Contenu multimédia à mémoire

R. Grigoraş, V. Charvillat, M. Douze
LIMA - IRIT – CNRS UMR 5505
ENSEEIHT, 2 rue Charles Camichel, 31071 Toulouse Cedex, France
{grig, charvi, douze}@enseeiht.fr

1 Introduction

La mise en ligne croissante de nouveaux contenus et services multimédias soulève le problème de la performance et de l'adaptabilité des applications déployées sur les réseaux actuels.

Bien entendu, ce contexte justifie des efforts de recherche dans les domaines suivants :

i. le codage objet, la compression, la scalabilité des contenus ;
ii. la synchronisation intra ou inter-flux, de groupe, logique, temporelle, discrète, continue ;
iii. la spécification de protocoles adaptés aux applications et aux contenus multimédias.

Mais au-delà des diverses technologies permettant de mesurer, d'apprécier, de négocier ou d'assurer la QoS, la tendance actuelle vise avant tout à poser le problème de la satisfaction des utilisateurs. C'est sans doute une lecture possible des travaux du groupe MPEG-21 [1]. Nous présentons une démarche cohérente avec ce point de vue.

Plutôt que de s'intéresser directement et indépendamment à chacun des points i, ii, iii précédents, nous proposons dans ce travail de donner aux contenus multimédia la possibilité d'apprendre comment *on se sert d'eux*, comment *on vient de se servir d'eux* ou comment *on se sert généralement d'eux*.
Cet apprentissage, basé sur la mémorisation des interactions subies par les contenus, permettra d'enrichir les techniques utilisées actuellement pour résoudre ces points jugés délicats. Notre approche bénéficie des travaux précédents traitant de la prédiction du comportement des utilisateurs et des sites web adaptatifs [2].

Nous décrivons donc l'intégration de nouveaux « flux élémentaires» au sein des contenus multimédias déjà structurés en flux indépendants dans l'approche MPEG-4 [3]. Ces flux correspondent à différents niveaux de mémorisation de l'histoire de l'utilisation d'un contenu. Ils permettent de résoudre certains problèmes critiques pour gérer la QoS (synchronisation, anticipation, hiérarchisation) et fournissent des idées intéressantes dans le cadre général de la description, de l'indexation, de l'accès et du requêtage de contenus multimédias.

Les contenus multimédias auxquels nous nous intéressons sont des documents hypermédias complexes et assez généraux (cf. Figure 1). Au niveau des expérimentations et des simulations nous privilégions dans ce papier l'étude de la navigation au sein d'hypervidéos.

