Cahiers GUTenberg

FORMATER DES DOCUMENTS AYANT DES FLOTTANTS : UN NOUVEL ALGORITHME POUR LaTeX 2ε

Frank MITTELBACH

Cahiers GUTenberg, nº 37-38 (2000), p. 86-108.

<http://cahiers.gutenberg.eu.org/fitem?id=CG_2000___37-38_86_0>

Formater des documents ayant des flottants : un nouvel algorithme pour LaTeX 2ε* †

Frank Mittelbach ‡

frank.mittelbach@latex-project.org

Résumé. Cet article décrit une approche pour le positionnement des éléments flottants dans un document à plusieurs colonnes.

La version actuelle de LaTeX a été écrite à l'origine pour des documents en une seule colonne puis étendue pour accepter les documents en deux colonnes, essentiellement en traitant chaque colonne indépendamment. Il en résulte de sévères limitations dans le système actuel, comme par exemple que les flottants qui s'étendent sur la largeur des deux colonnes sont au minimum retardé à la page suivante, ou que l'ordre de numérotation des flottants peut, dans ce cas là, ne pas être respecté.

Le nouvel algorithme a pour finalité de supprimer ces limitations et en même temps étendre les mises en page de ces classes de documents à du multi-colonnage avec des flottants pouvant s'étendre sur un nombre quelconque de colonnes.

Abstract. *This paper describes an approach to placement of floats in multicolumn documents. The current version of LaTeX was originally written for single-column documents and extended to support two-column documents by essentially building each column independently from the other. As a result the current system shows severe limitations in two column mode, such as the fact that spanning floats are always deferred to at least the next page or that numbering between column floats and spanning floats can get out of sequence.*

The new algorithm is intended to overcome these limitations and at the same time extend the supported class of document layouts to multiple columns with floats spanning an arbitrary number of columns.

† Cet article a été présenté lors du congrès GUTenberg à Toulouse en mai 2000, puis au congrès TUG à Oxford en août 2000. À paraître sous le titre « Formatting documents with floats — A new algorithm for LaTeX 2ε* », *TUGBoat*, vol. 21, 2000. Il est reproduit ici avec l'aimable autorisation de Frank Mittelbach.

‡ Traduit de l'anglais par Benjamin Bayart (bayartb@edgard.fdn.fr).

1. Introduction

Un des problèmes du formatage de documents contenant des flottants est le nombre potentiel de solutions à envisager pour la mise en forme. Le nombre de cas à envisager évolue comme un binôme de Newton en fonction du nombre de flottants et du nombre d'emplacements possibles pour les accueillir. Si n est le nombre de flottants en attente de placement et m le nombre de zones de placement sur la page en cours (sans compter la zone « à traiter plus tard »), alors le nombre de placements à examiner est donné par

$$\text{nb\_essais} = \binom{n+m}{m} = \frac{(n+m)!}{n!\,m!} \tag{1}$$

et ce en supposant que l'ordre des flottants doive être respecté, c'est-à-dire que si l'appel du flottant f_i est placé avant celui de f_j dans le source du document, alors le flottant f_i doit être placé « avant » f_j dans le document résultant, où « avant » est une relation définie entre les zones de placement possibles.

Par exemple, si 8 flottants sont en attente de placement dans 12 zones de placement possibles (ce qui correspond à un document en trois colonnes avec des zones de placement en haut et en bas de page permettant des flottants de toutes les largeurs possibles), alors on doit examiner 125 970 dispositions. Si deux flottants de plus doivent être traités, alors il faut examiner 646 646 possibilités.

Bien qu'un grand nombre de ces combinaisons ne soient pas acceptables et puissent être écartées très rapidement, après un test initial simple, le temps de calcul de l'algorithme resterait cependant trop élevé pour être acceptable. En supposant que l'on puisse examiner 1 000 combinaisons par seconde, ce qui est ridiculement élevé puisque nombre d'entre elles demanderont un essai de composition complet de la page, alors le cas des 646 646 combinaisons amènerait à près de 10 minutes de calcul avant de prendre une décision.

Il est donc important de trouver un algorithme qui ait une complexité au pire linéaire par rapport au nombre de flottants et au nombre de zones, même si cela signifie que dans certains cas un résultat intéressant aura pu échapper à l'examen. Il serait de plus intéressant que la redondance soit minimale.

Il convient de retenir que le temps réel de calcul pour du code TEX n'est pas simple à évaluer en ce sens que certains traitements sont bien plus rapides que d'autres. Par exemple, un test qui utilise un nombre raisonnable d'appels de macros et d'affectations de registres peut s'avérer bien plus lent que de composer une longue liste et réaliser ensuite un test simple.

L'algorithme que nous avons implémenté remplit la condition d'être (pratiquement) une fonction linéaire selon le nombre de flottants et le nombre de zones de placement.

2. Modèle du document source

Le document source est un simple flux de texte continu contenant des appels d'objets flottant. Pour l'instant, les appels sont symbolisés en placant intégralement les objets dans le flux mais il est envisageable de les fournir sous forme d'objets séparés. Les objets flottants se présentent (du moins actuellement) sous trois formes :

- Des objets où l'appel et le placement requièrent une relation stricte dans l'espace, par exemple sur la même ligne dans la marge. Un exemple peut être les notes en marge que LaTeX 2ε réalise avec `\marginpar`.
- Des objets où l'appel et le placement doivent être sur la même colonne, page ou double-page, par exemple les notes de bas de page.
- Des objets où il y a une relation définie entre le lieu de l'appel et le lieu de placement, par exemple « pas dans une colonne avant l'appel » ou « sur la même page ou après », etc. Ce sont les flottants traditionnels.

Les flottants du dernier groupe sont typés ; leur type est défini par le contenu logique de l'objet, par exemple « figure » ou « table », et ainsi de suite.

La mise en forme du document est effectuée en utilisant une quantité minimale, mais paramètrable, de lecture en avant (typiquement le texte composé dans la galée est équivalent à une page/double-page de texte pur en ne tenant pas compte de la place qui sera prise par les objets flottants présents dans le flux).

