# Optimisation de la présentation d'un document hypermédia

Bruno Bachelet, Philippe Mahey

# Optimisation de la présentation d'un document hypermédia

Bruno Bachelet* et Philippe Mahey†
Laboratoire LIMOS, FRE 2239 - CNRS, Université Blaise Pascal

**Résumé**

Cet article présente un algorithme pour l'ajustement de la présentation d'un document hypermédia dans le but de satisfaire au mieux les exigences de son auteur. Nous modélisons le problème sous la forme d'un problème de tension de coût minimum dans un graphe et proposons un algorithme basé sur la méthode de mise à conformité (*out-of-kilter*) pour le problème de flot de coût minimum. Dans un premier temps, nous proposons de résoudre le problème avec des coûts convexes linéaires par morceaux (2 morceaux). Ensuite, nous généralisons la méthode à des coûts convexes quelconques.

**Mots-clés :** document hypermédia synchronisé, tension de coût minimum, mise à conformité (*out-of-kilter*), optimisation convexe.

## Introduction

Tout le monde connaît maintenant le HTML qui est le langage favori pour la diffusion de documents sur Internet. Il apporte de nouvelles possibilités par rapport au support papier grâce à ses possibilités d'interactivité et à son intégration du multimédia. Mais comme toujours, les besoins augmentent et de nouveaux langages sont développés pour améliorer la structure et l'interactivité des documents. Notamment des langages comme SMIL (*Synchronized Multimedia Integration Language*) [13] ont fait leur apparition. Ils offrent une nouvelle dynamique aux documents hypermédia, et en particulier la possibilité d'y animer et synchroniser des composants multimédia.

La variété des composants qui forment un document (audio, vidéo, texte, image...) font de l'animation un problème compliqué. Comme l'expliquent [4] et [9], ces documents sont composés d'objets multimédia dont les durées de présentation doivent être ajustées afin de satisfaire un ensemble de contraintes temporelles qui traduit le déroulement de l'animation envisagée par l'auteur. Mais pour que ses contraintes soient satisfaites, l'auteur doit accepter une certaine flexibilité concernant la durée de chaque objet (durée que l'on appellera dorénavant *idéale*), les temps morts non exprimés explicitement étant complètement interdits.

Pour estimer la qualité d'un ajustement, une fonction de coût est introduite pour chaque objet. Si l'objet reste à sa durée idéale, le coût est nul, sinon il est positive et il augmente à mesure que l'ajustement s'éloigne de l'idéal. Ce qui se traduit généralement en pratique par une fonction de coût convexe (cf. figure 1). En résumé, le problème que l'on tente de résoudre ici consiste à trouver un ajustement de la meilleure qualité possible, autrement dit qui minimise la somme des coûts sur chaque arc. L'ajustement doit être calculé avant la présentation mais il peut être modifié ensuite au cours de la présentation, à cause de l'interaction avec l'utilisateur ou de retards liés aux transferts sur le réseau.

*bachelet@isima.fr - http://bruno.bachelet.net
†mahey@isima.fr

[5] et [9] proposent chacun un programme linéaire différent pour résoudre le problème avec la méthode du Simplex dans le cas de coûts convexes linéaires par morceaux avec seulement deux morceaux (cf. figure 1a). Cette méthode est adaptée uniquement à la préparation du document. [12] propose une solution basée sur une approche gloutonne pour un ajustement en temps réel pendant la présentation, mais il ignore complètement la notion de qualité.

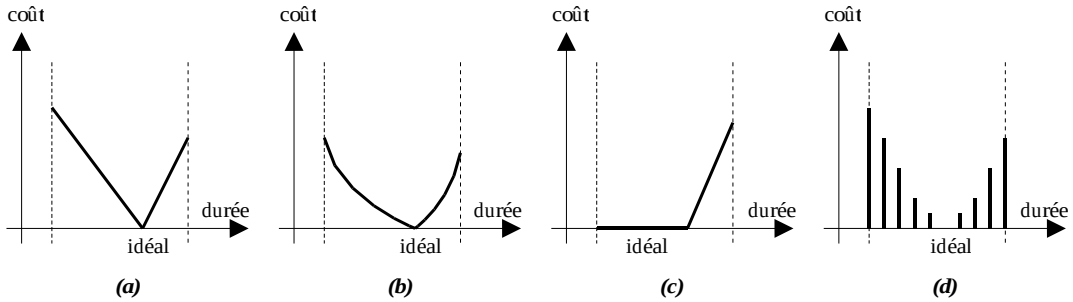


FIG. 1 – Exemples de fonctions de coût. a) Linéaire avec une valeur idéale. b) Non-linéaire avec une valeur idéale. c) Linéaire avec plusieurs valeurs idéales. d) Fonction discrète avec une valeur idéale.

