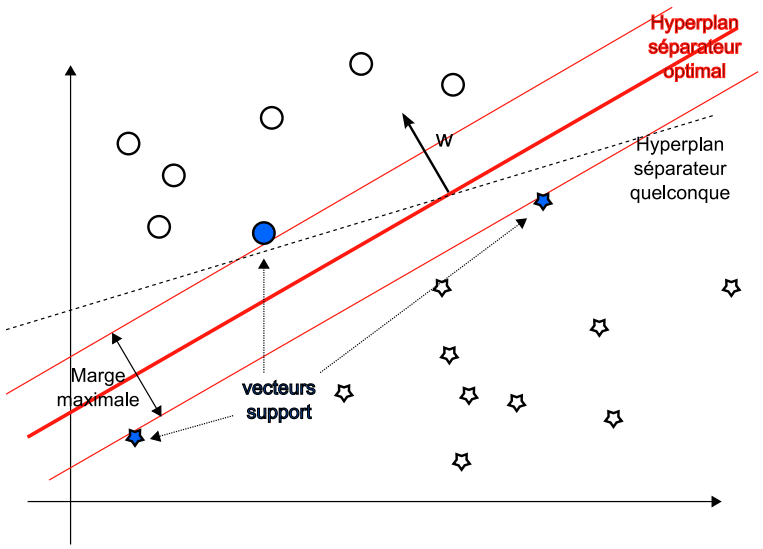


FIG. 5.1: Exemple d'un hyperplan optimal en dimension 2

Lorsque les deux classes sont non linéairement séparables, on peut chercher l'hyperplan qui classe bien le plus d'exemples. On se place dans le cadre de la recherche d'une solution approchée en utilisant les variables d'ajustement qui rendent les contraintes flexibles, menant au problème d'optimisation suivant :

$$\begin{array}{lll} \textit{Minimiser} & \|w\|^2 + C \sum\limits_{(a,b)\in\mathbb{D}} \xi_{ab} & \\ \textit{Sous les contraintes} & y_i \left(w^\top x_i + w_0\right) \geq 1 - \xi_{ab} & i = 1, \ldots, n \\ & \xi_{ab} \in \mathbb{R}^+ & \forall\,(a,b) \in \mathbb{D} \\ & C > 0 & \end{array}$$

Le paramètre C règle le compromis entre maximisation de la marge et minimisation des contraintes violées.

Appliquée à la problématique de l'ordonnancement, par exemple dans l'algorithme RankSVM ([Joa02]), le système devient :

$$\begin{array}{rcll} \text{minimiser} & : & \|w\|^2 + C \sum\limits_{(a,b)\in\mathbb{D}} \xi_{ab} & \\ \text{sous les contraintes :} & & & \\ u(a) - u(b) & \geq 1 - \xi_{ab} & \forall\,(a,b) \in \mathbb{D} & \\ \xi_{ab} & \in \mathbb{R}^+ & \forall\,(a,b) \in \mathbb{D} & \\ C & > & 0 & \end{array}$$

On peut également contourner le problème de non séparabilité linéaire en cherchant un séparateur linéaire dans un espace de redescription des données ayant plus de dimensions. En effet, on sait que plus le nombre de dimensions de l'espace de description des exemples est grand, plus l'existence d'un hyperplan séparateur entre deux classes est probable. Donc lorsqu'il n'existe pas de séparateur linéaire dans l'espace de description initial χ, l'idée est de chercher un hyperplan séparateur dans un espace de redescription des exemples $\Phi(\chi)$, de plus grande dimension. Supposons que cet hyperplan existe, on le note w. Le classifieur obtenu est alors

$$\forall x \in \chi \quad \begin{cases} w \cdot \Phi(x) > 0 & \Longrightarrow \quad classe\,1 \\ w \cdot \Phi(x) < 0 & \Longrightarrow \quad classe\,2 \end{cases}$$

Par exemple, supposons que $\chi = \mathbb{R}^2$ et qu'il n'existe pas de séparateur linéaire entre nos deux classes dans cet espace. On peut redécrire tout $x = (x_1, x_2) \in \chi$ dans $\mathbb{R}^3$ par $\Phi(x) = \left(x_1^2, x_2^2, \sqrt{2}x_1x_2\right)$ et chercher un séparateur linéaire dans ce nouvel espace.

Bien que très puissante, cette approche n'est pas adaptée dans notre cas. En effet, l'ordonnanceur généré n'est pas linéaire. Son utilisation dans notre programme de recherche de la configuration d'impression préférée est donc impossible.

Cependant, le principe de maximisation de la marge est particulièrement intéressant, du fait de son optimalité en terme de généralisation. En terme de performance, les différentes méthodes présentées dans cette section, notamment RankBoost et RankSVM, présentent généralement des résultats assez proches.

Nous avons choisi d'utiliser l'algorithme RankSVM, avec noyau linéaire, afin de résoudre notre problème d'apprentissage. Dans le cas d'un espace de description des données trop pauvre, nous devons donc utiliser un mécanisme de redescription, qui génère un nouvel espace également linéaire. C'est l'objet de la section suivante.

5.3 Structuration des utilités marginales

Le problème que nous examinons dans cette section est la découpe de l'ensemble de valeurs des fonctions d'utilité marginale. Cette découpe doit satisfaire deux types de contraintes contradictoires :

1. minimiser le nombre de segments de chaque fonction d'utilité marginale
2. maximiser le nombre de décisions observées validées par la fonction d'utilité globale

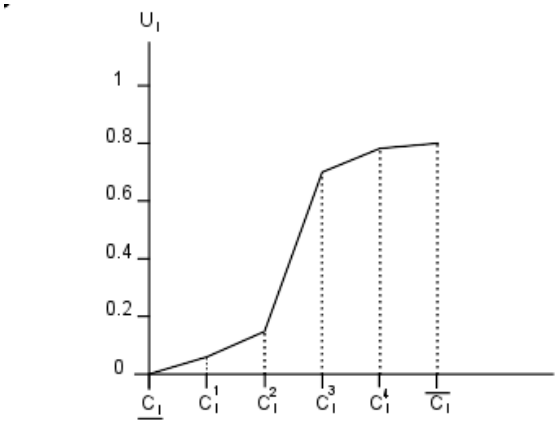


FIG. 5.2: Exemple d'utilité marginale

Il est évident que plus le nombre de segments d'une fonction affine par morceaux augmente, plus elle peut approcher finement n'importe quelle fonction, en particulier la fonction d'utilité de l'imprimeur. En revanche, plus le nombre de morceaux augmente, moins le calcul de l'alternative optimale est performant. En réalité, la performance décroît rapidement du fait de la nature exponentielle du problème.

Il est donc impératif de déterminer des points de rupture qui permettent, d'une part de maximiser l'adéquation de l'utilité globale aux décisions observées et, d'autre part de minimiser le nombre de morceaux constituant les utilités marginales.

5.3.1 Travaux relatifs

Dans la méthode UTA [JLS82], les fonctions d'utilité marginales $u_i : \left[\underline{A_i}, \overline{A_i}\right] \rightarrow \mathbb{R}^+$ sont modélisées par des fonctions continues linéaires par morceaux. Le nombre de segments est un paramètre α_i et les points de rupture $\left\{c_i^{k_j},\ i = 1 \ldots n,\ k_j = 1 \ldots \alpha_i\right\}$ sont répartis uniformément. De plus, chaque utilité marginale est supposée *croissante*.

Cette approche, bien qu'intéressante, pose deux problèmes majeurs, vis-à-vis de notre contexte applicatif :

1. Les fonctions d'utilité marginale ne sont pas forcément monotones. En particulier, considérons le zoom. Il peut varier sur l'intervalle $[0.01, 100]$ et le zoom préféré est généralement autour de 100%. On peut faire une remarque similaire pour les préférences sur chaque marge (que ce soit par rapport à l'image ou au média). Il est donc nécessaire de prendre en compte la non-monotonie des préférences de l'imprimeur.
2. Les domaines de définition des fonctions marginales ne sont pas uniformément

important. Pour reprendre l'exemple du zoom, la "zone" $[50\%, 300\%]$ sera généralement plus riche en "delta de préférence" que la zone $[500\%, 10000\%]$. D'où l'intérêt de découper la première avec un "grain plus fin", de façon à pouvoir approcher au mieux des préférences réelles de l'imprimeur.

Nous proposons une approche heuristique, basée sur la minimisation de la distance de Hausdorff modifiée, permettant d'estimer la position d'éventuels extrema sur les fonctions d'utilité marginales.

5.3.2 Intervalles contenant un extremum

Dans cette section, nous analysons l'existence d'extrema locaux, à partir de décisions globales observées. Nous montrons que si préférence globale et locale sont confondues (cas à un attributs), on peut déduire l'existence certaine d'extrema, borner leur positions et connaître leur nature (minimum ou maximum) suivant les décisions observées.

Nous montrons ensuite comment notre approche s'étend au cas général (nombre quelconque d'attributs), grâce à l'association d'une mesure de probabilité sur les préférences locales.

Cas à un attribut

Lorsque les alternatives sont définies sur un seul attribut, préférence globale et locale sont confondues. On supposera dans la suite que le domaine de définition A_1 de cet attribut est un intervalle fermé de $\mathbb{R}$. Dans ce cas, nous pouvons déduire l'existence d'extrema locaux, à l'intérieur d'intervalles, à partir de certaines observations.

Proposition 18 (Existence d'optimum local)**.** *Soit quatre alternatives* $a_1, a_2, b_1, b_2 \in \mathbb{A} = A_1$*, tel que* $u_1(a_1) \geq u_1(b_1)$ *et* $u_1(a_2) \geq u_1(b_2)$*, alors*

$$a_1 \leq b_1 \leq b_2 \leq a_2 \Rightarrow \exists c \in [a_1, a_2] \text{ tel que c est un miminum local de } u_1 \quad (5.6)$$
$$a_1 \leq b_2 \leq b_1 \leq a_2 \Rightarrow \exists c \in [a_1, a_2] \text{ tel que c est un miminum local de } u_1 \quad (5.7)$$
$$a_1 \leq b_2 \leq a_2 \leq b_1 \Rightarrow \exists c \in [a_1, b_1] \text{ tel que c est un extremum local de } u_1 \quad (5.8)$$
$$b_2 \leq a_1 \leq b_1 \leq a_2 \Rightarrow \exists c \in [b_2, a_2] \text{ tel que c est un extremum local de } u_1 \quad (5.9)$$
$$b_1 \leq a_1 \leq a_2 \leq b_2 \Rightarrow \exists c \in [b_1, b_2] \text{ tel que c est un maximum local de } u_1 \quad (5.10)$$
$$b_1 \leq a_2 \leq a_1 \leq b_2 \Rightarrow \exists c \in [b_1, b_2] \text{ tel que c est un maximum local de } u_1 \quad (5.11)$$

Preuve de l'implication 5.10. Par exemple, si u_1 est croissante sur $[b_1, b_2]$ alors elle est constante sur $[a_2, b_2]$ et admet donc une infinité d'extrema locaux sur cet intervalle, en particulier des maxima.

Si u_1 est décroissante sur $[b_1, b_2]$ alors elle est constante sur $[b_1, a_1]$ et admet donc une infinité d'extrema locaux sur cet intervalle, en particulier des maxima.

Par conséquent, si u_1 est monotone sur $[b_1, b_2]$, la proposition est vérifiée.

Supposons à présent que u_1 n'est pas monotone sur $[b_1, b_2]$, c'est-à-dire qu'il existe une partition

$$I_1, \ldots, I_k \subset [b_1, b_2]\,,\ k \geq 2,\ \bigcap_{j=1}^{k} I_j = \emptyset,\ et\ \bigcup_{j=1}^{k} I_j = [b_1, b_2]$$

$$avec\ \forall j \in 1 \ldots n-1,\ \forall x \in I_j,\ \forall y \in I_{j+1} \quad x < y$$

telle que si u_1 est croissante sur I_j alors u_1 est décroissante sur I_{j+1} et inversement.

Si $k > 2$ alors on a nécessairement au moins un intervalle croissant suivit d'un intervalle décroissant, ce qui implique l'existence d'un maximum local.

Si $k = 2$ alors on a soit une configuration <croissance, décroissance> et c'est terminé ; soit une configuration <décroissance, croissance>. Montrons par l'absurde que cette dernière configuration est impossible. On suppose donc que $k = 2$ et que u_1 est décroissante sur I_1 et croissante sur I_2. Comme $u_1(b_1) \leq u_1(a_1)$ on a nécessairement, ou bien $a_1 \in I_1$ ce qui implique que u_1 est constante sur $[b_1, a_1]$ et admet donc une infinité de maxima locaux, ou bien $a_1 \notin I_1$ et donc $[a_2, b_2] \subseteq [a_1, b_2] \subseteq I_2$, ce qui contredit $u_1(a_2) \geq u_1(b_2)$. □

Note. Nous omettons les preuves des autres implications, qui sont similaires.

On remarque toutefois que tout couple d'observations n'implique pas l'existence d'un extremum. Il est facile de voir qu'une configuration, telle que $a_1 \leq b_1 \leq a_2 \leq b_2$ par exemple, n'implique nullement l'existence d'un extremum. Plus généralement, on ne peut déduire l'existence d'un extremum local que si l'ordre entre a_1 et b_1 est opposé à celui entre a_2 et b_2.

Proposition 19 (Condition suffisante de l'existence d'un optimum local)**.** *Soit* $a_1, a_2, b_1, b_2 \in \mathbb{R}$*, et une fonction* $u : \mathbb{R} \to \mathbb{R}$*, continue sur* $\mathbb{R}$*, tel que* $u(a_1) \geq u(b_1)$ *et* $u(a_2) \geq u(b_2)$*, alors*

$$(a_1 - b_1)(a_2 - b_2) < 0 \quad \Rightarrow \quad \exists c \in [\min(a_1, a_2, b_1, b_2), \max(a_1, a_2, b_1, b_2)]$$

tel que u admet un extremum local en c.

Remark. L'inverse est faux en général.

Démonstration. On a

$$(a_1 - b_1)(a_2 - b_2) < 0 \quad \Leftrightarrow \quad \begin{cases} (a_1 - b_1) \geq 0 \quad et \quad (a_2 - b_2) \leq 0 \\ \qquad\qquad\qquad ou \\ (a_1 - b_1) \leq 0 \quad et \quad (a_2 - b_2) \geq 0 \end{cases}$$

$$\Leftrightarrow \quad \begin{cases} a_1 \geq b_1 \quad et \quad a_2 \leq b_2 \\ \qquad ou \\ a_1 \leq b_1 \quad et \quad a_2 \geq b_2 \end{cases}$$

Les conditions ($a_1 \geq b_1$ *et* $a_2 \leq b_2$) et ($a_1 \leq b_1$ *et* $a_2 \geq b_2$) sont symétriques, c'est pourquoi nous ne traitons dans la suite que le premier cas. On veut donc montrer

$$a_1 \geq b_1 \; et \; a_2 \leq b_2 \quad \Rightarrow \quad \exists c \in [\min(a_1, a_2, b_1, b_2), \max(a_1, a_2, b_1, b_2)]$$

tel que u admet un extremum local en c.

Sur l'intervalle $I = [\min(a_1, a_2, b_1, b_2), \max(a_1, a_2, b_1, b_2)]$, si la fonction u n'est pas monotone, elle admet nécessairement au moins un extremum local.

Considérons le cas où elle est monotone. Elle est soit croissante soit décroissante. Si u est croissante sur I alors elle est constante sur $[a_2, b_2]$ puisque $u(a_2) \geq u(b_2)$, d'où l'existence d'une infinité d'extrema sur $u(a_2) \geq u(b_2)$. De même si u est décroissante sur I alors elle est constante sur $[b_1, a_1]$ puisque $u(a_1) \geq u(b_1)$, d'où l'existence d'une infinité d'extrema sur $[b_1, a_1]$. □

Cas à n attributs

Dans le cas général d'un nombre d'attributs quelconque, la correspondance entre une préférence globale $a \succeq b$ c'est à dire $u(a) \geq u(b)$ et une préférence locale, par exemple $u_i(a_i) \geq u_i(b_i)$, n'est pas directe. Si l'on considère chaque décision globale ($a \succeq b$) *indépendamment*, on déduit facilement que :

$$\begin{array}{lll} \forall i \in 1 \ldots n \quad tel\, que & a_i \neq b_i \\ P(a_i \succeq b_i) & = & \dfrac{2^{n'-1}}{(2^{n'}) - 1} \end{array}$$

où $P(a_i \succeq b_i)$ est la probabilité que la valeur a_i soit préférée (au sens large) à la valeur b_i et n' est le nombre d'attributs sur lesquels a et b présentent des valeurs différentes. On voit que

Algorithm 5.1 Génération des intervalles "d'extremum"

Soit $\mathcal{I} = \{(I_k, P_k),\ k \in \mathbb{N}\}$ où $\forall k,\ I_k$ est un intervalle inclus dans A_1 et P_k est une mesure de probabilité que u_1 admette au moins 1 extremum sur I_k.
$\mathcal{I} \leftarrow \emptyset$
Soit $\mathbb{D}_i =$
$\{((a_i, b_i), p_i),\ a_i \succeq b_i,\ i = 1 \dots n\}$ la projection de l'ensemble des décisions globales observées sur l'attribut i, associé à une probabilité
Soit $\mathbb{D}_i^{\succeq} = \{((a, b), p) \in \mathbb{D}_i / a > b\}$
et $\mathbb{D}_i^{\preceq} = \{((c, d), p') \in \mathbb{D}_i / c < d\}$
$\forall ((a, b), p) \in \mathbb{D}_i^{\succeq}$
  $\forall ((c, d), p') \in \mathbb{D}_i^{\preceq}$
    $\mathcal{I} \leftarrow ([\min(a, b, c, d), \max(a, b, c, d)],\ p \cdot p')$

$$P(a_i \succeq b_i) \longmapsto_{n' \to \infty} \frac{1}{2}$$

Ce qui correspond bien à l'intuition que plus le nombre d'attributs augmente, moins une préférence globale implique une quelconque préférence locale.

En faisant l'hypothèse de l'indépendance des décisions globales (ce qui n'est pas gênant dans notre contexte) la probabilité de l'intersection des préférences locales se déduit facilement. Soit $(a \succeq b)$ et $(c \succeq d)$ deux décisions globales, on a :

$$\begin{array}{ccc} \forall i \in 1 \dots n \quad tel\, que & a_i \neq b_i \, et \, c_i \neq d_i & \\ P(a_i \succeq b_i \cap c_i \succeq d_i) & = & P(a_i \succeq b_i) \cdot P(c_i \succeq d_i) \end{array}$$

Ce simple calcul nous permet d'associer une probabilité à l'existence d'extrema locaux sur des intervalles particuliers, à partir des décisions globales.

Pour un attribut particulier, l'algorithme 5.1 permet d'obtenir un ensemble d'intervalles contenant au moins un extremum avec une certaine probabilité, en fonction des décisions globales observées.

5.3.3 Extrema des utilités marginales

Dans cette section, nous traitons du problème suivant : comment déterminer les extrema des fonctions d'utilité marginales, à partir des décisions observées ? Autrement dit, comment faire le lien entre, d'une part les préférences observées globale-

Algorithm 5.2 Construction d'une partition probabilisée de A_1

Soit $K = \left\{\underline{A_1}, \overline{A_1}\right\} \cup \left\{a \in A_1 / \exists\, (I, P) \in \mathcal{I}\ a = \underline{I}\ ou\ a = \overline{I}\right\}$
Soit $\widehat{K} = \left(a_1, \ldots, a_{|K|}\right)$ la liste croissante des éléments de K
Soit $P = \left(p_1, \ldots, p_{|K|-1}\right)$ où $\forall i =$
$1..|K|-1,\ p_i$ est une mesure de probabilités
que u_1 admette au moins 1 extremum sur $[a_i, a_{i+1}]$.
Initialisation des probabilités :

$\forall i = 1..|K|-1,\ p_i \leftarrow 0$

Mise à jour des probabilités :
$\forall k = 1 \ldots |K|-1$

$\forall\, (I,\, p_I) \in \mathcal{I}$

Si $[a_k, a_{k+1}] \subseteq I$ alors

$p_k \leftarrow p_k + \frac{a_{k+1}-a_k}{|I|}\,(p_I - p_k \cdot p_I)$

ment sur des couples d'alternatives et, d'autre part les valeurs extrêmes de préférence sur chaque attribut, composant ces alternatives ?

Nous basons notre approche sur une variante de la distance de Hausdorff modifiée [DJ94] qui est une mesure de similarité entre deux ensembles de points.

Une fois l'ensemble d'intervalles $\mathcal{I}$ calculé, nous combinons ses éléments afin de construire une partition de A_1 (voir l'algorithme 5.2) telle qu'à chaque partie, on associe une probabilité que u_1 admette au moins un extremum dedans.

Une fois la partition construite et une mesure de probabilité associée à chacune des parties, il reste à estimer un ensemble d'extrema. Pour y parvenir nous utilisons l'algorithme 5.3. La mise à jour de $\widehat{K}$ et P sachant l'ensemble E courant, utilisée dans l'algorithme revient à la reconstruction d'une partition probabilisée, dans laquelle on ne tient plus compte des intervalles contenant les points de E.

5.3.4 Résultats

La figure 5.3 montre empiriquement l'augmentation de la précision de la position ainsi estimée des extrema, en fonction du nombre de décisions observées. Dans cette expérience, les alternatives soumises à décision sont aléatoirement choisies, de façon indépendante et identiquement distribuée (i.i.d.), de même que les fonctions d'utilité utilisées pour décider. Seul est paramétré leur nombre d'extrema.

Pour calculer l'adéquation des extrema estimés avec les extrema réels, nous uti-

Algorithm 5.3 Détermination d'un ensemble d'extrema locaux

Etant donnés $\widehat{K}$ et P
Soit E l'ensemble des extrema à retourner.
$E \leftarrow \emptyset$
Faire
  $i^* \leftarrow \arg\max_{i=1..\left|\widehat{K}\right|-1} \frac{p_i}{a_{i+1}-a_i}$
  Si $p_{i^*} > seuil$ alors
    $E \leftarrow E \cup \left(a_i + \frac{a_{i+1}-a_i}{2}\right)$
    Mettre à jour $\widehat{K}$ et P sachant E
Tant que $p_{i^*} > seuil$

lisons la distance[7] de Hausdorff modifiée [DJ94]

Definition 20. Soit S et T deux ensembles (non vides) de points sur un espace métrique quelconque, et d une distance sur cet espace. La distance de Hausdorff modifiée, entre ces deux ensembles, est définie par

$$MHD(S,T) = \max\{g_d(S,T), g_d(T,S)\}$$

où g_d est la distance de Hausdorff modifiée relative, définie par

$$g_d(S,T) = \frac{1}{|S|}\sum_{p\in S}\min_{q\in T}\{d(p,q)\}$$

La figure 5.4 permet de comparer les performances de notre approche avec l'approche RankSVM munie d'un noyau RBF (pour "Radial Basis Function"). Les paramètres ont été réglés par une recherche en grille. Ces résultats sont des moyennes (sur dix exécutions) de résultats obtenus après apprentissage sur un ensemble de 50 exemples et testés sur un ensemble de 1000 exemples. Ils montrent que notre approche présente globalement des performance équivalentes à l'approche par noyau non linéaires, tout en gardant l'avantage de la linéarité.

