Création automatique de résumés vidéo par programmation par contraintes

Haykel Boukadida

ANNÉE 2015

THÈSE / UNIVERSITÉ DE RENNES 1
sous le sceau de l'Université Européenne de Bretagne

pour le grade de

DOCTEUR DE L'UNIVERSITÉ DE RENNES 1

Mention : Informatique

École doctorale MATISSE

présentée par

Haykel Boukadida

préparée à l'unité de recherche Inria de Rennes - Bretagne Atlantique
Équipes d'accueil : Orange Labs/MAS - INRIA/LINKMEDIA

Création automatique de résumés vidéo par programmation par contraintes.

Thèse soutenue à Rennes le 04 Décembre 2015

devant le jury composé de :

Georges QUÉNOT
Directeur de recherche CNRS / rapporteur

Bernard MERIALDO
Professeur Eurécom / rapporteur

Alexandre TERMIER
Professeur Université de Rennes 1 / examinateur

Julien PINQUIER
Maitre de Conférences Université Paul Sabatier / examinateur

Nicolas BELDICEANU
Professeur École des Mines de Nantes / examinateur

Pascale SÉBILLOT
Professeur INSA de Rennes / examinateur

Patrick GROS
Directeur de recherche Inria / directeur de thèse

Sid-Ahmed BERRANI
Resp. équipe de recherche Orange Labs / Co-directeur

Á mes chers parents,
qui m'ont permis d'arriver jusqu'ici.

Á Emna,
qui m'a soutenu pendant ces années ...

Remerciements

Cette thèse est le fruit d'un travail nécessitant le concours de nombreuses personnes. Il me sera très difficile de remercier tout le monde et de n'oublier personne. C'est pourquoi je tiens à remercier par avance ceux dont le nom n'apparaît pas dans cette page.

En premier lieu, mes remerciements s'adressent à mes deux responsables de thèse. Je tiens à remercier mon directeur de thèse Patrick Gros, directeur de recherche Inria. J'ai pris un très grand plaisir à travailler avec lui, je lui suis reconnaissant pour le temps conséquent qu'il m'a accordé pendant les moments difficiles de la thèse, ses qualités pédagogiques et scientifiques, ses précieux conseils et sa sympathie. J'ai pu développer mes compétences et j'ai beaucoup appris à ses côtés. Mes remerciements s'adressent également à mon co-directeur de thèse Sid-Ahmed Berrani, responsable équipe de recherche à Orange Labs. Je le remercie pour son encadrement et son suivi. Je leur exprime ici ma gratitude pour toute l'aide qu'ils m'ont apportée au cours des différentes étapes de cette thèse.

Je tiens ensuite à remercier les membres du Jury pour avoir accepté la charge d'évaluer mon travail. Je remercie tout d'abord M. Alexandre Termier, professeur à l'université de Rennes 1, d'avoir accepté de présider mon jury de soutenance. Je remercie également M. Georges Quénot, directeur de recherche CNRS et M. Bernard Merialdo, professeur à Eurécom, qui ont accepté de rapporter cette thèse et dont les remarques et les suggestions ont participé à améliorer la qualité de ce mémoire. Je tiens enfin à exprimer ma gratitude à Mme. Pascale Sébillot, professeur à l'INSA de Rennes, M. Julien Pinquier, maitre de conférences à l'université Paul Sabatier et M. Nicolas Beldiceanu, professeur à l'école des Mines de Nantes, pour avoir examiné ce travail, ainsi que pour les discussions enrichissantes pendant la soutenance.

Je remercie également toutes les personnes que j'ai eu le plaisir de côtoyer pendant ces années. Côté Orange Labs R&D, mes remerciements s'adressent aux membres de l'équipe MAS, au sein de laquelle ce travail a été effectué, Patrick, Olivier, Nicolas, Franck, Patrice, Benoît et les autres, ainsi que les doctorants Khaoula, Alina, Sonia, Moez, Chiheb, Clément et Grigory. Un remerciement spécial à Philippe, avec qui j'ai eu le plaisir de partager le même bureau pendant la thèse. Côté INRIA, les séminaires organisés avec les membres de l'équipe LINKMEDIA ont été très enrichissants et m'ont permis parfois de prendre du recul sur ma thèse. J'adresse de chaleureux remerciements à Marc Christie, pour son aide durant les différentes étapes de la thèse. Je tiens aussi à remercier les membres de l'équipe TASC, plus particulièrement Charles et Jean-guillaume.

Enfin, mes remerciements vont à mes parents Hechmi et Karima qui étaient toujours à mes côtés, sans eux je n'en serais pas là aujourd'hui, ma femme Emna qui m'a toujours soutenu, mes frères et sœurs, Ihsen en particulier, ma belle-famille, mon oncle qui était présent avec ma petite famille à la soutenance, mes amis, Youssef, Houssem, Moussa et Faysal, et tous ceux qui m'ont supporté pendant ces années de thèse.

Résumé

Cette thèse s'intéresse à la création automatique de résumés de vidéos. L'idée est de créer de manière adaptative un résumé vidéo qui prenne en compte des règles définies sur le contenu audiovisuel d'une part, et qui s'adapte aux préférences de l'utilisateur d'autre part. Nous proposons une nouvelle approche qui considère le problème de création automatique de résumés sous forme d'un problème de satisfaction de contraintes. La solution est basée sur la programmation par contraintes comme paradigme de programmation.

Un expert commence par définir un ensemble de règles générales de production du résumé, règles liées au contenu multimédia de la vidéo d'entrée. Ces règles de production sont exprimées sous forme de contraintes à satisfaire. L'utilisateur final peut alors définir des contraintes supplémentaires (comme la durée souhaitée du résumé) ou fixer des paramètres de haut niveau des contraintes définies par l'expert.

Cette approche a plusieurs avantages. Elle permet de séparer clairement les règles de production des résumés (modélisation du problème) de l'algorithme de génération de résumés (la résolution du problème par le solveur de contraintes). Le résumé peut donc être adapté sans qu'il soit nécessaire de revoir tout le processus de génération des résumés. Cette approche permet par exemple aux utilisateurs d'adapter le résumé à l'application cible et à leurs préférences en ajoutant une contrainte ou en modifiant une contrainte existante, ceci sans avoir à modifier l'algorithme de production des résumés.

Nous avons proposé trois modèles de représentation des vidéos qui se distinguent par leur flexibilité et leur efficacité. Outre les originalités liées à chacun des trois modèles, une contribution supplémentaire de cette thèse est une étude comparative de leurs performances et de la qualité des résumés résultants en utilisant des mesures objectives et subjectives.

Enfin, et dans le but d'évaluer la qualité des résumés générés automatiquement, l'approche proposée a été évaluée par des utilisateurs à grande échelle. Cette évaluation a impliqué plus de 60 personnes. Ces expériences ont porté sur le résumé de matchs de tennis.

Mots clés : Analyse de vidéos, Création automatique de résumés vidéos, Programmation par contraintes.

Abstract

This thesis focuses on the issue of automatic video summarization. The idea is to create an adaptive video summary that takes into account a set of rules defined on the audiovisual content on the one hand, and that adapts to the users preferences on the other hand. We propose a novel approach that considers the problem of automatic video summarization as a constraint satisfaction problem. The solution is based on constraint satisfaction programming (CSP) as programming paradigm.

A set of general rules for summary production are inherently defined by an expert. These production rules are related to the multimedia content of the input video. The rules are expressed as constraints to be satisfied. The final user can then define additional constraints (such as the desired duration of the summary) or enter a set of high-level parameters involving to the constraints already defined by the expert.

This approach has several advantages. This will clearly separate the summary production rules (the problem modeling) from the summary generation algorithm (the problem solving by the CSP solver). The summary can hence be adapted without reviewing the whole summary generation process. For instance, our approach enables users to adapt the summary to the target application and to their preferences by adding a constraint or modifying an existing one, without changing the summaries generation algorithm.

We have proposed three models of video representation that are distinguished by their flexibility and their efficiency. Besides the originality related to each of the three proposed models, an additional contribution of this thesis is an extensive comparative study of their performance and the quality of the resulting summaries using objective and subjective measures.

Finally, and in order to assess the quality of automatically generated summaries, the proposed approach was evaluated by a large-scale user evaluation. This evaluation involved more than 60 people. All these experiments have been performed within the challenging application of tennis match automatic summarization.

Keywords : Video analysis, Automatic video summarization, Constraint satisfaction programming.

Table des matières

Table des figures

Liste des tableaux

Chapitre 1

Introduction générale

Sommaire

1.1 Contexte et motivations

La croissance rapide de la quantité de documents audiovisuels disponibles sur différentes plateformes nécessite le développement de nouveaux outils avancés et spécifiques pour leur stockage, indexation, recherche, exploration ou visualisation. Certains de ces documents peuvent être des vidéos autoproduites (des enregistrements personnels, des présentations, des entrevues...), tandis que d'autres proviennent des satellites, de dispositifs médicaux ou de caméras de sécurité, ainsi que du flux audiovisuel diffusé par des milliers de chaînes de télévision. En particulier, les programmes TV diffusés sont de plus en plus présents dans notre vie quotidienne, et leur volume ne cesse d'augmenter constamment et très rapidement. Ils sont maintenant disponibles via les catalogues de vidéo à la demande (Netflix, la VoD d'orange...). Ils peuvent également être récupérés à partir des services de rattrapage TV (MyTF1, Pluzz, M6Replay...).