Avec quelques restrictions, nous proposons de modéliser le problème comme le problème de trouver une tension de coût minimum dans un graphe. Nous considérons dans un premier temps les fonctions de coût comme dans [5] et [9] (cf. figure 1a), et nous adaptons l'algorithme de mise à conformité présenté dans [11] et [8]. Ensuite nous proposons une variante de cet algorithme pour traiter le problème avec des coûts convexes différentiables (cf. figure 1b). La flexibilité de la modélisation proposée et la vitesse de résolution de l'algorithme rendent la méthode tout à fait adaptée pour une utilisation en temps réel pendant la présentation.

Dans la section 1, nous expliquons comment ce problème de synchronisation peut être vu comme un problème de tension de coût minimum dans un graphe. Ensuite, dans la section 2, nous présentons notre adaptation de la méthode de mise à conformité. Dans la section 3, nous exposons quelques résultats numériques. Nos conclusions et futurs travaux sont présentés dans la section 4.

# 1 Graphe temporel et tension de coût minimum

Le modèle que nous proposons pour notre problème est un graphe orienté $G = (X; U)$ où $X$ est un ensemble de noeuds et $U$ un ensemble d'arcs. Les noeuds représentent des événements (le début ou la fin de présentation d'un objet), et les arcs des relations de précédence entre ces événements. A chaque arc $u$ est associé un intervalle de temps $[a_u; b_u]$, une durée idéale $o_u$ et une fonction de coût $c_u$ définie sur l'intervalle. Un arc $u = (x; y)$ entre deux noeuds $x$ et $y$ signifie que l'événement $x$ précède l'événement $y$ et qu'ils sont séparés par une durée variant de $a_u$ à $b_u$. Deux noeuds particuliers que nous appelerons *tête* et *queue* sont ajoutés. La tête précèdera tous les noeuds sans précédesseurs et la queue succédera à tous les noeuds sans successeurs. Ainsi, la tête représente l'événement le plus tôt et la queue l'événement le plus tard. Par la suite un tel graphe $G$ sera appelé *graphe temporel*.

## 1.1 Des spécifications de l'auteur au graphe temporel

[2] présente différents types de relations temporelles. La figure 2 montre ceux que le graphe temporel est capable d'exprimer. Nous proposons ici quelques exemples de modélisation des spécifications de l'auteur à l'aide d'un graphe temporel.

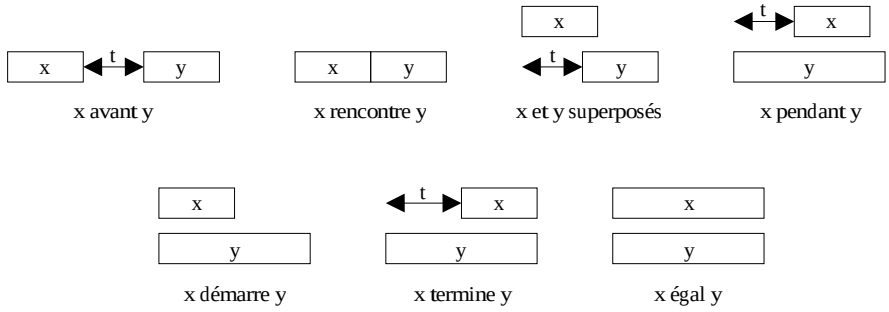


FIG. 2 – Relations d'Allen.

– Soit $i$ un objet multimédia. Dans le graphe temporel, $i$ est modélisé par deux événements (i.e. deux noeuds) $s_i$ et $e_i$ qui représentent respectivement le début et la fin de la présentation de l'objet $i$ (cf. figure 3a). Un arc $u_i$ est également nécessaire pour exprimer la relation temporelle entre $s_i$ et $e_i$.
– Considérons maintenant deux objets multimédia $i$ et $j$ qui doivent démarrer en même temps. Dans le graphe temporel, $i$ et $j$ sont modélisés comme précédemment, mais les noeuds $s_i$ et $s_j$ sont fusionnés en un seul noeud $s_{ij}$ (cf. figure 3b).
– Soit $i$ et $j$ deux objets multimédia qui doivent se terminer en même temps. Dans le graphe temporel, la situation de $i$ et $j$ est modélisée de manière similaire à la situation précédente. Cette fois ce sont les noeuds $e_i$ et $e_j$ qui sont fusionnés en un seul noeud $e_{ij}$ (cf. figure 3c).