[7] ce n'est pas une distance au sens mathématique du terme car elle ne vérifie pas le principe d'inégalité triangulaire

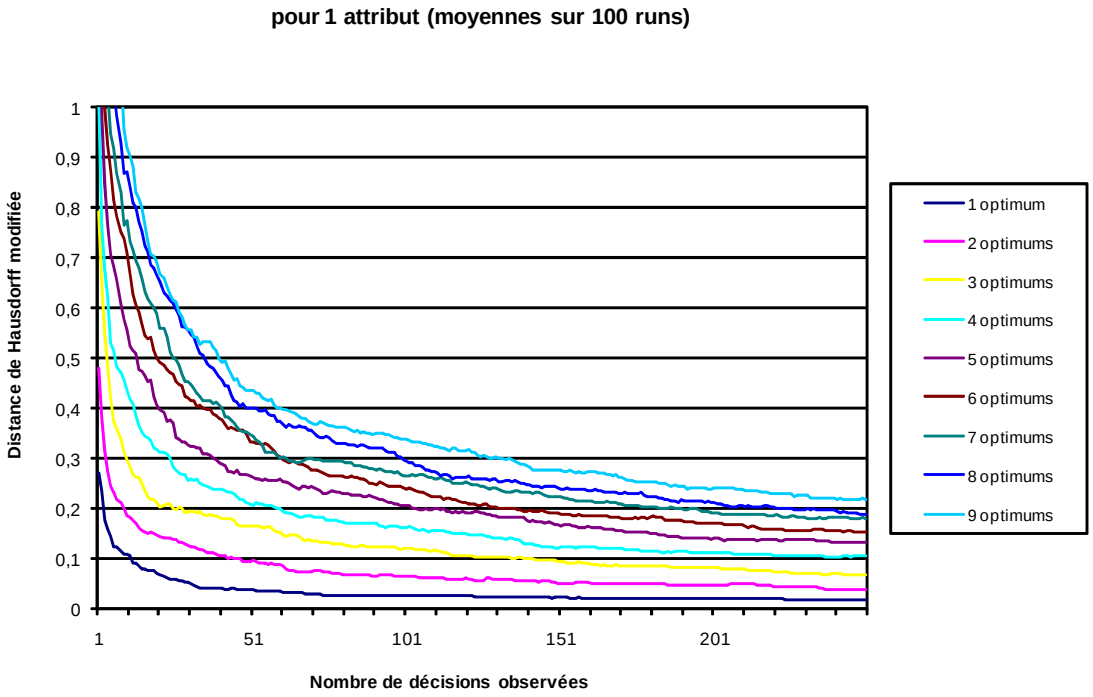


FIG. 5.3: Convergence de la distance de Hausdorff modifiée

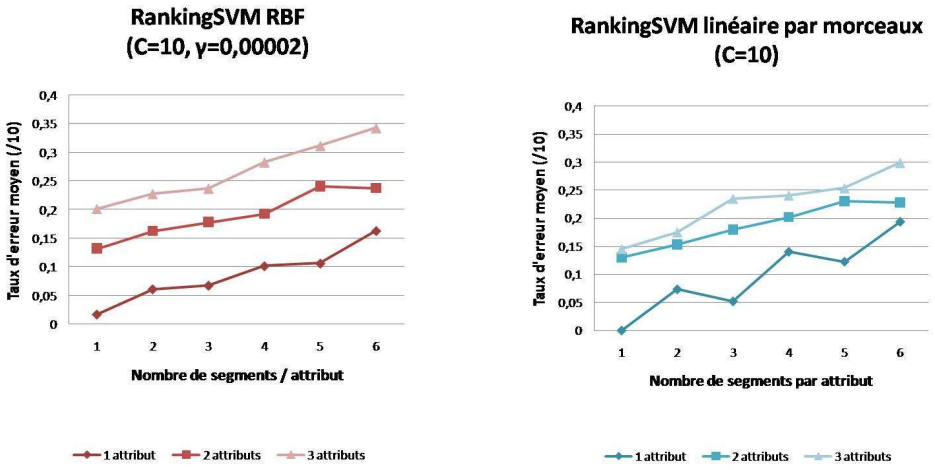


FIG. 5.4: Performances comparées

5.4 Application à la recommandation d'impression

Dans cette section, nous présentons un système de recommandation adaptatif de configuration d'impression. Basé sur le modèle de préférence de l'imprimeur proposé au chapitre 4, couplé avec l'algorithme d'apprentissage présenté dans les sections précédentes de ce chapitre, notre système de recommandation prend la forme d'un panneau graphique, inséré dans l'interface de l'application d'Océ présenté dans la section 1.1.2.

5.4.1 Interface utilisateur

Le système de recommandation s'insère dans l'interface utilisateur actuelle sous la forme d'un panneau rétractable proposant des configurations d'impression alternatives. Deux modes sont possibles :

1. le mode "recommandation simple"
2. le mode "recommandation multiple"

Recommandation simple

Le mode "recommandation simple" (voir Fig. 5.5) correspond à la proposition de différentes alternatives de configuration pour un document particulier. Ce panneau devient actif lorsque l'utilisateur sélectionne un seul document dans le panneau principal. Il se présente en deux parties :

1. à gauche : une représentation graphique de "grande" taille, permettant à l'utilisateur de se faire une idée plus précise du résultat.
2. à droite : des petites représentations graphiques de plusieurs configurations alternatives pour ce document.

Si aucune alternative du panneau de recommandation n'est sélectionnée par l'utilisateur, le graphique de gauche représente la configuration du document actuellement retenue (affichée en petit dans le panneau principal).

Lorsque l'utilisateur sélectionne une configuration alternative dans la partie droite du panneau de recommandation, son graphique s'affiche "en grand" dans la partie gauche. Une seule configuration alternative peut être sélectionnée à la fois.

Une seule des alternatives proposées correspond au modèle courant des préférences de l'utilisateur. Les autres correspondent à des politiques enregistrées[8] dont le comportement diffère de celui du modèle courant, sur le document sélectionné.

[8] Que ce soit des politiques d'impression pré-enregistrées ou apprises lors de sessions précédentes. Elles peuvent par exemple être classées par ordre de fréquence d'utilisation.

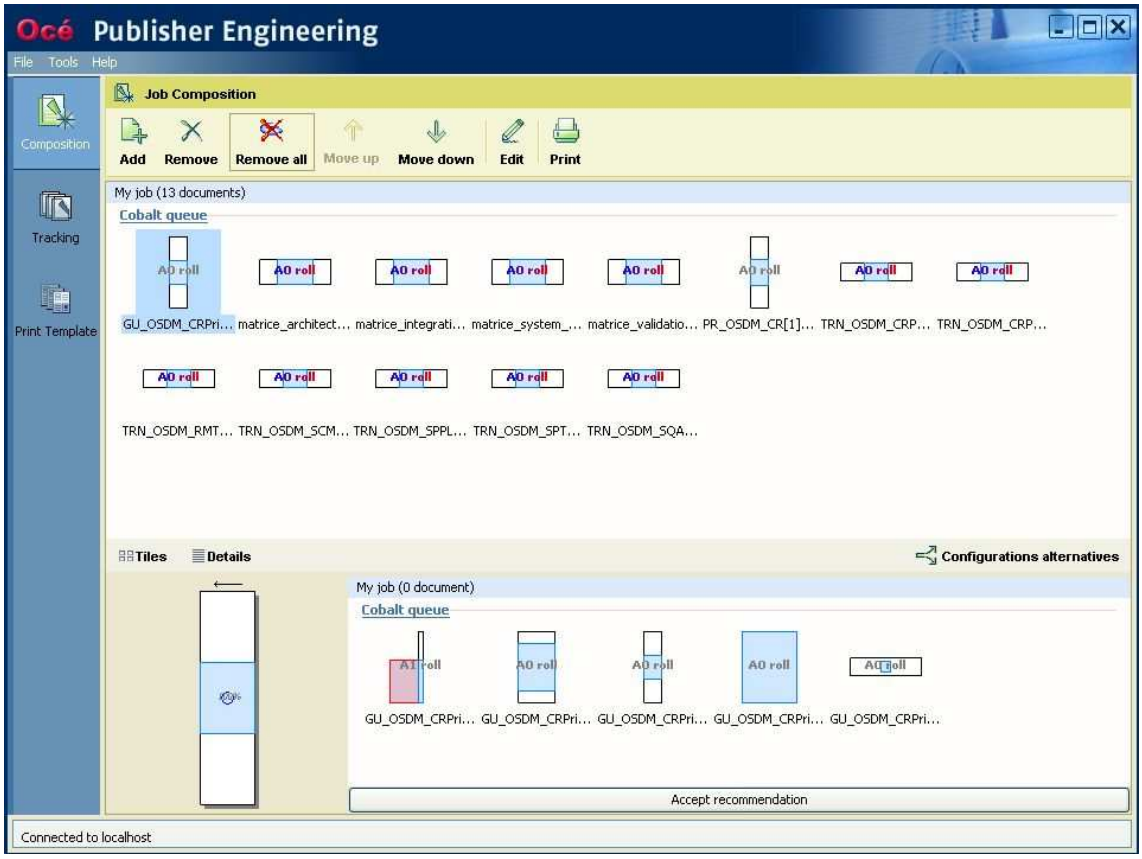


FIG. 5.5: Panneau de recommandation simple

Le bouton d'acceptation situé en bas de la partie droite permet à l'utilisateur de valider une configuration alternative. Dans ce cas, la représentation du panneau principal est mise à jour avec la nouvelle configuration sélectionnée.

Ce panneau permet donc à l'utilisateur d'explorer différentes configurations alternatives pour un document particulier et, éventuellement d'en choisir une. Utile dans la phase d'apprentissage des préférences de l'utilisateur, ce panneau s'avère peu efficace dès lors que les préférences sont apprises et qu'il faut mettre à jour les configurations d'impression restantes d'un grand nombre de documents.

C'est à cette tâche qu'est dédié le mode de recommandation multiple.

Recommandation multiple

Le mode "recommandation multiple" (voir Fig. 5.6) correspond à la proposition d'une alternative de configuration d'impression pour chaque document sélectionné par l'utilisateur dans le panneau principal. Le panneau de recommandation multiple devient actif dès que la sélection de l'utilisateur dépasse un document.

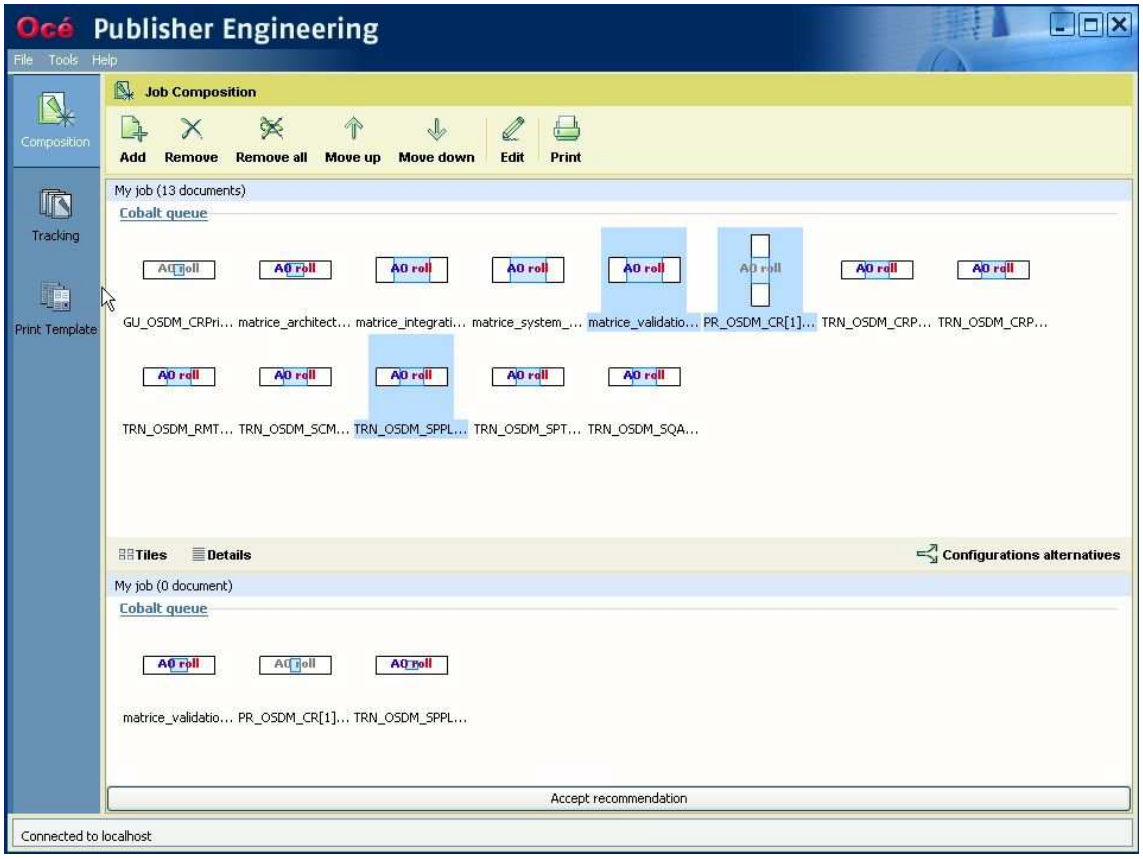


FIG. 5.6: Panneau de recommandation multiple

Il se présente comme une liste de configurations alternatives (sous forme de petites représentations graphiques). Chaque configuration alternative correspond à un et un seul document sélectionné dans le panneau principal. Ces configurations sont déterminées à partir du modèle de préférence courant de l'utilisateur.

L'utilisateur peut sélectionner tout ou partie des configurations alternatives proposées. Le bouton d'acceptation situé en bas permet à l'utilisateur de valider les configurations alternatives choisies. Dans ce cas, la représentation du panneau principal est mise à jour avec les nouvelles configurations sélectionnées.

Ce panneau permet donc à l'utilisateur de vérifier le comportement de la politique d'impression apprise par le système sur un grand nombre de documents à la fois et, éventuellement, d'en valider certains.

Scénario d'utilisation

Voici, en cinq points, un scénario "type" d'utilisation du système de recommandation pour configurer les paramètres d'impressions de plusieurs documents.

1. L'imprimeur charge ses documents dans l'application. Des représentations graphiques correspondant aux résultats de la politique d'impression courante s'affichent dans le panneau principal.
2. Si toutes les configurations d'impressions affichées dans le panneau principal correspondent aux souhaits de l'imprimeur, il imprime. Fin de l'utilisation.
3. Sinon il sélectionne un des documents dont la configuration n'est pas correcte et regarde si une des alternatives proposées dans le panneau de recommandation simple est conforme à ses préférences.
   (a) Si oui, il la valide. Le panneau principal et le modèle de préférence courant sont mis à jour. L'imprimeur vérifie ensuite, grâce au panneau de recommandation multiple, si la politique apprise par le système est valide (par rapport à ses préférences) sur les documents restant à configurer. Il valide les configurations conformes. Retour au point 2.
   (b) Sinon il configure manuellement les paramètres d'impression du document. Le panneau principal et le modèle de préférence courant sont mis à jour. Retour au point 2.

5.4.2 Exemple d'utilisation

Nous illustrons ici l'utilisation de notre système de recommandation. Dans notre exemple, l'imprimeur doit configurer les paramètres d'impression d'une quinzaine de documents. Ces documents, des images de grandes tailles, forment un échantillon assez représentatif d'un travail d'impression classique.

Etape 1

Une fois les documents chargés dans l'application, l'imprimeur obtient l'écran de la figure 5.7. Les documents chargés ont été automatiquement configurés avec la politique d'impression par défaut. On voit, à travers la représentation graphique des impressions de chaque document, que cette politique d'impression sélectionne un média à partir du rouleau A0, choisit un zoom de 100% et effectue un découpage synchrone.

Ce résultat ne correspond pas complètement aux desiderata de l'imprimeur de notre exemple, dont la politique d'ajustement de l'image est "zoomer pour ajuster".

Etape 2

L'imprimeur peut alors ouvrir le panneau de recommandation et obtenir l'écran de la figure 5.8. On voit que seul un document est sélectionné dans le panneau

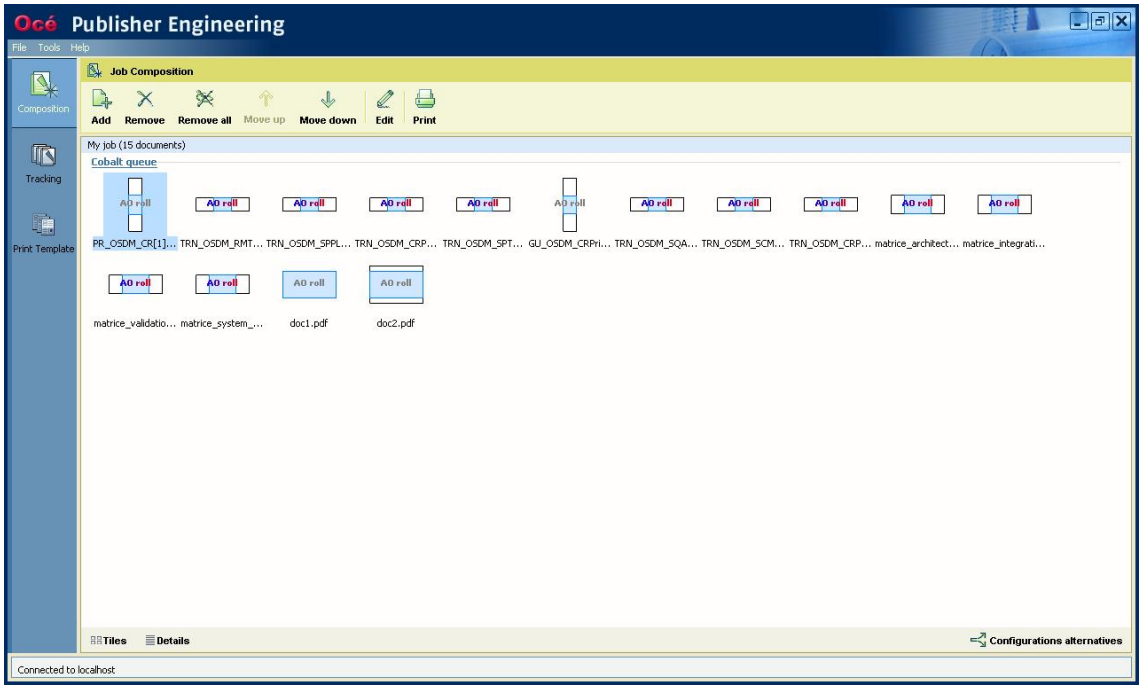


FIG. 5.7: Ecran initial : les documents sont chargés

principal, donc c'est le panneau de recommandation simple qui s'affiche, avec cinq alternatives de configuration. On remarque qu'une alternative correspond au résultat attendu : la seconde en partant de la droite. L'imprimeur qui valide ce choix met à jour la configuration du document et, implicitement, son modèle de préférence.

Etape 3

L'imprimeur peut ensuite vérifier si le modèle de préférence appris d'après son premier choix est valide sur le reste des documents du travail d'impression. Il lui suffit de sélectionner tous les documents restant à configurer et il obtient le panneau de recommandation multiple lui présentant la configuration alternative "préférée" par son modèle de préférence courant, pour chaque document sélectionné. L'imprimeur peut alors sélectionner dans ce panneau les documents bien configurés (figure 5.9). On voit ici que cinq documents (sélectionnés sur l'image) sur les douze restants ont été bien configurés. L'imprimeur peut les valider d'un coup.

Etape 4

L'imprimeur recommence le processus pour les documents restant à configurer. Comme à l'étape 2, il sélectionne un document restant et obtient l'affichage du

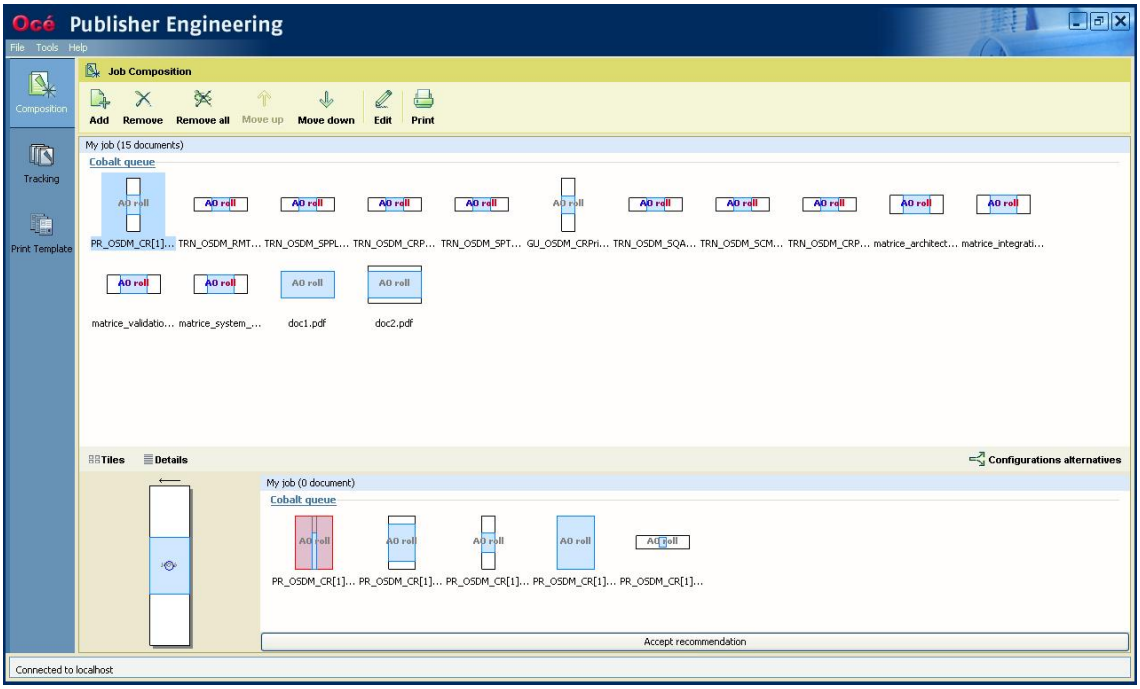


FIG. 5.8: Panneau de recommandation

panneau de recommandation simple. Une des alternatives proposées (sélectionnée sur l'image de la figure 5.10) correspond au résultat attendu. Son modèle de préférence est mis à jour une fois cette configuration validée.

Etape 5

Comme à l'étape 3, l'imprimeur vérifie l'adéquation du modèle de préférence courant sur l'ensemble des documents restant à imprimer. On voit sur l'image de la figure 5.11 qu'un seul document ne correspond pas au résultat attendu. Les autres sont validés d'un bloc.

Etape 6

Enfin, la dernière étape consiste à configurer le dernier document. L'imprimeur peut le faire à travers le panneau de recommandation simple, tel que montré dans la figure 5.12. Il ne lui reste plus alors qu'à lancer l'impression de l'ensemble des documents.

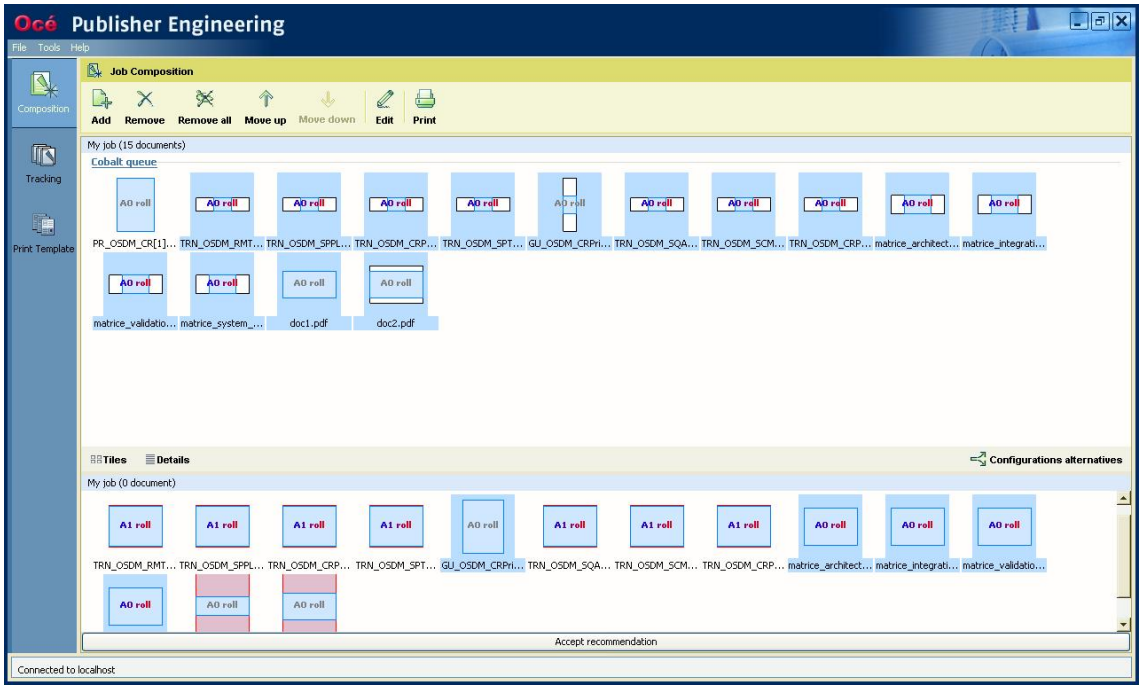


FIG. 5.9: Recommandations multiple

5.4.3 Génération de l'ensemble d'apprentissage

Dans notre contexte de recommandation adaptative, au début de l'interaction entre l'imprimeur et le système, notre ensemble d'apprentissage est vide. C'est à partir des différents choix de l'imprimeur (i.e. les configurations qu'il détermine soit manuellement, soit grâce au panneau de recommandation) que nous construisons un ensemble d'apprentissage.