Ces énormes quantités de contenu audiovisuel ont largement dépassé la capacité que possède l'être humain de les visualiser et ont rendu difficile la recherche de contenus intéressants pour un utilisateur. Le besoin d'outils efficaces permettant aux utilisateurs de choisir le contenu qu'ils vont regarder est donc manifeste. Ceci peut être réalisé en utilisant un moteur de recommandation et/ou de recherche à base de mots-clés ou de

similarité. Toutefois, en pratique, le contenu vidéo est très riche et très difficile à décrire finement en utilisant des caractéristiques sémantiques de haut niveau, et un moteur de recommandation ou de recherche peut être utilisé, dans un premier temps, pour filtrer et réduire l'espace de recherche à une courte liste de vidéos. Afin d'affiner la sélection et réaliser le choix final, l'utilisateur a besoin d'avoir un aperçu des vidéos présélectionnées qu'il souhaite regarder. Cet aperçu sera utile pour gagner un temps considérable et avoir une idée claire sur le contenu vidéo. Une solution à ce problème est donc de créer automatiquement un résumé de la vidéo sélectionnée. Le résumé vidéo permet de répondre à ce besoin en fournissant un aperçu général et rapide de l'ensemble du contenu audiovisuel de la vidéo originale et en présentant les parties intéressantes pour l'utilisateur.

En pratique, la création automatique de résumés de vidéos est une opération qui peut être utilisée dans un contexte professionnel comme pour un usage personnel :

Pour les professionnels, les résumés automatiques sont généralement utilisés pour parcourir des grandes bases de données vidéo et avoir rapidement un aperçu du contenu d'une vidéo. Celà est très important, notamment, pour les professionnels des médias ou pour les archivistes dans leur travail quotidien. Dans ce contexte, la création automatique de résumés fournit une assistance permettant aux professionnels d'économiser du temps et de réduire considérablement le coût de la recherche/annotation des vidéos. Les résumés générés automatiquement peuvent, également, être utilisés par les journalistes qui ont besoin d'avoir rapidement une idée claire sur ce qui a été diffusé dans les chaînes d'information par exemple.

Pour une utilisation personnelle, la création automatique de résumés permet à un utilisateur d'avoir un aperçu rapide d'une vidéo enregistrée sur son enregistreur vidéo personnel. Les résumés peuvent être utilisés, également, pour parcourir/récupérer des vidéos à partir d'une collection de vidéos personnelle. Ces vidéos ne sont généralement pas bien annotées. Leur utilisation ou la recherche d'une vidéo en particulier est très coûteuse en termes de temps. En outre, les résumés automatiques de vidéos sont utiles lorsqu'un utilisateur enregistre un flux TV via son terminal TV et ne dispose pas d'assez de temps pour regarder entièrement ce qu'il a enregistré. Dans ce cas, l'utilisateur peut demander au service de lui créer un résumé de l'enregistrement pour avoir une vue d'ensemble de ce qui s'est passé et, éventuellement, choisir la partie à regarder.

1.2 Problématique et objectifs

Plusieurs méthodes ont été proposées pour la création automatique de résumés de vidéos. L'objectif de certaines méthodes est de créer un résumé sous forme d'une vue d'ensemble sur la vidéo. Ce résumé est créé généralement par un expert indépendamment de ce qu'un utilisateur souhaiterait avoir. Cela consiste à présenter brièvement le contenu de la vidéo que ce soit pour avoir une idée sur le contenu audiovisuel ou bien pour aider un utilisateur à trouver une vidéo. D'autres méthodes prennent en considération des critères inférés globalement à partir du comportement observé des utilisateurs en terme général (les modèles d'attention par exemple). Ces résumés sont utilisés pour attirer l'attention d'un utilisateur potentiel. Les critères (tels que le mouvement des objets, le volume audio élevé...) permettent de repérer les parties de la vidéo susceptibles de correspondre à ce qui intéresse l'utilisateur ou à ce qu'un utilisateur souhaite voir dans le résumé.

Nous avons proposé une alternative à ces approches existantes qui consiste à créer un résumé personnalisable par l'utilisateur lui même. L'objectif de notre approche est d'assurer une consommation personnalisée des vidéos et de générer des résumés adaptés aux centres d'intérêts des utilisateurs. Nous considérons ainsi, qu'un résumé est dédié soit à une application cible soit à un utilisateur spécifique. Par exemple, un utilisateur souhaite avoir un résumé de deux minutes tandis qu'un autre voudrait un résumé de dix minutes. La capacité d'adaptation du résumé aux objectifs de l'application cible et aux attentes des utilisateurs est l'un des critères clés de l'efficacité d'une méthode de création automatique de résumés dans les applications du monde réel. En synthèse, notre méthode a pour objectif principal, non seulement de fournir une version plus courte présentant les parties intéressantes de la vidéo originale, mais aussi de produire un résumé adapté d'une vidéo donnée qui correspond aux préférences des utilisateurs.

Ainsi, pour qu'il soit utile dans la pratique, un résumé d'une vidéo devrait être créé en tenant compte des différentes règles de production relatives au **contenu vidéo** et aux **préférences de l'utilisateur** qu'il soit un **expert** ou un **utilisateur final**. La génération du résumé doit respecter des règles de production liées au type du contenu vidéo en question (un résumé d'un match de tennis ne partage pas les mêmes règles de production d'un résumé d'une émission musicale). Ce type de règles de production intervient au niveau du choix des segments vidéos à sélectionner et définit quel segment doit apparaître dans le résumé. Ces règles sont définies par un expert. Elles assurent que le résumé créé couvre les parties de la vidéo qui sont intéressantes et ce, pour un type de vidéo donné et pour un cas d'utilisation bien précis. En d'autre termes, elles impliquent

la structure finale du résumé à générer. Une analyse fine du contenu audiovisuel est requise dans ce cadre afin d'extraire les informations pertinentes à partir de différentes sources de données : images, audio et texte en utilisant différents algorithmes de détection d'attributs de bas niveau (visage, couleur, volume...). D'autres règles de production peuvent être définies, non pas par un expert, mais par l'utilisateur final lui-même. Un utilisateur peut spécifier ce qui doit être inclus dans le résumé en fonction de ses centres d'intérêts. Il est important de prendre en considération les préférences de l'utilisateur afin de produire un résumé adapté et personnalisé. Cela permet aux utilisateurs de demander (ou d'avoir en résumé) les parties de la vidéo qui les intéressent le plus (un utilisateur souhaite avoir un résumé qui couvre la fin d'une vidéo plutôt que son début, un autre souhaite avoir un résumé qui ne contient que les scènes d'action). D'autre part, l'utilisateur final peut définir un ensemble de paramètres exploités par les règles de production décrites précédemment (tel que la durée du résumé, la proportion de reportages et de commentaires par rapport à la durée totale d'un résumé d'un journal télévisé...). Ces paramètres reflètent la forme souhaitée du résumé et permettent de s'adapter, au mieux, aux préférences de l'utilisateur. En synthèse, l'adaptabilité du résumé généré est un critère important qui, comme on le verra plus tard, a reçu très peu d'attention dans les techniques existantes.

Les méthodes existantes de création automatique de résumés sont en quelque sorte des boites noires qu'il est compliqué de modifier pour un utilisateur. C'est l'un des problèmes qui se posent et que nous abordons dans cette thèse. Ces méthodes présentent à la fois l'algorithme de génération du résumé et la forme souhaitée de ce dernier (règles de production) sans aucune séparation entre les deux. Les règles de production sont totalement intégrées dans l'algorithme de génération du résumé. Cet aspect rend une éventuelle modification d'un résumé (par l'ajout ou la modification d'une règle) très difficile et nécessite des modifications sur tout le processus de création du résumé.

Dans cette thèse, nous abordons une autre problématique importante, c'est l'évaluation de la qualité des résumés générés automatiquement. L'aspect subjectif de la tâche rend difficile la définition d'un **bon résumé**. L'importance donnée à une information est souvent subjective, l'évaluation des résumés en est donc d'autant plus difficile. De nombreux résumés sont possibles pour une même vidéo. Deux professionnels prenant en considération les mêmes objectifs et la même application cible peuvent créer deux résumés différents. De plus, les formes attendues d'un résumé varient en fonction des objectifs et des domaines d'application. Le résumé d'un match de tennis n'est pas construit de la même façon qu'une bande-annonce de film. Un autre problème lié à l'évaluation de la qualité des résumés générés automatiquement concerne donc le choix des mé-

triques d'évaluation : peut-on seulement comparer les segments de la vidéo originale présents dans le résumé ? Cela soulève de nombreuses difficultés pour les chercheurs qui travaillent sur les techniques de création automatique de résumés.

Pour résumer, l'objectif de cette thèse est d'aborder le problème de la création de résumé automatiques de vidéos en considérant trois critères clés :

1. la capacité d'adaptation du résumé par rapport à la demande et aux attentes des utilisateurs ;
2. la séparation entre l'algorithme de génération du résumé et les règles de production impliquant sa forme souhaitée ;
3. l'évaluation de la qualité des résumés produits.

1.3 Contributions de la thèse

Dans cette thèse, nous nous sommes, dans un premier temps, focalisés sur l'adaptabilité des résumés. Les règles de production, décrites dans la section précédente, peuvent être exprimées sous forme de contraintes à satisfaire. Par conséquent, nous proposons de considérer le problème de création automatique de résumés comme un problème de satisfaction de contraintes (PSC), où la façon avec laquelle le résumé doit être créé est définie par un ensemble de contraintes.