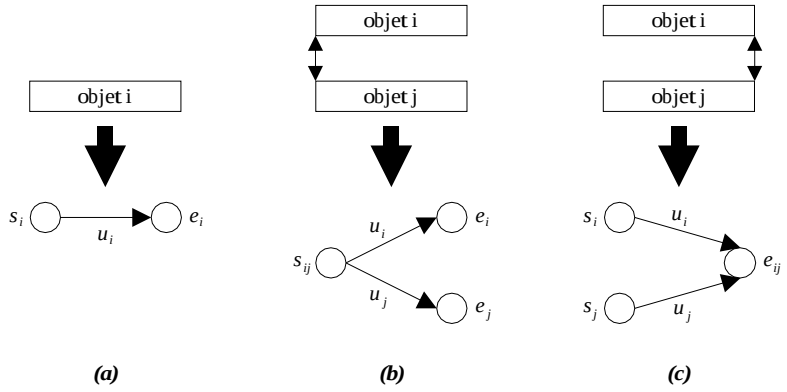


FIG. 3 – Représentation par un graphe temporel des spécifications de l'auteur.

## 1.2 Quelques définitions

Nous allons rappeler maintenant quelques définitions qui sont utiles pour la description du modèle mathématique (ces notions peuvent être retrouvées en détail dans [1]). Dans un graphe :

– Un cycle est une succession d'arcs tel que deux arcs successifs sont adjacents dans le graphe (i.e. ils ont au moins une extrémité en commun), et que le dernier arc est adjacent au premier. En choisissant un sens arbitraire de parcours d'un cycle $\gamma$, on notera $\gamma^+$ les arcs qui se trouvent dans le sens du parcours et $\gamma^-$ les arcs dans le sens opposé.
– Un cocycle est associé à un ensemble de noeuds. Il s'agit de la liste des arcs qui séparent l'ensemble du reste du graphe.
– Un flot est un vecteur de valeurs, chacune affectée à un arc, tel que la loi de conservation des noeuds est satisfaite (la somme des flots qui entrent dans un noeud est égale à celle des flots qui en sortent).

– Un potentiel est une valeur affectée à un noeud.
– Une tension est un vecteur de valeurs, chacune affectée à un arc, tel que la valeur de la tension pour un arc est la différence des potentiels de ses extrémités.

## 1.3 Tension de coût minimum

Dans un graphe temporel, chaque noeud $i$ représentant un événement, il est possible de lui affecter une date $\pi_i$ après avoir fixé arbitrairement une date au noeud tête. A chaque arc $u = (x; y)$ est associée une durée $\theta_u$ qui peut être vue comme la différence de potentiels $\pi_y - \pi_x$. Le vecteur $\theta = (\theta_u)_{u \in U}$ correspond alors à la définition d'un vecteur tension, et notre problème devient donc un problème de tension de coût minimum [3]. En utilisant une notation vectorielle et la matrice d'incidence du graphe, i.e. la matrice $A$ de dimensions $(m \times n)$ avec des éléments $a_{iu}$ égaux à $-1$ (si $u$ sort de $i$), $+1$ (si $u$ entre dans $i$) ou $0$ (dans les autres cas), le problème s'écrit simplement :

$$\text{minimiser} \sum_{u \in U} c_u(\theta_u)$$

$$\text{sous } \theta = A^T \pi,\ a \leq \theta \leq b$$

Les contraintes de synchronisation dans le graphe temporel peuvent également se traduire à l'aide de cycles et ainsi éliminer les variables de potentiel. En effet, dans un cycle, la somme des durées des arcs dans un sens doit être égale à la somme des durées des arcs dans le sens opposé. Si $S$ est une matrice de cycles où chaque colonne correspond à un cycle et chaque élément $s_{u\gamma}$ vaut $-1$ (si $u \in \gamma^-$), $+1$ (si $u \in \gamma^+$) ou $0$ (dans les autres cas), alors $\theta = A^T \pi \Leftrightarrow S^T \theta = 0$. Nous pouvons ainsi introduire un vecteur flot $\varphi$ qui est orthogonal au vecteur tension (i.e. $\varphi^T \theta = 0$) car un flot est une combinaison linéaire de cycles (i.e. $\varphi = S\delta$).

L'orthogonalité du flot et de la tension permet d'établir les conditions dites de *conformité* d'un arc $u$ [7]. On dira qu'un arc est *conforme* si :

– $a_u < \theta_u < b_u \Rightarrow \varphi_u = c'_u(\theta_u)$
– $\theta_u = a_u \Rightarrow \varphi_u \leq c'_u(\theta_u)$
– $\theta_u = b_u \Rightarrow \varphi_u \geq c'_u(\theta_u)$

Dans [7], il est prouvé que si tous les arcs sont conformes, alors la tension est optimale.