2 Principe général

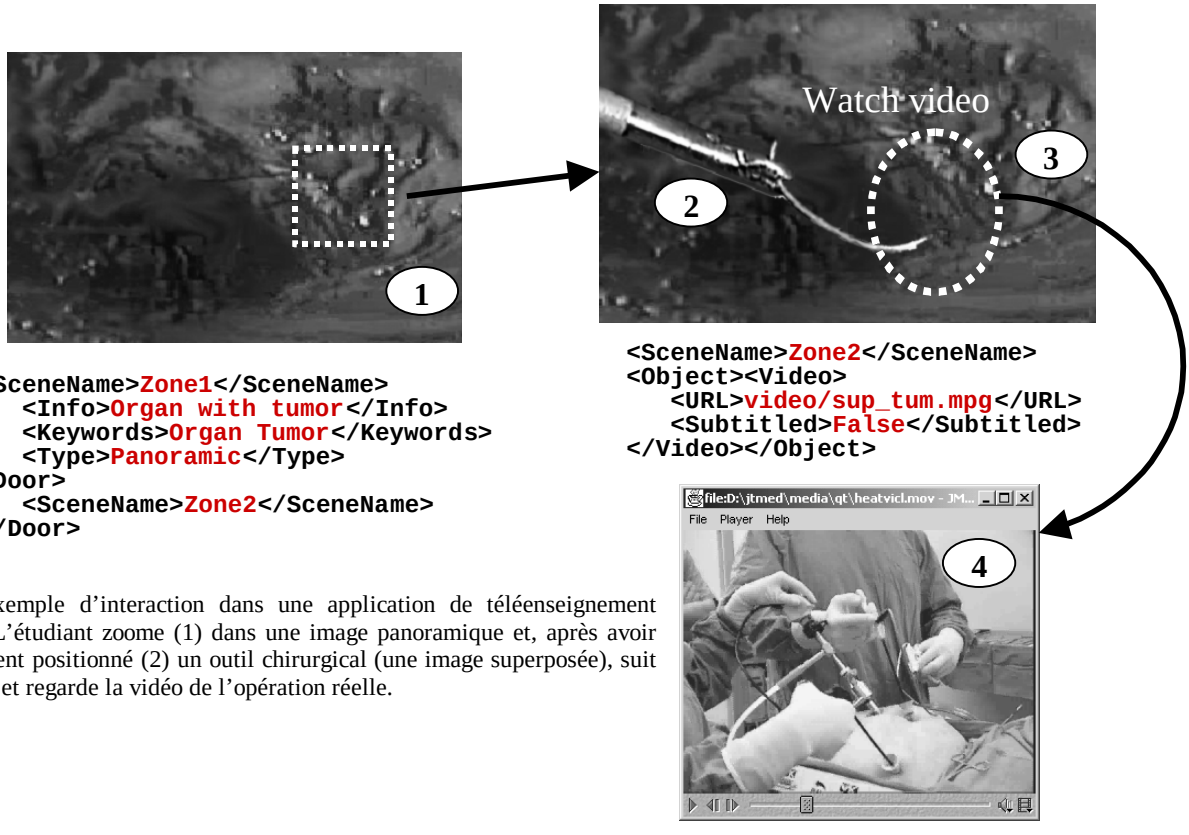


Fig. 1 Exemple d'interaction dans une application de téléenseignement médical. L'étudiant zoome (1) dans une image panoramique et, après avoir correctement positionné (2) un outil chirurgical (une image superposée), suit un lien(3) et regarde la vidéo de l'opération réelle.

Notre approche suppose que des utilisateurs similaires d'un contenu multimédia qui réalisent des suites similaires d'actions auront probablement des intentions semblables de navigation. Le système peut utiliser ceci pour anticiper les actions futures de l'utilisateur et offrir une navigation plus fluide.

2.1 Quelles interactions mémoriser ?

On peut enregistrer des variables appartenant aux domaines suivants:

- Audio/vidéo. Les actions comprennent stop/pause, accès ralenti/rapide à certaines séquences, demande d'enrichissement du contenu (zoom, augmentation de contraste), annotations partagées de certaines zones d'intérêt, changement de point de vue etc.
- Navigation hypermédia. Les actions peuvent être avance/retour, interruption d'un transfert (annulation de l'intention de suivre un lien) etc.
- Interactions 3D correspondantes aux senseurs VRML comme visibilité, orientation, toucher etc.
- Navigation dans des images panoramiques. Par exemple le zoom ou le changement d'orientation (QuickTimeVR [4]).

2.2 Comment les mémoriser ?

Au moins deux niveaux de mémoires peuvent être imaginés:

- *Mémoire a long terme* basée sur une analyse statistique (cumulative) des interactions des utilisateurs appartenant à la même « communauté ». Cette mémoire peut contenir des *événements influents*, des événements qui sont associés à un état d'utilisation du contenu au-delà duquel aucun utilisateur n'est allé ou à partir duquel les utilisateurs ne sont jamais revenus dans le contenu initial
- *Mémoire à court terme* contenant les actions les plus récentes de un ou plusieurs utilisateurs. Par exemple, si un utilisateur vient de suivre les n-1 premiers liens d'une page, il est très probable qu'il choisisse de suivre le n-ième lien. Un autre exemple : il est très probable qu'un utilisateur suive les n premiers liens d'une page si d'autres utilisateurs similaires (de la même communauté virtuelle ou de la même session) viennent de suivre ces liens.