Pour répondre aux besoins d'adaptation des résumés créés automatiquement, nous avons proposé une approche basée sur la programmation par contraintes (PPC). La PPC a pour objectif de trouver une ou plusieurs solutions qui satisfont des contraintes exprimées en utilisant un solveur de contraintes. La PPC vient de l'intelligence artificielle, c'est un outil puissant et efficace pour la résolution des problèmes NP-complets et pour la modélisation et la résolution des problèmes d'optimisation combinatoire. Elle a été utilisée pour résoudre les problèmes dans différents domaines, comme les logiciels d'optimisation pour le trafic aérien et la gestion des portes d'embarquement, la construction de véhicules, la planification d'horaires, la gestion de la rotation du personnel, etc.

Un des avantages de l'approche proposée est qu'elle assure une séparation claire entre la modélisation et la résolution du problème. En d'autres termes, entre les règles de production du résumé et l'algorithme de génération du résumé (ici, le solveur de contraintes). Différents types de résumés peuvent être générés simplement, en ajoutant de nouvelles contraintes ou en fournissant des paramètres de haut niveau pour les contraintes existantes. Notre méthode de création automatique de résumés n'est donc

pas une boîte noire qui doit être complètement revue si l'utilisateur souhaite adapter le résumé. Notre approche est souple et extensible, dans le sens où elle permet à l'utilisateur de configurer facilement le résumé final en fonction de ses besoins en ajoutant/modifiant une contrainte sans être amené à revoir l'ensemble du modèle ou à modifier n'importe quel paramètre interne. La figure 1.1 met en évidence cette séparation :

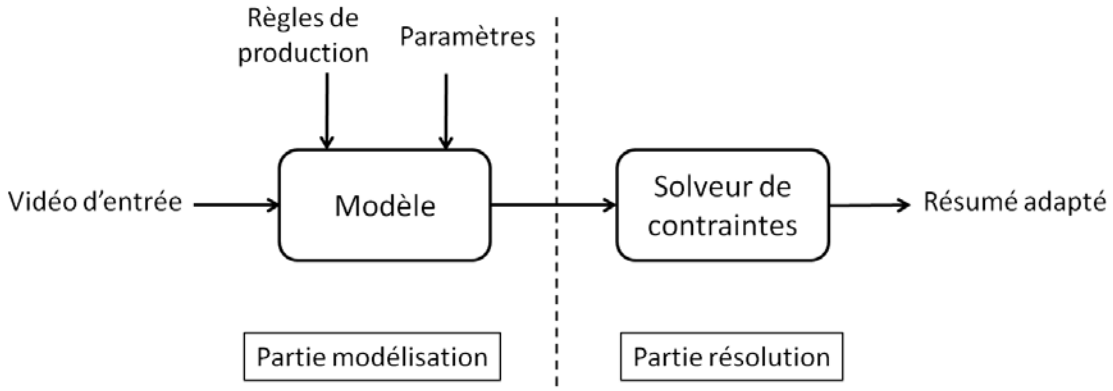

FIGURE 1.1 : Séparation entre la partie modélisation et la partie résolution du poroblème.

D'autre part, la programmation par contraintes permet de spécifier des contraintes et une fonction de coût à optimiser. Les contraintes peuvent être utilisées pour modéliser les règles de production qui doivent être obligatoirement respectées. Tandis que la fonction de coût à optimiser peut être utilisée pour minimiser ou maximiser une grandeur spécifique. En d'autres termes, cette fonction définit ce qu'il est souhaitable d'avoir (ou pas) dans le résumé final, sans que cela soit obligatoire.

En s'appuyant sur la programmation par contraintes, nous proposons trois modèles pour la création automatique de résumés de vidéos. Ces modèles se distinguent par leur flexibilité, c'est à dire leur capacité d'exprimer les règles de production du résumé. Ils se distinguent aussi par leur efficacité, c'est-à-dire leur capacité à trouver rapidement des solutions au problème de satisfaction de contraintes proposé. Aborder l'ensemble de ces aspects constitue les deux premières contributions de cette thèse (adaptation des résumés et séparation entre les règles de production et l'algorithme de génération du résumé).

La troisième contribution de cette thèse concerne la procédure d'évaluation de la qualité des résumés générés automatiquement. Nous avons évalué la méthode proposée, tout d'abord, en utilisant des mesures objectives et, ensuite, en lançant une campagne d'évaluation à large échelle effectuée par les utilisateurs.

Différents types de résumés ont été générés/récupérés sur un ensemble de 12 matchs de tennis (environ 28,5 heures de vidéo). L'évaluation a été réalisée avec la collabora-

tion de 61 personnes. Les résultats obtenus ont été soigneusement analysés en utilisant non seulement des mesures quantitatives, mais aussi des tests statistiques. Au-delà de l'évaluation de notre méthode, cette étude impliquant les utilisateurs nous permet de proposer des recommandations importantes concernant le protocole d'évaluation dans ce contexte.

1.4 Organisation du manuscrit

Le reste du présent manuscrit s'organise en six chapitres, qui s'articulent autour de deux parties :

Partie I - État de l'art : Les chapitres 2 et 3 présentent l'état de l'art des différents domaines de recherche afférents à cette thèse :

- Le chapitre 2 s'intéresse, dans une première partie, aux différentes approches de création automatique de résumés proposées dans les travaux existants. Ces travaux sont classés en quatre grandes catégories selon le principe sur lequel ils sont fondés et selon la forme souhaitée du résumé résultant. Nous présentons, également, quelques travaux proposés dans le cadre de la compagne TRECVID. Dans une deuxième partie de ce chapitre, nous nous intéressons aux différentes méthodes d'évaluation de la qualité des résumés générés, proposées dans le cadre de TRECVID et dans un contexte général. Nous présentons quelques protocoles et quelques critères d'évaluation. Nous examinons comment les travaux existants ont abordé ce sujet.

- Le chapitre 3 présente quant à lui, le paradigme de la programmation par contraintes. Nous rappelons quelques définitions des problèmes de satisfaction et d'optimisation de contraintes, des méthodes et des stratégies de résolution. Nous terminons par un exemple illustratif d'un problème de satisfaction de contraintes, nous présentons quelques modèles possibles utilisés pour sa résolution.

Partie II - Contributions : Les chapitres 4, 5 et 6 détaillent les différentes contributions de cette thèse.

Le chapitre 4 présente d'abord deux modèles différents du problème de création de résumés vidéo basés, tous les deux, sur une segmentation en plans de la vidéo d'entrée. Nous introduisons les modèles basés sur la programmation par contraintes, leurs avantages et leurs limites. Nous présentons une étude expérimentale sur l'efficacité des

modèles ainsi que sur la qualité des résumés générés. Ensuite, dans le chapitre 5, nous introduisons un nouveau modèle qui s'affranchit de toute pré-segmentation du flux audiovisuel et qui ne dépend d'aucune frontière de plan. Pour cela, nous avons été amenés à implémenter l'ensemble des relations de l'algèbre d'intervalles de Allen. Nous présentons le principe de ce modèle. Nous décrivons les nouvelles contraintes globales proposées et implémentées avec leurs algorithmes de filtrage. Nous présentons une étude comparative avec les modèles du chapitre 4 ainsi qu'une évaluation de la qualité des résumés résultants. Enfin, le chapitre 6 explique les difficultés liées à l'évaluation de la qualité des résumés générés. Nous présentons, par la suite, notre expérience menée par des tests utilisateurs ainsi que le protocole d'évaluation proposé.

Le dernier chapitre de ce manuscrit est enfin consacré aux conclusions, en dressant un bilan critique des principales contributions de cette thèse, et en proposant quelques pistes de travaux futurs ainsi qu'une liste des publications associées aux travaux de cette thèse.

Première partie

Définitions et état de l'art

Chapitre 2

Création automatique de résumés vidéo

Sommaire

2.1 Introduction

De nombreux travaux de recherche dédiés à la création automatique de résumés de vidéos ont été proposés [MA08, RK14]. Dans les travaux existants, les résumés de vidéos peuvent être de deux types. Le premier consiste à extraire un ensemble d'images, le deuxième consiste à créer une nouvelle vidéo à partir de la vidéo d'entrée. Le premier type, généralement appelé résumé statique de la vidéo, est une collection d'images représentatives qui sont soigneusement extraites à partir de la vidéo. Ces images sont appelées images-clés. Chacune d'entre elles représente le contenu visuel d'une partie

de la vidéo. La visualisation de ce type de résumé est rapide et la complexité de son algorithme de création est généralement faible. Le second type, également connu sous le nom de résumé dynamique de vidéo, est un ensemble de segments vidéo sélectionnés à partir de la vidéo d'entrée. Ce type de résumés conserve les propriétés dynamiques de la vidéo d'entrée et par conséquent il est plus agréable à regarder qu'un résumé statique. Il est également plus expressif, car il comprend à la fois de l'information visuelle et audio. Mis à part quelques cas d'utilisation spécifiques où la sélection d'un ensemble d'images représentatives est suffisante, un résumé sous forme d'une vidéo est généralement plus utile dans la pratique. Même si l'approche que nous avons proposée peut s'appliquer pour générer, à la fois, les résumés statiques et les résumés dynamiques de vidéos, nous avons mis l'accent, dans ce chapitre, plutôt sur ce dernier type de résumés (résumés dynamiques).

Dans ce chapitre, nous décrivons les techniques existantes de création de résumés vidéo regroupés en quatre catégories sur la base de leurs principes. La dernière section de ce chapitre est dédiée au problème de l'évaluation de la qualité des résumés générés.