## 1.4 Réalisabilité des contraintes

Avant de chercher un ajustement de qualité optimale, il est intéressant de se demander d'abord s'il existe un ajustement réalisable. En effet, pendant la présentation où il faut réagir très rapidement aux événements imprévus comme l'intervention de l'utilisateur ou des retards réseaux, la première chose à satisfaire, c'est la possibilité de continuer la présentation en satisfaisant les contraintes de l'utilisateur. Seulement ensuite, on pourra s'intéresser à la qualité de la présentation. La condition pour qu'un ajustement réalisable existe est la suivante :

$$\forall \gamma \text{ un cycle de } G,\ \sum_{u \in \gamma^-} b_u - \sum_{u \in \gamma^+} a_u \geq 0$$

Un algorithme simple et efficace peut être utilisé pour résoudre ce problème. Il commence avec une tension nulle et, pour chaque arc $u$ où $\theta_u < a_u$ (arc dit *incompatible*), un cocycle contenant $u$ peut être trouvé pour lequel la tension de tous les arcs peut être augmentée (pour les arcs dans le sens de $u$) ou diminuée (pour les arcs dans le sens opposé à $u$) tout en gardant compatibles les arcs qui l'étaient déjà (cf. le lemme de Minty [10]). La valeur maximum de l'augmentation est déterminée et appliquée au cocycle. Si $u$ est encore incompatible, un autre cocycle est recherché pour $u$, sinon l'algorithme passe à un autre arc incompatible.

## 2 Mise à conformité

Nous allons présenter maintenant une adaptation de la méthode de mise à conformité introduite par [7] pour les problèmes de flot de coût minimum, et appliquée pour la première fois à des problèmes de tension de coût minimum dans [11]. Des algorithmes polynômiaux et fortement polynômiaux ont été étudiés récemment dans [8] mais ils restent trop lents pour nos besoins. La mise à conformité est une méthode qui modifie itérativement à la fois le flot et la tension du graphe jusqu'à obtenir l'optimalité exprimée par les conditions de conformité (cf. paragraphe 1.3). Pour chaque arc, ces conditions peuvent être représentées par une courbe, dite de *conformité*. Nous considérons ici deux types de coûts : un coût linéaire par morceaux (cf. figure 4a) et un coût convexe (cf. figure 4b).

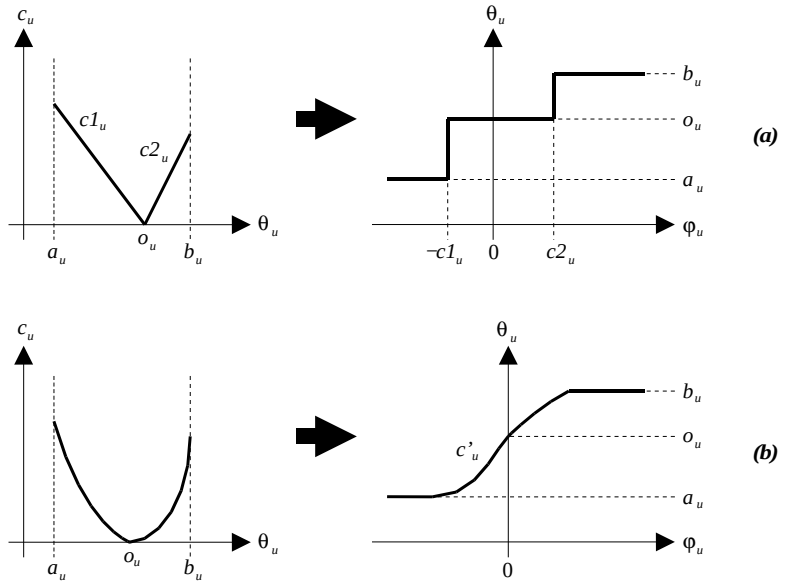


FIG. 4 – Exemples de courbes de conformité.

On constate qu'un arc $u$ avec un flot $\varphi$ et une tension $\theta$ sera conforme si son couple $(\theta;\varphi)$ est sur sa courbe de conformité. Dans le paragraphe 1.3, nous avons vu que si tous les arcs sont conformes, alors la tension est optimale. Donc toute la difficulté consiste à placer tous les arcs sur leur courbe de conformité. Nous expliquons d'abord comment faire avec des coûts linéaires par morceaux (cf. figure 4a) et ensuite nous détaillons la méthode pour des coûts différentiables (cf. figure 4b). La stratégie employée est inspirée de [11] et [8] pour résoudre le problème avec des coûts linéaires.