3 De nouveaux besoins au-delà du web

3.1 Travaux précédents

Cette étude bénéficie des travaux précédents dans le domaine de la modélisation prédictive statistique (predictive statistical modelling, [7]) qui hérite à son tour de deux disciplines : la modélisation de l'utilisateur (user modelling) et l'apprentissage automatique (machine learning). La plupart des études précédentes ont presque exclusivement traité des systèmes hypertextes [9, 10, 11]. Le résultat de ses travaux permet d'identifier des agents Web de deux types :

- les systèmes de recommandation qui proposent à l'utilisateur des informations ou des facilités (fonctionnent généralement hors -ligne) et
- les systèmes qui réalisent des actions à la place de l'utilisateur (en ligne ou hors-ligne)

Plusieurs modèles ont été proposés, les plus utilisés étant : les chaînes de Markov discrètes ou continues [9, 10, 11] et l'exploration des séquences (sequence mining) [2, 9]. Ces modèles produisent les meilleures prédictions des requêtes de pages Web.

La solution d'un problème dans le domaine hypertexte n'en est pas nécessairement une pour l'hypermédia, et plus particulièrement pour l'hypermédia streamable (en flux continu). Par rapport au Web, l'hypermédia streamable doit gérer des objets multimédia de taille importante dont la présentation est soumise à des contraintes temporelles strictes. Actuellement on dispose d'une quantité importante de statistiques Web mais très peu de statistiques d'hypermédia streamable (cela est en train de changer, avec l'acceptation de standards comme SMIL et RTSP).

Prédire les actions futures n'est toutefois pas suffisant si on veut améliorer la fluidité de la navigation. Il faudrait en plus réaliser des actions appropriées. Il est naturellement question de faire du préchargement lorsqu'on prend en considération l'hypermédia streamable.

3.2 Notre approche

Un utilisateur navigue dans un contenu hypermédia et nous voulons prédire le contenu auquel il voudra accéder prochainement afin de lui offrir une navigation fluide. Pour réaliser cet objectif nous devons trouver

- un modèle pour décrire l'utilisation du contenu hypermédia (les interactions possibles)
- des algorithmes de prédiction des interactions futures et de préchargement des contenus associés.

Le préchargement permet de réduire les latences[1] tout en prévenant la sous-utilisation de la bande passante. La latence est un facteur de QoS important, qui provient de deux sources potentielles :

- les latences réseau, causées par la surcharge des serveurs ou des réseaux ou par les délais de propagation ;
- les latences de traitements multimédia qui incluent le temps de démarrage d'une présentation (analyse d'entête, décodage d'un minimum de flux, etc.)

[1] On peut facilement observer qu'on attend assez souvent des secondes avant de pouvoir jouer un contenu streamé avec RTSP. Cette latence inclut le temps nécessaire pour la connexion au serveur, la négociation des paramètres de transport, le préchargements d'une certaine quantité de données ainsi que le temps nécessaire au démarrage de la présentation.

Diminuer ces deux types de latence ne peut pas se faire sans introduire des coûts. Le préchargement réduit les latences réseaux mais un préchargement agressif implique une charge réseau accrue et un trafic en rafales [10]. Le prélancement d'une présentation hypermédia diminue les latences de traitement mais induit une charge importante sur la machine cliente.

Le présent article propose un mécanisme efficace pour réduire le premier type de latence. Une amélioration ultérieure du système prendra en compte également les latences de traitement.