La difficulté réside dans les choix implicites de l'imprimeur. En effet, lorsque celui-ci configure les paramètres d'impression d'un document, on peut supposer que la configuration obtenue est préférée à *toutes* les autres possibles. Ne pouvant pas construire un ensemble d'apprentissage infini, nous devons sélectionner un sous-ensemble des configurations non élues, afin de générer nos exemples d'apprentissage.

Pour ce faire, nous proposons l'algorithme incrémental 5.4, guidé par les résultats d'apprentissages successifs. A partir d'un modèle de préférence, structurellement le plus simple possible et dont les poids sont initialisés aléatoirement, nous comparons les choix du modèle avec ceux de l'imprimeur. Tant qu'ils sont différents, nous ajoutons les "différences" à l'ensemble d'apprentissage et adaptons la structure et les poids du modèle.

On voit que la construction de l'ensemble d'apprentissage n'est pas séparée de l'apprentissage du modèle. Ainsi, lorsque l'algorithme s'arrête, l'ensemble d'appren-

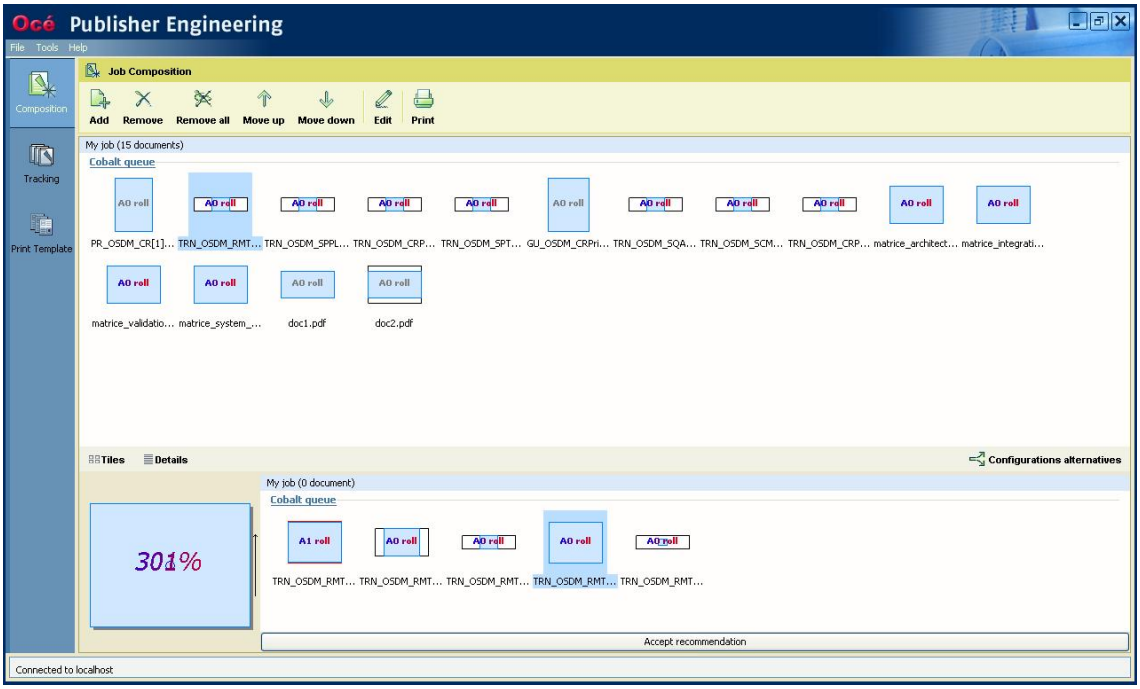


FIG. 5.10: Dernier document à configurer

tissage est généré et le modèle appris. C'est pourquoi nous désignons cet algorithme sous le terme de général.

Remark. Bien qu'apparenté, cet algorithme ne constitue pas un *apprentissage actif*. L'apprentissage actif (voir par exemple [CAL94]) consiste à fournir incrémentalement des exemples à l'utilisateur, de sorte que ses décisions successives (dans le cas de l'ordonnancement, l'ordre entre les deux possibilités) permettent de faire converger au mieux l'algorithme d'apprentissage. Notre algorithme ne fournit pas à l'utilisateur un couple de configurations en lui demandant de les ordonner.

5.4.4 Résultats

Nous présentons ici les résultats de performance de l'algorithme d'apprentissage dans le cadre de la recommandation d'impression.

Expérience n°1

La première expérience présentée illustre le comportement de l'algorithme d'apprentissage dans un mode de fonctionnement "en ligne", c'est-à-dire dans un fonctionnement similaire à son utilisation par un imprimeur à travers l'interface de recommandation.

Algorithm 5.4 Algorithme général

Soit $u_{S,W} : \mathbb{A} \rightarrow [0,1]$ la fonction d'utilité représentant les préférences de l'imprimeur, avec S la structure et W les poids du modèle.
Soit EA l'ensemble d'apprentissage :
EA $\leftarrow \emptyset$
Soit ECO l'ensemble des n choix observés :
ECO $\leftarrow ((document_i, configuration_i),\ i = 1..n)$
Initialisation de la structure du modèle S sans points de rupture.
Initialisation aléatoire des poids du modèle W
Tant que $\exists i \in 1..n \, tel \, que \ (d_i, c_i) \in ECO$ et

$$c_i \neq a* = \arg\max_{a \in \mathbb{A}_{d_i}} u_{S,W}(a)$$

où $\mathbb{A}_{d_i}$ est l'ensemble des configurations alternatives admissibles avec le document i
EA $\leftarrow EA \cup (c_i, a^*)$
Mise à jour du modèle

Adaptation de la structure S (algo. 5.1, 5.2 et 5.3)
Adaptation des poids W (rankSVM linéaire)

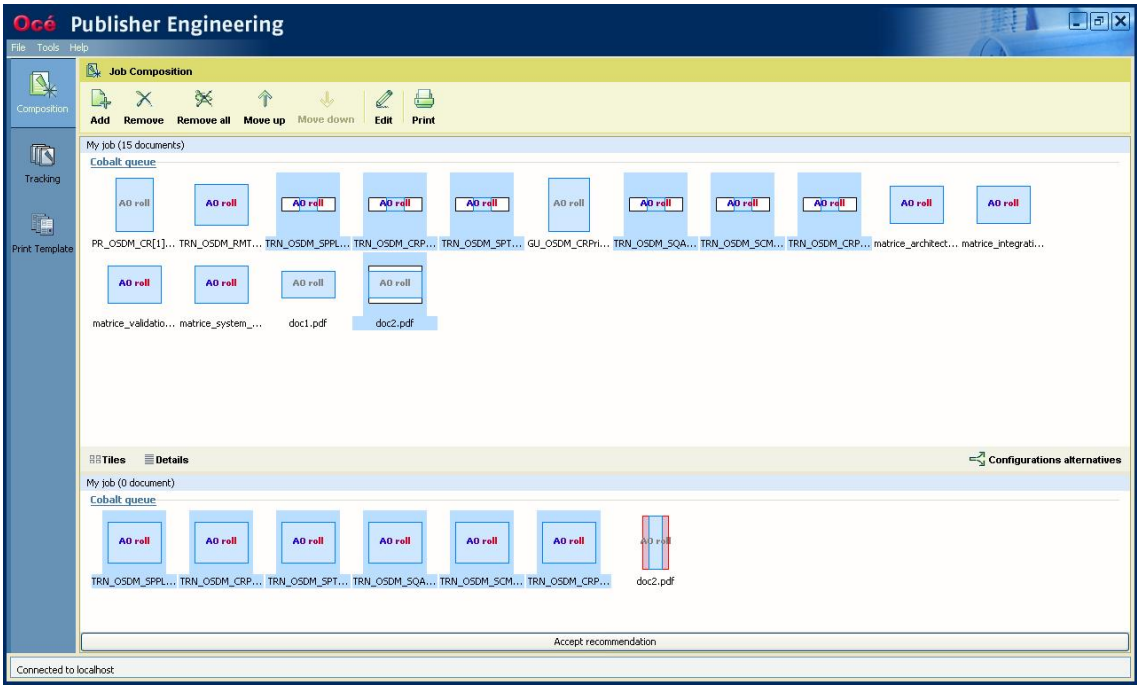


FIG. 5.11: Panneau de recommandation multiple : 6/7 documents validés

Nous avons testé le nombre de configurations manuelles requises, afin de configurer entièrement des travaux d'impression constitués de 25 documents.

La démarche est la suivante : la politique courante est appliquée à l'ensemble des documents restant à configurer. Le résultat de ces configurations est comparé avec le résultat attendu de la politique d'impression témoin. Les documents bien configurés sont extraits du lot. Tant qu'il reste au moins un document mal configuré, on en sélectionne un et on simule une configuration manuelle de l'imprimeur en ajoutant à l'ensemble d'apprentissage un nouveau choix. On recommence jusqu'à ce que l'ensemble des documents à configurer soit vide.

Notons que l'environnement, c'est-à-dire que l'ensemble des médias disponibles, est constant, comme dans une situation d'utilisation normale.

La figure 5.13 montre le taux moyen de configurations manuelles, par politique d'impression, pour configurer un travail d'impression de 25 documents. Ces taux sont moyennés sur 25 tests.

Les dix politiques testées sont les suivantes (choix du média/ajustement/zoom désiré/découpe) :

1. plus proche / ajuster au média / 100% / standard
2. plus proche / réduire pour ajuster / 100% / synchrone
3. plus proche / couper pour ajuster / 100% / standard

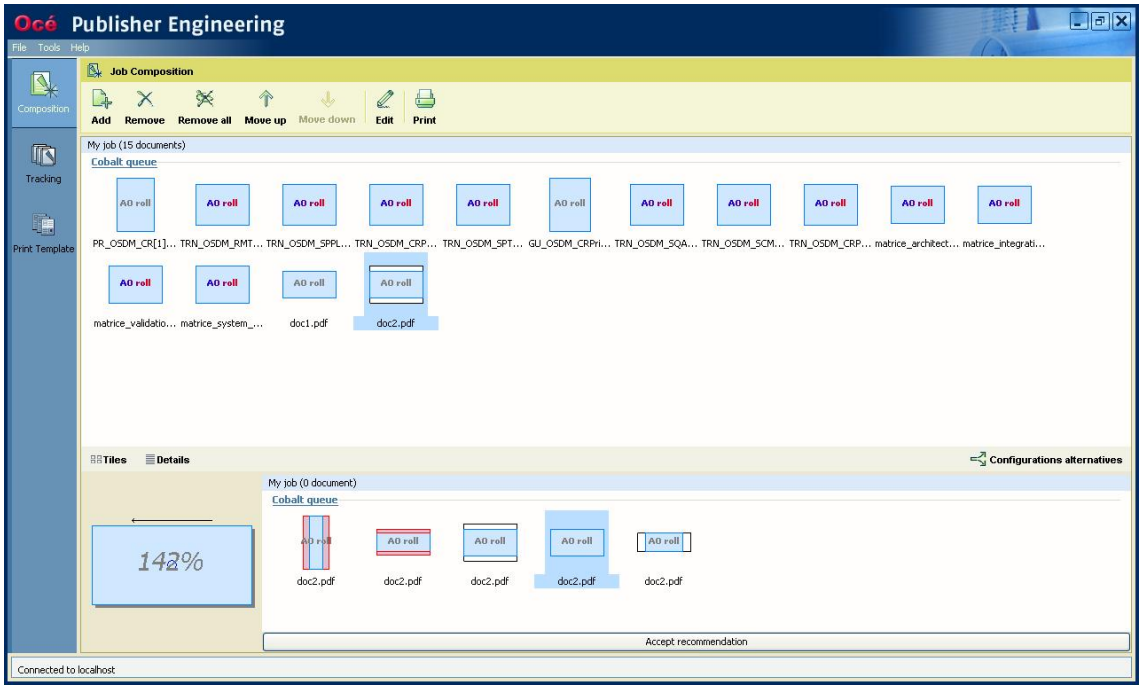


FIG. 5.12: Dernier document

4. plus proche / réduire pour ajuster / 50% / synchrone
5. plus proche / couper pour ajuster / 50% / standard
6. plus grand sinon plus petit / couper pour ajuster / 100% / synchrone
7. plus grand sinon plus petit / ajuster au média / 100% / standard
8. plus grand sinon plus petit / réduire pour ajuster / 100% / synchrone
9. plus grand sinon plus petit / réduire pour ajuster / 50% / standard
10. plus grand sinon plus petit / couper pour ajuster / 50% / synchrone

Le taux moyen sur l'ensemble des politiques est d'environ 22%, c'est-à-dire en moyenne, 5 ou 6 configurations manuelles sur 25. Ce taux semble élevé mais il faut noter que les dimensions des documents composant les travaux d'impression testés ont été générés au hasard, donnant des ensembles très hétérogènes, par rapport à une situation réelle.

On remarque de fortes disparités entre politiques. La politique n°1 est la plus simple à apprendre avec en moyenne deux à trois configurations manuelles pour réussir à configurer l'ensemble des 25 documents. A l'inverse, la politique n°8 est la plus difficile, avec presque une configuration manuelle sur deux nécessaire.

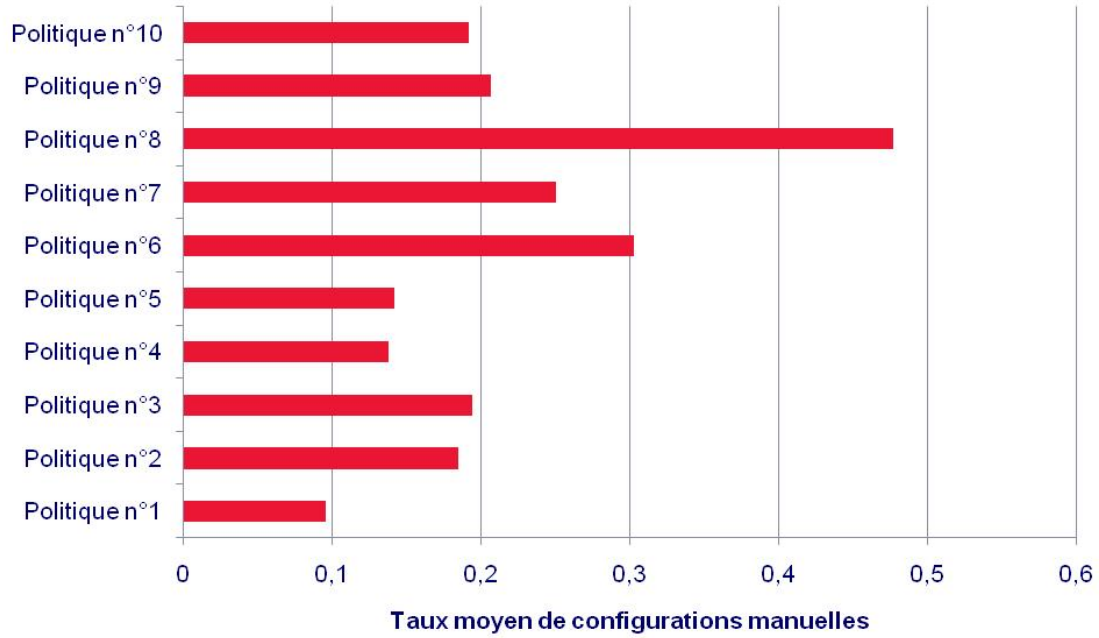


FIG. 5.13: Performance de l'apprentissage sur 10 politiques

On voit qu'en moyenne, les politiques basées sur la stratégie de choix de média "plus grand sinon plus petit" sont plus difficiles à apprendre que celles basées sur la stratégie "aussi proche que possible". Cela s'explique par l'effet discontinu de la première stratégie, qui est plus difficile à représenter dans notre modèle.

On peut noter également qu'entre les politiques 2 et 4, ainsi qu'entre les numéros 6 et 10, le seul paramètre qui change est le zoom désiré : 50% ou 100%. Les résultats montrent que les politiques présentant un zoom désiré de 100% sont plus difficiles à apprendre qu'avec un zoom à 50%. Nous proposons d'expliquer ce phénomène de la façon suivante : la réduction systématique d'une image à 50% permet d'éviter plus souvent la problématique de l'ajustement au média, que ce soit en réduction supplémentaire ou en coupe, ce qui simplifie la politique.

Expérience n°2

Dans la seconde expérience, nous nous intéressons aux performances de l'algorithme d'apprentissage dans une utilisation "hors ligne", correspondant par exemple à un entrainement lors des phases de conception, afin de pré-enregistrer les politiques standards dans notre nouveau modèle de préférence.

Nous considérons cette fois les performances de l'algorithme sur un ensemble de test constitué de 1000 documents, en fonction du nombre de configurations validées

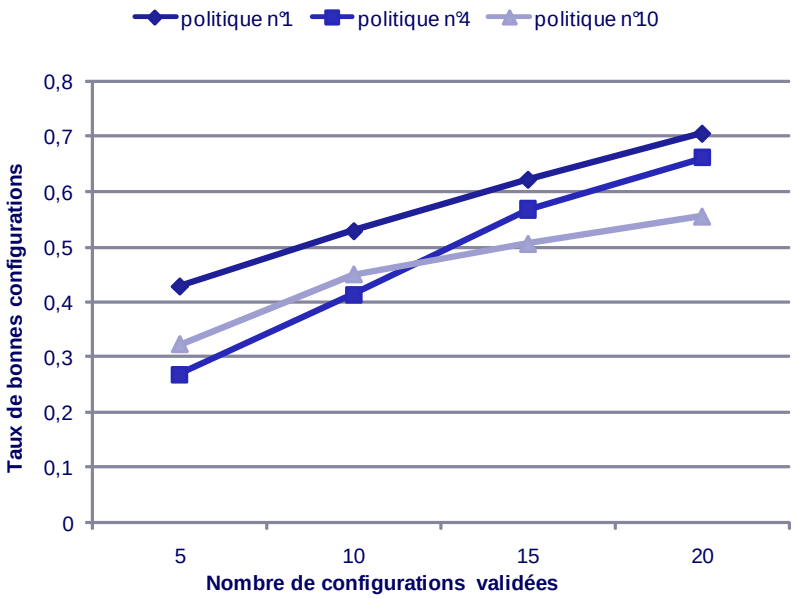


FIG. 5.14: Performance de l'apprentissage "hors ligne"

de l'ensemble d'apprentissage. De plus, chaque document de l'ensemble de test est associé avec un environnement particulier (i.e. un ensemble de médias disponibles), ce qui signifie que deux documents ayant les mêmes caractéristiques mais des environnement différents n'auront pas forcément la même configuration attendue.

La figure 5.14 montre les résultats des tests sur trois politiques précédentes, les n°1, n°4 et n°10. Ces résultats sont des moyennes sur 10 tests.

On voit que le taux de réussite croît rapidement avec la taille de l'ensemble d'apprentissage. Les trois politiques présentent des résultats assez similaires.

5.5 Conclusion

Nous avons formalisé le problème de l'apprentissage automatique du modèle de préférences d'un utilisateur, à partir de ses choix, dans le cadre de l'apprentissage d'ordonnancement supervisé.

Nous proposons une approche couvrant l'apprentissage des poids du modèle ainsi que la structure même du modèle. Concernant l'apprentissage des poids du modèle, après un état de l'art sur les techniques de la littérature, nous optons pour l'utilisation de l'algorithme RankSVM.

Notre apport original vient de l'adaptation de la structure du modèle, à travers l'estimation des extrema des fonctions d'utilité marginales. Cette estimation est réalisée par une méthode heuristique, basée sur la projection des préférences globales sur chaque axe de préférence, associée à une mesure de probabilité.

Enfin, nous présentons l'application de nos travaux au domaine de l'impression professionnelle grand format, à travers un système de recommandation inséré dans une application professionnelle.

Conclusion

Partant d'une problématique industrielle, nos travaux constituent une approche globale de l'automatisation du paramétrage des impressions grand format. Nous traitons cette question à travers ses deux aspects complémentaires : la conception et l'utilisation. Derrière ces aspects, émergent les différentes problématiques scientifiques abordées au long de ce mémoire :

- l'évaluation de l'utilisabilité d'une application
- la modélisation mathématique de la configuration d'impression, comme un problème d'optimisation sous contraintes
- la modélisation des préférences appliquée au domaine de l'impression
- l'apprentissage des préférences dans un contexte de recommandation interactive

A travers l'application de méthodes scientifiques à un nouveau domaine et la proposition d'un algorithme original adapté aux contraintes du problème, nous présentons une nouvelle approche pour traiter la question de l'automatisation de la configuration des paramètres d'impression grand format.

Synthèse

Dans le premier chapitre, nous avons présenté la problématique industrielle à la base de nos travaux : l'automatisation de la configuration des paramètres d'impression d'un document. Nous avons introduit les différents aspects de la configuration d'impression grand format et montré l'intérêt et la difficulté de son automatisation.

Dans la seconde partie de ce chapitre, nous avons présenté l'approche générale de nos travaux : améliorer l'utilisabilité grâce à un système de recommandation capable d'apprendre uniquement sur la base des choix précédents de l'imprimeur. Nous avons positionné notre approche dans un état de l'art sur les systèmes de recommandations.

Nous avons présenté dans le chapitre 2 la notion d'utilisabilité d'une application informatique et les différentes méthodes permettant de l'évaluer. Nous traitons par-

ticulièrement des techniques d'évaluation automatique, basées sur la récupération et l'analyse automatique de données d'interaction entre l'utilisateur et l'application.

Pour illustrer notre propos, nous avons mené une étude d'utilisabilité d'une application en phase beta, conçue par Océ. Nous en présentons les résultats, obtenus à l'aide d'un outil d'analyse automatique de logs, que nous avons développé. Notre approche s'inscrit dans une perspective complémentaire aux méthodes actuellement en usage chez Océ pour l'étude d'utilisabilité des futurs produits.

Ces travaux préliminaires illustrent la difficulté de la tâche de configuration d'impression et renforce la conviction du besoin de méthodes automatiques pour aider à la résolution de ce problème.

Dans le chapitre 3, nous avons énuméré, expliqué et formalisé l'ensemble des données formant l'entrée de notre problème de configuration d'impression grand format.

Nous avons modélisé l'espace des paramètres d'impression, afin de définir l'ensemble des configurations admissibles, connaissant les caractéristiques du document à imprimer, les contraintes et directives de l'imprimeur, ainsi que les ressources d'impression disponibles.

Dans le chapitre 4, nous avons présenté le modèle des préférences de l'imprimeur, actuellement en vigueur dans les produits d'Océ. Nous en avons proposé une formalisation, mis en lumière les relations entre les différents paramètres. Nous avons également pointé certaines limites comportementales.

Nous répondons ensuite à un des objectifs industriels de nos travaux de recherche : faciliter le développement et la maintenance de la gestion des politiques d'impression, pour les développeurs. Notre démarche générale est de considérer le problème sous un angle permettant au maximum l'utilisation d'algorithmes efficaces connus.

Nous proposons d'abord une modélisation mathématique du problème du choix automatique d'une configuration d'impression, d'après une politique d'impression. Ce modèle permet de considérer le choix d'une configuration comme un problème d'optimisation sous contraintes. De plus, il met en lumière les symétries qui réduisent la taille de l'espace de recherche. Cependant, le problème d'optimisation modélisé est non linéaire et présente notamment des contraintes d'implication, ce qui en fait un mauvais candidat à une résolution efficace. Nous montrons alors comment passer du modèle initial, qui correspond à un problème d'optimisation non linéaire, à une série minimale de problèmes linéaires mixtes en nombres entiers. Cette reformulation permet d'attaquer la résolution avec des algorithmes efficaces existants. De plus, elle

reste très déclarative, ce qui rend plus facile aux développeurs la manipulation de l'aspect logique de la fonctionnalité, tout en laissant le contrôle de côté.

Dans la troisième partie de ce chapitre, nous examinons diverses modélisations de préférence, dans le but d'améliorer le modèle actuel. Après avoir donné les hypothèses sous lesquelles on peut formuler la relation de préférence de l'imprimeur comme une fonction d'utilité, nous passons en revue :

– le modèle classique : l'utilité additive

– quatre modèles graphiques : les CP-Nets, TCP-Nets, GAI-Nets et UCP-Nets.

Nous concluons sur le choix du modèle additif, sous réserve d'adapter les attributs sur lesquels s'expriment les préférences. Cela permet de nous affranchir des dépendances préférentielles présentes initialement.