Pendant la composition des pages, le principal critère de « qualité » évalué par l'algorithme est de réussir à placer chaque flottant le plus tôt possible sans être en contradiction avec certaines contraintes définies.

3. Modèle de mise en page du document

3.1. Colonnes

La mise en page admise par le nouvel algorithme admet un nombre arbitraire de colonnes de texte de largeurs égales.

Le nombre de colonnes par page ainsi que leur largeur peuvent être changés aux changements de page imposés, comme les changements de chapitre par exemple.

3.2. Colonnes équilibrées

L'équilibrage des colonnes (comme on l'obtenait avec le package multicol) est prévu mais non implémenté. Le problème majeur réside dans la gestion des flottants présents dans les colonnes pendant le processus d'équilibrage.

3.3. Zones de placement des flottants

Les objets flottants sont disposés dans des zones de placement qui sont de forme rectangulaire. Chaque zone de placement recouvre une ou plusieurs colonnes, leur largeur est donc donnée par la formule suivante, où c est le nombre de colonnes recouvertes :

$$\langle \textit{larg-zone} \rangle = c \times \big(\langle \textit{larg-col} \rangle + \langle \textit{sep-col} \rangle \big) - \langle \textit{sep-col} \rangle$$

La convention de désignation des zones de placement est la suivante :

⟨*identifiant*⟩⟨*col-départ*⟩⟨*nb-cols*⟩.

L'⟨*identifiant*⟩ est une lettre unique dénotant le type de zone, par exemple t pour le haut, b pour le bas. Le ⟨*nb-cols*⟩ est un chiffre indiquant le nombre de colonnes couvertes par la zone. La ⟨*col-départ*⟩ est un chiffre [1] indiquant la colonne de départ de la zone. Ainsi t23 est une zone qui est en haut de page, qui commence à la colonne deux et recouvre trois colonnes, à savoir les colonnes deux, trois et quatre. Une restriction liée à la convention de désignation est qu'il ne peut pas y avoir plus de 9 colonnes [2].

Seul un sous-ensemble des zones de placement peut être rempli sur une page. De par sa nature, le nouvel algorithme ne permet pas un placement qui supposerait d'interrompre une colonne de texte à cause d'un flottant (sauf bien entendu les flottants avec l'option « *here* » [3]) [4]. Cela impose que le remplissage de certaines zones doit être empêché, plus précisément celles remplissant ces

1. Avec un peu d'attention dans le code on peut étendre à plus d'un chiffre.
2. La convention est différente de celle envisagée dans les premiers temps où t23 désignait une zone commençant à la colonne deux et s'étendant jusqu'à la colonne trois.
3. NdT : « ici », en français, cette option positionne le flottant *dans* la colonne, sans le faire flotter.
4. Un jour cette restriction sera peut-être supprimée.

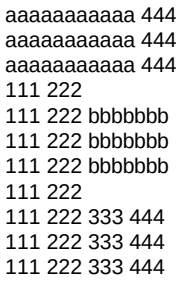

FIGURE 1 – *Zones de placement avec recouvrement*

conditions, sachant que pcl (où p = pos, c = colonne, l = largeur) vient d'accueillir un flottant :

$$pij \text{ où } i < c \leq i + j < c + l$$

ou

$$pij \text{ où } i \leq c + l < i + j \leq \langle \textit{nombre-de-colonnes} \rangle$$

La première formule décrit les zones qui recouvrent partiellement pcl sur la gauche, et la seconde celles sur la droite. Les zones qui sont des sous- ou surzones, par exemple t13 et t22, ne s'influencent pas l'une l'autre. La restriction ci-dessus est nécessaire pour éviter une situation comme celle présentée à la figure 1, c'est-à-dire quand la zone t32 (figurée par des b) imposerait le découpage de la quatrième colonne en deux zones de texte indépendantes.

Les possibilités, de même que les restrictions, sont identiques que l'on s'intéresse aux zones de haut ou de bas de page. Cela implique que, dans le nouveau schéma, on peut avoir des zones de bas de page recouvrant plusieurs colonnes [5].

3.4. Pages et colonnes de flottants

Les pages de flottants, c'est-à-dire les pages ne contenant que des flottants, seront possibles, de même que les colonnes de flottants.

3.5. Types de flottants

Le type de flottants à une influence sur la mise en page, par exemple pour le positionnement du sous-titre par rapport au corps de l'objet flottant, comment cette légende est mise en forme, quels textes fixes sont ajoutés, etc. Cela

5. NdT: Chose impossible avec l'algorithme utilisé par LaTeX 2ε.

restreint également l'algorithme de placement aux zones de placement qui peuvent être exploitées comme expliqué plus loin.

3.6. Marges

Les zones en marge peuvent recevoir les notes marginales, qui sont alignées avec la ligne de texte correspondante. Dans les documents de plus de deux colonnes, les notes en marge ne sont actuellement pas autorisées, bien que l'on puisse concevoir d'en utiliser même dans ce cas. Si la place venait à manquer pour le placement de deux flottants en marge, le plus tardif serait décalé vers le bas s'il y a assez de place dans la page, sinon la ligne contenant l'appel serait déplacée sur la colonne/page suivante[6].

Les marges peuvent aussi servir à y placer les notes de bas de page. Un prototype de cet usage est déjà disponible, voir à la section 3.7.

Un autre usage possible des marges est de les exploiter (au moins partiellement) comme zone de placement à part entière. Le problème de cet usage est que ces zones de placement auront une largeur qui ne sera pas liée à la largeur des colonnes, imposant donc qu'on y place une catégorie particulière de flottants.

Une autre extension intéressante serait d'autoriser les zones de placement qui sont bordées par une marge à utiliser la marge, autorisant ainsi le placement de flottants qui sont plus larges que la largeur nominale de la zone. Un cas particulier de cet usage, à savoir le placement de la légende dans la marge à coté du corps du flottant, est d'ores et déjà fourni en utilisant le mode de formatage de légende qui convient.