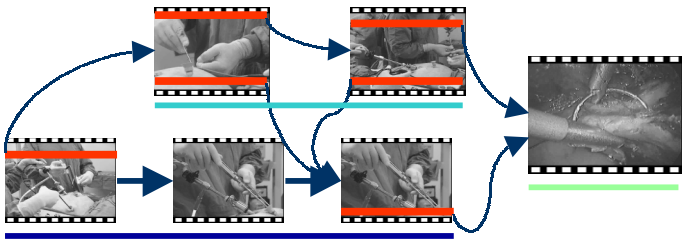
4 Modélisation intuitive

Nous nous intéressons dans cette section à un type précis de contenu multimédia : les hypervidéos. En outre, nous modélisons de manière intuitive et simple la prise en compte de la mémoire d'utilisation de ces vidéos.

4.1 Hypervidéos

Les hypervidéos [5,6] sont des ensembles structurés de vidéos liées entre elles (Figure 2). L'hypervidéo permet une navigation non-linéaire mais, à l'inverse de l'hypertexte, son contenu n'est pas statique. Des opportunités de sauter à d'autres vidéos (hyperliens) apparaissent et disparaissent avec le temps. Les hyperliens sont de deux types : temporels et spatio-temporels. Les liens *temporels* sont représentés par des annotations superposées à la vidéo pendant une certaine période. Les liens *spatio-temporels* exploitent la structure hiérarchique de la vidéo et associent des liens à certains objets à un certain moment.

Fig. 2 Exemple d'hypervidéo médicale contenant deux séquences



à moyenne résolution et une séquence courte à haute définition

4.2 Modèles de Markov

Intuitivement une navigation au sein de l'hypervidéo de la Figure 2 peut être modélisé par un graphe à 6 états. La prise en compte de la mémoire à long terme (statistiques de navigations) permet d'étiqueter chaque transition[2] d'un état vers ses successeurs avec une certaine probabilité. Ces éléments sont résumés dans la Figure 3.

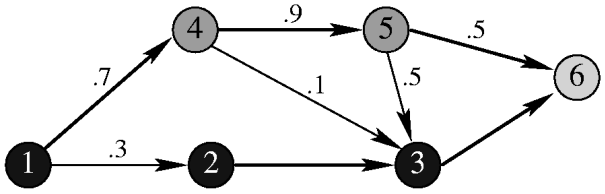


Fig. 3 Modélisation à états grossiers

On peut néanmoins imaginer d'autres types d'états allant du plus fin au plus grossier. Les états grossiers (Figure 3) mettent en évidence la structure logique du contenu. Par contre, des états plus fins (Figure 4) reflètent la structure interne des flux (typiquement un état = un GOP). Pour des raisons de simplicité mais aussi pour disposer d'un contenu mobile nous avons choisi la modélisation à états grossiers pour nos expérimentations. Disposer de moins d'états implique un contenu moins lourd et donc plus mobile, car les flux de mémoire sont indissociables du contenu.

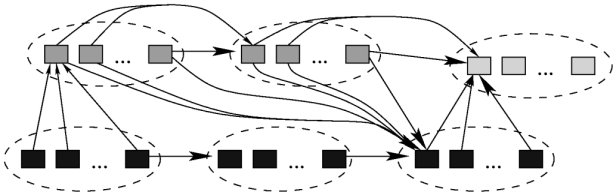


Fig. 4 Modélisation à états fins

En début de navigation seule la mémoire à long terme contient des informations, la mémoire à court terme est vide. Nous utilisons la prédiction markovienne afin de trouver la séquence des états les plus probables. L'utilisation d'un certain nombre d'heuristiques visant à modifier dynamiquement les probabilités des transitions permettra ensuite de prendre en compte la mémoire à court terme.

Les états ne seront plus des états de Markov au sens strict (sans mémoire), car nous intégrons aussi des informations de la mémoire à court terme. Typiquement, si un utilisateur suit un hyperlien (à partir d'un état donné), et s'il revient ensuite à cet état, la probabilité qu'il re-suive immédiatement le même lien devrait diminuer.

Malgré leur simplicité, l'utilisation des chaînes de Markov nous permet de montrer facilement l'intérêt du préchargement ou des heuristiques.