Dans le chapitre 5, nous avons formalisé le problème de l'apprentissage automatique du modèle de préférence d'un utilisateur, à partir de ses choix, dans le cadre de l'apprentissage d'ordonnancement supervisé.

Nous proposons une approche couvrant l'apprentissage des poids ainsi que la structure du modèle. Concernant l'apprentissage des poids du modèle, après un état de l'art sur les techniques de la littérature, nous optons pour l'utilisation de l'algorithme RankSVM.

Notre apport original vient de l'adaptation de la structure du modèle, à travers l'estimation des extrema des fonctions d'utilité marginales. Cette estimation est réalisé par une méthode heuristique, basée sur la projection des préférences globales observées sur chaque axe de préférence, associée à une mesure de probabilité.

Enfin, nous présentons l'application de nos travaux au domaine de l'impression professionnelle grand format, à travers un système de recommandation intégrée dans une application professionnelle et discutons de ses performances.

Perspectives

Seuls les aspects de mise en page et de finition de l'impression ont fait l'objet du travail de recherche relaté dans ce mémoire. Nous n'avons pas traité la problématique de la qualité d'impression pour la raison suivante : les critères de qualité sont très dépendants du contenu de chaque document. Prendre en compte les critères de qualité lors de l'automatisation de la configuration d'impression demande une analyse du contenu de chaque document, ce qui est hors du champ de ce travail de recherche mais pourrait constituer une extension à nos travaux.

La qualité d'impression se traduit en terme de résolution, qualité du papier et des encres, gestion de la couleur, des à-plats, de la netteté des lignes, etc. Il existe diffé-

rents algorithmes pour reconstituer "au mieux" les couleurs que l'on perçoit à l'écran sur le papier, c'est le problème du *gamut mapping*. De même, plusieurs méthodes sont possibles afin de distribuer les gouttes d'encre sur le papier (*halftoning*), avec chacune leurs avantages et inconvénients comme l'apparition d'effets de bandes ou de "vers", désagréables à l'oeil. Certaines techniques favorisent ainsi l'impression des lignes, ce qui convient parfaitement à l'impression de dessins techniques ; d'autres rendent mieux avec l'impression de photos, ou encore des textes.

Comme les documents sont souvent constitués d'éléments de types divers, le choix des algorithmes définissant le rendu est complexe. Automatiser ce choix à partir d'informations extraites des documents à imprimer et d'une politique d'impression incluant les critères de qualité, est une extension naturelle de la fonctionnalité étudiée dans ce mémoire.

En ce sens, il serait intéressant d'étudier les possibilités d'extension de notre modèle de préférence à la prise en compte de ces nouveaux critères. Cela permettrait de les intégrer directement[9] dans notre système de recommandation.

Une autre problématique connexe pourrait bénéficier de l'extension de notre approche. Il s'agit du problème de l'agencement de plusieurs images sur un même média, de manière à minimiser le gaspillage de papier. Ce problème d'optimisation sous contraintes, corrélé à notre problématique initiale, ouvre des perspectives intéressantes d'un point de vue industriel et un défi passionnant du point de vue scientifique. Probablement de la catégorie NP difficile, il nécessite des algorithmes recherchant, de façon heuristique, une solution proche de l'optimale. Des méthodes telles que les algorithmes génétiques, la recherche tabou, le recuit simulé, pourrait permettre d'aborder la résolution de ce problème. L'articulation de cette résolution avec les méthodes présentées dans ce mémoire semble également être un problème en soi. Une autre solution serait de résoudre une version simplifié du problème dans notre modèle linéaire mixte en nombre entier. Cela nécessite certainement une étude approfondie.

Le problème dual à celui présenté dans le paragraphe précédent est l'impression d'un document en plusieurs morceaux. Ce problème se pose notamment pour l'impression des très grandes affiches publicitaires. Permettre d'imprimer un document en plusieurs morceaux pourrait parfaitement faire partie d'une politique d'impression et rentrer dans notre modélisation. Dans ce cas, d'autres questions doivent également être résolues, comme le choix des lignes de coupes et la répartition des morceaux sur les médias.

[9] sous réserve de pouvoir présenter à l'imprimeur, de façon pertinente, toutes les facettes des alternatives recommandées

Concernant l'extension de notre algorithme d'apprentissage, on peut envisager la piste de recherche suivante : utiliser les résultats de l'algorithme RankSVM, afin de sélectionner un sous-ensemble de décisions posant problème au modèle. Cela permettrait de baser la recherche d'extrema locaux sur un petit nombre de décision, augmenterait ses performances et éventuellement la pertinence de sa solution, en diminuant le bruit. De plus, cette extension ne paraît pas nécessiter une grande adaptation de l'algorithme initial. Pour ces raisons nous considérons que cette extension constitue un axe de recherche intéressant.

Bibliographie

[ACZ03] A. Anglada, P. Codognet, and L. Zimmer. Nscsp, définition and resolution by transformation. In *Proceedings of the 5th International Workshop on Soft Constraints in CP03*, 2003.

[ACZ04] A. Anglada, P. Codognet, and L. Zimmer. An adaptive search for the NSCSPs, 2004.

[AQM99] G. Al-Qaimari and D. McRostie. KALDI : a computer-aided usability engineering tool for supporting testing and analysis of human-computer interaction. In *Proceedings of the third international conference on Computer-aided design of user interfaces table of contents*, pages 337–355. Kluwer Academic Publishers Norwell, MA, USA, 1999.

[Bal96] S. Balbo. EMA : automatic analysis mechanism for ergonomic evaluation of user interface. *CSIRO*, 1996.

[BBB01] C. Boutilier, F. Bacchus, and R.I. Brafman. UCP-Networks : A directed graphical representation of conditional utilities. In *Proceedings of the Seventeenth Conference on Uncertainty in Artificial Intelligence*, pages 56–64, 2001.

[BBHP99] C. Boutilier, R.I. Brafman, H.H. Hoos, and D. Poole. Reasoning with conditional ceteris paribus preference statements. In *Proceedings of the Fifteenth Annual Conference on Uncertainty in Artificial Intelligence*, pages 71–80, 1999.

[BCOP07] C. Bessiere, R. Coletta, B. O'Sullivan, and M. Paulin. Query-driven constraint acquisition. In *The 20th International Joint Conference on Artificial Intelligence (IJCAI 2007)*, pages 50–55, 2007.

[BD02] R.I. Brafman and C. Domshlak. Introducing Variable Importance Tradeoffs into CP-Nets. In *Proceedings of the 18th Conference in Uncertainty in Artificial Intelligence*, 2002.

[BF76] E. M. L. Beale and J. J. H. Forrest. Global optimization using special ordered sets. *Mathematical Programming*, 10(1) :52–69, 1976.

[BG95] F. Bacchus and A. Grove. Graphical models for preference and utility. In *Proceedings of the Eleventh Conference on Uncertainty in Artificial Intelligence*, pages 3–10, 1995.

[BHY97] Robin D. Burke, Kristian J. Hammond, and Benjamin C. Young. The findme approach to assisted browsing. *IEEE Expert*, 12(4) :32–40, 1997.

[Bly02] J. Blythe. Visual Exploration and Incremental Utility Elicitation. In *Proceedings of the National Conference on Artificial Intelligence*, pages 526–532. Menlo Park, CA ; Cambridge, MA ; London ; AAAI Press ; MIT Press ; 1999, 2002.

[BMR+99] S. Bistarelli, U. Montanari, F. Rossi, T. Schiex, G. Verfaillie, and H. Fargier. Semiring-Based CSPs and Valued CSPs : Frameworks, Properties, and Comparison. *Constraints*, 4(3) :199–240, 1999.

[BT70] E. M. L. Beale and J. A. Tomlin. Special facilities in a general mathematical programming system for nonconvex problems using ordered sets of variables. *Operational Research*, 69 :447–454, 1970.

[Bur99] Robin D. Burke. The Wasabi Personal Shopper : A Case-Based Recommender System. *Proceedings of the National Conference on Artificial Intelligence*, pages 844–849, 1999.

[Bur00] Robin D. Burke. Knowledge-based recommender systems. *Encyclopedia of Library and Information Systems*, 69(Supplement 32) :175–186, 2000.

[Bur02] Robin D. Burke. Hybrid recommender systems : Survey and experiments. *User Model. User-Adapt. Interact*, 12(4) :331–370, 2002.

[CAL94] D. Cohn, L. Atlas, and R. Ladner. Improving generalization with active learning. *Machine Learning*, 15(2) :201–221, 1994.

[CBQ] R. Coletta, C. Bessiere, and J. Quinqueton. Modelisation semi-automatique par acquisition de contraintes. *9iemes Journees nationales sur la resolution pratique de problemes NP-complets (JNPC'03)*, pages 129–143.

[CC57] A. Charnes and WW Cooper. Management models and industrial applications of linear programming. *Management Science*, pages 38–91, 1957.

[CNT05] Marc Christie, Jean-Marie Normand, and Charlotte Truchet. Computing inner approximations of numerical maxcsp. In *Interval Analysis, Constraint Propagation, Applications (IntCP 2005)*, 2005.

[CP04] L. Chen and P. Pu. Survey of Preference Elicitation Methods. Technical report, EPFL Technical Report IC/2004, 2004.

[Dam08] Marc Damez. *De l'apprentissage artificiel pour l'apprentissage humain : de la récolte de traces à la modélisation utilisateur*. PhD thesis, Université Pierre et Marie Curie, Paris, 2008.

[Den03] B. Dennis. A Survey of Preference Elicitation. *Manuscript, Computer Science Department, North Carolina State University*, 2003.

[DJ94] M.P. Dubuisson and A. Jain. A modified Hausdorff distance for object matching. In *Proceedings of 12th International Conference on Pattern Recognition*, volume 1, pages 566–568, 1994.

[DR08] M. Damez and S. Renaud. Représentation des données pour l'aide à l'analyse cognitive de parcours. *Numero special de la Revue d'Intelligence Artificielle, Visualisation et extraction de connaissances. RSTI-RIA*, 22, 2008.

[DSLDCT06] A. Da Silva, Y. Lechevallier, F. De Carvalho, and B. Trousse. Mining web usage data for discovering navigation clusters. In *11th IEEE Symposium on Computers and Communications, 2006. ISCC'06. Proceedings*, pages 910–915, 2006.

[Fal94] Boi Faltings. Arc consistency for continuous variables. *Artificial Intelligence*, 65 :363–376, 1994.

[FISS98] Yoav Freund, Raj Iyer, Robert E. Schapire, and Yoram Singer. An efficient boosting algorithm for combining preferences. In Jude W. Shavlik, editor, *Proceedings of ICML-98, 15th International Conference on Machine Learning*, pages 170–178. Morgan Kaufmann Publishers, San Francisco, US, 1998.

[Fre86] S. French. *Decision Theory*. Ellis Horwood, 1986.

[FS97] Y. Freund and R.E. Schapire. A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting. *Journal of Computer and System Sciences*, 55(1) :119–139, 1997.

[FW92] E.C. Freuder and R.J. Wallace. Partial Constraint Satisfaction. *Artificial Intelligence*, 58(1-3) :21–70, 1992.

[GJP95] F. Girosi, M. Jones, and T. Poggio. Regularization theory and neural networks architectures. *Neural Computation*, 7(2) :219–269, 1995.

[GP04] C. Gonzales and P. Perny. GAI networks for utility elicitation. In *KR2004 : Principles of Knowledge Representation and Reasoning*, pages 224–234, 2004.

[GP05] C. Gonzales and P. Perny. GAI networks for decision making under certainty. In *IJCAI 2005–Workshop on Advances in Preference Handling*, 2005.

[GW05] K. Gajos and D.S. Weld. Preference elicitation for interface optimization. *Proceedings of the 18th annual ACM symposium on User interface software and technology*, pages 173–182, 2005.

[HH97] V. Ha and P. Haddawy. Problem-focused incremental elicitation of multi-attribute utility models. *Proceedings of the Thirteenth Conference on Uncertainty in Artificial Intelligence*, pages 215–222, 1997.

[HL99] B. Helfrich and J.A. Landay. QUIP : quantitative user interface profiling. *Unpublished manuscript. Available at http ://home.earthlink.net/ bhelfrich/quip/index.html*, 1999.

[HR00] David M. Hilbert and David F. Redmiles. Extracting usability information from user interface events. *ACM Computing Surveys*, 32(4) :384–421, December 2000.

[IH01] Melody Y. Ivory and Marti A. Hearst. The state of the art in automating usability evaluation of user interfaces. *ACM Computing Surveys*, 33(4) :470–516, December 2001.

[JLS82] E. Jacquet-Lagreze and J. Siskos. Assessing a set of additive utility functions for multicriteria decision-making, the UTA method. *EUROP. J. OPER. RES.*, 10(2) :151–164, 1982.

[JLS01] E. Jacquet-Lagrèze and Y. Siskos. Preference disaggregation : 20 years of MCDA experience. *European Journal of Operational Research*, 130(2) :233–245, 2001.

[Joa02] T. Joachims. Optimizing Search Engines Using Clickthrough Data. In *Proceedings of the ACM Conference on Knowledge Discovery and Data Mining (KDD)*, 2002.

[LHL97a] G. Linden, S. Hanks, and N. Lesh. Interactive assessment of user preference models : The automated travel assistant. *Proceedings of the Sixth International Conference on User Modeling*, 6778, 1997.

[LHL97b] Greg Linden, Steve Hanks, and Neal Lesh. Interactive assessment of user preference models : The automated travel assistant. In *Proceedings, User Modeling*, 1997.

[Lho93] Olivier Lhomme. Consistency techniques for numeric csps. pages 232–238, 1993.

[LL05] A. Lallouet and A. Legtchenko. Two contributions of constraint programming to machine learning. *Lecture notes in computer science*, 3720 :617, 2005.

[LLY08] Marie-Jeanne Lesot, Nicolas Labroche, and Lionel Yaffi. Analyse et visualisation interactive de sessions web. *Revue d'Intelligence Artificielle*, 22(3-4) :369–382, 2008.

[LPCC98] A. Lecerof, F. Paterno, I. CNUCE, and P. CNR. Automatic support for usability evaluation. *IEEE Transactions on Software Engineering*, 24(10) :863–888, 1998.

[MN68] M. Maňas and J. Nedoma. Finding all vertices of a convex polyhedron. *Numerische Mathematik*, 12(3) :226–229, 1968.

[Mou03] Vincent Mousseau. Elicitation des préférences pour l'aide multicritère à la décision. *HDR, Université Paris Dauphine*, 2003.

[MR93] M. Macleod and R. Rengger. The Development of DRUM : A Software Toolfor Video-assisted Usability Evaluation. *People and Computers VIII*, 1993.

[MTT04] F. Masseglia, D. Tanasa, and B. Trousse. Diviser pour decouvrir. Une methode d'analyse du comportement de tous les utilisateurs d'un site web. *INGENIERIE DES SYSTEMS D INFORMATION.*, 9 :61–84, 2004.

[NGCB08] J.M. Normand, A. Goldsztejn, M. Christie, and F. Benhamou. A Branch and Bound Algorithm for Numerical MAX-CSP. In *Proceedings of the 14th international conference on Principles and Practice of Constraint Programming*, pages 205–219. Springer-Verlag Berlin, Heidelberg, 2008.

[Nie93] J. Nielsen. *Usability Engineering*. Morgan Kaufmann, 1993.

[Nie94] J. Nielsen. Enhancing the explanatory power of usability heuristics. In *Proceedings of the SIGCHI conference on Human factors in computing systems : celebrating interdependence*, pages 152–158. ACM Press New York, NY, USA, 1994.

[OH88] D.R. Olsen and B.W. Halversen. Interface usage measurements in a user interface management system. In *Proceedings of the 1st annual ACM SIGGRAPH symposium on User Interface Software*, pages 102–108. ACM New York, NY, USA, 1988.

[Rau93] M. Rauterberg. AMME : an Automatic Mental Model Evaluation to analyse user behaviour traced in a finite, discrete state space. *Ergonomics*, 36(11) :1369–1380, 1993.

[RS04] F. Rossi and A. Sperduti. Acquiring both constraint and solution preferences in interactive constraint systems. *Constraints*, 9(4) :311–332, 2004.

[Rut94] Z. Ruttkay. Fuzzy constraint satisfaction. In *IEEE World Congress on Computational Intelligence, Proceedings of the Third IEEE Conference on Fuzzy Systems*, pages 1263–1268, 1994.

[RVW09] F. Rossi, K.B. Venable, and T. Walsh. Preferences in constraint satisfaction and optimization. *AI Magazine*, 29(4) :58, 2009.

[Sav51] LJ Savage. The theory of statistical decision. *Journal of the American Statistical Association*, pages 55–67, 1951.

[Sch92] T. Schiex. Possibilistic constraint satisfaction problems, or "how to handle soft constraints ?". In *Proc. 8th Conf. of Uncertainty in AI*, pages 269–275, 1992.

[Sch00] T. Schiex. Réseaux de contraintes. *HDR, INRA*, 2000.

[SFV95] T. Schiex, H. Fargier, and G. Verfaillie. Valued Constraint Satisfaction Problems : Hard and Easy Problems. In *International Joint Conference on Artificial Intelligence*, volume 14, pages 631–639. LAWRENCE ERLBAUM ASSOCIATES LTD, 1995.

[SH91] A.C. Siochi and D. Hix. A study of computer-supported user interface evaluation using maximal repeating pattern analysis. In *Proceedings of the SIGCHI conference on Human factors in computing systems : Reaching through technology*, pages 301–305. ACM New York, NY, USA, 1991.

[SH01] A.A. Salo and R.P. Hamalainen. Preference ratios in multiattribute evaluation(PRIME)-elicitation and decision procedures under incomplete information. *IEEE Transactions on Systems, Man, and Cybernetics, Part A : Systems and Humans*, 31(6) :533–545, 2001.

[SH04] A. Salo and R.P. Hämäläinen. Preference programming. *Manuscript.(Downloadable at http ://www. sal. hut. fi/Publications/pdf-files/msal03b. pdf)*, 2004.

[ShF96] D. Sam-haroud and B. Faltings. Consistency techniques for continuous constraints. *Constraints*, 1 :85–118, 1996.

[SKR99] J. Ben Schafer, Joseph A. Konstan, and John Riedi. Recommender systems in e-commerce. In *ACM Conference on Electronic Commerce*, pages 158–166, 1999.

[SL01] S. Shearin and H. Lieberman. Intelligent profiling by example. In *Proceedings of the 6th international conference on Intelligent User Interfaces*, pages 145–151. ACM New York, NY, USA, 2001.

[Tsa93] E. Tsang. *Foundations of constraint satisfaction*. Academic Press San Diego, 1993.

[Vap95] V.N. Vapnik. The nature of statistical learning theory. 1995.

[Wal50] A. Wald. *Statistical Decision Functions*. 1950.

[ZAD98] L.S. Zettlemoyer, R.S. Amant, and M.S. Dulberg. IBOTS : agent control through the user interface. *Proceedings of the 4th international conference on Intelligent user interfaces*, pages 31–37, 1998.

Annexes

Annexe A

Constitution d'un fichier de logs d'interaction

Malgré ce qu'apporte l'étude des enregistrements d'interaction, cette méthode est encore peu utilisée (pour des travaux récent sur le sujet, voir par exemple [Dam08]). L'analyse des fichiers ne constitue pas la seule difficulté de ce procédé. D'une part des problèmes d'ordre technique apparaissent dès que l'on se pose la question de la méthode d'enregistrement des évènements utilisateur. D'autre part, devant le flot énorme des évènements générés pendant une interaction, il est crucial de filtrer à l'enregistrement un sous-ensemble d'évènements.

Nous présentons dans la première section différentes techniques d'enregistrement des évènements d'une interface graphique utilisateur. Dans la seconde section, nous concentrons notre propos sur les évènements issus d'une interface JAVA de type AWT et SWING, permettant de guider la sélection des évènements pertinents.

A.1 Récupération des logs

A.1.1 L'instrumentation

Pour voir ce qui se passe au sein d'une application, traditionnellement, on doit « instrumenter » le code c'est-à-dire insérer des lignes dans le code source pour suivre son exécution, ou pour écrire ces données dans un fichier de logs. Instrumenter le code est en théorie un bon moyen d'obtenir des informations précises : il est facile de spécifier quelles informations écrire dans les logs, en choisissant leurs formats et leurs emplacements. Cela peut constituer la base pour diverses formes d'évaluation ou de debugging, et pas seulement pour les tests d'utilisabilité. La difficulté de l'instrumentation est dans le fait de placer la bonne ligne de code au bon endroit.

Cela implique d'avoir une connaissance détaillée de la structure de l'application et de l'information à enregistrer, un savoir dont ne dispose pas forcément l'ergonome. Celui-ci opère souvent en tant que conseiller durant la conception et le développement, et non en tant que programmeur durant l'implémentation. Par conséquent, l'insertion de ces lignes se fait par l'intermédiaire d'un développeur, dirigé par l'expert, laissant ainsi la porte ouverte aux malentendus et erreurs de programmation. L'instrumentation entraîne aussi la nécessité d'avoir à disposition le code source ou un développeur impliqué dans le projet, ce qui n'est pas toujours le cas. Le plus souvent, les développeurs sont réticents à l'annotation de leur code pour des tests d'utilisabilité, qui n'affectent pas directement leur travail, contrairement au debugging des fonctionnalités de l'application. Le temps passé à instrumenter le code, ainsi que le risque accru d'ajouter des erreurs, expliquent pourquoi cela est rarement réalisé dans le cadre d'étude d'utilisabilité. Une autre limite à cette méthode est qu'elle n'est valide que pour un seul programme. Autrement dit, le même procédé d'insertion de lignes, incluant le même lot de problèmes, doit être répété pour chaque application que l'on veut évaluer ou déboguer. D'autres moyens plus pertinent existent pour l'enregistrement d'évènements, exploitant des "points d'entrée" des librairies graphiques. Nous en présentons deux dans la suite.

A.1.2 Windows Event Logging API

Si on se limite aux tests de programmes tournant sous une plate-forme Windows, on peut obtenir quelques informations sur les évènements d'une interface, comme les saisies clavier et les clics de souris, par le biais d'une API fournie par Windows. De cette manière on peut intercepter des évènements envoyés au système d'exploitation, venant des périphériques d'entrée/sortie : claviers, souris, imprimantes. De ce fait, il est possible d'obtenir la touche du clavier qui a été pressée et ainsi enregistrer ce qu'écrit l'utilisateur. On peut aussi intercepter des informations concernant les clics venant de la souris et sa position à l'écran, en plus du titre de la fenêtre active (i.e. qui possède le focus). Dans certains cas, il est également possible d'obtenir le label de boutons, suite au clic d'un utilisateur mais uniquement sur des fenêtres dépendantes de Windows, comme les fenêtres d'ouverture de fichier ou de "Préférences". Ces données relèvent donc principalement d'évènements de bas niveau, comme les saisies clavier ou clics de souris, mélangées à quelques évènements de haut niveau, comme le nom de la fenêtre active. Entre les deux, il y a un "trou", ne donnant pas à l'évaluateur un suivi précis de la session de test. Pour savoir ce qu'est réellement en train de faire l'utilisateur, nous devons chercher des informations qui ne peuvent être obtenues par l'intermédiaire de la bibliothèque Windows Event Logging, puisqu'elle ne fournit aucune données sur l'interaction réelle au sein du programme.

A.1.3 JAVA Accessibility API

Si l'on se limite aux applications JAVA, il existe un moyen d'obtenir tous les évènements d'une interface graphique sans instrumenter le code. La solution est donnée par l'utilisation de JAVA Accessibility API qui est traditionnellement utilisée pour adapter l'interface aux personnes ayant divers handicaps. Les technologies d'accessibilité offrent une variante de l'interface graphique pour aider ces personnes à interagir avec le logiciel. Des exemples de ces applications sont les lecteurs d'écrans, programmes lisant à haute voix ce qui est montré à l'écran ou la loupe, qui agrandit la zone où se situe le pointeur de souris. Des applications comme celles-ci doivent posséder des informations précises sur l'interface pour connaître ce qu'elles ont à présenter dans leur interface modifiée. La possibilité d'utiliser les technologies d'accessibilité existe dans la JVM (Java Virtual Machine) depuis la version 1.2. Cela limite donc l'enregistrement des évènements sur les interfaces Java développées depuis cette version. Un réel avantage est que Java est indépendant de la plate-forme donc l'enregistrement des évènements fonctionne sur tous les systèmes.