3.7. Notes de bas de page

Les notes de bas de page peuvent être considérées comme un type spécial de flottants. Ce sont des objets qui sont associés à des lignes de texte (par leur point d'appel) mais contrairement aux flottants classiques tels que « figure » et « table » leur placement est nettement plus contraint, par exemple, elles doivent apparaître en bas de la colonne où se trouve l'appel, ou au moins sur la même page que l'appel.

Dans la version actuelle, le modèle gère soit les notes en bas de la colonne d'appel (comportement habituel) ; toutes les notes en bas de la dernière co-

6. Ceci n'est pas encore implémenté — dans la version actuelle les deux flottants se chevauchent !

lonne (comme avec le package tfnright pour le mode deux colonnes) ; ou toutes les notes dans la marge intérieure ou extérieure.

Sans extension de l'algorithme de mise en forme (mais en redéfinissant convenablement les commandes de gestion des notes) elles peuvent être traitées comme des notes en marge ou comme des notes de fin d'ouvrage.

3.8. En-têtes et pieds de page

Les zones de tête et de pied de page peuvent être utilisées pour recevoir des données venant de chaque colonne. Une version étendue du mécanisme de marques de TEX permet la définition d'un nombre arbitraire de classes de marques. Dans chaque classe de marque, des informations à propos de la marque de haut de colonne (c.-à-d. la marque active au début de la colonne), de la premièrei et de la dernière marque sont mises à disposition.

Cela permet la production correcte d'en-têtes et de pieds de pages courrants pour différents types d'applications comme les dictionnaires, les manuels, etc.

4. Modèle de traîtement

4.1. Concepts du placement des flottants

Pour construire une (double-)page l'algorithme commence par assembler et collecter suffisament de matériel pour pouvoir remplir la page sans placer aucun flottant. Dans le même temps, tous les flottants dont le point d'appel est dans le texte collecté sont mis en forme et mémorisés. Ils forment, avec les flottants restant en attente des pages précédentes, une liste ordonnée de flottants à essayer.

Les zones de placement des flottants dans la page en cours de construction sont également ordonnées.

L'algorithme procède en prenant le premier flottant dans la liste des « à essayer » et en essayant de le positionner dans la première zone de placement. Ensuite il vérifie que toutes les contraintes (voir ci-après) sont validées et, sinon, l'algorithme essaiera de placer le flottant dans la prochaine zone, et ce soit jusqu'à ce que les contraintes soient satisfaites, soit que la liste des zones de placement soit épuisée. Un essai qui n'échoue pas indique que cette distribution de flottants sur la page devient la meilleure solution du moment, et que tous les autres essais se feront en essayant d'ajouter un flottant à cette solution (pas de retour arrière). Si l'algorithme n'arrive pas à placer le flottant

dans quelque zone que ce soit, cela signifie que le flottant sera renvoyé dans une prochaine page.

À chaque fois qu'un flottant est ajouté à une zone, les contraintes pour les essais suivants sont modifiées. Il y a plusieurs raisons à cela : d'une part, le point d'appel des différents flottants se déplace puisque le flottant occupera de la place dans la page ; d'autre part, placer un flottant dans une zone peut conduire à interdire le placement d'autres flottants dans la même zone ou dans d'autres zones.

4.2. Pages et colonnes de flottants

Pour le moment il n'y a qu'une gestion rudimentaire des pages de flottants : au début de chaque page l'algorithme essaiera de construire une page en utilisant tous les flottants qui sont en attente. Cependant il n'y a pas de contrôle de la mise en page permettant de définir les conditions selon lesquelles l'essai est concluant ou non.

4.3. Stockage des flottants

Les corps des flottants sont composés dans des boîtes au point d'appel, comme pour les environements figure et table du LaTeX standard ; il serait possible également de spécifier au point d'appel un pointeur logique vers un flottant dont la composition est spécifiée ailleurs (par exemple dans un fichier externe).

Cependant, les sous-éléments textuels tels que les légende, etc (par exemple indiquées par \caption) ne sont pas composés si tôt mais sont stockés dans des listes de *tokens* ; cela permet d'essayer différentes mises en forme possibles, par exemple pour sa taille, durant les essais de placement du flottant. À l'heure actuelle, on est limité à un seul élément de légende par flottant.

4.4. Traitement des légendes

Quant un flottant est placé dans une zone, le formatage de la légende est réalisé, pour essai, et rattaché au corps du flottant. Ce processus peut prendre en compte quelques informations sur la placement du flottant en cours de test, comme la zone à utiliser, le fait d'être en mode recto-verso ou non, etc. L'algorithme peut envisager plusieurs possibilités avant de trancher sa décision, par exemple, si une mise en forme de la légende[7] se traduit en une violation de certaines contraintes, une approche différente pourra être utilisée.

7. NdT : dans l'article original, il est dit « *float* » c.-à-d. flottant, ce qui pourrait laisser à croire que le formatage du corps du flottant n'est pas encore réalisé.

4.5. Éjection des flottants

Il est possible d'indiquer dans le source du document des endroits limites au delà desquels les flottants dont les points d'appels sont déjà passés ne peuvent pas aller. En d'autres termes un « point d'éjection » impose à l'algorithme de positionner tous les flottants concernés (c-à-d en attente) dans des zones qui sont « avant » le point d'éjection.

Si à cause d'autres contraintes le flottant n'a pas pu être placé dans une zone satisfaisante, l'algorithmes réessaye en premier lieu de placer le flottant dans toutes les zones valides en utilisant un ensemble de contraintes moins rigide (par exemple, les restrictions sur le nombre maximum de flottants par zone sont suprimées), s'il ne trouve toujours pas de solution, il essaiera en dernier recours de placer le point d'éjection sur une prochaine colonne, ce qui signifie couper la colonne de texte avant le point d'éjection.

L'éjection des flottants peut se faire soit pour tous les flottants, soit sur une base typée, par exemple il est possible d'éjecter uniquement les flottants de type « figure ».