[2] Une transition correspond à un suivi d'un lien ou à un changement automatique de segment vidéo, en mode lecture, le long du parcours par défaut.

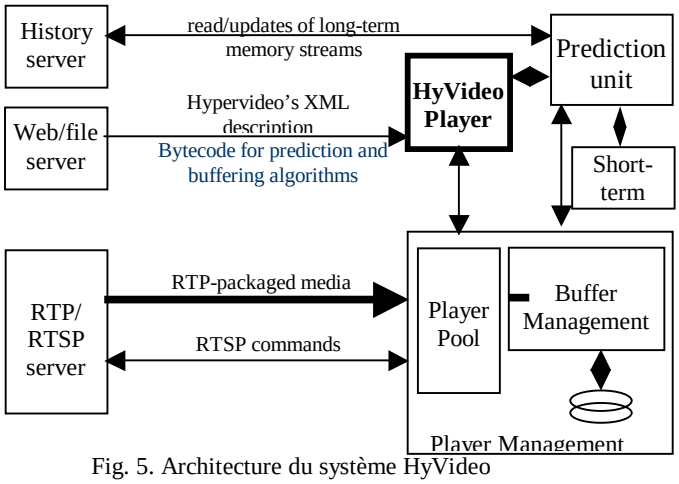


Fig. 5. Architecture du système HyVideo

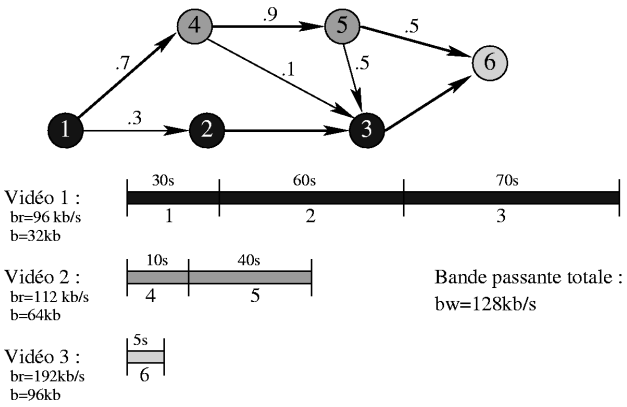


Fig. 6 Modélisation à états grossiers de l'hypervidéo de la figure 3. Les vidéos ont des débits (br - *bitrate*) et des minima b différents.

5 Premières experimentations

5.1 Contexte expérimental

Nous effectuons des simulations en Python et nous développons un prototype basé sur la plate-forme Java et ses extensions multimédia. Nous avons choisi comme domaines d'application la télévision interactive, la téléformation médicale et les visites virtuelles. Nous utilisons principalement deux types de contenu hypermédia : les *hypervidéos* et les *hyperpanoramiques*.

Dans la figure 5 nous présentons HyVidéo, une architecture client/serveur pour la navigation hypervidéo. Le client est basé sur un player JMF/RTSP. Il utilise des unités de prédiction et de gestion de tampons et il démarre les players de manière anticipée. Les informations de mémoire et le bytecode Java des algorithmes de prédiction sont fournies par un serveur de fichiers. Un serveur RTP/RTSP met en flux le contenu hypermédia.

5.2 Un exemple simple

Sur un modèle simple à 6 états représentant des segments de vidéos différentes (Cf. figure 6, br = le bitrate, b = quantité minimum de données nécessaire avant de pouvoir jouer) des simulations qui prennent en compte quelques politiques de préchargement montrent des réductions importantes des latences (Tableau 1).

Deux politiques de préchargement sont proposées :

- politiques P* (préchargement *proportionnel*). On précharge tous les états atteignables depuis l'état courant en allouant à chaque flux une bande passante proportionnelle aux probabilités associées aux états
- politiques B* (préchargement prioritaire pour l'état le plus probable (*best-first*))

Chacune de ses politiques dispose de deux versions. La version *conservative* (C) cherche à utiliser un minimum de bande passante et arrêter le préchargement d'un état dès que la quantité minimum b de flux nécessaire pour jouer est téléchargée. Par contre, la version *agressive* (A) continue le préchargement et cherche à utiliser toute la bande passante disponible.