Tous les évènements générés se retrouvent dans une "file d'attente" de la JVM en attendant d'être traités, cette file étant contrôlée par le programme d'accessibilité. Pour y avoir accès, l'application d'accessibilité doit tourner sur la même JVM que le programme principal. Dans ce but, il faut ajouter les classes de Java Accessibility dans le JRE puisqu'elles n'y sont pas par défaut. Elles sont téléchargeables sur le site de Sun sous la forme d'un jar, nommé jaccess.jar que l'on placera dans le répertoire « jre<n° de version>/lib/ ». De plus, il faut placer dans le répertoire « ext » situé au même endroit, un fichier nommé « accessibility.properties » contenant la ligne suivante :

assistive_technologies = "Nom de la classe de l'enregistreur d'évènements".

A la suite de cela, toutes les applications Java ayant une interface graphique AWT/Swing et tournant sur cette JVM seront « écoutées » et éventuellement adaptées par le programme.

La file est remplie par les évènements venant des packages AWT et Swing de Java. Par exemple, un clic sur un bouton engendre un ActionEvent et l'ouverture d'un menu insère un MenuEvent dans la file des évènements. En tout, il y a 28 types d'évènements pouvant être présents dans la file, offrant ainsi une riche source d'information utilisable pour l'analyse. Ces évènements peuvent servir dans un contrôle temps réel ou être sauvegardé sous forme de fichier de logs.

L'énorme quantité d'évènements générés et récupérés dans la file pose une première difficulté de traitement. Tous ne sont pas pertinents pour l'étude d'utilisabilité. Seuls certains fournissent des indications sur ce que fait l'utilisateur. Les autres ne révèlent rien, ou dans la plupart des cas, sont redondants. Nous pouvons, par exemple,

mettre de côté tout évènement relatant la perte du focus par un composant, cette information pouvant être induite par le fait que d'autres l'obtiennent. Chaque action de l'utilisateur peut produire un à plus d'une douzaine d'évènements. Le défi se situe donc dans l'élaboration de filtres permettant l'enregistrement des seuls évènements utiles à une étude d'utilisabilité. Définir des filtres généraux, applicables sur plusieurs interfaces, est délicat. En effet, un évènement peut paraître sans importance pour une application tandis qu'il donne une indication précieuse pour une autre. Les évènements relatifs à la souris, par exemple, ne sont pas forcément utiles pour une application du type "Éditeur de texte". En revanche, ils se relèvent indispensables si le programme est une application de cartographie routière (utilisation de la fonction de déplacement sur la carte, etc.).

Pour résumer, il n'est pas nécessaire que le programme testé soit spécialement conçu, pour utiliser cette technique d'enregistrement des évènements. Mais le programme d'enregistrement, lui, doit être adapté au type d'interface à étudier.

Il faut ajouter qu'utiliser les évènements d'une interface graphique pour des tests d'utilisabilité suppose qu'une bonne conduite de programmation ait été faite en amont, c'est-à-dire que les objets soient tous nommés et que les informations soient préalablement initialisées. Autrement il sera difficile d'extraire des informations utiles des fichiers de logs.

A.2 Évènements de l'interface graphique Java

Voici un bref listing des évènements d'interface générés par les librairies AWT et Swing de Java. Cette liste n'est pas exhaustive puisque certains évènements ne retranscrivent pas les actions de l'utilisateur. Mais ces tableaux permettent de se faire une idée sur les évènements pertinents à enregistrer.

En observant la définition de ces évènements, on peut établir différentes catégories basées sur le niveau d'abstraction où ils se situent. Par exemple, le ActionEvent, généré par un bouton, englobe :

- la pression sur le bouton gauche de la souris situé sur un composant de type JButton
- le relâchement de la pression exercée sur le bouton

A.2.1 Évènements de base

Ces tableaux rassemblent différents évènements intervenant dans l'utilisation d'une interface. Les "méthodes à implémenter" sont celles de l'écouteur (listener) ayant le même préfixe que l'évènement qu'il écoute, auquel on ajoute le suffixe "Lis-

tener" (sauf dans les cas précisés) à la place du suffixe "Event". Par exemple pour un évènement de la classe FocusEvent, l'écouteur approprié sera un FocusListener.

Nom	Utilisation	Méthodes à implémenter
ComponentEvent	[awt] Se produit quand un composant change de taille, bouge ou devient visible/invisible	componentHidden componentMoved componentResized componentShown
FocusEvent	[awt] Généré quand un composant perd ou gagne le focus	focusGained focusLost
KeyEvent	[awt] Se produit quand une saisie clavier a été faite sur un composant. Il s'agit des évènements envoyés quand une touche est pressée, relâchée ou les deux (« tapée »).	keyPressed keyReleased keyTyped
MouseEvent	[awt] Évènement relatif aux actions de la souris. Comme un clic, une pression sur un des boutons, l'entrée et la sortie du pointeur sur le composant écouteur, un relâchement de la pression sur le bouton. Mais encore le déplacement de la souris avec un bouton pressé ou non.	mouseClicked mouseEntered mouseExited mousePressed mouseReleased MouseMotionListener mouseDragged mouseMoved
MouseWheelEvent	[awt] Envoyé lorsque la souris est utilisée avec sa molette	mouseWheelMoved

Nom	Utilisation	Méthodes à implémenter
WindowEvent	[awt] Évènements relatifs aux modifications d'une fenêtre	windowActivated windowClosed windowClosing windowDeactivated windowDeiconified windowIconified windowOpened WindowFocusListener windowGainedFocus windowLostFocus WindowStateListener windowStateChanged

A.2.2 Évènements riches

Voici le tableau des évènements qui représentent plus clairement une action de l'utilisateur, elle représente donc des éléments de plus haut niveau regroupant un ensemble d'évènements atomiques. Certains de ces évènements sont spécifiques à un composant Java particulier, tandis que d'autres sont générés par une multitude de composants.

Nom	Utilisation	Méthodes à implémenter
ActionEvent	[awt] Généré par des composants comme les Boutons ou les JComboBox.	actionPerformed
AdjustementEvent	[awt] Généré par les composants chargé de l'ajustement des composants (ScrollBar)	adjustementValueChanged
AncestorEvent	[swing] Envoyé aux enfants quand la hiérarchie des composants est changée.	ancestorAdded ancestorMoved ancestorRemoved
CaretEvent	[swing] Généré quand dans un composant textuel le curseur a été déplacé.	caretUpdate

Nom	Utilisation	Méthodes à implémenter
ChangeEvent	[swing] Si on modifie le contenu d'un composant	stateChanged CellEditorListener editingCancel editingStopped
ContainerEvent	[awt] Envoyé quand le contenu du Container est changé (par un ajout ou une suppression)	componentAdded componentRemoved
DocumentEvent	[swing] Généré si un document est modifié (texte simple, HTML, XML).	changeUpdate insertUpdate removeUpdate
HyperLinkEvent	[swing] Evènement généré par l'activation d'un lien HyperText dans un JEditorPane.	HyperLinkUpdate
ItemEvent	[awt] Evènement qui est généré quand un nouvel item est sélectionné (dans une liste du type ComboBox, par exemple)	itemStateChanged
InternalFrameEvent	[swing] Evènement relatif aux InternalFrame (ayant les mêmes capacités que les Windows)	internalFrameActivated internalFrameClosing internalFrameClosed internalFrameDeactivated internalFrameDeiconified internalFrameIconified internalFrameOpened
ListDataEvent	[swing] Généré lorsqu'un élément d'une liste est modifié	contentsChanged intervalAdded intervalRemoved
ListSelectionEvent	[swing] Produit lorsque la sélection au sain d'une liste a changé	valueChanged columnSelectionChanged
MenuDragMouseEvent	[swing] Généré par un menu qui autorise le « Drag » de ces items.	menuDragMouseDragged menuDragMouseEntered menuDragMouseExited menuDragMouseReleased
MenuEvent	[swing] Évènements relatifs aux menus	menuCanceled menuDeselected menuSelected

Nom	Utilisation	Méthodes à implémenter
MenuKeyEvent	[swing] Évènements relatifs aux menus sollicités par une action saisie clavier	menuKeyPressed menuKeyReleased menuTyped
PopupMenuEvent	[swing] Évènements relatifs aux PopupMenu	popupCanceled popupWillBecomeInvisible popupWillBecomeVisible
TableColumnModelEvent	[swing] Produit lorsqu'on modifie les colonnes d'une table (ajouter, modifier, effacer)	columnAdded columnMarginChanged columnMoved columnRemoved columnSelectionChanged
TableModelEvent	[swing] Généré quand le contenu du tableau change.	tableChanged
TextEvent	[awt] Envoyé par un textComponent qui subi une modification de son contenu.	textValueChanged
TreeExpansionEvent	[swing] Produit quand un nœud du JTree est étendu ou rétracté.	treeCollasped treeExpanded treeWillExpandListener treeWillCollapse treeWillExpand
TreeModelEvent	[swing] Evènement généré quand le modèle du JTree est modifié	treeNodeChanged treeNodeInserted treeNodeRemoved treeStructureChanged
TreeSelectionEvent	[swing] Généré quand la sélection courante à été changée	valueChanged
UndoableEditEvent	[swing] Envoyé quand une action de type Undo/Redo est exécutée.	undoableEditHappened

A.2.3 Autres évènements

Évènements n'appartenant pas aux packages awt et swing ou utilisés pour un type d'interaction bien précis comme le Drag&Drop.

Nom	Package	Utilisation	Méthodes à implémenter

Nom	Package	Utilisation	Méthodes à implémenter
DragGestureEvent	java.awt.dnd	Envoyé par un composant quand un geste de drag est réalisé sur un objet Transferable	dragGestureRecognized
DragSourceEvent	java.awt.dnd	Généré quand un composant accepte une action de drag	dragExit
DragSourceDragEvent	java.awt.dnd	Produit lorsqu'une action drag continue à être sur la source	dragEnter dragOver dropActionChanged
DragSourceDropEvent	java.awt.dnd	Produit lorsque le geste de drag&drop est complet	dragDropEnd
DropTargetEvent	java.awt.dnd	Produit lorsqu'un composant qui pouvait supporter des actions de Drop est sollicité.	dragExit
DropTargetDragEvent	java.awt.dnd	Envoyé lorsqu'une action de drag arrive sur un composant supportant les dépôts.	dragEntrer dragOver

Nom	Package	Utilisation	Méthodes à implémenter
DropTargetDropEvent	java.awt.dnd	Envoyé lorsqu'une action de drop est faite sur un composant récepteur	Drop
PropertyChangeEvent	java.beans	Généré par des modifications de taille ou de contraintes dans les composants	propertyChange vetoableChangeListener vetoableChange

Annexe B

LogHandler

Nous présentons dans ce chapitre l'application nommée LogHandler, développée dans le but de faciliter le traitement des fichiers de logs récupérés. Ce traitement s'inscrit dans le schéma classique du processus de fouille de données (fig. B.1), de l'étape de prétraitement et nettoyage des données à l'étape d'analyse. Après une présentation générale de l'application, nous détaillons les fonctionnalités fournit pour chacune de ces étapes.

B.1 Présentation de l'application

Réalisée en Java, l'application LogHandler permet de charger des données d'interaction sous forme de logs quelconque, de les nettoyer et de les stocker dans une base de données (fig. B.2). A partir de cet état, LogHandler permet d'exécuter différentes analyses, comme le calcul de statistiques globales ou la recherche de séquences particulières d'évènements. Les résultats des analyses sont présentés au format HTML.

De plus, l'application permet de traduire les logs nettoyés dans le format ARFF[1], les rendant compatibles avec l'application WEKA, une librairie JAVA open source, qui offre un ensemble de méthodes de fouille de données.

B.1.1 Interface principale de l'application

L'interface graphique est divisée en quatre sections qui correspondent aux étapes d'une première analyse d'un fichier de logs. L'affichage tente de partager les fonctions de chargement/correction d'un fichier, de sélection des données, du paramétrage des analyses et de l'affichage des résultats (fig. B.3).

[1]ARFF (Attribute Relation File Format) : Format de fichiers utilisé par la librairie WEKA consacrée au Data Mining (http ://www.cs.waikato.ac.nz/ml/weka/)

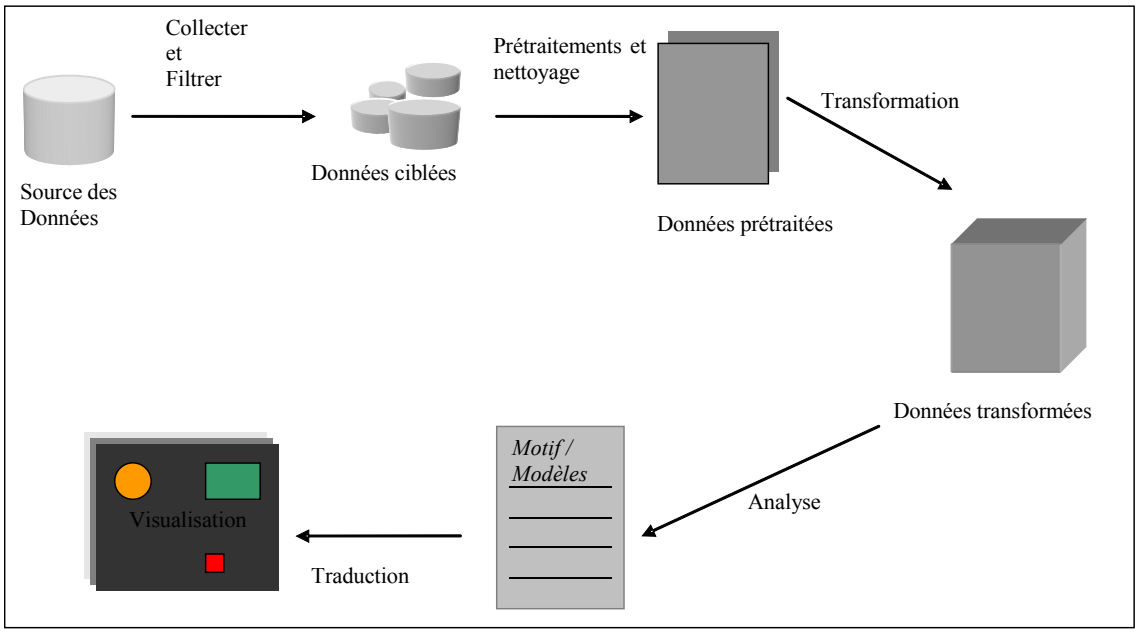


FIG. B.1: Schéma général de la fouille de données

Les fonctions de chargement et de correction d'un fichier de logs sont accessibles par l'option « Load File » du menu « File ». Les autres fonctionnalités sont directement visibles sur l'interface. La sélection des données se fait par l'intermédiaire d'un arbre dont la sélection a été redéfinie. L'utilisateur ne peut sélectionner que les fichiers d'une seule interface à la fois car nous avons fait l'hypothèse que les fichiers d'une même interface avaient tous le même format. Les onglets de résultats restant affichés même après un changement de sélection, on peut comparer les résultats de deux interfaces.

Arbre de sélection

Nous avons vu dans la description de l'interface, la place importante qu'occupait l'arbre de sélection. En effet, c'est par le biais de ce composant que l'utilisateur va déterminer quels fichiers seront analysés. Une sélection de tel ou tel fichier dans l'arbre va par la suite affecter la saisie des paramètres des analyses (cette caractéristique est décrite en détail dans la suite). L'arbre compte en fait trois niveaux, le premier n'est autre que le nœud de la base de données. Le second niveau est celui des interfaces des applications étudiées. Nous avons choisi de considérer que chacun de ces nœuds représente une version particulière d'une même interface (i.e. application). Le troisième niveau est celui des fichiers issus de l'utilisation de ces

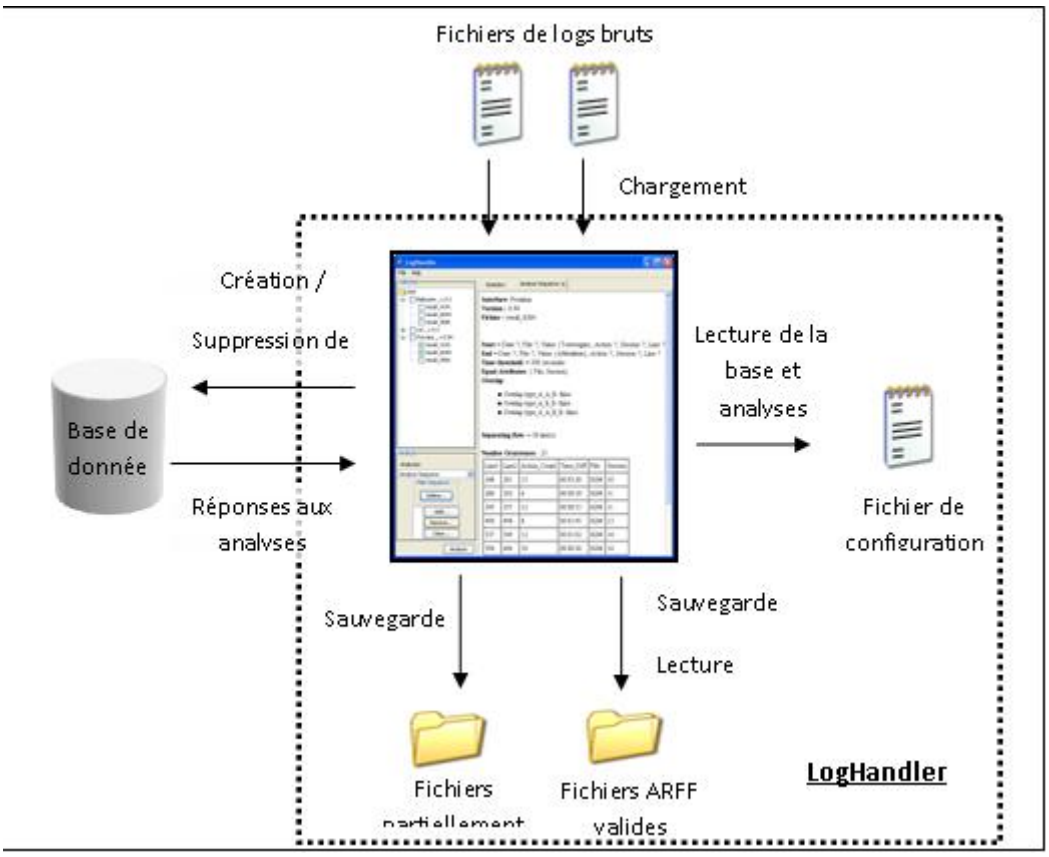
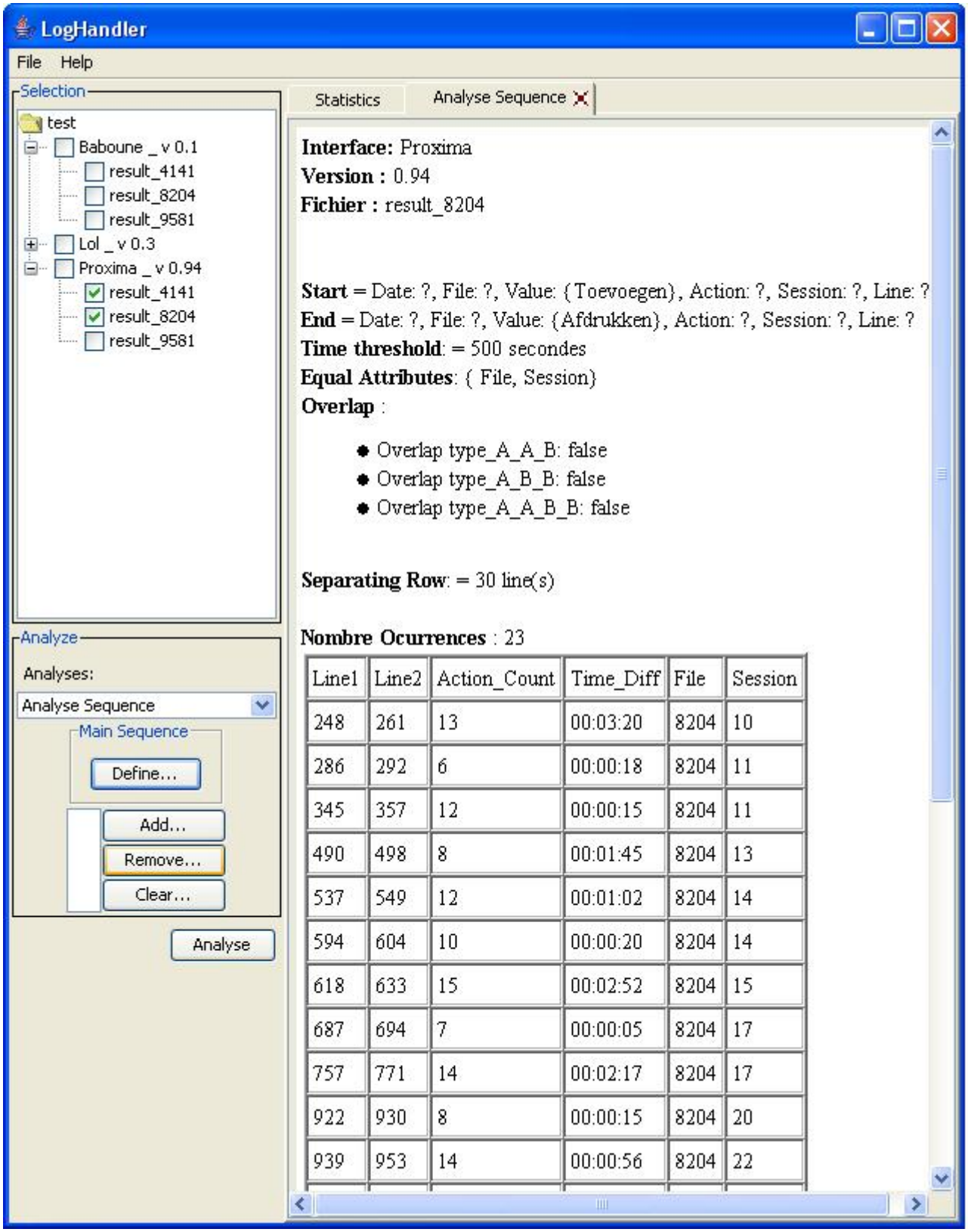


FIG. B.2: Schéma détaillant les ressources utilisées par l'application



Line1	Line2	Action_Count	Time_Diff	File	Session
248	261	13	00:03:20	8204	10
286	292	6	00:00:18	8204	11
345	357	12	00:00:15	8204	11
490	498	8	00:01:45	8204	13
537	549	12	00:01:02	8204	14
594	604	10	00:00:20	8204	14
618	633	15	00:02:52	8204	15
687	694	7	00:00:05	8204	17
757	771	14	00:02:17	8204	17
922	930	8	00:00:15	8204	20
939	953	14	00:00:56	8204	22

FIG. B.3: Interface principale du LogHandler

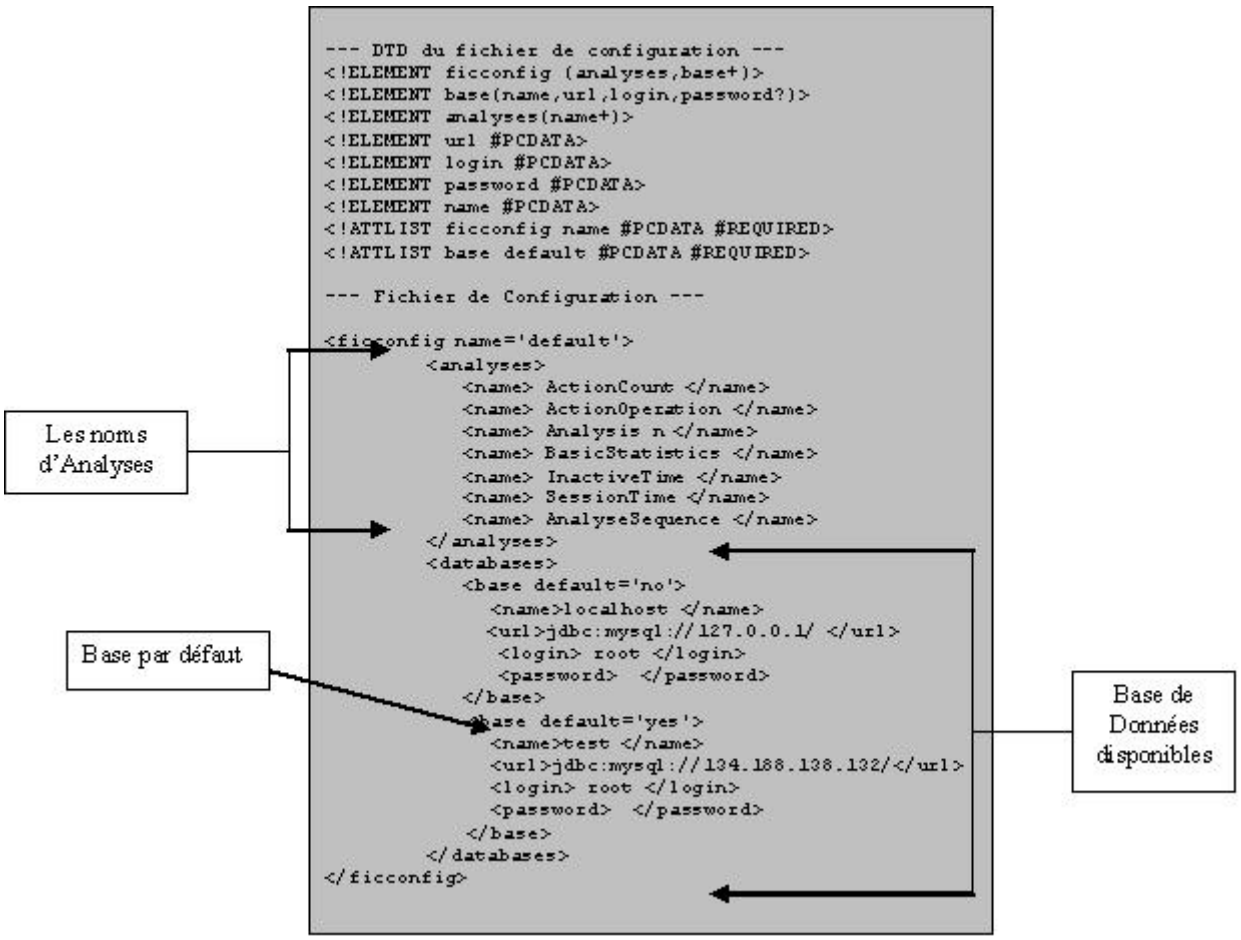


FIG. B.4: Exemple d'un fichier de configuration et de sa DTD

interfaces. La sélection dans l'arbre est contrainte de la façon suivante : tous les nœuds « fichier » sélectionnés doivent avoir le même nœud parent, c'est-à-dire être issus de la même interface. Cette contrainte est due aux algorithmes d'analyses qui ne peuvent s'appliquer que sur un format de log à la fois.