À un point d'éjection peut également être attribué un valeur qui contrôle la « rigidité » utilisée par l'algorithme. Par défaut, l'algorithme est en mode strict décrit ci-avant. L'attribut column autorise l'algorithme à ne pas respecter le point d'éjection du moment que le flottant reste dans la bonne colonne. De la même manière, les valeurs page et spread assureront que le flottant n'est pas renvoyé plus loin que la page ou la double-page courante. De cette manière on peut s'assurer qu'un flottant est toujours visible depuis son point d'appel.

4.6. Relations entre flottant et point d'appel

L'algorithme garde une trace de la relation entre un flottant et son point d'appel. Cela permet de définir des contraintes utilisées par l'algorithme dans la phase de placement. Il est toujours acceptable de placer un flottant « après » son point d'appel, c.-à-d. dans une colonne ou page apparaissant plus tard. Pour le moment, les contraintes suivantes peuvent être précisées :

none[8] qui indique que la relation entre le point d'appel et le placement du flottant n'est d'aucune importance,

page qui indique que le flottant peut être placé n'importe où dans la page contenant le point d'appel (il est visible depuis le point d'appel),

column qui indique que le flottant peut être placé avant le point d'appel sous condition qu'il soit dans la même colonne,

8. NdT : « aucune » en français.

after[9] qui indique que le flottant doit être placé strictement après le point d'appel.

Quand l'algorithme sera étendu pour gérer directement les double-pages, une option sera ajoutée à cette liste pour autoriser un flottant à être placé avant le point d'appel pourvu que ce soit sur la même double-page.

4.7. Relations entre flottants larges et point d'appel

Pour les flottants qui s'étendent sur deux colonnes ou plus, il y a plusieurs façons de considérer la relation spatiale entre le point d'appel et la zone de placement. Par exemple, si un flottant dont le point d'appel est dans la deuxième colonne a été placé dans la zone `b12`, le flottant est-il avant ou après son point d'appel? La réponse change selon qu'on considère que le flottant est en première ou en deuxième colonne, les deux ayant du sens.

Pour le moment, les comportements suivants peuvent être spécifiés :

strict qui indique que la colonne la plus à gauche de la zone est considérée comme la colonne où le flottant à été placé,
flexible qui indique que c'est la colonne de droite de la zone.

Ces règlages n'ont de sens que si les relations principales entre flottant et point d'appel sont `column` ou `after`.

4.8. Relations entre flottants et notes de bas de page

Il est possible d'imposer à l'algorithme de vérifier pour chaque colonne la présence de notes de bas de page et, si oui, de l'empêcher de placer des flottants dans les zones du bas. En théorie il est possible qu'une configuration interdite disparaisse d'elle-même une fois que l'algorithme aura placé d'avantage de flottants, ce serait par exemple le cas où en ajoutant des flottants dans la page, la note encombrante changerait de colonne. Cependant, vérifier cela induirait des retour-arrières potentiellement considérables dans l'algorithme ; en conséquence l'algorithme utilise une méthode plus conservatrice et considère simplement qu'un essai de positionnement a échoué si des notes et des zones basses se rencontrent.

9. NdT : « après » en français.

Il est prévu d'ouvrir le choix au *designer*[10] de préciser où les notes de bas de page doivent être placées par rapport aux flottants des zones basses (si la combinaison est autorisée). À l'heure actuelle ce n'est pas possible et les notes apparaissent toujours juste après le texte de la colonne, c'est-à-dire au dessus des flottants des zones basses.

4.9. États des zones

Pour chaque zone, l'algorithme garde trace du fait que la zone est fermée ou non à un type de flottant précis, par exemple n'accepte plus de flottant de type « figure » ou est fermée à tous les types. L'état d'une zone peut changer selon que des flottants sont placés dans d'autres zones (cela peut par exemple fermer toutes les zones précédentes, ou des zones en recouvrement) ou selon que cette zone devienne trop pleine (par exemple avec une contrainte de taille ou de nombre de flottants pour la zone).

Certaines de ces contraintes peuvent être assouplies dans certains cas ; par exemple, si l'algorithme doit éjecter les flottants en attente avant un point donné dans le flot du texte, il ne tiendra pas compte des contraintes sur la taille de la zone ni sur le nombre de flottants. Cependant, si une zone a été fermée à cause d'un autre flottant qui a été placé dans une autre zone, elle restera fermée pour assurer que les flottants apparaissent dans l'ordre et pour s'assurer que les zones en conflit pour cause de recouvrement, comme expliqué à la section 3.3, ne seront pas utilisées simultanément.

4.10. Contraintes sur les zones

L'algorithme offre plusieurs possibilités au *designer* de spécifier comment et dans quelles circonstances un flottant est autorisé à être ajouté dans une des zones de placement.

Ainsi que nous l'avons expliqué précédemment toutes les zones d'une page sont testées dans un ordre donné. Cet ordre peut être spécifié et changé pour des parties spécifiques du document. Les zones qui sont fermées pour un type donné ne seront pas envisagées, tout comme les zones qui ne font pas la bonne largeur pour contenir le flottant. Si ces premiers tests sont validés, le flottant

10. NdT : l'article original distingue *designer* et utilisateur, *designer* étant pris au sens de la personne qui fait le dessin d'une mise en page, par opposition à la personne qui utilise cette mise en page. Il s'agit donc plutôt de la distinction entre le développeur d'une classe de documents et l'utilisateur de cette classe. Le terme de *designer* a été conservé dans cette traduction.

peut cependant ne pas être plaçable dans la zone considérée s'il ne remplit pas les contraintes suivantes :

- il y a une limite supérieure du nombre de flottants qu'une page peut accueuillir,
- chaque zone a une limite supérieure du nombre de flottants qu'elle peut contenir,
- après le placement du flottant l'espace restant dans les colonnes de texte doit être plus grand qu'une valeur spécifique.

Toutes ces contraintes sont paramètrables.

Des contraintes supplémentaires seront probablement ajoutées quand on aura accumulé suffisament d'expérience sur les contrôles réellement utiles pour permettre la spécification d'un nombre raisonnable de mises en forme.

Par exemple, LaTeX 2ε permet au *designer* de contraindre la taille maximum d'une zone, mais doit-on contraindre la hauteur d'une zone, ou de l'ensemble des zones empilées? Ou bien doit-on contraindre les deux?