La figure 7 illustre la politique proportionnelle conservative. Au moment initial, ne disposant pas de la quantité b_4 minimum pour commencer à jouer, l'utilisateur doit attendre que cette quantité soit téléchargée. Ensuite, pendant que la vidéo s'affiche, la bande passante restante est employée pour précharger, proportionnellement à leurs probabilités, les états suivants (5 et 3). La politique étant conservative, le préchargement s'arrête dès que la quantité b_3 est disponible.

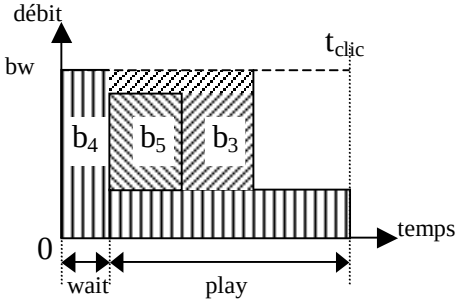


Fig. 7 : Exemple de politique PC appliquée pendant l'état courant 4 et en considérant que les états suivants ont des probabilités de 0.9 et 0.1

Le tableau suivant montre des latences calculées pour le chemin de navigation 1-4-5-6 en considérant que le moment auquel l'utilisateur décide de suivre le lien se situe à mi-longueur du segment vidéo courant.

Latence (en sec.)	**Pas de prédict.**	**PC**	**PA**	**BC**	**BA**
Précharge ment	**3.75** 1.25+2.5	**2.55** .8+1.75	**0.99** .25+.74	**2.25** .5+1.75	**0.85** .25+.6
Connexion	**0.9**	**0.3**	**0.3**	**0.3**	**0.3**
Total	**4.65**	**2.85**	**1.29**	**2.55**	**1.15**

Tableau 1. Latences calculées pour des simulations sans préchargement ou avec différentes politiques de préchargement

Pour ce parcours particulier (qui est d'ailleurs le « plus probable ») on observe une réduction significative des latences pour les quatre politiques. Afin de mieux comprendre pourquoi dans ces conditions une politique donne des meilleurs résultats qu'une autre nous présentons

séparément la somme des latences associées aux états 1,4,5 et la latence associée à l'état 6 seul. Les politiques *agressives* réussissent ainsi à précharger, pendant la présentation de l'état 5, une fraction importante de l'état 6, dont le débit dépasse la bande passante disponible.

Les latences de connexion, présentées séparément, sont non-négligeables[3]. Nous en déduisons qu'une réduction des latences réseaux est possible. On peut noter que, sans aller jusqu'au préchargement, la simple pré-initialisation des connexions est suffisante pour réduire les latences.

5.3 Utilisation des heuristiques

Des heuristiques peuvent réduire davantage les latences. En effet, le trajet déjà parcouru nous permet d'ajuster, d'une manière adaptative, la matrice des probabilités des transitions (prise en compte de la mémoire à court terme).

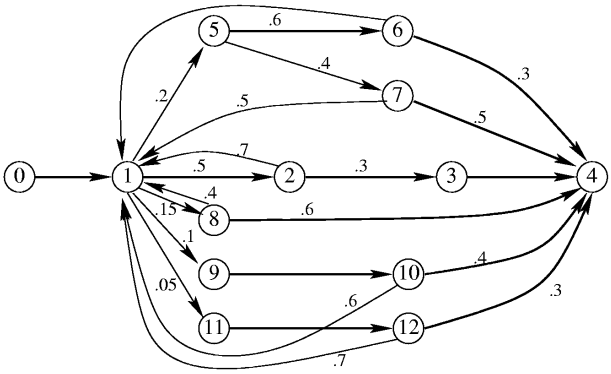


Fig. 8 : Exemple de navigation dans une hypervidéo comprenant une table de matières (état 1) renvoyant à plusieurs chapitres.