B.1.2 Configuration de l'application

L'application LogHandler utilise un fichier de configuration pour déterminer quelle base de données utiliser mais aussi quelles analyses proposer dans l'interface de l'utilisateur. Ce fichier de configuration n'est lu qu'une seule fois à l'initialisation de l'application ; cela permet d'accéder aux données nécessaires à la mise en place de certains mécanismes de l'interface. Voici un exemple de fichier de configuration avec sa DTD :

L'application réagira au fait qu'une base ne soit pas ou plus accessible, en effet une erreur sera générée au moment de la connexion (après le lancement de l'application) et les fonctionnalités de l'application ne seront pas disponibles. La mention «No DataBase » sera précisée en haut de la zone de sélection.

Dans la définition des analyses sont précisées toutes les analyses qui doivent être disponibles dans l'interface. Simplement le nom ou le chemin (des packages) amenant à la classe d'analyse est demandé pour ajouter une analyse. Si une analyse est marquée dans le fichier alors qu'aucune classe n'y correspond, elle ne sera tout simplement pas proposée à l'utilisateur.

Ce mécanisme augmente l'évolutivité de l'application LogHandler, en permettant l'ajout de nouvelles méthodes d'analyse de manière simple.

B.1.3 Préparation des paramètres d'analyses

Une fois que les méthodes d'analyses présentes dans l'application sont déterminées et que l'arbre est construit, différents mécanismes visant à faciliter la saisie des paramètres des analyses sont mis en œuvre. Durant l'initialisation, les en-têtes des fichiers stockés dans la base sont chargés. Les valeurs distinctes des attributs de type « string » ou « nominal » sont également chargées, permettant ainsi de proposer une liste des noms des attributs et des valeurs de certains attributs. Grâce à cela, on peut utiliser des composants du type « JComboBox » ou « JList » plutôt qu'un champ de texte qui demanderai plus de travail à l'utilisateur de LogHandler.

Si le chargement d'un fichier ne pose apparemment pas de problèmes, il est nécessaire de rappeler que la taille des fichiers de logs est souvent conséquente (les 20 000 lignes sont en général dépassées). La mise en place de tous ces mécanismes de saisies entraîne donc un certain coût. C'est pour cela que nous avons décidé de charger les données nécessaires pendant la phase d'initialisation de l'application. En effet, réalisé au moment de la sélection, un chargement entraînerai un "gel" de l'interface pendant le temps du chargement. Pendant la phase d'initialisation, ce traitement est caché jusqu'à l'affichage de l'interface.

B.2 Nettoyage des données

Dans cette section, nous introduisons les étapes correspondant au chargement et au nettoyage des données d'interaction.

Attribut	Type	Valeur
Line	Numeric	
Files	Numeric	
Session	Numeric	
Date	Date	'yyyy-MM-dd HH :mm :ss'
Event	Nominal	WINDOW_DEACTIVATED WINDOW_ACTIVATED WINDOW_ICONIFIED ACTION ITEM_STATE_CHANGED WINDOW_OPENED WINDOW_DEICONIFIED WINDOW_CLOSED WINDOW_CLOSING
Value	String	

TAB. B.1: Format de logs récupérés

B.2.1 Chargement d'un fichier

Des logs vers un format plus souple

Nous avons présenté les fichiers log qui ont été récupérés après les bêta-tests du produit d'Océ (Tab. B.1). Ces fichiers logs sont constitués de 6 attributs dont un de type Date. Nous allons dresser le tableau des types WEKA correspondants à ces attributs, ce qui nous servira à comprendre comment générer l'en-tête du fichier ARFF correspondant à un fichier de logs quelconque.

Les logs enregistrés sont le fruit d'un premier filtrage des évènements. Nous avons précisé plus haut que seuls les évènements ActionEvent, ItemEvent et WindowEvent ont été conservés. Les valeurs du champ Event ont donc un nombre de possibilités très réduit par rapport à ce qu'on peut obtenir. C'est typiquement un attribut de type "nominal".

Transformation des logs vers un format hybride

Pour traiter les fichiers de logs nous avons besoin du séparateur d'attributs. Dans nos logs ce séparateur n'est autre que le « ; ». Nous avons donc toutes les informations pour décomposer chaque ligne de notre fichier. Le premier traitement réalisé est une analyse sur un échantillon des lignes de logs, afin de "deviner" le nombre d'attributs et leurs types.

Une ligne est un vecteur dont l'élément contenu à l'indice j est la valeur de l'at-

yyyy-dd-MM HH :mm :ss	HH :mm :ss yyyy-MM-dd
yyyy-MM-dd HH :mm :ss	HH :mm :ss yyyy-dd-MM
MM-yyyy-dd HH :mm :ss	HH :mm :ss MM-yyyy-dd
dd-yyyy-MM HH :mm :ss	HH :mm :ss dd-yyyy-MM
MM-dd-yyyy HH :mm :ss	HH :mm :ss MM-dd-yyyy
dd-MM-yyyy HH :mm :ss	HH :mm :ss dd-MM-yyyy

TAB. B.2: Types de dates reconnues par le parseur

tribut j dans cette ligne. Jusqu'à présent nous avons stocké uniquement les données d'interaction contenues dans le fichier. Il faut préparer le fichier pour la génération et la modification éventuelle de son en-tête. Pendant le scan des lignes, nous avons stocké les valeurs distinctes de chaque attribut. Ceci dans le but d'offrir à l'utilisateur la possibilité de changer le type de son attribut en type nominal. De plus cela permet de donner une prévision du type de chaque attribut. On va déterminer les types des champs d'un log, grâce à un parcours d'un nombre fixe de lignes. Nous avons choisi de procéder à la vérification dans cette ordre : numeric, date, nominal et string. Le type string est le type par défaut d'un attribut, s'il n'a pas validé les premiers tests. La vérification d'un champ numérique est simple, il suffit de regarder si la chaîne ne contient rien d'autre que des chiffres ou une virgule. Un type nominal peut-être considéré comme un attribut ayant un nombre de valeurs distinctes assez faible (constante fixée à 15 valeurs). Le type date impose une vérification plus rigoureuse que les autres types. En effet, nous tentons de reconnaître 12 formats de date référencés dans le tableau B.2.

Cependant il existe des cas où le format sélectionné n'est pas le bon. En effet, certains formats sont équivalents sur une période bien précise du mois. Par exemple, du 01 mai 2006 au 12 mai 2006, les deux formats suivants sont équivalents : « yyyy-dd-MM HH :mm :ss », « yyyy-MM-dd HH :mm :ss ». Mais arrivé le 13 Mai, seul l'un des deux reste valide. Le fait de déterminer le format sur un échantillon va mener à ce type de cas. Une des solutions possibles serait de tirer les lignes étudiées aléatoirement dans le fichier. Mais il n'est pas garanti que ce cas de figure ne se reproduise pas. Reconnaître tous ces formats va nous permettre de manipuler une bonne partie de fichiers de logs ayant un champ date. Cependant nous n'utiliserons par la suite qu'un seul format de date afin de nous simplifier la manipulation des données.

La détermination du type va permettre d'avoir un premier en-tête du fichier qui va nous servir à repérer certaines erreurs du log. Ces erreurs seront stockées lors d'une seconde lecture du fichier, après que l'en-tête soit défini. Pour le moment, on considère que les erreurs sont les indices des lignes contenant au moins un attribut

non valide, manquant ou supplémentaire.

Le contenu d'un fichier de logs est maintenant :

- Noms des attributs
- Nombre d'attributs
- Un tableau de lignes
- Le type des attributs
- L'ensemble des valeurs distinctes des attributs
- Indices des lignes contenant des erreurs.

Cette structure ressemble à celle d'un fichier ARFF ; elle permet de modifier l'en-tête et de repérer les lignes erronées.

Le parseur ARFF ad hoc (raisons et mise en oeuvre)

Nous avons vu que l'application utilisait des fichiers ARFF contenant encore des erreurs. Nous aurions pu utiliser le parseur ARFF fourni avec WEKA si celui-ci ne s'arrêtait pas à la première erreur. De plus, ce parseur ne conserve pas les valeurs distinctes de tous les attributs mais seulement celles de type « string » et « nominal ». Nous avons implanté notre propre parseur ARFF capable de parcourir tout un fichier malgré les erreurs qu'il contient et donnant la possibilité de changer l'en-tête. Le principe de ce parseur ne diffère pas beaucoup de celui du parseur précédent. Cependant il doit prendre en compte la présence des commentaires, la signification du symbole « ? »[2] et la présence de la balise « @relation » qui est le nom de la base de données contenues dans le fichier. Nous allons encore changer notre représentation du fichier log pour y inclure cette balise. N'ayant pas cette information dans nos logs bruts, on considéra que la valeur par défaut est le nom du fichier.

Les étapes du chargement

A cette étape, deux possibilités s'offrent à l'utilisateur : soit le fichier est un log quelconque, soit les données d'interaction sont exprimées au format ARFF (contenant des erreurs ou non). Dans le premier cas, l'utilisateur passe par un écran de saisie du caractère séparateur avant d'obtenir le panneau de correction. Dans le cas d'un fichier au format ARFF, on accède directement au panel d'édition de l'entête et de correction après le chargement du fichier (Fig. B.5).

[2]Symbole « ? » : le point d'interrogation dans le format ARFF exprime le fait que la valeur pour ce champs soit inconnue.

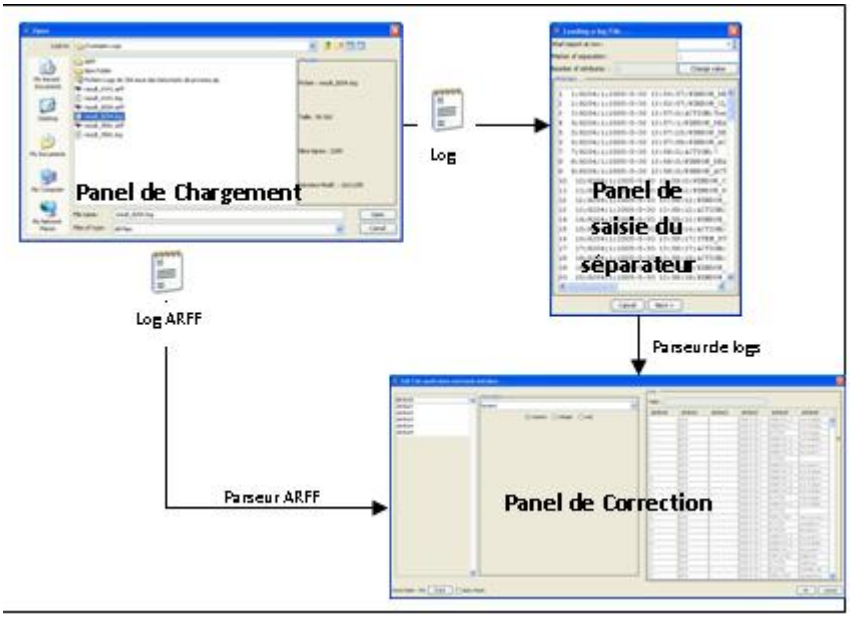


FIG. B.5: Chargement d'un fichier de logs dans l'application

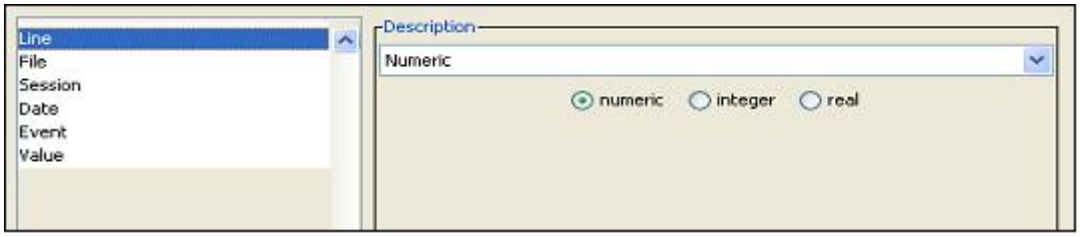


FIG. B.6: Sélection d'un type numérique

B.2.2 Définition de l'en-tête du fichier

Maintenant il est possible de modifier l'en-tête de notre fichier grâce au panneau de correction. La première détermination du type des attributs peut ne pas être valide et c'est, dans ce cas, à l'utilisateur d'opter pour le type ou le format qui lui convient le mieux. Avec le panel de modification d'en-tête, on peut modifier le type de tous les attributs. Les possibilités liées à chaque type ont quasiment toutes été décrites.

Pour les attributs numériques (Fig. B.6), nous avons le choix entre :

– « integer » : pour des valeurs entières
– « real » : pour les valeurs réelles
– « numeric » : pour toutes valeurs « integer » ou « real »

Pour le type « Date » (Fig. B.7), l'utilisateur a le choix entre les douze formats décrits dans le tableau B.2.

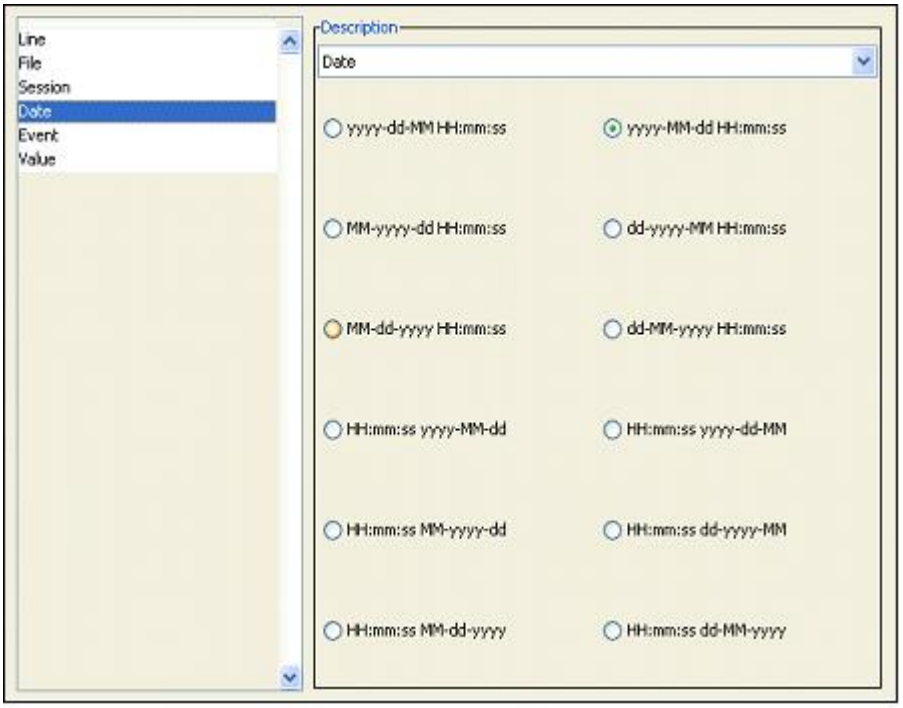


FIG. B.7: Sélection du format de type Date

Avec le type « Nominal », on a la possibilité de rajouter ou d'enlever un élément de l'ensemble ou de rafraîchir la liste des symboles pour prendre en compte des modifications qui ont eu lieu dans le panel de correction (Fig. B.8).

Quant au type « String », aucune option particulière n'est disponible.

Une fonction "accessoire" de ce panneau est la saisie des noms des attributs. A la place des labels donnés par défaut, l'utilisateur peut le renommer afin de les désambigüiser et de faciliter le paramétrage des analyses par la suite. A côté de cet écran, nous avons celui de correction qui permettra de visualiser et de corriger les erreurs du fichier. Cette seconde partie de l'écran dépend fortement de l'en-tête qui a été saisi avec les éléments présentés dans cette partie.

B.2.3 Correction et réévaluation des erreurs

Pour la phase de correction, nous allons présenter sur la notion d'erreur laissée en suspens. C'est pour cela que, dans un premier temps, nous verrons comment se définit une erreur mais aussi comment nous allons l'utiliser pour déterminer les lignes potentiellement intéressées par cette modification.

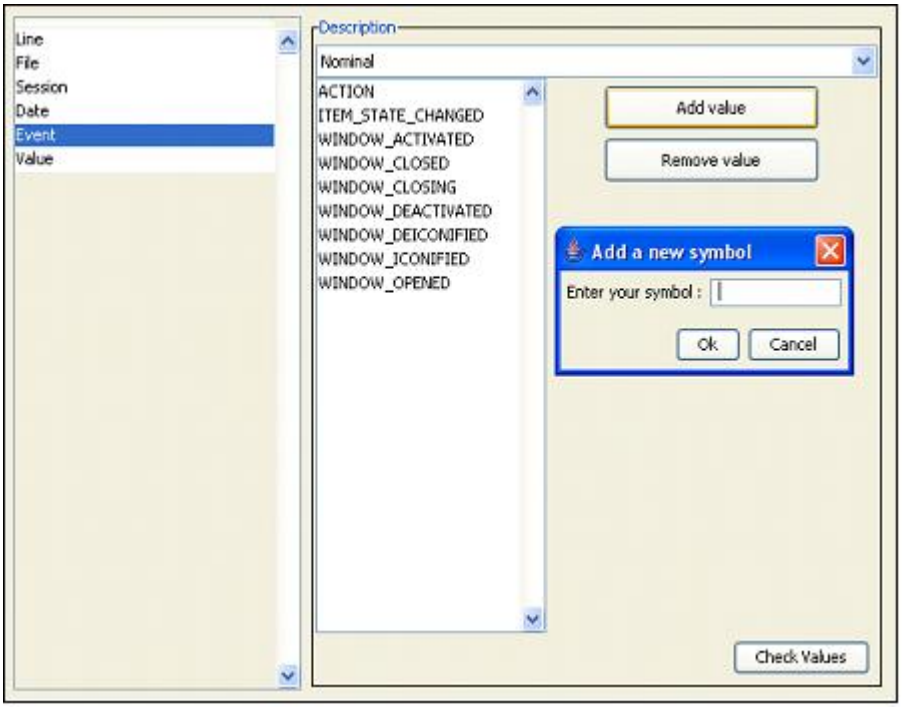


FIG. B.8: Affichage du panneau pour le type nominal

Définition d'une ligne d'erreur

Avant de définir une ligne d'erreur, nous allons expliquer quels sont les quatre types d'état possible pour la valeur d'un attribut dans une ligne de log :

- VALUE_OK : qui valide la valeur par rapport au type attendu définit dans l'en-tête
- ERROR_TOO_MUCH_ATTRIBUTE : cet état est présent lorsque la ligne comporte trop d'attribut
- ERROR_TOO_FEW_ATTRIBUTE : lorsque la ligne ne possède pas assez d'attribut
- ERROR_VALUE : quand la valeur d'un champ ne correspond pas à l'en-tête du fichier.

Ces états vont nous permettre de représenter une ligne d'erreur et de localiser les différents problèmes la concernant. Une ligne d'erreur se représente par :

- numéro de ligne (indice de la ligne dans le fichier)
- la valeur de la ligne
- La liste des états de la ligne.

Prenons comme exemple, une ligne incorrecte de nos logs :

2376 ; 9581 ; 21 ; 2005-5-28 ;WINDOW_DEACTIVATED ; Océ Publisher Engineering 0.9.2.4 -127.0.0.1

La liste des états correspondant à cette ligne est :

VALUE_OK, VALUE_OK, VALUE_OK, *ERROR_VALUE*, VALUE_OK, VALUE_OK

En regardant la spécification de nos logs (voir Tab. B.1), il est normal de trouver une erreur pour le quatrième champ de cette ligne.

Nous allons regarder une ligne comportant plusieurs erreurs et voir quel sera le schéma généré :

1494 ; 9581 ; ; 2005-5-22 6 :20 :53 ; ITEM_STATE_CHANGED ;

Cette ligne contient une valeur de mauvais type et un attribut manquant, elle est classée de la façon suivante :

VALUE_OK, VALUE_OK, *ERROR_VALUE*, VALUE_OK, VALUE_OK , *ERROR_TOO_FEW_ATTRIBUTE*

Comme on le voit sur cet exemple, on distingue le fait qu'une valeur ne soit pas précisée et le fait qu'un attribut soit manquant. Posséder le schéma d'une ligne fautive va nous permettre de déterminer quels outils mettre à disposition pour la corriger en y ajoutant les traitements spécifiques à la case.

Processus de correction

Une fois les lignes identifiées, elles sont affichées en rouge sur le panel de correction (Fig. B.9). L'utilisateur a plusieurs options : faire des traitements globaux ou établir une correction sur les valeurs pris en défaut.

Pour les traitements dits globaux, nous avons :

- La suppression d'une colonne
- La séparation des valeurs d'une colonne
- La fusion de deux colonnes

Dans les traitements relatifs à une ligne et à une case sont :

- la suppression de la ligne
- la fusion de deux valeurs
- la modification d'une valeur

La fusion de valeurs et de colonnes n'est possible que si le nombre d'attributs dépasse le nombre d'attributs prévu par l'entête du fichier ARFF ou déterminé à la saisie du séparateur de champs. On observe donc si la ligne ou toutes les lignes d'erreurs possèdent l'état ERROR_TOO_MUCH_ATTRIBUTE. Il est alors demandé de préciser le séparateur qui va unir les valeurs des deux colonnes fusionnées. On en fera de même dans le cas où les lignes ne possèdent pas assez d'attributs et contiennent l'état ERROR_TOO_FEW_ATTRIBUTE. La suppression d'une colonne ne se fait

que si le nombre d'attributs dépasse le nombre prévu ou si cette colonne est vide. La modification d'une valeur et la suppression d'une ligne est réalisable sur chaque ligne mais leur traitement est différent en cas d'erreur ou de validité. En effet malgré le fait que l'on dispose de certains outils agissant sur plusieurs lignes à la fois, on n'a pas encore la possibilité de corriger toutes les fautes de même type en une seule passe. C'est pour cela que nous allons définir le concept d'opération.

Définition d'une opération

Une opération est produite lorsqu'une ligne erronée est corrigée. Cet élément va nous permettre d'appliquer une correction sur plusieurs lignes. On définit une opération comme :

- le type de traitement (fusion, correction, suppression)
- le schéma d'état de la ligne corrigée
- la valeur avant opération
- la valeur après opération
- la(les) attribut(s) concerné(s)
- le séparateur (si besoin)

Une fois la première correction faite, on va filtrer les lignes d'erreurs avec le schéma de la ligne corrigée. Puis selon le type d'opération, nous allons vérifier si la valeur du champ correspond à celle qui a été modifiée. Ainsi nous avons les erreurs potentiellement concernées par notre correction et l'utilisateur peut modifier ou non les lignes d'erreurs affichées.