4.11. "Here" ou pas "Here" [11]

LaTeX 2ε permet à l'utilisateur de contrôler le placement d'un flottant donné dans une ou plusieurs zones spécifiées en utilisant des lettres. Une notation spéciale, h désignerait un « *here* » flottant. Sa sémantique annoncée est d'essayer de placer le flottant « à l'emplacement dans le texte où le flottant apparaît » [1, p. 197]. Si ce n'est pas possible LaTeX 2ε essaiera les autres possibilités proposées sur la page suivante, ainsi un flottant avec l'indication ht apparaitra-t-il soit dans le texte soit en haut de la prochaine page, ou d'une suivante [12].

Dans de nombreux cas on préfère un « *here* » qui s'ignifie toujours « ici ». Ce second cas est implémenté dans des paquetages additionnels pour LaTeX 2ε, souvent cependant quitte à autoriser les flottants à apparaître dans le désordre.

Le nouveau modèle gère uniquement le « *here* » absolu pour les flottants ; cependant, l'ordonancement correct des flottants dans le document est garanti (si la commande générant le flottant « here » impose un point d'éjection pour les flottants du même type). S'il n'y a pas assez de place pour mettre le flottant

11. NdT : dans le titre anglais, *To "Here" or not to "Here"* (allusion à *To be or not to be*), *Here* (avec le sens de « ici ») désigne une option classique de placement des flottants.

12. En mode bicolonne cela peut se traduire par le placement du flottant en haut de la seconde colonne même si le point d'appel est finalement dans cette colonne.

dans la colonne, le flottant et la ligne précédente [13] sont déplacés a la prochaine colonne/page.

5. Contrôle par l'utilisateur

5.1. Changement de colonne et de page

Le changement de colonne ou de page peut être contrôlé dans le source du document en y plaçant certaines commandes. La commande \columnbreak termine la colonne courante après la ligne en cours (si utilisée en mode horizontal). Similairement \pagebreak termine la page courante [14].

5.2. Éjection manuelle des flottants

L'éjection des flottants peut être demandée dans le source du document par la commande \flushfloats. Cette commande prend deux arguments optionnels qui, si présents, indiquent le type de flottant à éjecter (par défaut, tous) et la rigidité de l'éjection (par défaut strict). Les autres valeurs possibles pour la rigidité sont column, page et spread.

5.3. Indiquer les zones préférées

Au moment d'écrire cet article, l'interface pour spécifier dans le source du document le groupe de zones dans lesquelles un flottant peut apparaître n'est pas arrêtée. On peut envisager de conserver l'interface originale de LaTeX d'environnements flottants avec un argument optionnel. Dans ce cas quelque chose comme [t] serait interprété en interne comme « toute zone haute existante » et traduit en une liste comme t12 t11 t21. Mais d'autres interfaces sont également envisageables.

5.4. Placer les flottants à la main

Tout algorithme qui place automatiquement les flottants peut, dans certains cas, échouer à produire un résultat adéquat. Avec LaTeX 2_ε, l'utilisateur ne disposait que de l'argument optionnel de l'environnement de flottant et, avec

13. Plus précisément la colonne est coupée au dernier point de coupure autorisé précédent la position courante, qui se trouve généralement une ligne avant, mais peut se trouver avant (ou après).

14. Pour le moment ces commandes forcent la coupure ; il n'y a pas de possibilité, comme avec LaTeX 2_ε, de suggérer uniquement que ce point est bon ou mauvais pour la coupure.

```
Page: 1 (1)
   Area: t13
      Float: 4 (figure 4) []
   Area: b21
      Float: 2 (figure 2) [mylab:fig1]
   Area: t31
      Float: 3 (figure 3) [mylab:fig2]

   Area: hhh
      Float: 11 (table 1) []

Page: 2 (2)
   Area: t13
      Float: 8 (figure 8) []
   Area: t22
      Float: 5 (figure 5) []
   Area: b11
      Float: 6 (figure 6) [mylab:fig3]
   Area: b31
      Float: 7 (figure 7) [mylab:fig4]
```

FIGURE 2 – *Exemple de fichier* fpl

cette méthode, en déplaçant légèrement le corps du flottant dans le source du document, il pouvait finalement changer la mise en forme selon ses souhaits.

C'était une tâche manuelle, source d'erreur, et le moindre changement dans le texte du document était succeptible de ruiner ses efforts.

Pour améliorer cette situation, le nouvel algorithme peut écrire un fichier contenant toutes ses sélections de flottants [15] (voir l'exemple figure 2). Par simple déplacement de lignes l'utilisateur peut produire des altérations sur cette sélection. Si un fichier modifié est présent sour le nom \jobname.fpc l'algorithme l'utilisera sans essayer d'appliquer aucune de ses règles internes. Ainsi la mise en forme se fera exactement comme spécifié [16].

Outre déplacer les flottants entre zones de placement, il sera possible de les placer dans (ou en dehors de) zones spéciales appelées hhh qui représentent une liste de tous les flottants « *here* » sur une page. Si un flottant est mis dans

15. Dans ce contexte on entend par flottant les flottants traditionnels, et non les notes de bas de page ou en marge.

16. Si les flottants sont stockés dans le document source à leur point d'appel, l'algorithme ne sera capable de positionner un flottant que si il a déjà passé ce point dans le source. Cela signifie qu'on ne peut pas déplacer un flottant indéfiniment tôt dans le document mais seulement dans une certaine limite. Si les flottants sont stockés à l'extérieur du source du document cette restriction ne s'applique pas.

```
=====================================================================
 STATS: floats waiting = 37 on page 2
=====================================================================
Float: \bx@A   {figure}  {1.1}   {a special one}
 area trial: t13  -> failed: span count t13 /= 1
 area trial: t12  -> failed: span count t12 /= 1
 area trial: t22  -> failed: span count t22 /= 1
 area trial: t32  -> failed: span count t32 /= 1
 area trial: t11  -> failed: t11 float not allowed by user control (t21)
 area trial: b11  -> failed: b11 float not allowed by user control (t21)
 area trial: t21  -> accepted
Float: \bx@B   {figure}  {1.2}   {This is a figure caption for (1)}
 area trial: t13  -> failed: span count t13 /= 1
 area trial: t12  -> failed: span count t12 /= 1
 area trial: t22  -> failed: span count t22 /= 1
 area trial: t32  -> failed: span count t32 /= 1
 area trial: t11  -> failed: area closed for type figure
 area trial: b11  -> failed: area closed for type figure
 area trial: t21  -> accepted
```

FIGURE 3 – *État d'avancement de l'algorithme*

une zone « *here* » cela signifie qu'il sera positionné comme un flottant « *here* » à son point d'appel.