Dans un exemple plus complexe, dont le graphe de navigation est représenté dans la Figure 8, nous pouvons illustrer le fonctionnement d'une de ces heuristiques. Considérons par exemple le parcours de navigation 0, 1, 2, 1, 5, 7, 1, 8, 1, 9, 10, 1, 11, 12.

Une heuristique, qu'on appelle *BETA* par la suite, détecte une navigation exhaustive : après avoir suivi les trois premiers liens de l'état 1 (1->2, 1->5 et 1->8), les probabilités des liens 1->9 et 1->11 sont mises à jour (multipliées par un facteur β de valeur croissante). Ceci va favoriser le préchargement des données des états 9 et 11.

Tableau 1 Latences calculées pour l'hypervidéo de la figure 8, différentes politiques de préchargement et avec/sans utilisation de l'heuristique *BETA*

Politique	**PC**	**PA**	**BC**	**BA**
Sans *BETA* t_{clic}=0.1	3.81	3.80	3.88	3.87
Avec BETA t_{clic}=0.1	3.81	3.80	3.88	3.87
Sans *BETA* t_{clic}=0.5	6.10	4.71	5.91	5.91
Avec BETA t_{clic}=0.5	6.10	4.70	5.91	4.54

En conjonction avec la politique PA, l'heuristique *BETA* induit une petite réduction de la latence. Cette réduction est beaucoup plus appréciable si la politique BA est utilisée, (si le temps de clic est situé à mi-longueur du segment vidéo courant, cf. Tableau 2).

6 Nouvelles approches

Intuitivement on peut penser qu'il n'y a pas de meilleure politique globale. Les simulations le confirment (voir par exemple le Tableau 2). Il faudrait donc trouver un moyen pour choisir la bonne politique pendant la navigation. De plus, le temps de suivi d'un lien (ou le temps passé dans un état avant de suivre un lien) n'est pas pris en compte d'une manière satisfaisante. Une modélisation à l'aide de chaînes de Markov de 2nd ordre nous permet de considérer tous ces aspects et intégrer aussi une partie de l'histoire de navigation récente dans le modèle même.

6.1 Modélisation à l'aide des chaînes de Markov de 2nd ordre

Une chaîne de Markov de 2nd ordre permet une analyse plus fine des probabilités de transitions. Dans l'exemple de la figure 9, la probabilité de réaliser la transition 3->4, qui est de 0.6 (Markov 1er ordre), peut être « décomposée » dans une probabilité de 0.8 si on vient de l'état 1 et de 0.2 si on vient de l'état 2 (Markov 2nd degré). En d'autres mots, si on est dans l'état 3, on pourrait croire qu'il est plus probable d'aller en 4 qu'en 5 mais, sachant qu'on vient de 1, on a plus de chances d'aller en 5.

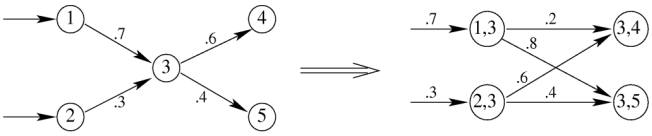


Fig. 9 Un ensemble d'états représenté à l'aide de chaînes de Markov de 1er et 2nd ordre.

Nous pouvons enrichir davantage notre modèle avec des informations temporelles[4]. Ainsi, à partir de la mémoire à long terme on peut extraire $\tilde{t}_{ij}$, le temps moyen passé dans l'état i avant de suivre le lien vers l'état j. Soit Π l'ensemble de politiques dont on dispose. Soit $L_{ij}(\pi)$ la latence calculée pour un suivi de lien de l'état courant i vers l'état j après un temps $\tilde{t}_{ij}$ et pour une politique $\pi \in \Pi$.