Auto-check

Sous cette appellation se cache la fonction permettant de visualiser directement les erreurs causées/résolues par le changement d'un type ou la correction de lignes dans le fichier. Cette option est décochée par défaut puisque elle cause des pertes de performances gênantes sur les fichiers de grande taille. En effet, si l'utilisateur change le type d'un attribut nous devons vérifier les valeurs de chaque ligne pour cet attribut afin de déterminer les nouvelles erreurs. C'est pour cette raison que nous avons essayé d'améliorer la mise à jour des erreurs dans le panel de correction. Ainsi pour les traitements locaux (correction d'une valeur, par exemple) seules les lignes concernées seront vérifiées.

B.2.4 Devenir des fichiers corrigés

La phase de correction prend fin avec la sauvegarde des données corrigées dans la base de données. Cette étape se traduit par la conversion d'un fichier de logs

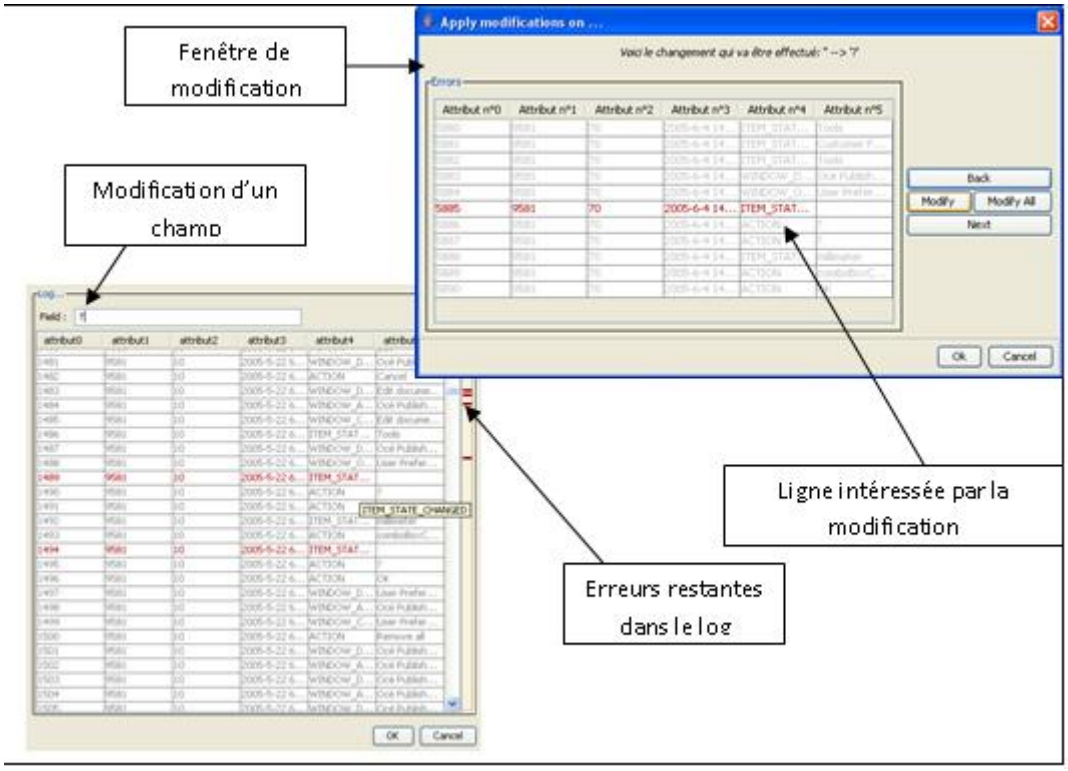


FIG. B.9: Modification d'une ligne et fenêtre de modification

Type ARFF	Type SQL
Numeric	Numeric –> Double Real –> Double Integer –> Int
String	LongText
Date	DateTime
Nominal	Enum

TAB. B.3: Correspondance des types ARFF et SQL

devenu valide au sens d'un fichier ARFF. Utiliser ce format, nous apporte plus que de pouvoir disposer des méthodes de Data Mining de WEKA. En trouvant les correspondances entre les types ARFF et ceux de SQL, nous allons pouvoir créer la table qui va recevoir les données de notre fichier.

Les seules difficultés rencontrées pour les insertions des lignes dans la table récemment crée sont dues au format Date et aux caractères sémantiquement parlants pour le langage SQL. Dans ce dernier cas, seul la protection du caractère en question est nécessaire (par exemple, le « ' » qui est la marque d'un début/fin de chaîne de caractères en SQL, sera remplacé par « \' » dans la requête). Toutes les dates reconnues dans nos fichiers de logs seront converties vers un seul format utilisé dans SQL : 'yyyy-MM-dd HH :mm :ss' Le stockage des données dans la base constitue la première partie de la sauvegarde. En effet, on va sauver les fichiers rajoutés dans la base en local dans le répertoire contenant les fichiers corrigés. Son organisation copie exactement la composition de l'arbre de sélection. Ce dernier sera aussi mis à jour en rajoutant le nœud correspondant au fichier que l'on vient de sauver.

Maintenant nous avons tous les éléments nécessaires pour réaliser nos analyses. Nous allons donc pouvoir faire des analyses de type SQL et WEKA. L'utilisation de ces deux éléments va nous permettre de mélanger les analyses « classiques » et celles issues des méthodes de Data Mining.

B.3 Analyses

Les analyses constituent la deuxième partie de la phase d'implémentation. Nous attendons de ces dernières deux types de résultats :

- analyses globales : statistiques générales sur l'interaction
- détection de séquences

B.3.1 Statistiques globales

Concernant les statistiques globales de l'interaction, sont implémentées entre autres :

- le calcul du nombre d'occurrences d'une valeur,
- le calcul du temps d'improductivité,
- le calcul du temps d'exécution par sessions.

Rappelons que le temps d'exécution d'une application peut-être divisé en :

- temps d'activité
- temps d'absence : périodes où l'application étudiée n'a plus le focus : l'utilisateur fait "autre chose" sur son poste
- temps d'inactivité : périodes où l'utilisateur reste sur l'application mais n'agit plus pendant un certain temps.

Le temps d'improductivité est la somme du temps d'absence et du temps d'inactivité. Dans le calcul du temps d'improductivité, on ne peut considérer tout temps entre deux actions comme improductif. C'est pour cela que cette notion est accompagnée d'un seuil que l'utilisateur de LogHandler doit paramétrer. Au-delà de ce seuil, le temps passé est jugé improductif et est comptabilisé. La plupart de ces résultats peuvent être obtenus grâce à une modélisation sous forme de séquences. C'est le cas pour le temps d'improductivité mais aussi d'inactivité que nous avons définis plus haut.

B.3.2 Détection des séquences

La détection de séquences dans les fichiers de logs va permettre une analyse plus spécifique de certaines tâches en apportant toujours les informations concernant le temps d'exécution et le nombre d'actions effectué pour l'accomplir. Nous allons définir ce qu'est une séquence, présenter le composant qui permet de la saisir et illustrer l'utilisation d'une séquence sur un exemple.

Définition d'une séquence

Nous ne concevons pas la détection d'une séquence comme une suite d'évènements précise à retrouver dans le fichier de logs. Une séquence est définie comme une suite d'évènements décrite par un certain nombre de contraintes. Nous avons donc dû trouver un compromis entre souplesse et expressivité.

Dans LogHandler, une séquence est initialement définit par la ligne de début et celle de fin. Une fois qu'elle est bornée, il est possible de lui affecter des contraintes de temps, de préciser quels attributs doivent être équivalents ou encore le nombre de lignes séparant ces bornes. Pour les bornes de la séquence, chaque attribut pourra

contenir une liste de valeurs possibles. On peut aussi lui affecter des contraintes de chevauchement qui permettront de préciser si l'on tolère la présence ou non d'une des bornes à l'intérieure de la séquence trouvée. Mais aussi si nous devons considérer la plus grande des séquences trouvée ou celle de taille minimale. Cependant, quelles que soient les options cochées, aucune des séquences trouvées ne contiendra une chaîne commençant par la borne de fin et se terminant par celle de début de la séquence principale.

Se limiter à cette définition des séquences est cependant réducteur. C'est pour cette raison que nous avons ajouté le fait qu'une séquence puisse comporter des sous-séquences. La présence de celles-ci pouvant être purement informative ou bien déterminer si l'on doit tenir compte ou non du motif trouvé.

Le panneau de saisie

Le panneau de saisie des séquences est la représentation concrète de la définition faite ci-dessus (Fig. B.10). Dans l'interface d'analyse, nous avons clairement fait la distinction entre la séquence principale et les sous-séquences. Ceci est dû au fait que les sous-séquences peuvent conditionner les résultats obtenus de par leur existence ou au contraire par leur absence. De plus, nous n'attendons pas de ces sous-séquences les mêmes résultats. En effet, ici nous allons retourner le nombre d'apparitions du sous-motif en question dans la chaîne principale et non pas le temps d'exécution ou encore le nombre d'évènements. Ce choix s'explique par le fait que ces sous-séquences sont étudiées pour retracer le cheminement que l'utilisateur a suivi dans le but d'accomplir sa tâche. On veut, par exemple, déterminer si un utilisateur a utilisé la fenêtre d'aide, la fonction de recherche, la sauvegarde de sa configuration avant une impression, etc.

Pour ce panneau, nous avons dû réaliser un composant permettant de sélectionner plusieurs valeurs dans une liste d'éléments. Ce composant n'a rien d'innovant puisqu'il utilise des composants de Swing, mais nous voulions garder l'aspect « linéaire » des bornes de la séquence et surtout utiliser les données que nous avons chargées pendant la phase d'initialisation. Contrairement aux autres analyses qui n'utilisaient que la liste des noms des attributs pour leur paramétrage, les séquences demandent de connaître les différentes valeurs pour un attribut donné. Cependant seuls les champs de type string et nominal pourront bénéficier de ce nouveau composant. Nous allons maintenant montrer comment saisir une séquence, et présenter la requête correspondant aux données saisies.

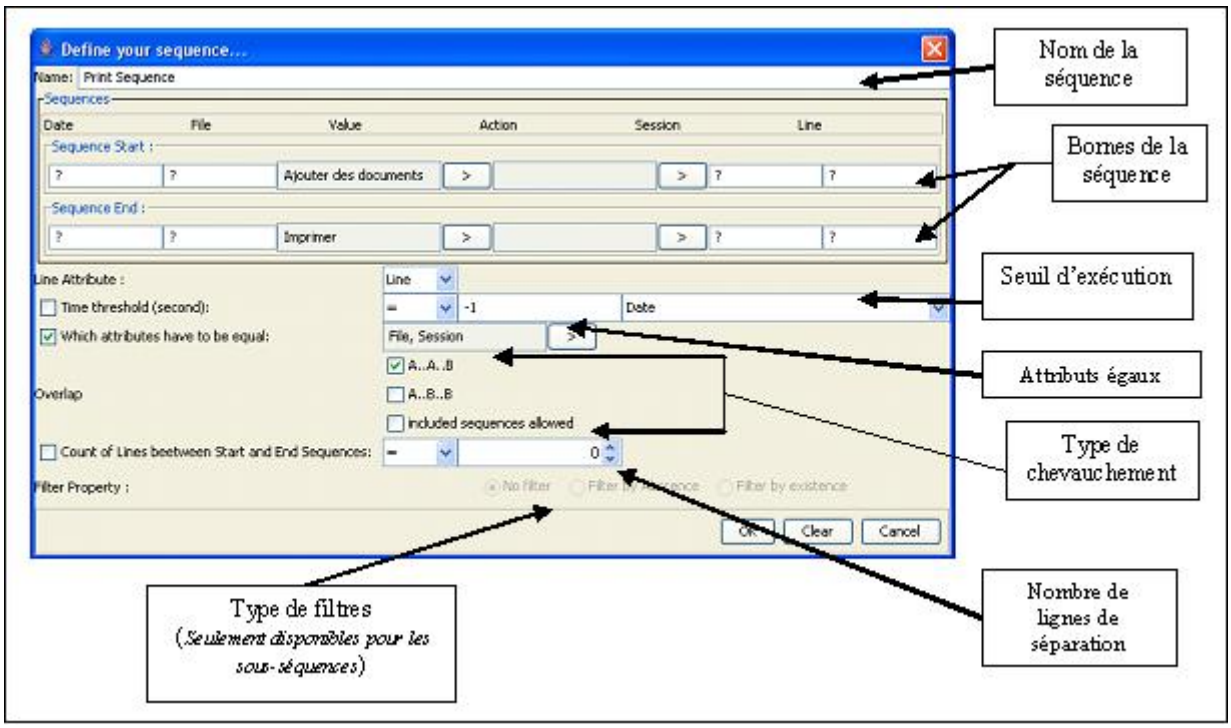


FIG. B.10: Fenêtre de paramétrage d'une séquence

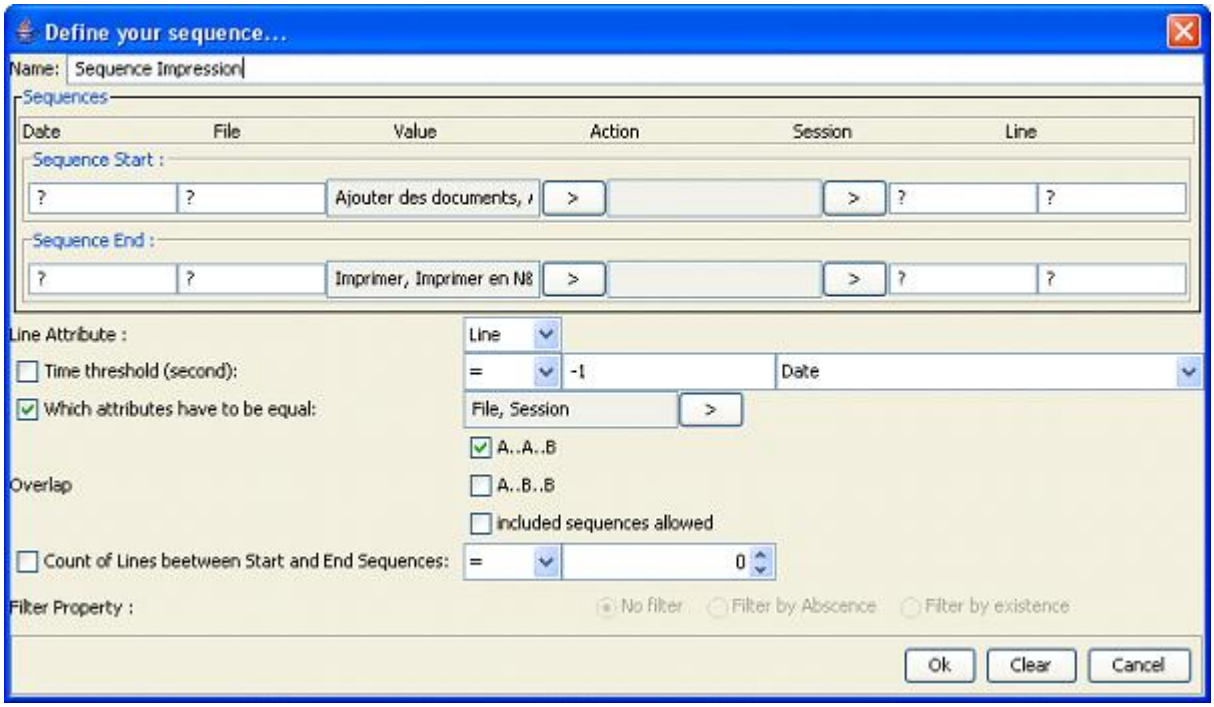


FIG. B.11: Panneau après la saisie de la séquence d'impression

Exemple de saisie

Nous illustrons notre propos sur la saisie d'une séquence représentant la réalisation d'un travail d'impression avec une consultation de l'aide en ligne B.11. Un travail d'impression commence par le chargement d'un ou plusieurs documents et finit lorsque l'utilisateur actionne le bouton « Imprimer ». Avec cette définition, nous allons, dans la première ligne, entrer toutes les valeurs qui ont un rapport avec l'ajout d'un document. Ici nous avons « Ajouter un document » et « Ajouter des documents ». Dans la seconde, nous allons entrer toutes les valeurs relatives à une Impression, soit : « Imprimer », « Imprimer en couleurs », « Imprimer en N&B ». Puis nous acceptons le fait de pouvoir avoir plusieurs ajouts de documents avant l'action « Imprimer » et de considérer la plus grande séquence possible respectant nos contraintes. Nous précisons aussi le fait que les attributs « File » et de « Session » doivent être identiques pour les deux bornes (à des fins d'optimisation du temps d'exécution de la requête).

Maintenant passons à la consultation de l'aide. Dans l'interface, cette aide est disponible dans le menu « Aide ». Malheureusement dans la version testée, la documentation d'aide n'a pas été faite et la consulter n'engendre pas l'ouverture d'une fenêtre. Mais seulement, une ligne où la valeur est « Aide » et l'évènement «

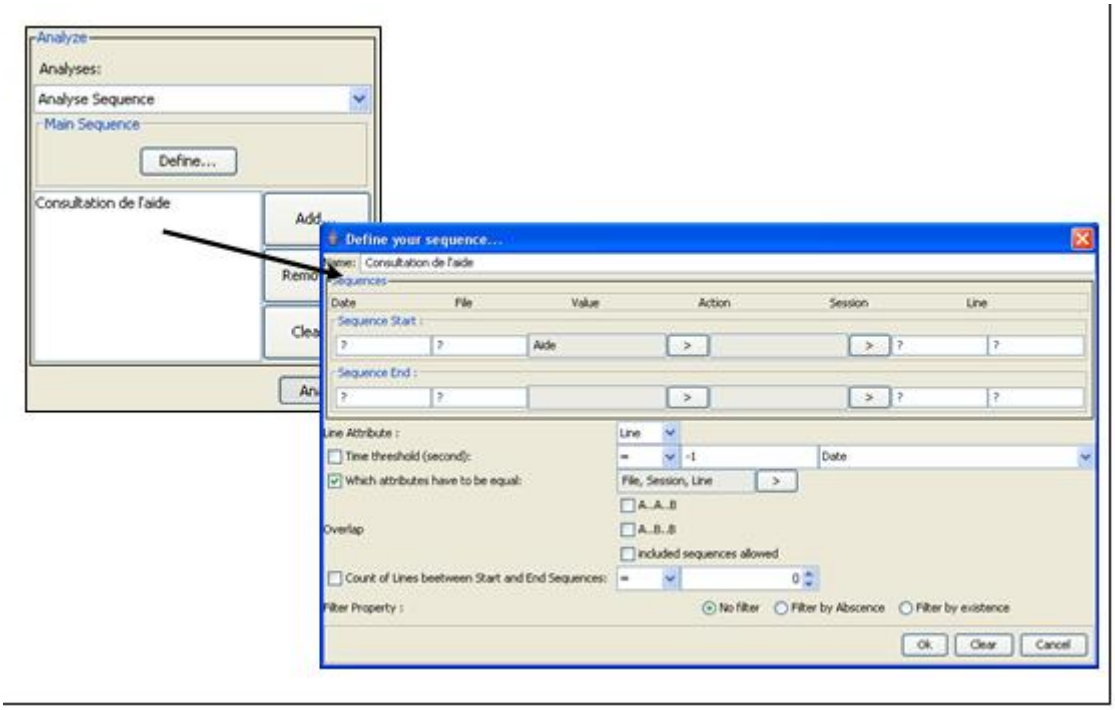


FIG. B.12: Ajout de la sous-séquence “consultation de l’aide”

ITEM_STATE_CHANGED ». Dans le panneau, définir une “séquence” ne comportant qu’une ligne implique de préciser que le champ « ligne » des deux bornes est identique. Une telle sous-séquence s’exprime donc comme sur la figure B.12.

Après avoir saisie ces paramètres dans le panneau d’édition, une requête SQL est générée, permettant de réaliser la recherche de séquences. La partie correspondant à la recherche de la « séquence d’impression » est représentée par la figure B.13.

Dans cette requête, il manque le fait de récupérer la plus grande séquence possible. En effet, cette option ne peut être directement incluse dans la requête principale ce qui nous oblige à stocker le résultat dans une table temporaire, choisir le plus grand intervalle retourné et faire la jointure avec les résultats précédents. Pour les sous-séquences, la requête traduisant le panneau d’édition est la même. Nous allons appliquer cette requête sur la table contenant toutes les lignes concernées par les séquences principales détectées. Puis nous calculons le nombre d’occurrences de chaque sous-motif dans chaque séquence. Enfin, nous ne retournons que les séquences correspondant au filtre choisi (filtrage par existence ou absence de la sous-séquence ou aucun filtrage).

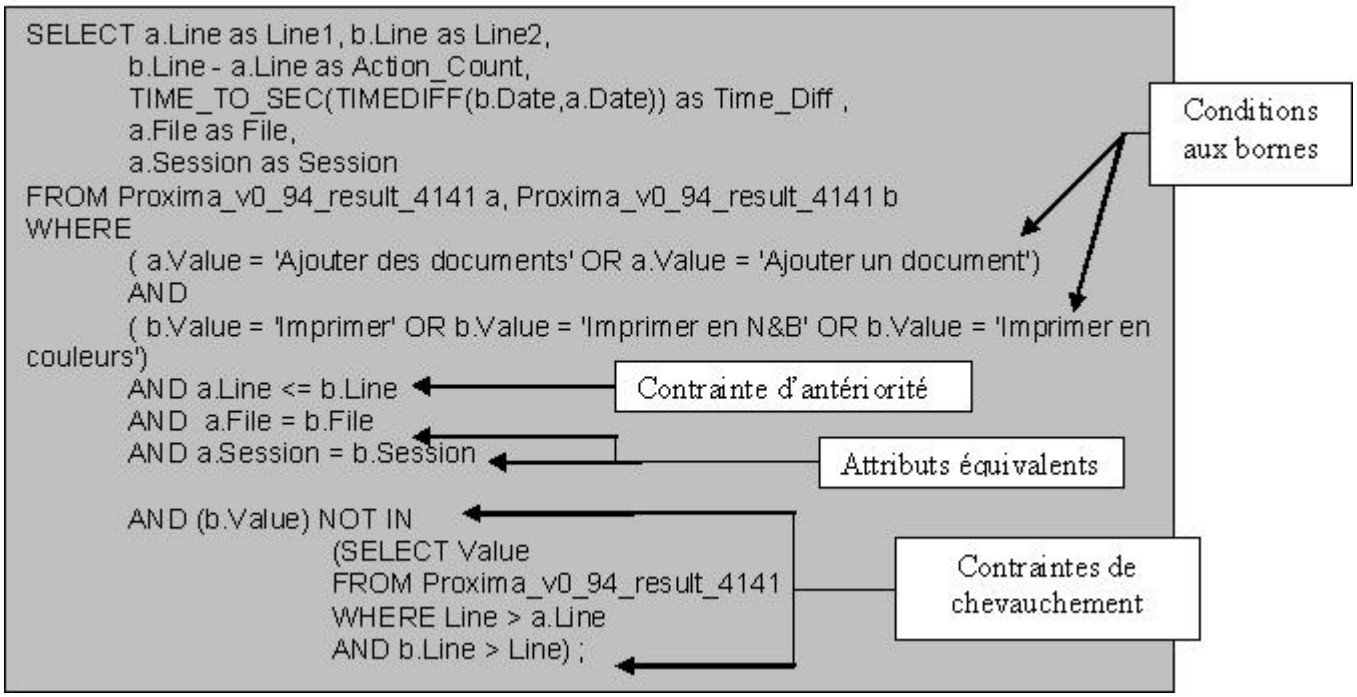


FIG. B.13: Requête correspondant à la recherche de la séquence principale

Performance

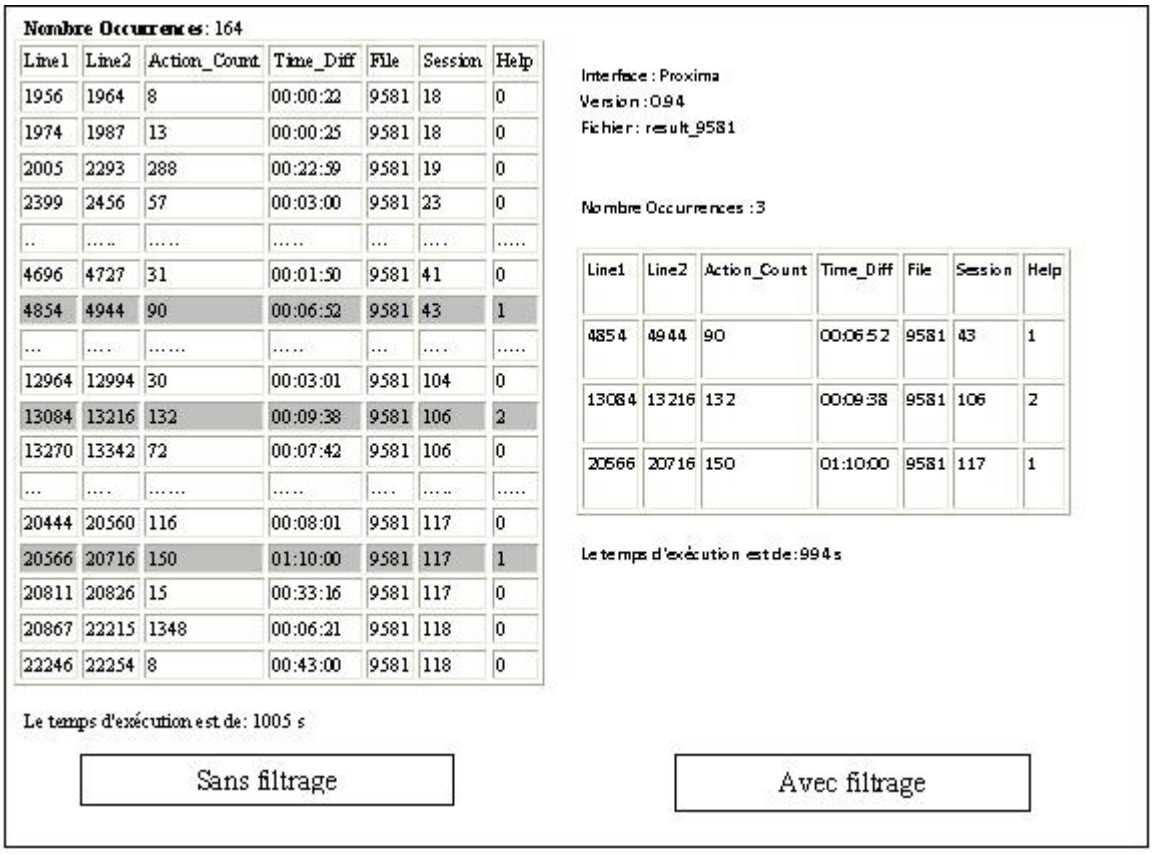
Nous avons consacré un certain temps à l'élaboration d'un algorithme permettant d'obtenir les résultats dans un temps acceptable et cela sur de grandes instances de fichiers. C'est dans ce but que nous avons utilisé la création de tables temporaires contenant les résultats d'une recherche d'un sous-motif. Ainsi à la fin de la requête, nous faisons une jointure de toutes ces tables temporaires afin de rassembler le fruit des différentes sous requêtes. Bien que le temps d'exécution des analyses ait été sensiblement amélioré, nous constatons un temps moyen de quinze minutes pour des recherches de séquences renvoyant 200 résultats avec la détection de deux sous-motifs. Cette perte de temps dans l'exécution est due aux contraintes de chevauchement qui s'expriment par des requêtes imbriquées et l'utilisation d'un « NOT IN ».