### 2.1 Coûts linéaires par morceaux

Pour résoudre le problème, nous classons les arcs du graphe en 4 catégories (cf. figure 5) :

- noir : les arcs dont une augmentation de la tension ou une diminution du flot les rapproche de la courbe (i.e. les arcs qui sont en dessous de la courbe) ou les maintient sur la courbe ;
- bleu : les arcs dont une diminution de la tension ou une augmentation du flot les rapproche de la courbe (i.e. les arcs qui sont en dessus de la courbe) ou les maintient sur la courbe ;
- rouge : les arcs conformes dont une diminution ou une augmentation de la tension les maintient sur la courbe (i.e. les arcs qui sont sur les parties verticales de la courbe) ;

– vert : les arcs conformes dont une diminution ou une augmentation du flot les maintient sur la courbe (les arcs qui sont sur les parties horizontales de la courbe).

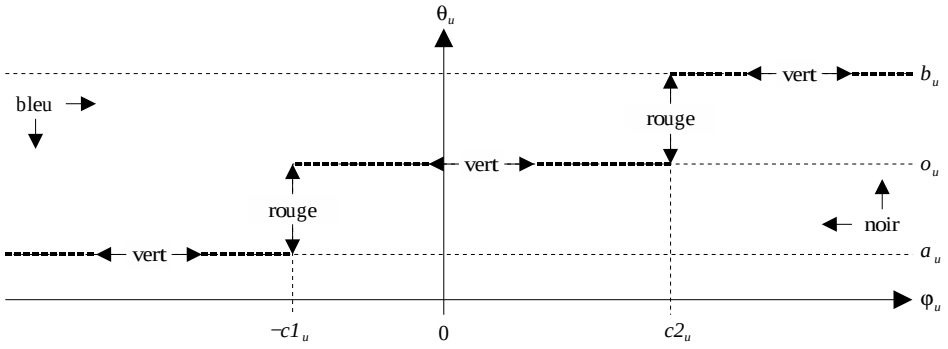


FIG. 5 – Coloration d'un arc.

D'après le lemme de Minty [10], on peut affirmer :

– pour un arc $u$ noir, il existe soit un cycle contenant $u$ pour lequel le flot peut être diminué, i.e. avec des arcs noirs (dans le sens de $u$), bleus (dans le sens opposé à $u$) et verts, soit un cocycle contenant $u$ pour lequel la tension peut être augmentée, i.e. avec des arcs noirs (dans le sens de $u$), bleus (dans le sens opposé à $u$) et rouges.
– pour un arc $u$ bleu, il existe soit un cycle contenant $u$ pour lequel le flot peut être augmenté, i.e. avec des arcs bleus (dans le sens de $u$), noirs (dans le sens opposé à $u$) et verts, soit un cocycle contenant $u$ pour lequel la tension peut être diminuée, i.e. avec des arcs bleus (dans le sens de $u$), noirs (dans le sens opposé à $u$) et rouges.

Cette affirmation, notée *(*)*, nous permet de présenter un algorithme (cf. figure 6) qui améliore la position d'un arc par rapport à sa courbe de conformité. A la manière du paragraphe 1.3, un cycle ou un cocycle y est représenté par un vecteur de valeurs $+1$, $-1$ ou $0$.

```
si (u est noir) alors
 trouver un cycle γ ou un cocycle ω selon (*);

 si (cycle trouvé) alors
  trouver λ la diminution maximum du flot sur γ;
  φ ← φ − λγ;
 sinon
  trouver λ l'augmentation maximum de la tension sur ω;
  θ ← θ + λω;
 fin si;
sinon /* u est bleu */
 trouver un cycle γ ou un cocycle ω selon (*);

 if (cycle trouvé) alors
  trouver λ l'augmentation maximum du flot sur γ;
  φ ← φ + λγ;
 sinon
  trouver λ la diminution maximum de la tension sur ω;
  θ ← θ − λω;
 fin si;
fin si;
```

FIG. 6 – Algorithme *ameliorerArc*.

Ensuite, nous proposons une méthode pour rendre tous les arcs conformes, ce qui détermine une tension optimale. L'algorithme (cf. figure 7) applique successivement la procédure précédente d'amélioration sur chaque arc du graphe et recommence jusqu'à ce qu'ils soient tous conformes.

```
trouver une tension θ compatible ;
φ ← 0 ;

tant que (∃u ∈ U, u non conforme) faire
 pour tout u ∈ U faire
  si (u non conforme) alors ameliorerArc(u) ;
 fin pour ;
fin tant que ;
```

FIG. 7 – Algorithme *rendreConforme*.