2.2 Approches basées sur les modèles d'attention

L'objectif de ces approches est de simuler et de modéliser l'attention des utilisateurs pour créer le résumé vidéo. Ces approches procèdent généralement en trois étapes.

Segmentation : une première étape consiste à segmenter la vidéo en un ensemble d'unités de base. Une unité de base peut être temporelle (une seconde, plusieurs secondes, une minute, ...). Elle peut être sous forme de segments (plans, scènes, ...) ou même d'une image.

Calcul de scores et création de courbes : la deuxième étape consiste à calculer un score pour chaque unité de base de la vidéo d'entrée. Ce score reflète *l'importance* qui peut être attribuée à l'unité de base et donne une indication sur la probabilité pour qu'elle soit sélectionnée et incluse dans le résumé. En pratique, les scores sont calculés en se basant sur des algorithmes de détection de caractéristiques de bas niveau (le volume de l'audio, l'intensité de mouvement, la luminosité...). Ces caractéristiques permettent d'identifier des moments susceptibles d'attirer l'attention humaine. Les scores calculés sont utilisés par la suite pour créer une courbe modélisant l'attention des utilisateurs. Par exemple, l'attention de l'utilisateur est souvent captée par des événements visuels, acoustiques ou textuels. Par conséquent, des caractéristiques visuelles, acoustiques et textuelles sont détectées pour

créer respectivement des courbes d'attention visuelles, acoustiques et textuelles. Une courbe de synthèse modélisant l'attention humaine est obtenue par la fusion de ces différentes courbes.

Sélection d'extraits : la dernière étape consiste à analyser la courbe d'attention finale et à sélectionner les extraits à inclure dans le résumé. La sélection d'extraits est basée sur la détection des images (ou des séquences) qui correspondent aux pics de la courbe moyennant l'application d'un seuil (les extraits ayant des scores supérieurs à ce seuil sont sélectionnés) ou l'utilisation d'un intervalle de temps (des extraits d'une durée de 30 secondes ayant un pic au centre sont sélectionnés).

Dans [LYG+08, LL09], la génération de résumés est basée sur la création d'un modèle d'attention visuelle uniquement. Le modèle proposé est calculé en utilisant des histogrammes de couleurs et de luminosité qui servent à détecter les régions d'intérêt dans une image (personnes ou objets). Cette méthode est limitée puisqu'elle n'utilise ni l'information audio ni l'information textuelle pour générer les résumés vidéo. Cette limitation a été traitée en partie par Evangelopoulos et al. [ERP+08] qui ont proposé un modèle de saillance audiovisuelle en combinant deux modèles d'attention visuelle et acoustique. Le modèle de saillance acoustique est créé en se basant sur l'extraction des composantes AM-FM (modulations temporelles d'amplitude et de fréquence) du signal audio. Le signal audio est modélisé par la somme du changement de fréquence et d'amplitude. Les structures saillantes sont alors les signaux de modulation sous-jacents. Quant au modèle de saillance visuelle, il est créé en se basant sur la couleur, l'intensité de l'image et le mouvement des objets. Les deux modèles sont combinés en un modèle de saillance audiovisuelle M_{av} définit par :

$$M_{av} = w_a.M_a + w_v.M_v \tag{2.1}$$

où w_a et w_v sont les poids attribués respectivement au modèle de saillance acoustique M_a et au modèle de saillance visuelle M_v. Les poids sont déterminés par le moindre carré de leurs valeurs individuelles dans l'intervalle $[0, 1]$ suivant la normalisation des courbes.

Les mêmes auteurs ont proposé une extension de leurs travaux en introduisant, en complément des modèles de saillance visuelle et acoustique déjà décrits, un modèle de saillance textuelle [EZS+09]. L'information textuelle est récupérée à partir de la transcription de la piste audio. Pour cela, le flux audio est segmenté en utilisant la technologie de reconnaissance automatique de la parole. Les segments audio sont délimités par l'image de début et l'image de fin de chaque mot. Les scores sont attribués pour

chaque mot en utilisant l'étiquetage morpho-syntaxique (*POS tagger : Part-of-speech tagger*). L'équation 2.1 est étendue et utilisée pour la fusion des trois modèles de saillance (visuelle, acoustique et textuelle).

Une étape très importante dans le processus de création de résumés vidéo en utilisant les modèles d'attention, est la fusion des modèles d'attention partiels donnant naissance à un modèle d'attention final. Dans [MHL+05], la combinaison des différents modèles d'attention partiels peut être effectuée, soit par des systèmes de fusion linéaires (donnés par l'équation 2.1) où un poids est affecté à chaque modèle calculé (les poids sont ajustés en fonction de l'importance du modèle), soit par une combinaison non-linéaire [HZ05] pour obtenir de meilleurs résultats de décision en ce qui concerne le processus de fusion.

Dans [NMZ05], Ngo et al. ont introduit une approche basée sur la construction de graphes. Cette approche est décomposée en deux étapes. La première étape consiste à classer les plans en utilisant l'algorithme de coupes normalisées (*normalized cuts algorithm*) [SM00]. Dans la deuxième étape, la vidéo est représentée sous la forme d'un graphe temporel orienté de plans. Le graphe est similaire au graphe STG (*scenes transition graph*) présenté dans [YYW+95]. Les valeurs de l'attention basée sur le mouvement sont attribuées à chaque nœud du graphe. Le résumé est généré à partir du graphe en tenant en compte à la fois la structure de la vidéo et les valeurs de l'attention.

Par ailleurs, un autre travail original a été proposé dans [LGF+10]. Les auteurs décrivent un processus expérimental qui applique l'imagerie par résonance magnétique fonctionnelle (fMRI) pour étudier la dynamique et l'interaction entre le flux multimédia et la réponse du cerveau humain pour estimer les poids utilisés dans la fusion des modèles. Ils ont identifié 36 régions d'intérêt dans le cerveau humain à partir desquelles une série de signaux est extraite et utilisée pour modéliser la réponse du cerveau.

Dans [LK03], des scores sont calculés pour chaque image de la vidéo d'entrée. Chaque score est associé à un attribut tel que l'occurrence des visages, l'occurrence du texte, le *zooming* de la caméra et le volume audio. Des conditions sont exprimées par un expert sur ces différents attributs sous forme d'un ensemble d'inégalités. Les inégalités impliquent les attributs et un ensemble de seuils fixés par l'expert. Le résumé final est composé des images qui satisfont toutes les conditions exprimées. Pour éviter une segmentation excessive, un algorithme glouton est utilisé pour maximiser le score total et réduire le nombre d'extraits sélectionnés à partir de la vidéo d'origine.

Dans [GGR+14], Gygli et al. ont proposé une approche de création de résumés de vidéos personnelles. La vidéo est tout d'abord segmentée en se basant sur le mouvement dans les images. L'importance d'un segment est ensuite estimée en utilisant un ensemble d'attributs de bas et de haut niveau tels que la couleur, la luminosité, la dé-

tection de monuments, de personnes et le suivi des objets. Enfin, en utilisant les scores reflétant l'attention humaine pour chaque segment, le problème de création du résumé est formalisé sous forme d'un problème de sac à dos (*Knapsack problem*) qui maximise l'importance totale dans le résumé.

Bien que les approches basées sur des modèles d'attention soient simples et offrent un bon formalisme pour introduire et résoudre le problème de création de résumés vidéo, les résultats obtenus présentent des limitations. Les caractéristiques de bas niveau détectées sont essentiellement pondérées et additionnées pour construire la courbe modélisant d'attention humaine, ce qui ne garantit pas forcément une corrélation avec les intérêts réels de l'utilisateur. D'autre part, ces approches impliquent trop de paramètres fixés de façon heuristique ou bien empirique. Ces paramètres sont difficiles à mettre en place, en particulier les cœfficients (poids) de pondération des différents caractéristiques de bas niveau ainsi que le seuil final appliqué sur la courbe d'attention résultante. Enfin, une fois que le modèle est formé, il est très difficile de le modifier afin de prendre en considération un nouveau type de vidéo ou un nouveau critère (un nouveau attribut de bas niveau par exemple visage, volume,...).

2.3 Le résumé sous forme d'une vue d'ensemble

D'autres travaux visent à créer des résumés sous forme d'aperçus utilisés pour donner aux utilisateurs une idée générale sur l'ensemble du contenu vidéo.

Une façon triviale de créer des résumés sous forme d'aperçu de la vidéo est de l'accélérer. Le principe consiste à condenser la vidéo originale en accélérant tout simplement sa lecture. Le système *Video Cue* proposé dans [PAS+99] est un exemple qui adopte une vitesse de lecture plus rapide lors de la lecture de la vidéo pour produire une vue d'ensemble. Bien que le temps de visualisation soit réduit, les propriétés de la vidéo sont déformées et la compréhension de l'audio est affectée. Une autre façon triviale de créer ce genre de résumés (vue d'ensemble) est d'ignorer des images à partir de la vidéo d'origine [LSK05], ce qui permet de réduire la durée de la vidéo, mais en contre partie, permet d'introduire des distorsions. Ce compromis est formulé en utilisant une optimisation *MINMAX* qui minimise la distorsion maximale par segment.

D'autres approches sont basées sur la mesure de la similarité entre les différentes parties de la vidéo et l'élimination de la redondance. Afin de sélectionner des images représentatives qui soient assez différentes les unes des autres et qui représentent bien la totalité du contenu de la vidéo, une comparaison de toutes les images de la vidéo entre elles, est effectuée.