À titre d'extension de cette méthode nous faisons des essais de contrôle manuel restreint à certaines parties du document, par exemple permettre à l'utilisateur de contrôler manuellement un seul chapitre tout en laissant l'algorithme s'occuper du reste. Nous prévoyons également d'intégrer un contrôle de la longueur des colonnes par ce biais, de telle sorte qu'il soit possible de rallonger ou raccourcir une (double-)page par des spécifications externes au document plutôt qu'en modifiant le source.

5.5. Surveiller le comportement de l'algorithme

Par opposition à la routine de sortie de LaTeX 2_ε, qui, vue de l'utlisateur, est une boîte noire, le nouvel algorithme essaie le plus possible de rendre son mode de décision compréhensible. La figure 3 montre un exemple d'état d'avancement ainsi produit. Il indique pour chaque flottant chacune des zone qui a été envisagée, pourquoi elle a été écartée, etc. Il y a également une option qui produit à peu près 1000 fois plus d'informations, mais elle ne semble utile que pour chercher les erreurs dans le code.

6. Spécifications de mise en page

Dans la classe de documents le *designer* garde le contrôle du comportement de l'algorithme sur tous les aspects décrits précédemment (et quelques autres).

Les spécifications sont faites selon le nouveau concept de patron [17] et d'instance, voir [2]. Des informations supplémentaires (code expérimental, documentation plus détaillée, etc.) se trouvent sur le site web du projet LaTeX à l'adresse :

http://www.latex-project.org

Contrairement à l'algorithme lui-même, qui dans ses fonctionnalités de base semble maintenant stable et fiable, l'interface de *design* est beaucoup plus expérimentale. Aussi les exemples de déclarations donnés ici ne sont que le reflet des idées actuelles (ou de l'implémentation) et sont succeptibles d'être modifiés à tout instant.

6.1. Déclaration de type de flottants

Les types de flottants sont déclarés par la commande \DeclareFloatType qui pour le moment prend deux paramètres : le nom du type auquel on fera référence à plusieurs endroits et une lettre utilisée pour produire les noms des fichiers externes, par exemple pour les listes de figures, etc.

```
\DeclareFloatType{figure}{f}
\DeclareFloatType{table}{t}
\DeclareFloatType{algorithms}{a}
```

Il est clair que d'autres informations doivent être indiquées avec le type, par exemple le mode de numérotation, etc. Ces informations seront probablement ajoutées en changeant le deuxième argument en une liste d'affectations clef/valeur comparable à la façon dont sont déclarées les zones.

La déclaration d'un nouveau type de flottant définit automatiquement les environnement nécessaires pour l'utilisateur.

6.2. Déclaration des zones

Toute zone de placement qui sera utilisée à un certain moment par l'algorithme doit être préalablement déclarée. Cela se fait avec la commande \DeclareFloatArea

17. NdT : *template*, en anglais.

qui prend deux arguments : le nom de la zone (qui doit respecter la convention exposée section 3.3) et une liste d'affectations clef/valeur qui décrit les principales caractéristiques de cette zone.

```
\DeclareFloatArea{t22}
  {
    type-close-list = {t11,b11},
    all-close-list  = {t12,t32},
    max-float-num   = 2,
  }
```

Pour l'intant une zone est caractérisée par le nombre maximum de flottants qu'elle peut accueuillir (max-float-num) et par deux listes qui indiquent à l'algorithme quelles autres zones sont affectées par le placement d'un flottant dans cette zone. La liste type-close-list indique quelles zones sont fermées au type de flottant qui est ajouté à la zone courante (respect de l'ordre), tandis que la liste all-close-list indique quelles zones doivent être totalement fermées si celle-ci accueuille un flottant (gestion du recouvrement).

La clef type-close-list est supposée, au départ, spécifier un ordre partiel des zones pour s'assurer que les flottants n'apparaissent pas dans le désordre dans le document. Par exemple, la déclaration précédente nous dit : si un flottant est placé en zone t22 (c'est-à-dire la zone haute commençant en colonne deux et faisant deux colonnes de large), alors les zones d'une colonne de largeur t11 et b11 (c'est-à-dire celles de la première colonne) sont fermées pour les flottants de ce type. Cependant, en supposant que cet exemple apparaisse dans la définition d'une mise en page en quatre colonnes qui prévoit des zones comme t14 et t13, il n'est pas fait mention de la fermeture de ces zones. Ainsi, dans ce cas particulier, un flottant s'étendant sur trois ou quatre colonnes serait-il toujours autorisé à apparaître dans une zone haute.

D'autre part, la clef all-close-list est là pour indiquer des contraintes plus visuelles, par exemple « si t12 est utilisée on ne veut pas utiliser b12, on se restreint à b22 dans ce cas ». De plus, elle doit satisfaire la restriction à propos des zones en recouvrement décrite à la section 3.3 ; ainsi, dans l'exemple, t12 et t32 sont fermées puisqu'elles recouvrent partiellement t22. [18]

6.3. Déclarations de mise en forme des notes de bas de page

La mise en forme des notes de bas de page est indiqué en déclarant des instances du type footnotesetup. Pour le moment trois patrons sont disponibles,

18. Comme mentionné précédemment, cette restriction pourrait être levée dans une version future de l'algorithme ; dans la mesure où c'est obligatoire, on pourrait alternativement ajouter ces zones automatiquement pour éviter les problèmes à la compilation.

bien qu'ils ne doivent être considérés que comme des prototypes : le patron std produit des notes conventionnelles sous chaque colonne, le patron ftnright place toutes les notes sous la colonne de droite, et le patron margin renvoit les notes dans la marge extérieure.