## 2.2 Coûts convexes

La méthode est un peu différente dans le cas général de coûts convexes. En effet, nous avons vu précédemment que pour améliorer un arc, soit la tension, soit le flot d'autres arcs est modifiée, mais jamais les deux en même temps, parce qu'il est très difficile de gérer cette situation. Cependant, nous avons parfois besoin de déplacer un arc déjà conforme le long de sa courbe. Comme ici la courbe n'est plus constituée de segments horizontaux et verticaux, nous devrions modifier en même temps le flot et la tension d'un arc conforme pour le déplacer. Afin d'éviter cela, nous approximons la courbe de conformité par une courbe en escalier (cf. figure 8) :

$$c_u'^{\varepsilon}(x) = \frac{c_u\big((n+1)\varepsilon\big) - c_u\big(n\varepsilon\big)}{\varepsilon} \text{ où } n \in \mathbb{Z},\ n\varepsilon \leq x < (n+1)\varepsilon$$

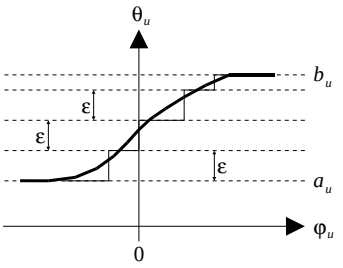


FIG. 8 – Approximation d'une courbe de conformité.

Nous utilisons la méthode présentée dans le cas linéaire pour amener tous les arcs sur leur courbe approximée, et ensuite nous affinons l'approximation de la courbe. La procédure est répétée jusqu'à obtenir la précision souhaitée. Il est recommandé de commencer avec une précision très faible et de l'affiner progressivement au lieu de démarrer directement avec une précision fine. L'algorithme de la figure 9 présente cette méthode. La variable $p$ est la précision de la courbe.

```
ε ← 0.1 × sup{b_u − a_u,  u ∈ U} ;

tant que (ε > p) faire
 rendre tous les arcs ε-conformes ; /* utiliser rendreConforme */
 ε ← sup{0.1ε, p} ;
fin tant que ;
```

FIG. 9 – Algorithme *rendreEpsilonConforme*.

# 3 Résultats numériques

Nous présentons maintenant les résultats numériques obtenus par l'algorithme de mise à conformité. Nous analysons tout d'abord les résultats dans le cas de coûts linéaires par morceaux et ensuite dans le cas de coûts convexes. Dans les deux cas, les résultats sont exprimés en secondes. Ils proviennent de tests effectués sur un processeur RISC 6000 à 166 MHz sous un système d'exploitation Unix AIX. Nous avons utilisé le langage C++ et ses capacités orientées objet pour programmer les algorithmes. Pour notre méthode, nous considérons la procédure qui trouve un cycle ou un cocycle comme une itération simple. Pour le logiciel CPLEX, une itération simple sera une itération de la méthode du Simplex. Enfin, tous les résultats (temps et nombre d'itérations) sont des moyennes de séries de 10 tests sur des graphes générés aléatoirement. Nous comparons ensuite ces résultats avec ceux obtenus sur des graphes plus structurés appelés *graphes série-parallèles*.

| Dimension graphe | | CPLEX | | Conformité | |
|---|---|---|---|---|---|
| Noeuds | Arcs | Itérations | Temps | Itérations | Temps |
| 50 | 200 | 165 | 0,5 | 313 | 0,2 |
| 50 | 400 | 300 | 0,9 | 520 | 0,4 |
| 100 | 400 | 415 | 1,2 | 615 | 0,7 |
| 100 | 800 | 703 | 2,6 | 1053 | 1,7 |
| 500 | 2000 | 3261 | 33,7 | 3033 | 21,3 |
| 500 | 4000 | 5854 | 103,3 | 5274 | 63,3 |
| 1000 | 4000 | 8990 | 271,6 | 5983 | 103,1 |
| 1000 | 8000 | 14007 | 670,2 | 10629 | 297,9 |

TAB. 1 – Tension maximum = 1000.