La méthode proposée dans [CF02], permet de sélectionner des extraits qui maximisent la similarité moyenne entre l'extrait sélectionné et le reste de la vidéo. La sélection des extraits est basée sur le calcul de la factorisation non-négative d'une matrice de similarité. Le résumé final peut être généré sous forme d'une combinaison de plans, comme dans la méthode proposée dans [GL01]. Les plans sont regroupés en un ensemble de *clusters* en fonction de leur similarité visuelle. Le plan le plus long de chaque *cluster* est retenu pour représenter celui ci. Dans d'autre méthodes, le résumé final peut être généré sous forme d'une combinaison de sous-plans, comme dans [NT99]. La méthode proposée découpe la vidéo en un ensemble de sous-plans en calculant l'activité de mouvement local. Une fois les sous-plans détectés, l'algorithme proposé calcule l'indice de l'intensité de mouvement pour chaque sous-plan et ne retient que ceux qui ont l'indice le plus élevé. L'indice de l'intensité de mouvement correspond à la quantité de l'activité visuelle au sein de chaque sous-plan. Il est calculé en appliquant la fonction de filtrage de transformée en ondelettes 1D.

Dans [Gon03], les plans sont classés en utilisant un arbre couvrant de poids minimal (*minimum spanning tree* en anglais) où les nœuds représentent les plans de la vidéo et chaque arête représente la distance entre deux plans dans l'espace des attributs. Un graphe biparti est utilisé pour éliminer les doublons et garder les plans qui sont visuellement distincts. Un résumé vidéo est ainsi créé. Un autre résumé est créé en utilisant la technique d'analyse sémantique latente (LSA, *Latent semantic analysis* en anglais) appliquée sur les transcriptions de parole. Un algorithme d'alignement audiovisuel à base de graphes est utilisé pour aligner les deux résumés. Plusieurs autres méthodes basées sur des graphes orientés sont proposées notamment dans [LLK04, LKL05]. Les nœuds correspondent aux différents plans de la vidéo. À chaque nœud est associé un poids qui est la durée du plan correspondant. Les arêtes du graphe représentent la distance entre les plans (calculée par une fonction combinant la similarité visuelle et la distance temporelle). Étant donné que la durée totale du résumé est la somme des poids des nœuds (plans à sélectionner), les auteurs ont proposé de trouver le plus long chemin du graphe ayant la somme des poids des nœuds appartenant à un intervalle de tolérance $[dmin..dmax]$.

L'algorithme TV-MMR (*Text Video Maximal Marginal Relevance*) proposé dans [LMR+11] est basé sur le calcul de la similarité visuelle en plus de l'information textuelle pour la création des résumés vidéo. L'algorithme permet de sélectionner, de manière itérative, des images dont le contenu visuel est le plus similaire au contenu vidéo, mais en même temps le plus différent des images déjà sélectionnées dans le résumé. L'information textuelle est récupérée à partir de la transcription de la piste audio en utilisant le

système ASR *Automatic Speech Recognition*. Le résumé est le résultat de la concaténation d'un ensemble de segments audiovisuels courts de durée prédéfinie, sélectionnés à partir de la vidéo d'entrée. Le résumé optimal est le résumé qui maximise la quantité totale du contenu visuel tout en préservant la cohérence de l'information textuelle.

Dans [DR14], le problème de création automatique de résumés vidéo est considéré comme un problème de sélection d'un ensemble optimal d'extraits qui minimise la distance entre la vidéo d'entrée et le résumé généré. Cette distance est mesurée en utilisant une courbe d'indices HIP (*Heterogeneity Image Patch*) qui représente la dissemblance entre la vidéo d'entrée et le résumé vidéo. Le calcul des valeurs des indices HIP est basé sur les pixels et ne nécessite pas l'information sémantique ni l'estimation complexe du mouvement de la caméra.

Selon les travaux présentés dans [STC10], un résumé vidéo doit répondre à deux critères essentiels qui sont « la couverture » : le résumé doit représenter la totalité de la vidéo d'origine et « la diversité » : les différentes parties du résumé sont assez distinctes et assez différentes. Un aperçu rapide de la vidéo est fourni en sélectionnant des extraits à partir de la vidéo qui maximisent « la couverture » et « la diversité ». Les auteurs ont proposé une formulation mathématique de ces deux critères et une fonction de coût à optimiser.

Ces approches permettent de couvrir la totalité du contenu audiovisuel et de fournir une vue d'ensemble de la vidéo, mais ils ne garantissent pas que le résumé généré puisse attirer l'attention des utilisateurs.

2.4 Approches basées sur l'extraction d'événements intéressants

Une autre façon de créer des résumés vidéo est de détecter les moments forts (appelés en anglais *highlights*) de la vidéo d'entrée.

L'un des premiers travaux utilisant ce principe est le projet *Informedia* [SK95]. ce projet se base sur la détection de caractéristiques audiovisuelles de bas niveau. Un résumé audio est créé en correspondance avec les mots-clés qui sont extraits à partir de la transcription audio en utilisant la technique TF-IDF. D'une autre part, un résumé vidéo est créé en utilisant la détection de visages, la détection de texte et la détection du mouvement de la caméra. Ce résumé est créé en utilisant un système de classement qui considère que les images présentant les visages ou le texte sont les plus importantes, les images statiques qui suivent le mouvement de la caméra sont moins importantes... etc. Bien que l'intégration de l'information visuelle, audio et textuelle soit la meilleure façon de comprendre une vidéo, la génération de résumés basés sur une telle technique

nécessite encore une intervention manuelle.

Le système *Vabstract* présentée dans [PLF+96], propose de créer une bande annonce comprenant des moments forts d'un film en sélectionnant des scènes vidéo intéressantes. Dans cette méthode, les auteurs ont choisi les scènes comme unités à sélectionner. Une scène est un ensemble d'images consécutives tournant autour d'une même action et liées sémantiquement. Les scènes contenant des objets, des personnes, des scènes d'action, du dialogue... sont extraites et classées comme intéressantes.

Un nouvel algorithme de classification a été proposé dans [PN05]. Cet algorithme utilise une approche de *clustering* basée sur l'utilisation de graphes non orientés mesurant la similarité visuelle entre toutes les paires d'événements d'une vidéo. Les moments forts sont détectés en sélectionnant le clip représentatif de chaque *cluster* et en se basant sur les propriétés des *clusters* : la taille du *cluster* et la *globalité* d'un événement. La globalité d'un événement est un critère calculé en utilisant le nombre de chaînes diffusant cet événement et la fréquence de diffusion de ce dernier. La sélection des clips respecte bien évidemment la contrainte de la durée autorisée du résumé final.

Dans [LHC10], Liu et al. ont proposé une méthode basée sur la suppression des images non pertinentes en fonction de l'intérêt de l'utilisateur afin de garder les moments forts de la vidéo. Pour définir son intérêt, l'utilisateur fournit quelques images qui contiennent un objet d'intérêt (sans localiser précisément l'objet) et d'autres images qui ne contiennent pas l'objet d'intérêt. C'est une méthode qui repose sur l'intervention humaine dans la boucle de détection d'objets adapté aux intérêts des utilisateurs. La méthode de création de résumés est semi-automatique et repose sur l'apprentissage faiblement supervisé.

D'autres approches ont identifié les rôles principaux et les communautés de rôles afin de détecter les événements intéressants dans une vidéo. Dans ce contexte, une technique de création de résumés de films a été proposée dans [TKL+13]. Elle est basée sur l'exploration des relations entre les rôles qui sont regroupées en communautés. Les rôles sont détectés en utilisant la détection de scène et la classification de visages.

Le système *LiveLight* proposé dans [ZX14], décrit une approche de création de résumés présentant le contenu audiovisuel le plus important et le plus intéressant de la vidéo d'entrée. L'approche est basée sur l'apprentissage d'un dictionnaire d'une vidéo donnée en utilisant la méthode *group sparse coding* [BPS+09]. Cela consiste à construire un dictionnaire en utilisant la quantification vectorielle sur un grand ensemble de descripteurs visuels (couleur, texture, angles et forme) à partir d'un ensemble d'apprentissage. Un algorithme du plus proche voisin est utilisé pour compter le nombre d'occurrences de chaque mot du dictionnaire dans la vidéo. Le dictionnaire est mis à jour au fil de l'avan-

cement de la vidéo. Le résumé est ensuite généré en combinant les segments qui sont à l'origine de la mise à jour du dictionnaire.

La création de résumés basés sur la détection des événements intéressants est une tâche difficile dans l'absence d'une connaissance a *priori* du type de la vidéo en question. Dans ce contexte, plusieurs techniques ont été proposées pour détecter les moments forts dans des vidéos de types spécifiques où certaines caractéristiques particulières peuvent être utilisées, comme les vidéos de sport et les vidéos de journaux télévisés. Pour les vidéos de sport en terme général (quel que soit la discipline), des travaux basés sur la détection d'événements tel que dans [XRD03] ont été proposés. D'autres qui abordent en particulier des vidéos de basket-ball [ZE01], des vidéos de baseball [CHG02, RGA00] ou encore des vidéos de football [CDV11, ETM03, SC03], où un modèle de chaîne de Markov, intégré par un algorithme EM (espérance-maximisation) est utilisé pour détecter les pauses et les phases de jeu. Quant aux vidéos de journaux télévisés, un système de questions-réponses est proposé dans [YCZ+03], introduisant une approche de pondération pour classifier les plans de la vidéo qui contiennent des mots associés très fréquents en utilisant la transcription de la parole. Afin de détecter les événements intéressants, d'autres travaux ciblent un autre type spécifique de vidéo, les vidéos égocentriques [LG13, LG15]. Les vidéos égocentriques sont des vidéos acquises avec des caméras portées par des personnes donnant une vue rapprochée sur les actions de la personne qui porte la caméra. Ces méthodes sont basées sur la détection de personnes et d'objets d'intérêt.