Les clefs des patrons ci-dessus fournissent seulement une flexibilité rudimentaire (pour dire ça positivement), dans une version de production chacune nécessiterait un grand nombre d'extensions. Par exemple

```
\DeclareInstance{footnotesetup}
  {mainmatter}{std}
  {
   text-sep    = 14pt plus 3pt,
   max-height  = 8in,
  }
```

déclarerait l'instance nommée mainmatter qui fournit des notes sous les colonnes avec une séparation de 14pt+ et une hauteur maximale de notes de 8in par colonne.

Des instances comme celle-là peuvent alors être utilisées dans la déclaration d'une mise en page particulière comme nous le verrons. Alternativement on pourrait utiliser des instances non-nommées par le biais de \UseTemplate.

6.4. Déclarations de mise en page

Au cœur de la déclaration de la mise en page on trouve les instances du type pagesetup2.[19] La figure 4 donne un exemple de réglage utilisant toutes les variables disponibles pour le moment. Les quatre premières clefs (column-num, column-width, column-height, et column-sep) décrivent le colonnage de la page, à savoir ici une page à deux colonnes.

Les quatre clefs suivantes définissent les contraintes standard pour l'algorithme quand il placera les flottants : max-float-num est le nombre maximum de flottants qui peuvent apparaître sur une page normale, float-callout-constraint est le type de relations autorisées entre flottant et point d'appel[20], float-callout-span-constraint choisit l'interprétation du positionnement des flottants multicolonnes, et bottom-float-footnote-constraint indique si oui ou non les flottants en bas de page sont aurotisés s'il y a des notes.

Les trois dernières contraintes sont remplacées par flush-float-callout-constraint, flush-float-callout-span-constraint, et flush-bottom-float-footnote-constraint si une

19. Le nombre 2 à une origine historique et disparaîtra dans les versions futures.
20. NdT : *call-out* a été traduit par point d'appel.

```
\DeclareInstance{pagesetup2}{mainmatter}{std}
 {
  column-num                          = 2,
  column-width                        = 220pt,
  column-height                       = 610pt,
  column-sep                          = 20pt,
  max-float-num                       = 3,
  float-callout-constraint            = after,
  float-callout-span-constraint       = strict,
  bottom-float-footnote-constraint    = forbidden,
  flush-float-callout-constraint      = page,
  flush-float-callout-span-constraint = flexible,
  flush-bottom-float-footnote-constraint = none,
  area-list                           = {t12,t11,b11,b12,t21,b21},
  defer-type-close-list               = {t12,t11,b11,b12,t21,b21},
  defer-all-close-list                = ,
  footnote-setup                      = mainmatter,
 }
```

FIGURE 4 – *Exemple de déclaration du patron* pagesetup2 *montrant toutes les clefs*

éjection ne peut pas être effectuée sans réduire les contraintes (max-float-num est ignorée automatiquement dans ce cas).

La clef area-list définit l'ensemble des zones de placement autorisées ainsi que l'ordre dans lequel ces zones sont testées pour positionner les flottants. Les clefs defer-type-close-list et defer-all-close-list définissent ce qui doit être fait dans le cas précis où un flottant n'a pas pu être placé. Par exemple, si un flottant d'un type donné ne peut pas être placé, alors toutes les zones listées dans defer-type-close-list seront fermées pour ce type. En d'autres termes, ces deux clefs sont comparables à celles utilisées pour la déclaration des zones.

Ces clefs, ainsi que celles de la déclaration de zone, sont donc particulièrement importantes pour garantir un ordre intelligent des flottants dans la page après mise en forme.

Dans une implémentation précédente de l'algorithme on utilisait un modèle plus simple : il y avait une seule liste de zones qui était raccourcie à chaque fois qu'un flottant ne pouvait pas être placé, restreignant ainsi les flottants suivants à cette liste limitée de zones. Cela fonctionnait bien tant que l'on avait principalement des flottants mono-colonne, puisque dans ce cas les zones pouvaient être raisonnablement ordonnées en une séquence simple. Cependant, sitôt que l'on gère les flottants multi-colonnes la situation devient moins immédiatement décidable. Est-il autorisé de placer un flottant à venir en t12 s'il y en a déjà un en t11?[21]

21. NdT : Le paragraphe original n'est pas plus clair, malheureusement.

Il est assez probable que les règles prévues pour le moment se révèleront trop primitives. Nous étudierons cela une fois que des mises en page en nombre suffisant auront été produites avec ce modèle (ou n'auront pas pu être produites parce qu'elles se seront avérées ne pas être assez spécifiées).

Finalement, la clef footnote-setup recevra une instance du patron footnotesetup.

6.5. Déclarations des mises en forme de flottants

Il existe un prototype d'interface utilisant les patrons du type buildfloat pour le rattachement des légendes aux corps des flottants. Au moment de la rédaction de cet article, les patrons disponibles sont centeredbelow, centeredabove, et bottomright, qui centrent la légende au-dessous ou au-dessus du corps du flottant ou la place à sa droite, alignée avec le bas du corps du flottant. Il faudrait généraliser pour un système de production pour qu'ils soient plus flexibles.

Quand il essaie de mettre en forme un flottant, l'algorithme vérifie l'existence d'instances de buildfloat et utilise la première pour construire le flottant. Plus précisément, il vérifie d'abord s'il en existe une avec le nom ⟨*area*⟩-⟨*type*⟩, ensuite il cherche ⟨*area*⟩, puis ⟨*type*⟩ et finalement, si aucune n'existe, il cherche une instance nommée default. Donc il faut au minimum que la dernière soit déclarée par la classe de documents.

```
\DeclareInstance{buildfloat}{default}
  {centeredbelow}{}
\DeclareInstance{buildfloat}{table}
  {centeredabove}{}
\DeclareInstance{buildfloat}{t31}
  {bottomright}{}
\DeclareInstance{buildfloat}{t22}
  {bottomright}{}
```

Cet exemple de déclaration indique que les légendes doivent être au dessus pour les tables, et en dessous pour les autres types, sauf pour les zones t31 et t22 où la légende est sur le côté.