Les résultats de la méthode pour des coûts linéaires par morceaux sont comparés avec la résolution des programmes linéaires équivalents. Le logiciel utilisé pour les résoudre est CPLEX 6.0. Les temps présentés regroupent le temps de résolution par CPLEX et le temps nécessaire pour générer le programme linéaire. Le tableau 1 montre l'évolution du temps de résolution en fonction de la dimension du graphe. Le tableau 2 montre plutôt l'effet de l'échelle des tensions sur le temps de résolution.

| Tension | CPLEX | | Conformité | |
|---|---|---|---|---|
| maximum | Itérations | Temps | Itérations | Temps |
| 1000 | 4756 | 66,1 | 4147 | 40,4 |
| 10000 | 4763 | 71,1 | 4733 | 60,6 |
| 100000 | 4767 | 68,6 | 5081 | 66,7 |

TAB. 2 – Noeuds = 500, Arcs = 3000.

Nous pouvons observer que les temps de calcul avec la méthode de mise à conformité sont globalement deux fois plus rapides que CPLEX pour une petite échelle des tensions et qu'il rejoint les performances de CPLEX pour des échelles des tensions plus importantes. Ce qui est encourageant puisque notre méthode n'utilise pas tous les mécanismes lourds qui font de CPLEX l'un des solveurs de programmes linéaires les plus efficaces. De plus, notre méthode est très flexible et facile à implémenter dans un outil de conception et de présentation de documents hypermédia, ce qui est loin d'être le cas de CPLEX. Finalement, nous avons montré que notre algorithme est capable de traiter des coûts non linéaires. Nous avons utilisé CPLEX comme référence pour comparer les solutions optimales dans des graphes de dimension importante et pour évaluer le comportement numérique de l'algorithme proposé.

| Dimension graphe | | Conformité | |
|---|---|---|---|
| Noeuds | Arcs | Itérations | Temps |
| 50 | 200 | 2521 | 5 |
| 50 | 400 | 5084 | 16,6 |
| 100 | 400 | 5532 | 18,2 |
| 100 | 800 | 11682 | 67,9 |
| 500 | 2000 | 32871 | 579,1 |
| 500 | 4000 | 71163 | 2326,7 |

TAB. 3 – Précision = 1/1000.

| Précision | Conformité | |
|---|---|---|
| | Itérations | Temps |
| 1 | 2665 | 8 |
| 10 | 3620 | 10,5 |
| 100 | 4554 | 13,6 |
| 1000 | 5506 | 18,9 |
| 10000 | 6511 | 60,4 |

TAB. 4 – Noeuds = 100, Arcs = 400.

Pour tester la méthode avec des coûts convexes, nous utilisons la fonction quadratique $k_u(\theta_u - o_u)^2$ ($k_u \in \mathbb{R}^+$) sur chaque arc $u$. Le tableau 3 montre l'évolution du temps de résolution en fonction de la dimension du graphe. Le tableau 4 montre l'effet de la précision sur le temps de calcul. Les tests sont arrêtés à une précision de $1/1000$ car le temps de résolution augmente de manière exponentielle et qu'une précision de $1/10000$ donne des temps beaucoup trop longs. Nous supposons que cela est partiellemnt dû à limitation de l'ordinateur dans la représentation des nombres réels. Le nombre d'itérations semble linéaire quand la précision décroît, ce qui signifie que les cycles et les cocycles deviennent de plus en plus durs à trouver.

Les résultats précédents portaient sur des graphes générés aléatoirement. Nous avons constaté que dans la pratique, les graphes qui représentent les contraintes temporelles d'un document hypermédia sont très structurés et sont très proches de la structure des graphes série-parallèles. Ces graphes sont cependant une idéalisation des cas pratiques. Formellement, un graphe série-parallèle est un graphe qui ne peut pas être réduit à un graphe complet de 4 noeuds. Plus intuitivement, ces graphes peuvent être construits en appliquant itérativement l'une des opérations suivantes à partir d'un seul arc [6] :

- sérialisation : un arc est séparé en deux arcs consécutifs par l'introduction d'un noeud intermédiaire (cf. figure 10a),
- parallélisation : un arc est remplacé par deux arcs ayant les mêmes extrémités (cf. figure 10b).

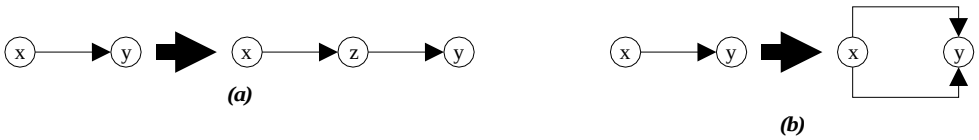


FIG. 10 – Construction d'un graphe série-parallèle. a) Sérialisation. b) Parallélisation.