Ces approches sont très spécifiques et exigent généralement une phase d'apprentissage pour chaque type de vidéos.

2.5 Approches basées sur les Tweets

Ces approches sont basées sur les messages courts des microblogues récupérés à partir des réseaux sociaux, tel que Twitter, qui est un service de microblogage (service de messages courts), permettant à l'utilisateur de poster des brefs messages appelés *tweets*.

Plusieurs travaux de recherche ont exploré l'association d'un contenu textuel au contenu audiovisuel afin de déterminer les segments importants d'une vidéo. Twitter représente une source intéressante de telle information textuelle. Dans le cas de programmes de télévision diffusés en direct, *Twitter* permet aux utilisateurs de discuter, d'exprimer leur opinion et de réagir en temps réel.

Le système *EpicPlay* [TB12] calcule, durant une émission de sport diffusée en direct, le nombre de *tweets* postés par minute. Cela permet l'élaboration d'une courbe dont les

pics correspondent probablement à des événements intéressants. Le même principe a été utilisé par le système *TwitInfo* [MBB+11] où une étiquette est attribuée à chaque pic de la courbe. Les pics sont étiquetés par un terme, choisi à partir du texte des tweets, en utilisant le critère TF-IDF (*term frequency inverse document frequency*) [SJ72].

L'outil *Statler* présenté dans [SKC10], examine les *tweets* qui sont reliés à la diffusion en direct d'un évènement médiatique (pas forcément du sport). La relation entre le flux d'annotations Twitter et le flux du contenu médias est établie en se basant sur deux métriques. La première métrique est le taux de conversation (*retweet*) entre les auteurs des *tweets*. L'outil considère que les moments durant lesquelles les utilisateurs s'échangent le plus (en utilisant le caractère « @ ») sont des moments intéressants. La deuxième métrique est la taille des *tweets* : plus les *tweets* sont longs, plus l'événement correspondant est intéressant.

Une autre méthode est proposée dans [TYO11]. Le flux des *tweets* est considéré comme un flux de documents textuels. Les auteurs proposent une technique de sélection de *tweets* représentatifs. Elle s'appuie sur une méthode de *clustering* qui tient compte de l'information temporelle des documents. Cette technique est inspirée du problème *Facility Location Problem* (figure 2.1). La similarité entre deux documents est calculée en tenant compte de la fréquence des mots communs entre eux et aussi de la distance temporelle qui les sépare. Le résumé est composé de l'ensemble des tweets sélectionnés avec leurs horodateurs (*timestamps*). Cette méthode peut générer à la fois un résumé textuel et un résumé vidéo puisque chaque tweet sélectionné correspond à un événement intéressant dans la vidéo d'entrée.

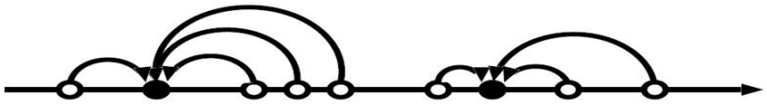

FIGURE 2.1 : *Facility Location Problem* : sélection de *tweets* représentatifs.

Une autre méthode qui n'aborde pas explicitement la création de résumés vidéo, mais qui pourrait être utilisée comme une caractéristique supplémentaire et intéressante dans la création de ce type de résumés, est la méthode proposée par Zhao et al. dans [ZWV11]. Le but de cette méthode est d'extraire la réaction émotionnelle des téléspectateurs en temps réel envers un événement télévisé diffusé en direct. Les événements sont détectés par une augmentation significative du nombre de *tweets* pendant une courte durée en utilisant une fenêtre temporelle glissante. Pour chaque événement, un ensemble de mots-clés caractérisant les *tweets* dans le voisinage de l'événement sont

extraits. Ces mots-clés sont utilisés pour extraire un sentiment en le comparant à un ensemble de mots prédéfinis et représentant des classes de sentiments.

Ces approches, basées sur les *tweets*, sont puissantes, mais elles sont limitées aux programmes TV diffusés en direct qui sont très populaires et qui génèrent un grand nombre de tweets en temps réel. Les tweets doivent être associés à d'autres caractéristiques. Seuls, ils peuvent détecter des événements socialement importants, mais ne garantissent pas qu'un résumé basé sur ces événements soit intéressant et couvre l'ensemble des parties intéressantes de l'émission TV.

2.6 Travaux effectués dans le cadre de TRECVID

2.6.1 Contexte

TRECVID (*TREC Video Retrieval*) est une campagne internationale annuelle d'évaluation proposée par le NIST (*National Institute of Standard and Technology*). Son objectif principal est de promouvoir la recherche dans l'analyse de contenus multimédia. TRECVID est souvent considéré comme le projet d'évaluation le plus complet de ces dernières années. Une évaluation des résultats des participants est effectuée sur un corpus de très grande taille (plus de 35000 vidéos) en fournissant deux corpus vidéo différents : un corpus d'apprentissage et un corpus de test ainsi que des méthodes de mesure standards permettant de comparer les performances des différents systèmes proposés.

La campagne TRECVID comprend chaque année différentes tâches auxquelles les participants peuvent participer :

- détection de plans ;
- détection de copies vidéo ;
- détection des événements à partir de données de vidéo-surveillance ;
- indexation sémantique et extraction d'attributs de haut niveau (parole, intérieur/extérieur, ...) ;
- recherche d'informations (personne, lieu, objet, ...) ;
- création de résumés vidéo à partir des épreuves de tournage de BBC (*BBC rushes*).

Nous nous intéressons ici à la dernière tâche, celle de la création automatique de résumés vidéo à partir des épreuves de tournage provenant de l'archive de la chaîne TV BBC. Cette problématique était le sujet de nombreux travaux de recherche sur plusieurs

années (notamment 2007 [OSK07] et 2008 [OSA08]). Les épreuves de tournage (*rushes*) sont les vidéos originales produites lors des tournages et qui serviront par la suite pour le montage. Autrement dit, il s'agit de vidéos à l'état brut présentant une redondance importante. Contrairement aux vidéos éditées, les épreuves de tournage contiennent d'autres types d'informations redondantes comme par exemple les plans inutiles qui peuvent être insérés pendant l'enregistrement de la vidéo (mouvement intermédiaire de la caméra), ou encore les scènes enregistrées à maintes reprises. Le but principal de cette tâche est d'évaluer la capacité des méthodes, qui ont été développées dans ce cadre, à éliminer les parties répétitives et peu intéressantes. L'idée derrière ces méthodes est de sélectionner un ensemble de segments non redondants à partir des épreuves de tournage et de les assembler pour former un résumé. Ce dernier devrait présenter les objets principaux (fixes ou en mouvement) et les événements intéressants (principalement visuels) des épreuves de tournage tout en minimisant la durée totale qui respectera un rapport de compression minimal (inférieur ou égal à 4% de la durée initiale des épreuves).

Pour cette tâche, TRECVID fournit un corpus de résumés des épreuves de tournage de BBC générés avec un rapport de compression donné. Une base de vidéo MPEG-1 est aussi mise à disposition pour tester et évaluer les méthodes proposées. Le corpus d'apprentissage est composé de 57 vidéos (une totalité de 35 heures). Le corpus de test quant à lui contient 100 vidéos (une totalité de 48 heures).

2.6.2 Travaux des participants

De nombreuses méthodes ont été évaluées dans le cadre des compagnes TRECVID. La majorité des travaux effectués autour de la création de résumés vidéo à partir des épreuves de tournage s'appuient sur trois phases : (1) une phase de segmentation de la vidéo en un ensemble d'unités de base, (2) une phase de *clustering* des unités détectées et (3) une phase de sélection d'extraits à inclure dans le résumé. Les unités de base peuvent être des scènes, des plans, des sous-plans, des segments vidéo de durée fixe ou même des images de la vidéo.

Le plan est souvent considéré comme l'unité de base la plus appropriée pour la création de résumés. Dans ce cas, un *clustering* des plans est utilisé dans le but de détecter et éliminer les plans redondants. Certaines caractéristiques sont utilisées pour classer heuristiquement l'importance des plans telles que la couleur [NDM+08], les visages [BT08], la longueur des plans [SRT08], le mouvement de la caméra [KSV+08] ou l'intensité de mouvement que présente le plan [RJ09].