[3] Des expérimentations réalisées avec un serveur de streaming QuickTime et un client Java Media Framework/RTSP montrent un temps moyen de connexion d'environ 300 ms.

[4] Une modélisation ultérieure analysera l'opportunité d'utiliser des chaînes de Markov à temps continu si on dispose d'une distribution adéquate pour le temps de clic.

La latence potentielle totale correspondant à une politique donnée est donc $L_i(\pi)=\sum_j L_{ij}(\pi)$. Le système choisira la politique qui donne la plus petite latence :

$$\pi_o=\arg\min_{\pi\in\Pi}(L_i(\pi))$$

6.2 Récompense immédiate

Si la bande passante n'est pas gratuite ou si on veut l'utiliser avec parcimonie nous pouvons introduire une fonction de récompense. On suppose que le prix de la bande passante s'exprime en €/octet et qu'on peut aussi associer un coût en €/s à la latence.

La récompense associée à une politique est

$$R_i(\pi)=-L_i(\pi)-C_i(\pi)$$

où L et C sont les coûts associés respectivement à la latence et au préchargement. On choisit donc la politique qui maximise $R_i(\pi)$:

$$\pi_o=\arg\max_{\pi\in\Pi}(R_i(\pi))$$

7 Conclusion

Nous venons de présenter des contenus multimédia à mémoire. Des simulations et des expérimentations montrent l'utilité de la prise en compte de l'historique des interactions. La réduction des latences de navigation semble possible grâce au couple prédiction-préchargement.

Les utilisations possibles de ces flux de mémoire sont nombreuses :

- anticipation du chargement
- anticipation des rendus
- identification de priorités entre composants
- identification de caractéristiques de fiabilité et d'ordre pour les protocoles de transport
- recommandation d'utilisation par défaut

Plus généralement ce type d'approche permet d'imaginer un système adaptatif de navigation hypermédia qui tient compte de la structure hiérarchique des contenus multimédias, des besoins de synchronisation et surtout des comportements des utilisateurs.

Bibliographie

[1] MPEG-21, the "Multimedia Framework" MPEG Group, URL: http://www.cselt.it/mpeg.

[2] B. Trousse, *Evaluation of the Prediction Capability of a User behaviour Mining Approach for Adapative Web Sites*. In RIAO 2000, 6th Conference on Content-Based Multimedia Information Access, Collège de France, Paris, France, April 12-14, 2000.

[3] MPEG-4 standard, the MPEG Group, URL: http://www.cselt.it/mpeg/standards/mpeg-4/mpeg-4.htm

[4] QuickTimeVR, URL: http://quicktime.apple.com

[5] N. Sawhney, D. Balcom, I.Smith, *Authoring and navigating video in space and time : A Framework and Approach towards Hypervideo.* IEEE Multimedia, Vol. 4, No. 4, October / December 1997. URL: http://www.lcc.gatech.edu/gallery/hypercafe

[6] J. Tolva, *MediaLoom: An Interactive Authoring Tool for HyperVideo*, M.S. Project Paper, 1998

[7] I. Zukerman, and D. Albrecht, *Predictive Statistical Models for User Modeling*. User Modeling and User-Adapted Interaction, Anniversary Issue, 2001

[8] M. Crovella and P. Barford. *The Network Effects of Prefetching*. Proc. of Infocom'98, IEEE, April 1998.

[9] Y. Aumann et all, *Predicting Event Sequences: Data Mining for Prefetching Web-pages*, in Proc. of KDD'98

[10] K. Aberer, S. Hollfelder, *Resource Prediction and Admission Control for Interactive Video Browsing Scenarios Using Application Semantics*, GMD Report No. 40, September 1998

[11] D. Albrecht, I. Zukerman, A. Nicholson, *Pre-sending Documents on the WWW : A comparative Study*, Proc. IJCAI'99, August 1999.