Le tableau 5 présente des résultats pour ces graphes série-parallèles. Les temps de calculs sont significativement inférieurs à ceux des graphes quelconques. Cependant CPLEX tire mieux avantage de leur structure particulière que notre méthode. Une modification possible serait d'adapter la méthode de recherche des cycles et des cocycles à une structure série-parallèle.

| Dimension graphe | | CPLEX | | Conformité | |
|---|---|---|---|---|---|
| Noeuds | Arcs | Itérations | Temps | Itérations | Temps |
| 50 | 200 | 90 | 0,4 | 254 | 0,1 |
| 50 | 400 | 158 | 0,8 | 476 | 0,2 |
| 100 | 400 | 177 | 0,7 | 489 | 0,3 |
| 100 | 800 | 339 | 1,5 | 937 | 0,7 |
| 500 | 2000 | 912 | 4,2 | 2148 | 6,2 |
| 500 | 4000 | 1416 | 11,4 | 4197 | 13,6 |
| 1000 | 4000 | 1641 | 10,9 | 4079 | 23,8 |
| 1000 | 8000 | 2593 | 30,4 | 8127 | 51,3 |

TAB. 5 – Tension maximum = 1000.

# 4 Conclusion

Nous avons proposé ici une méthode rapide et facile à implémenter pour répondre au problème de trouver un ajustement réalisable et de la meilleure qualité possible. Cette méthode semble tout à fait compétitive. La répartition du calcul sur chaque arc du graphe temporel en fait une méthode attractive pour un réajustement en temps réel. Mais l'introduction dans le problème d'événements imprévisibles comme l'interaction avec l'utilisateur ou des retards dans le réseau ouvre la voie à de futures recherches.

Les graphes série-parallèles semblent un bon point de départ pour l'étude de problèmes plus compliqués comme l'introduction de durées discrètes pour les objets multimédia, ou une définition de la qualité différente comme le nombre d'objets qui n'ont pas leur durée idéale. La structure particulière de ces graphes facilite amplement la résolution des problèmes. Ces graphes ont été largement utilisés dans la littérature sur la théorie des graphes [6], mais c'est, à notre connaissance, la première fois qu'ils sont analysés dans ce contexte.

# Références


[1] R.K. AHUJA, T.L. MAGNANTI et J.B. ORLIN (1993) : *Network Flows - Theory, Algorithms, and Applications*. Prentice-Hall.

[2] J. ALLEN (1983) : *Maintaining Knowledge about Temporal Intervals*. Communications of the ACM, volume 26, numéro 11, pages 832-843.

[3] C. BERGE et A. GHOUILA-HOURI (1962) : *Programmes, jeux et réseaux de transport*. Dunod.

[4] M. CECELIA BUCHANAN et POLLE T. ZELLWEGER (1992) : *Specifying Temporal Behavior in Hypermedia Documents*. Proceeding of the European Conference on Hypertext '92, Milan, Italie, pages 262-271.

[5] M. CECELIA BUCHANAN et POLLE T. ZELLWEGER (1993) : *Automatically Generating Consistent Schedules for Multimedia Documents*. Multimedia System, Springer-Verlag, pages 55-67.

[6] R.J. DUFFIN (1965) : *Topology of Serie-Parallel Networks*. Journal of Mathematical Analysis and Applications, volume 10, pages 303-318.

[7] D.R. FULKERSON (1961) : *An Out-of-Kilter Method for Minimal Cost Flow Problems*. SIAM Journal on Applied Mathematics, volume 9, pages 18-27.

[8] MALIKA HADJIAT (1996) : *Problèmes de tension sur un graphe - Algorithmes et complexité*. Thèse de Doctorat, Université de la Méditerranée, Marseille, France.

[9] MICHELLE Y. KIM et JUNEHWA SONG (1995) : *Multimedia Documents with Elastic Time*. Multimedia '95, pages 143-154.

[10] G. MINTY (1961) : *Solving Steady-State Non Linear Networks of Monotone Elements*. IRE Transactions on Circuit Theory, pages 99-104.

[11] J.M. PLA (1971) : *An Out-of-Kilter Algorithm for Solving Minimum Cost Potential Problems*. Mathematical Programming, volume 1, pages 275-290.

[12] L. SABRY-ISMAIL, N. LAYAÏDA et C. ROISIN (1999) : *Dealing with Uncertain Durations in Synchronized Multimedia Presentations*. Multimedia Tools and Applications Journal, Kluwer Academic Publishers.

[13] W3C (1998) : *Synchronized Multimedia Integration Language (SMIL) 1.0 Specification*. W3C Recommendation. Disponible sur `http://www.w3.org/TR/REC-smil`.