Dans [SRT08], le *clustering* des plans est effectué en formant un graphe dont les

nœuds représentent tous les plans de la vidéo. Une arête non orientée est ajoutée entre chaque paire de nœuds et le poids de l'arête est la distance entre les deux images clés des deux plans en question. Un arbre couvrant de poids minimal est ensuite calculé. Toutes les arêtes restantes ayant un poids supérieur à un seuil sont supprimées faisant apparaître un ensemble de *clusters* de nœuds (plans). Le plan le plus long de chaque *cluster* est sélectionné. Le nombre de visages, l'intensité de mouvement et la taille du *cluster* sont utilisés pour classer et sélectionner parmi les plans candidats ceux à inclure dans le résumé final. Truong et Venkatesh [TV06] ont classé les plans selon des descripteurs calculés par la méthode des SIFT (*Scale-invariant feature transform*) qui est une méthode permettant de transformer une image en un ensemble de vecteurs de caractéristiques qui sont invariants par transformations géométriques usuelles [Low99]. Une image clé est sélectionnée à partir de chaque *cluster* de plans en se basant sur un ensemble de règles comme par exemple la sélection du visage le plus grand ou bien, dans l'absence de visages, la sélection de la plus longue distance relative avec la caméra. Dans [BT08], une première étape consiste à éliminer les plans courts (durée inférieure à 10 images). Le *clustering* effectué sur les plans restants, se base sur l'extraction de caractéristiques visuelles telles que l'occurrence des visages et l'intensité du mouvement. Une version modifiée de l'algorithme de la plus longue sous-séquence commune est appliquée. Les liens qui se chevauchent sont supprimés. Un extrait est sélectionné à partir de chaque *cluster* en se basant sur un ensemble de règles prédéfinies comme par exemple la durée minimale et maximale d'un extrait sélectionne, la durée entre deux extraits et la durée totale maximale de tous les extraits sélectionnés. Dans [WFW+08], un *clustering* hiérarchique est effectué afin de sélectionner les images représentatives et éliminer les parties répétitives. La vidéo est segmentée tout d'abord en un ensemble de plans. Chaque plan est divisé en sous-plans. Un sous-plan inclut les images consécutives du plan qui sont très similaires. La similarité est calculée en utilisant un histogramme d'intersection des couleurs. Un seuil de similarité est fixé empiriquement, au dessus duquel deux plans sont considérés similaires. Enfin, chaque sous-plan est divisé en un ensemble de segments vidéo ayant chacun une durée égale à une seconde. Le résumé final est la concaténation d'un ensemble de ces segments d'une seconde. Une étape de filtrage des segments redondants et non utiles est réalisée : les segments d'une seconde contenant des images de couleur uniforme et les plans de durée très courts sont éliminés. Un score est calculé pour chaque segment d'une seconde en se basant sur des caractéristiques de bas niveau comme la couleur, les bordures, la détection de visages, le mouvement dominant et l'information audio. Après avoir été trié selon leurs scores, un ensemble de segments d'une seconde, ayant un score supérieur à un seuil, est sélectionné et concaténé

pour former le résumé final. Un *clustering* est également effectué dans [LLR+08] pour la sélection des images clé en utilisant un histogramme local de couleur. L'approche consiste à décomposer une vidéo en plans et à appliquer, par la suite, un alignement global entre toutes les paires de plans. L'alignement est effectué en utilisant un algorithme d'alignement de séquences. Dans le système PICSOM [KSV+08], la sélection des extraits vidéo s'appuie sur une pondération linéaire pour favoriser les images à proximité du centre de chaque plan de la vidéo. Les scores attribués aux extraits contenant des visages, de la parole, des objets ou du mouvement caméra sont par la suite incrémentés selon des poids heuristiques. Ren et Jiang [RJ09] ont proposé une structure hiérarchique pour modéliser les épreuves de tournage (plan, sous-plans...) et ont utilisé le *clustering* basé sur l'algorithme des k plus proches voisins (*K-NN*) pour l'élimination de la redondance. Le plan le plus représentatif est sélectionné à partir de chaque *cluster* en fonction de sa longueur (le plan le plus long du *cluster* est sélectionné afin de maximiser la conservation du contenu visuel dans le résumé) et l'intensité de mouvement qu'il présente (maximiser la quantité de mouvement dans le résumé). Tang et al. [TZL+06] ont proposé un *clustering* basé sur la détection de caractéristiques de haut niveau comme les activités des personnages d'une scène, le dialogue entre les différents personnages et les la reconnaissance des bâtiments. L'élimination des plans répétitifs s'appuie sur l'extraction d'une tranche spatiotemporelle à partir de chaque plan plutôt que sur l'extraction d'uniquement une image représentative. Une tranche spatiotemporelle est un ensemble d'images consécutives extraites sur une période de temps. La similarité entre les tranches spatiotemporelles est calculée en comparant leurs histogrammes respectifs définis sur les différentes caractéristiques détectées. Afin d'effectuer un *clustering* sur les plans et éliminer la redondance, deux méthodes sont utilisées dans [QBPM+08] : l'algorithme des k plus proches voisins (K-NN) dans un premier temps et un *clustering* hiérarchique dans un second temps. Pour les deux méthodes, plusieurs attributs de bas et de moyen niveau sont détectés à partir des épreuves de tournage : activité de l'audio, activité du mouvement, images indésirables, visage et mouvement de la caméra. Une méthode ad-hoc de fusion des ces attributs hétérogènes est utilisée.

Contrairement aux méthodes déjà présentées dans cette section, dans [ISo8], le *clustering* est effectué sur les scènes et non pas sur les plans. Des histogrammes de couleurs et de texture sont calculés pour chaque image de la vidéo. La vidéo est segmentée en un ensemble de scènes en calculant la distance euclidienne des couleurs moyennes des images successives. Les scènes dupliquées sont, par la suite, éliminées en se basant sur la couleur moyenne.

Le système cost292 [NDM+08] propose une méthode basée à la fois sur la détection

de scènes et le *clustering* des plans. Outre la détection de visages et de mouvement de la caméra, les descripteurs de couleur MPEG-7 sont calculés pour chaque image de la vidéo et utilisés pour le *clustering* des plans et la détection de scènes. Les *clusters* sont formés à partir d'une matrice de similarité entre les images de la vidéo. Compte tenu des différents *clusters* formés et des différentes scènes détectées, des extraits minimisant les répétitions sont sélectionnés et ajoutés au résumé final.

Le problème de ce type d'approches réside dans le fait que ces approches sont locales et utilisent uniquement l'information visuelle pour éliminer la redondance. Ces approches ne rentrent pas dans la compréhension de la scène et ne font pas une analyse approfondie qui va au delà de la redondance et de l'aspect visuel. De ce fait, seuls les plans visuellement similaires sont élagués. Deux cas d'utilisation peuvent illustrer ce problème :

- étant donné un ensemble de plans continus constituant une scène de la vidéo, l'utilisateur peut comprendre ce qui se passe dans la scène en regardant seulement un (ou quelques) plans. Toutefois, les plans d'une même scène peuvent ne pas être similaires et par conséquent appartenir à différents *clusters*. Des plans potentiellement non pertinents sont ainsi favorisés et inclus dans le résumé final ;

- étant donné deux plans visuellement similaires appartenant à deux scènes différentes ou à deux événements différents, l'élimination de l'un de ces plans via le *clustering* des plans entraînera une compréhension incomplète de l'événement en question dans le résumé.

Simplement accélérer la vitesse de lecture de la vidéo est une autre technique utilisée pour la création de résumés vidéo. L'être humain est capable de comprendre les événements d'une vidéo accélérée jusqu'à 25 fois la vitesse initiale [HCL+07]. Tout comme la méthode de [PAS+99] présentée au début de la section 2.3, Christel et al. [CHL+08] ont participé à la tâche de TRECVID et ont présenté des travaux basés aussi sur l'accélération de la vidéo. Une première étape consiste à générer un résumé vidéo par accélération de 50 fois la vitesse de lecture de la vidéo originale. Les segments non pertinents sont par la suite détectés et supprimés. L'audio est ajouté au résumé pour améliorer sa compréhension. Une reconnaissance automatique de la parole est tout d'abord effectuée. Le résultat du module de reconnaissance est divisée en phrases en fonction de la durée de silence dans le discours et de la détection d'un changement d'orateur. Un ensemble de phrases est sélectionné et ajouté au résumé sans tenir compte de la synchronisation entre la vidéo et l'audio. Cette approche permet de créer un résumé qui couvre la majorité

de l'épreuve de tournage mais qui contient en contre partie beaucoup de redondance. Bien que cette approche nous permette de créer un résumé de durée réduite, ce dernier inclut inévitablement des parties redondantes ou inutiles.

2.6.3 Évaluation dans le cadre de TRECVID

TRECVID a défini une méthodologie d'évaluation de la qualité des résumés qui se base sur une combinaison de critères objectifs et subjectifs. Une évaluation subjective consiste à juger si le résumé créé inclut des segments intéressants de la vidéo originale et si le résumé contient des redondances. La subjectivité des ces critères vient en partie du fait que les segments jugés intéressants sont annotés par 5 évaluateurs qui sont chargés de créer une vérité terrain et d'attribuer un score pour chaque critère d'évaluation. D'un autre coté, une évaluation objective consiste à mesurer les performances des systèmes proposés tel que la durée du résumé créé et le temps pris par le système pour générer ce résumé.

Deux méthodes de base utilisant des techniques simples de création de résumés sont proposées pour comparer et évaluer les différents systèmes. La première méthode, basique, est une méthode uniforme qui consiste à sélectionner périodiquement un extrait d'une seconde toutes les 25 secondes. Les extraits sont concaténés pour former le résumé final. La durée du résumé est de 4% la durée totale de la vidéo. La deuxième méthode est basée sur la détection des plans en utilisant un seuil sur la différence de la quantité de mouvement entre deux images consécutives. La méthode consiste à effectuer un *clustering* des K-moyennes sur les plans. Le nombre de *clusters* est égal au nombre de secondes dans le résumé de durée 4% de la durée totale de la vidéo. À partir de chaque *cluster*, le plan le plus proche du centroïde est sélectionné. Le résumé vidéo est le produit de la concaténation d'une seconde extraite du milieu de chaque plan sélectionné.