B.3.3 Affichage et enregistrement des résultats

Comme nous l'avons précisé plus haut, les résultats sont écrits en HTML. Ceci est possible grâce à un composant de Swing qui peut servir d'interpréteur rudimentaire : le « JEditorPane ». Ce choix n'est pas anodin puisqu'il demande une certaine rigueur dans l'écriture du flux résultat. En effet, contrairement à certains navigateurs, ce composant est peu laxiste sur la forme du document ; les erreurs de balise étant sanctionnées par l'affichage du code HTML lui-même. Cependant avec la possibilité d'utiliser ce langage, nous pouvons afficher les résultats facilement sous forme de tableau et les accompagner d'une mise en page simple. En plus d'afficher les résultats, l'application LogHandler permet de les sauvegarder sous forme de fichiers. A l'instar des navigateurs à onglets actuels, c'est l'onglet courant qui est enregistré. Nous ne proposons actuellement que deux formats de stockage :

- Le HTML : déjà généré, il est simple de le mettre sous forme de fichier
- un format matriciel ressemblant à la partie "données" d'un fichier ARFF.

Si enregistrer en HTML peut paraître évident, le faire dans un format matriciel l'est moins. Avec ce format, il sera possible d'utiliser un tableur pour afficher les résultats sous forme de graphiques. En effet, pour le moment, il n'est pas possible dans l'application de voir les résultats sous forme de diagramme. C'est pour combler ce manque que la traduction des tables HTML en un format matriciel a été implantée. La figure B.14 montre le type de résultat retourné pour l'exemple de séquence formulée plus haut. Nous ne montrons ici qu'une partie de l'affichage, la requête ayant plus d'une centaine de résultats. On observe aussi les différents types de filtrage par la sous-séquence d'aide.



Nombre Occurrences: 164

Line1	Line2	Action_Count	Time_Diff	File	Session	Help
1956	1964	8	00:00:22	9581	18	0
1974	1987	13	00:00:25	9581	18	0
2005	2293	288	00:22:59	9581	19	0
2399	2456	57	00:03:00	9581	23	0
..	....	.....	.....	...	....	.....
4696	4727	31	00:01:50	9581	41	0
4854	4944	90	00:06:52	9581	43	1
...	....	.....	.....	...	....	.....
12964	12994	30	00:03:01	9581	104	0
13084	13216	132	00:09:38	9581	106	2
13270	13342	72	00:07:42	9581	106	0
...	....	.....	.....	....	....	.....
20444	20560	116	00:08:01	9581	117	0
20566	20716	150	01:10:00	9581	117	1
20811	20826	15	00:33:16	9581	117	0
20867	22215	1348	00:06:21	9581	118	0
22246	22254	8	00:43:00	9581	118	0

Le temps d'exécution est de: 1005 s

Sans filtrage

Interface : Proxima
Version : 0.9.4
Fichier : result_9581

Nombre Occurrences : 3

Line1	Line2	Action_Count	Time_Diff	File	Session	Help
4854	4944	90	00:06:52	9581	43	1
13084	13216	132	00:09:38	9581	106	2
20566	20716	150	01:10:00	9581	117	1

Le temps d'exécution est de: 994 s

Avec filtrage

FIG. B.14: Résultats de l'analyse de la séquence exemple

Annexe C

Exemple de configuration d'impression

C.1 Un exemple concret

Nous donnons ici un exemple numérique concret, afin de fixer les idées.
Les dimensions sont données en mm.

C.1.1 Les données du problème

Nous reprenons ici les paramètres présentés dans la section 3.1, en y ajoutant la politique d'impression utilisée pour cet exemple.

Contexte

- Formats standards :

$$FS = \{(A0, 1189, 841), (A1, 841, 594), (A2, 594, 420), (A3, 420, 297), (A4, 297, 210)\}$$

- Seuil de tolérance : seuil standard qui borne les marges matérielles $seuil = 5$.
- Document à imprimer : image en orientation paysage avec demande de pliage en tenant compte du cartouche situé en bas à droite.

$$d = (510, 740, 3)$$

Directives de l'imprimeur

- Zoom désiré : 100%

$$z^* = 1$$

– Marges supplémentaires : l'imprimeur souhaite ajouter une estampille en haut à gauche de l'image, ce qui implique une marge supplémentaire particulière au-dessus de l'image. De plus les marges matérielles sont prises en compte.

$$G^{sup} = (50, auto, auto, auto)$$

– Marges d'extrémité : pas de marges d'extrémité puisque le document doit être plié.

$$G^{ext} = (0, 0)$$

Ressources du système d'impression

L'imprimeur dispose de deux imprimantes, pourvues d'un rouleau chacune, de formats A3 et A2. La première imprimante ne permet pas le pliage automatique alors que la seconde permet de plier des deux coins. Il y a donc deux sources de média s_1 et s_2 telles que :

$$\begin{aligned} s_1 &= (297, 210, 15000, \{1\}, (3, 3, 3, 3)) \\ s_2 &= (420, 210, 15000, \{1, 2, 3\}, (3, 5, 3, 5)) \end{aligned}$$

on a

$$\hat{S} = \{s_1, s_2\}$$

Politique d'impression

Nous décrivons ici la politique d'impression de l'imprimeur qui va servir à configurer le document.

– choix d'un média : <Plus_grand_sinon_plus_petit>
– ajustement de l'image : <Réduire_pour_ajuster>
– gestion d'erreur : ne s'applique pas ici car l'imprimeur n'a pas sélectionné un média particulier
– ajustement de la découpe : <Synchrone>

C.1.2 Problèmes d'optimisation sous contraintes

Fonction objectif

$$minimiser \qquad f_{obj} = \alpha\left(g_{\Uparrow} + g_{\Leftarrow} + g_{\Downarrow} + g_{\Rightarrow}\right) + \beta\left(c_{\Uparrow} + c_{\Leftarrow} + c_{\Downarrow} + c_{\Rightarrow}\right) + \gamma\left\|z^* - z\right\|$$

Conditions de politique d'impression (première phase)

$$
\begin{aligned}
r = 0 \; ou \; r = 180 \implies & \begin{cases} z^* \cdot i_h < m_h \\ z^* \cdot i_l < m_l \end{cases} \\
r = 90 \; ou \; r = 270 \implies & \begin{cases} z^* \cdot i_h < m_l \\ z^* \cdot i_l < m_h \end{cases}
\end{aligned}
$$

$$
\begin{aligned}
z &\leq z^* \\
c_{\Uparrow} &\leq seuil \\
c_{\Leftarrow} &\leq seuil \\
c_{\Downarrow} &\leq seuil \\
c_{\Rightarrow} &\leq seuil
\end{aligned}
$$

S'il n'y a pas de solution, les conditions de politique d'impression deviennent les suivantes.

Conditions de politique d'impression (éventuelle seconde phase)

$$
\begin{aligned}
z &\leq z^* \\
c_{\Uparrow} &\leq seuil \\
c_{\Leftarrow} &\leq seuil \\
c_{\Downarrow} &\leq seuil \\
c_{\Rightarrow} &\leq seuil
\end{aligned}
$$

Dans les deux phases, on a les mêmes conditions d'admissibilité.

Conditions d'admissibilité

Deux sources de média :

$$s = s_1 \quad ou \quad s = s_2$$

Contraintes relatives à la source choisie :

$$\begin{cases} s_{H_{min}} & \leq & m_h & \leq & s_{H_{max}} \\ & & m_l & = & s_L \\ & & m_f & \in & s_F \end{cases}$$

Contraintes relatives aux marges supplémentaires et d'extrémité :

$$r = 180 \implies \begin{cases} g_{\Uparrow}^{min} & = & g_{\Uparrow}^{sup} & + & g_{\Downarrow}^{ext} \\ g_{\Leftarrow}^{min} & = & s_{g_{\Rightarrow}^{mat}} & & \\ g_{\Downarrow}^{min} & = & s_{g_{\Uparrow}^{mat}} & + & g_{\Uparrow}^{ext} \\ g_{\Rightarrow}^{min} & = & s_{g_{\Leftarrow}^{mat}} & & \end{cases}$$

$$r = 270 \implies \begin{cases} g_{\Uparrow}^{min} & = & g_{\Uparrow}^{sup} & & \\ g_{\Leftarrow}^{min} & = & s_{g_{\Uparrow}^{mat}} & + & g_{\Uparrow}^{ext} \\ g_{\Downarrow}^{min} & = & s_{g_{\Leftarrow}^{mat}} & & \\ g_{\Rightarrow}^{min} & = & s_{g_{\Downarrow}^{mat}} & + & g_{\Downarrow}^{ext} \end{cases}$$

Possibilités de rotation et implication sur la finition :

$$\begin{cases} r = 180 & ou & r = 270 \\ r = 180 & \implies & m_f = 3 \\ r = 270 & \implies & m_f = 2 \end{cases}$$

Contraintes sur la position de l'image :

$$\begin{cases} g_{\Rightarrow} & = & g_{\Rightarrow}^{min} \\ g_{\Downarrow} & = & g_{\Downarrow}^{min} \\ c_{\Rightarrow} & = & 0 \\ c_{\Downarrow} & = & 0 \end{cases}$$

Respect des marges minimales :

$$\begin{cases} g_{\Uparrow} & \geq & g_{\Uparrow}^{min} \\ g_{\Leftarrow} & \geq & g_{\Leftarrow}^{min} \\ g_{\Downarrow} & \geq & g_{\Downarrow}^{min} \\ g_{\Rightarrow} & \geq & g_{\Rightarrow}^{min} \end{cases}$$

Contraintes générales du système

$$\begin{cases} c_{\Uparrow} + c_{\Downarrow} & < & z \cdot i_h \\ c_{\Leftarrow} + c_{\Rightarrow} & < & z \cdot i_l \end{cases}$$

$$r = 180 \Longrightarrow \begin{cases} z \cdot d_l - m_l + g_{\Leftarrow} + g_{\Rightarrow} - c_{\Leftarrow} - c_{\Rightarrow} & = & 0 \\ z \cdot d_h - m_h + g_{\Uparrow} + g_{\Downarrow} - c_{\Uparrow} - c_{\Downarrow} & = & 0 \\ g_{\Uparrow} + g_{\Downarrow} & < & m_h \\ g_{\Leftarrow} + g_{\Rightarrow} & < & m_l \end{cases}$$

$$r = 270 \Longrightarrow \begin{cases} z \cdot d_l - m_h + g_{\Leftarrow} + g_{\Rightarrow} - c_{\Leftarrow} - c_{\Rightarrow} & = & 0 \\ z \cdot d_h - m_l + g_{\Uparrow} + g_{\Downarrow} - c_{\Uparrow} - c_{\Downarrow} & = & 0 \\ g_{\Uparrow} + g_{\Downarrow} & < & m_l \\ g_{\Leftarrow} + g_{\Rightarrow} & < & m_h \end{cases}$$

C.1.3 Solution

Il est aisé de voir que la première phase n'admet pas de solution car aucune source de média ne permet d'imprimer une feuille plus grande que l'image d'origine. En revanche, le second système admet la solution optimale $(m, r, z, g, c) \in E$ suivante :

média	$m = (420, 535, 2)$
rotation	$r = 270$
zoom	$z = 71,5\%$
marges	$g = (50, 3, 5, 3)$
coupes	$c = (0, 0, 0, 0)$

C.2 Traduction linéaire de l'exemple

Dans cette section, nous reprenons l'exemple présenté dans la section C.1 et nous montrons comment l'exprimer comme une suite de problèmes linéaires mixtes.

D'après la politique de sélection du média <Plus_grand_sinon_plus_petit>, nous devons considérer la possibilité d'une seconde passe, au cas où la première n'aurait pas donné de solution. De plus, pour chaque passe, nous devons considérer les angles de rotation de 180° et 270°. Notre problème se traduit donc en quatre programmes linéaires mixtes. Certaines parties étant identiques, pour plus de clarté nous découpons les contraintes en deux groupes : celles qui sont constantes et celles qui changent d'une phase à l'autre.

C.2.1 Fonction objectif

$$min \qquad f_{obj} = \alpha \left(g_{\Uparrow} + g_{\Leftarrow} + g_{\Downarrow} + g_{\Rightarrow}\right) + \beta \left(c_{\Uparrow} + c_{\Leftarrow} + c_{\Downarrow} + c_{\Rightarrow}\right) + \gamma \cdot \triangle z$$

où $\triangle z$ est la fonction linéaire par morceaux représentant la valeur absolue de l'écart entre le zoom effectif et le zoom désiré.

C.2.2 Les contraintes constantes

$$
\begin{aligned}
1 \cdot z_1 + 0 \cdot z_2 + 99 \cdot z_3 &= \triangle z \\
c_1, c_2 &\in \{0, 1\} \\
c_1 + c_2 &= 1 \\
z_1 &\geq 0 \\
z_2 &\geq 0 \\
z_3 &\geq 0 \\
z_1 &\leq 1 \cdot c_1 + 0 \cdot c_2 \\
z_2 &\leq 1 \cdot c_1 + 1 \cdot c_2 \\
z_3 &\leq 0 \cdot c_1 + 1 \cdot c_2 \\
0,01 \cdot z_1 + z^* \cdot z_2 + 100 \cdot z_3 &= z
\end{aligned}
$$

$$
\begin{aligned}
s_1, s_2 &\in \{0, 1\} \\
s_L &= 297 \cdot s_1 + 420 \cdot s_2 \\
s_{H_{min}} &= 210 \cdot s_1 + 210 \cdot s_2 \\
s_{H_{max}} &= 15000 \cdot s_1 + 15000 \cdot s_2 \\
s_{g^{mat}_{\Uparrow}} &= 3 \cdot s_1 + 3 \cdot s_2 \\
s_{g^{mat}_{\Leftarrow}} &= 3 \cdot s_1 + 5 \cdot s_2 \\
s_{g^{mat}_{\Downarrow}} &= 3 \cdot s_1 + 3 \cdot s_2 \\
s_{g^{mat}_{\Rightarrow}} &= 3 \cdot s_1 + 5 \cdot s_2 \\
s_{F_1} &= 1 \cdot s_1 + 1 \cdot s_2 \\
s_{F_2} &= 0 \cdot s_1 + 1 \cdot s_2 \\
s_{F_3} &= 0 \cdot s_1 + 1 \cdot s_2
\end{aligned}
$$

$$
\begin{cases}
s_{H_{min}} \leq m_h \leq s_{H_{max}} \\
m_l = s_L \\
m_f \in s_F
\end{cases}
$$

$$\begin{aligned} g_{\Rightarrow} &= g_{\Rightarrow}^{min} \\ g_{\Downarrow} &= g_{\Downarrow}^{min} \\ c_{\Rightarrow} &= 0 \\ c_{\Downarrow} &= 0 \end{aligned}$$

$$\begin{cases} r = 270 \implies m_f = 2 \\ g_{\Rightarrow} = g_{\Rightarrow}^{min} \\ g_{\Downarrow} = g_{\Downarrow}^{min} \\ c_{\Rightarrow} = 0 \\ c_{\Downarrow} = 0 \end{cases}$$

$$\begin{cases} c_{\Uparrow} + c_{\Downarrow} < z \cdot i_h \\ c_{\Leftarrow} + c_{\Rightarrow} < z \cdot i_l \end{cases}$$

$$\begin{cases} g_{\Uparrow} \geq g_{\Uparrow}^{min} \\ g_{\Leftarrow} \geq g_{\Leftarrow}^{min} \\ g_{\Downarrow} \geq g_{\Downarrow}^{min} \\ g_{\Rightarrow} \geq g_{\Rightarrow}^{min} \end{cases}$$

C.2.3 Les contraintes de la première phase

Sous un angle de 180°

$$\begin{aligned} z^* \cdot i_h &< m_h \\ z^* \cdot i_l &< m_l \end{aligned}$$

$$\begin{aligned} z &\leq z^* \\ c_{\Uparrow} &\leq seuil \\ c_{\Leftarrow} &\leq seuil \\ c_{\Downarrow} &\leq seuil \\ c_{\Rightarrow} &\leq seuil \end{aligned}$$

$$\begin{array}{rcl} g^{min}_{\Uparrow} & = & g^{sup}_{\Uparrow} + g^{ext}_{\Downarrow} \\ g^{min}_{\Leftarrow} & = & s_{g^{mat}_{\Rightarrow}} \\ g^{min}_{\Downarrow} & = & s_{g^{mat}_{\Uparrow}} + g^{ext}_{\Uparrow} \\ g^{min}_{\Rightarrow} & = & s_{g^{mat}_{\Leftarrow}} \end{array}$$

$$\begin{array}{rcl} z \cdot d_l - m_l + g_{\Leftarrow} + g_{\Rightarrow} - c_{\Leftarrow} - c_{\Rightarrow} & = & 0 \\ z \cdot d_h - m_h + g_{\Uparrow} + g_{\Downarrow} - c_{\Uparrow} - c_{\Downarrow} & = & 0 \\ g_{\Uparrow} + g_{\Downarrow} & < & m_h \\ g_{\Leftarrow} + g_{\Rightarrow} & < & m_l \\ m_f & = & 3 \end{array}$$

Sous un angle de 270°

$$\begin{array}{rcl} z^* \cdot i_h & < & m_l \\ z^* \cdot i_l & < & m_h \end{array}$$

$$\begin{array}{rcl} z & \leq & z^* \\ c_{\Uparrow} & \leq & seuil \\ c_{\Leftarrow} & \leq & seuil \\ c_{\Downarrow} & \leq & seuil \\ c_{\Rightarrow} & \leq & seuil \end{array}$$

$$\begin{array}{rcl} g^{min}_{\Uparrow} & = & g^{sup}_{\Uparrow} \\ g^{min}_{\Leftarrow} & = & s_{g^{mat}_{\Uparrow}} + g^{ext}_{\Uparrow} \\ g^{min}_{\Downarrow} & = & s_{g^{mat}_{\Leftarrow}} \\ g^{min}_{\Rightarrow} & = & s_{g^{mat}_{\Downarrow}} + g^{ext}_{\Downarrow} \end{array}$$

$$\begin{array}{rcl} z \cdot d_l - m_h + g_{\Leftarrow} + g_{\Rightarrow} - c_{\Leftarrow} - c_{\Rightarrow} & = & 0 \\ z \cdot d_h - m_l + g_{\Uparrow} + g_{\Downarrow} - c_{\Uparrow} - c_{\Downarrow} & = & 0 \\ g_{\Uparrow} + g_{\Downarrow} & < & m_l \\ g_{\Leftarrow} + g_{\Rightarrow} & < & m_h \\ m_f & = & 2 \end{array}$$

C.2.4 Les contraintes de la seconde phase

Sous un angle de 180°

$$\begin{aligned} z &\leq z^* \\ c_{\Uparrow} &\leq seuil \\ c_{\Leftarrow} &\leq seuil \\ c_{\Downarrow} &\leq seuil \\ c_{\Rightarrow} &\leq seuil \end{aligned}$$

$$\begin{aligned} g_{\Uparrow}^{min} &= g_{\Uparrow}^{sup} + g_{\Downarrow}^{ext} \\ g_{\Leftarrow}^{min} &= s_{g_{\Rightarrow}^{mat}} \\ g_{\Downarrow}^{min} &= s_{g_{\Uparrow}^{mat}} + g_{\Uparrow}^{ext} \\ g_{\Rightarrow}^{min} &= s_{g_{\Leftarrow}^{mat}} \end{aligned}$$

$$\begin{aligned} z \cdot d_l - m_l + g_{\Leftarrow} + g_{\Rightarrow} - c_{\Leftarrow} - c_{\Rightarrow} &= 0 \\ z \cdot d_h - m_h + g_{\Uparrow} + g_{\Downarrow} - c_{\Uparrow} - c_{\Downarrow} &= 0 \\ g_{\Uparrow} + g_{\Downarrow} &< m_h \\ g_{\Leftarrow} + g_{\Rightarrow} &< m_l \\ m_f &= 3 \end{aligned}$$

Sous un angle de 270°

$$\begin{aligned} z &\leq z^* \\ c_{\Uparrow} &\leq seuil \\ c_{\Leftarrow} &\leq seuil \\ c_{\Downarrow} &\leq seuil \\ c_{\Rightarrow} &\leq seuil \end{aligned}$$

$$\begin{aligned} g_{\Uparrow}^{min} &= g_{\Uparrow}^{sup} \\ g_{\Leftarrow}^{min} &= s_{g_{\Uparrow}^{mat}} + g_{\Uparrow}^{ext} \\ g_{\Downarrow}^{min} &= s_{g_{\Leftarrow}^{mat}} \\ g_{\Rightarrow}^{min} &= s_{g_{\Downarrow}^{mat}} + g_{\Downarrow}^{ext} \end{aligned}$$

$$
\begin{array}{rcl}
z \cdot d_l - m_h + g_{\Leftarrow} + g_{\Rightarrow} - c_{\Leftarrow} - c_{\Rightarrow} & = & 0 \\
z \cdot d_h - m_l + g_{\Uparrow} + g_{\Downarrow} - c_{\Uparrow} - c_{\Downarrow} & = & 0 \\
g_{\Uparrow} + g_{\Downarrow} & < & m_l \\
g_{\Leftarrow} + g_{\Rightarrow} & < & m_h \\
m_f & = & 2
\end{array}
$$