7. Performances de l'algorithme

Pour mesurer les performances de l'algorithme, on a utilisé un fichier de test un peu ridicule qui contient trois types de flottants (figures, tables et algorithmes) ayant au total 47 flottants. La mise en forme choisie compte 3 colonnes

```
STATS: floats waiting = 24 on page 1
STATS: trials = 286
STATS: floats waiting = 19 on page 2 (float page)
STATS: trials = 159
STATS: floats waiting = 37 on page 2
STATS: trials = 397
STATS: floats waiting = 19 on page 3 (float page)
STATS: trials = 166
STATS: floats waiting = 7 on page 4 (float page)
STATS: trials = 41
STATS: floats waiting = 20 on page 4
STATS: trials = 204
STATS: floats waiting = 5 on page 5 (float page)
STATS: trials = 27
STATS: floats waiting = 12 on page 5
STATS: trials = 108
STATS: floats waiting = 0 on page 6 (float page)
STATS: trials = 0
STATS: floats waiting = 6 on page 6
STATS: trials = 57
    ...
STATS: floats waiting = 6 on page 12 (float page)
STATS: trials = 26
STATS: floats waiting = 6 on page 12
STATS: trials = 37
STATS: floats waiting = 0 on page 13
STATS: trials = 0
```

FIGURE 5 – *Statistiques sur l'algorithme*

et 11 zones potentielles. Les légendes des figures sont placés sous le flottant, elles sont placées au-dessus pour les tables et les algorithmes [22]. Exception faite des zones hautes touchant la marche extérieure : les flottants placés ici ont leur légende sur leur droite et partiellement dans la marge. Les notes de bas de page de toutes les colonnes sont réunies dans la marge extérieure.

Les flottants devaient être strictement placés après leur point d'appel, et un maximum de dix flottants par page a été imposé, soit à peu près trois par colonne.

Dans la mesure où le document contenait beaucoup de flottants très tôt (24 sur la première page) et où les premiers étaient spécialement conçus pour ne pas être plaçables au premier essai, l'algorithme à dû beaucoup travailler pour placer tous les flottants en attente. La figure 5 montre quelques statistiques telles que l'algorithme les fournit sur le nombre d'essais nécessaires (le plus élevé fut 397 pour 37 flottants ; pour comparaison, la formule (1) indiquait 22 595 200 368 essais ce qui aurait certainement pris un peu plus de temps à

22. Cette règle ne semble pas usuelle en français, nous écrivons donc les légendes d'algorithmes après ceux-ci dans ce *Cahier*. [Ndlr]

	PIII (650MHz)	486DX4 (75MHz)
	pas de surveillance	
real	0m1.533s	0m27.633s
user	0m1.460s	0m26.940s
sys	0m0.050s	0m0.690s
	état d'avancement	
real	0m3.116s	0m36.885s
user	0m1.740s	0m34.470s
sys	0m0.080s	0m2.420s
	surveillance complète	
real	0m7.833s	1m22.480s
user	0m2.720s	1m7.890s
sys	0m0.280s	0m12.360s

FIGURE 6 – *Temps de calculs de l'algorithme*

évaluer !). À retenir que sur la troisième page l'algorithme à réussi à produire une page de flottants, pour toutes les autres pages cette tentative a échoué.

La figure 6 montre les temps de calcul pour produire le document final de 13 pages selon que l'algorithme est utilisé avec différents réglages du mode de surveillance. Les machines de test étaient un Pentium III 650 et un portable plus ancien avec un 486. Dans les deux cas TEX était celui directement installé par un CD TEX Live 4.

Ces mesures de temps montrent que l'algorithme a un temps de traitement très acceptable puisque même sur un 486 le temps moyen pour produire une page est de l'ordre de 2 secondes.

8. Perspectives

Bien que l'algorithme courant se comporte bien il y a plusieurs domaines dans lesquels ses fonctionalités pourraient, et même devraient, être étendues. Les points les plus importants sont recencés ici.

- L'équilibrage partiel des pages, comparable à la façon de procéder du paquetage multicol, devrait être implémenté pour permettre des mises en forme où, par exemple, un titre peut s'étendre sur plusieurs colonnes.
- Nous souhaitons fournir plus de contrôle sur les zones en marge, permettant les flottants en marge ainsi que d'autres objets en marge, le tout intéragissant convenablement.

– Sans trop d'effort, l'algorithme pourrait être étendu pour gérer convenablement les doubles-pages, donc cela devrait être ajouté un jour prochain.
– Une fois que l'algorithme a décidé quels flottants il allait placer sur une page, on devrait pouvoir ajouter un post-traitement dans lequel on reconsidèrerait le placement exact en fonction de certaines règles. Si, par exemple, la relation au point d'appel est `page` alors les flottants vont tendre à être dans les colonnes de gauche. C'est fort bien tant que l'on a beaucoup de flottants à placer, mais sur une page où il n'y a que peu de flottants on pourra préférer les redistribuer différemment une fois que l'on sait clairement quels flottants peuvent aller sur le page.
– Puisque l'on sait à l'avance combien de flottants sont en attente de placement, on pourrait utiliser un autre algorithme qui testerait tous les placements possibles aussi longtemps que l'on a un nombre limité de flottants à placer. La limite à partir de laquelle on changerait d'algorithme pourrait alors être paramétrable pour que les gens avec plus de puissance machine (ou plus de patience) puissent obtenir le résultat optimal avec autant de flottants qu'ils le souhaitent.

Bibliographie

[1] Leslie Lamport. *LaTeX: A Document Preparation System*. Addison-Wesley, Reading, Massachusetts, seconde édition, 1994.

[2] Frank Mittelbach, David Carlisle, and Chris Rowley. New interfaces for LaTeX class design. *TUGboat*, 20(3):214–216, septembre 1999.