2.7 Évaluation de la qualité des résumés automatiques

Il est important de pouvoir évaluer la qualité des résumés générés automatiquement. Cependant, l'évaluation de la qualité des résumés générés est une tâche délicate. C'est l'une des parties les plus difficiles à mettre en place dans le processus de développement de méthodes de création de résumés vidéo. Il est extrêmement difficile de donner une définition formelle de ce qui est un bon résumé.

Les résumé attendus varient généralement selon le type de la vidéo, l'application cible et les préférences des utilisateurs. Par exemple, un résumé vidéo de sport et une bande d'annonce d'un film ne sont pas créés de la même manière. En outre, l'évaluation

de la qualité d'un résumé est très subjective car l'importance accordée au contenu vidéo peut varier d'un utilisateur à un autre, en fonction des centres d'intérêt de chacun. Le problème de création de résumés n'admet pas une solution unique. En effet, étant donné une ou plusieurs vidéos, chacun créera un résumé différent. Même les résumés qui sont créés manuellement par des experts sont généralement différents.

L'un des problèmes liés à l'évaluation de la pertinence d'un résumé vidéo est que l'évaluation manuelle est très coûteuse et que l'évaluation automatique est intrinsèquement peu fiable. Un autre problème dans ce contexte, c'est qu'il n'existe pas de base de vidéos de taille réaliste, commune et publique qui peut être utilisée pour comparer différentes solutions. Un cadre d'évaluation cohérent et largement accepté est toujours indisponible.

Dans cette section, nous revenons sur les différentes méthodes d'évaluation de méthodes de résumés automatiques qu'elles soient manuelles ou automatiques, traitant du fond (le contenu audiovisel du résumé) ou de la forme (temps d'exécution, flexibilité...). Une analyse approfondie des méthodes d'évaluation de résumés automatiques est présentée dans [TV07]. Ces méthodes sont classées en trois catégories :

1. description du résultat où l'évaluation consiste en une analyse des résultats obtenus et l'influence de paramètres internes ;
2. les paramètres objectifs qui sont généralement appliquées sur les approches statiques ;
3. les études des utilisateurs où des utilisateurs indépendants analysent les résumés obtenus et évaluent leur qualité.

Cette dernière catégorie (une évaluation subjective) est la méthode la plus acceptée. Dans ce contexte, un certain nombre d'évaluateurs sont invités à visualiser un certain nombre de résumés vidéo, puis à répondre à des questions. Le but de ces questions est d'évaluer le degré de satisfaction de l'évaluateur par rapport à certains critères liés au résumé. Parmi ces critères, on peut citer la cohérence, si le résumé est agréable à regarder, s'il est informatif et s'il est facile à comprendre [NMZ03, JHvD08]. Les questions peuvent être également liées à la compréhension du contenu du résumé en question, et peuvent concerner donc des objets ou des événements spécifiques [San07] tel que combien de personnes apparaissent dans le résumé ? qui a gagné la première manche ? ...etc. D'autres questions peuvent concerner des critères liés à la tâche de l'évaluation elle même, par exemple, le temps nécessaire pour effectuer l'évaluation ou le nombre de clics sur l'interface. Dans [TPA+06], après avoir regardé un résumé, l'évaluateur est

invité à répondre à trois questions sur la qualité de résumé, puis à répondre à dix questions à choix multiples issus de la vidéo originale. Deux juges humains indépendants créent deux listes de segments intéressants de la vidéo sans regarder les résumés, et l'intersection des deux listes a été utilisée pour produire les questions.

Un autre type d'évaluation subjective effectuée par des utilisateurs est présenté dans [YMH03]. Après avoir regardé tous les résumés, l'utilisateur est invité à regarder un extrait choisi au hasard d'une vidéo complète choisie également au hasard et on lui demande de deviner de quelle vidéo l'extrait provienne. La qualité du résumé est quantifiée par le pourcentage de réponses correctes que l'utilisateur fournit en regardant tous les extraits possibles de toutes les vidéos.

Les évaluations subjectives peuvent être influencées par des facteurs humains, tels que l'hétérogénéité des profils des utilisateurs, les problèmes d'interface, la fatigue pour les résumés longs... etc. Un autre problème majeur des évaluations subjectives est qu'une fois l'évaluation effectuée, elle n'est pas reproductible ni réutilisable. Si l'algorithme de création de résumés est modifié, l'évaluation doit être refaite. Par ailleurs, l'évaluation subjective doit impliquer un grand nombre d'évaluateurs et d'évaluations pour être statistiquement significative. Ceci rend la tâche très coûteuse en terme de temps. Dans d'autres situations, les utilisateurs sont invités à regarder la vidéo d'origine avec le résumé généré afin que l'évaluation soit plus fiable, ce qui augmente de manière considérable le coût de la tâche d'évaluation (notamment la durée totale nécessaire pour mener ce type d'évaluation).

Concernant la deuxième catégorie des méthodes d'évaluation, plusieurs mesures objectives ont été proposées. Pour l'évaluation de résumés sous forme d'une vue d'ensemble, la métrique la plus appropriée est la mesure de la similarité entre l'ensemble des extraits et la vidéo d'origine [LZF+04]. Quant à l'évaluation de résumés basés sur la détection d'événements intéressants, la présence de segments intéressants peut être quantifiée dans le résumé en la comparant avec une vérité terrain [CHG02, AKT03].

Par ailleurs, un système, appelé *SUPERSIEV*, a été proposé en 2004 dans [HMD04]. Il permet, non seulement d'évaluer la qualité d'un résumé vidéo, mais aussi de comparer les différents algorithmes de création automatique de résumés. Tout d'abord, des résumés de référence (vérité de terrain) sont rassemblés dans une étude utilisateurs effectuée par de nombreux évaluateurs et sur plusieurs vidéos. Pour chaque vidéo, ces résumés sont combinés pour générer une vidéo unique de référence qui représente la majorité des opinions des évaluateurs. Dans une deuxième étape, le système détermine la meilleure image de référence pour chaque image de la vidéo entière et calcule les scores correspondants pour former une table de recherche qui évalue chacune de ces

images. Étant donné un résumé obtenu par un algorithme candidat de création de résumés, le système peut alors évaluer ce résumé de différents aspects en calculant le rappel, la précision moyenne cumulée, le taux de redondance et la proximité de la moyenne.

2.8 Conclusion

Dans ce chapitre, nous avons présenté des méthodes existantes de création automatique de résumés. Nous avons également introduit la compagne TRECVID, notamment la tâche de l'évaluation des résumés vidéo, et nous avons présenté quelques travaux évalués dans ce cadre. Dans une deuxième partie, nous avons discuté les difficultés liées à l'évaluation de la qualité des résumés générés et nous avons discuté les différentes catégories de méthodes d'évaluation ainsi que quelques travaux existants dans ce contexte.

Chapitre 3

Programmation Par Contraintes

Sommaire

3.1 Introduction

La programmation par contraintes (PPC) [RVBW06] est un paradigme de programmation qui relève du domaine de l'intelligence artificielle et de la recherche opérationnelle.

C'est une technique de résolution de problèmes combinatoires et de complexité élevée. Elle est utilisée pour une très grande variété de problèmes combinatoires ainsi que de nombreuses applications réelles [BSR10, BPS99], telles que l'acheminement de véhicules, l'ordonnancement, la configuration, la planification, l'affectation de ressources, et la bioinformatique. Plus récemment, la programmation par contraintes a été considérée comme un outil essentiel dans le domaine de l'exploration de données [DRGN08].

La programmation par contraintes s'intéresse aux problèmes définis en termes de contraintes de temps, d'espace,... ou plus généralement de ressources. Elle est également considérée comme un outil efficace pour la résolution des problèmes NP-complets.

Le principe de la programmation par contraintes est de parcourir un espace de recherche pour trouver une (ou plusieurs) solution(s) à un problème de satisfaction de contraintes (PSC). Une contrainte est une propriété qui doit être vérifiée par la solution trouvée.

Notre motivation pour utiliser la programmation par contraintes pour la résolution du problème de création automatique de résumés de vidéos, est que ce paradigme de programmation, comme mentionné précédemment, est considéré comme une approche puissante pour la modélisation et la résolution de problèmes d'optimisation combinatoire. Notant que le problème de création automatique de résumés de vidéos tel que nous l'avons défini est un problème d'optimisation combinatoire vu la taille énorme de l'espace de recherche, le nombre élevé de combinaisons d'extraits qui peuvent former le résumé final et surtout le nombre élevé de comparaisons des bornes des extraits à sélectionner avec les différents segments d'attributs de bas niveau détectés à partir de la vidéo d'entrée.

L'avantage de la programmation par contraintes est qu'elle présente une séparation claire entre la modélisation et la résolution des PSC. L'intérêt principal de cette séparation est de proposer à l'utilisateur de modéliser un problème sans s'intéresser à la façon avec laquelle le problème sera résolu. En effet, l'utilisateur se concentre uniquement sur la spécification du problème lui-même et le solveur est responsable de le résoudre. Les différentes parties sont clairement identifiées dans la figure 3.1. La première partie qui est la partie de modélisation consiste à modéliser le problème comme un PSC. A cet effet, les variables du problème, leurs domaines et les contraintes qui doivent être satisfaites, doivent être définis. Tandis que la deuxième partie qui est la partie de la résolution, consiste à résoudre le problème modélisé : lire le modèle, définir la méthode de résolution (section 3.4) et définir la stratégie de recherche (section 3.5).

Lors de la résolution de problèmes à contraintes, on peut distinguer deux types de solutions :

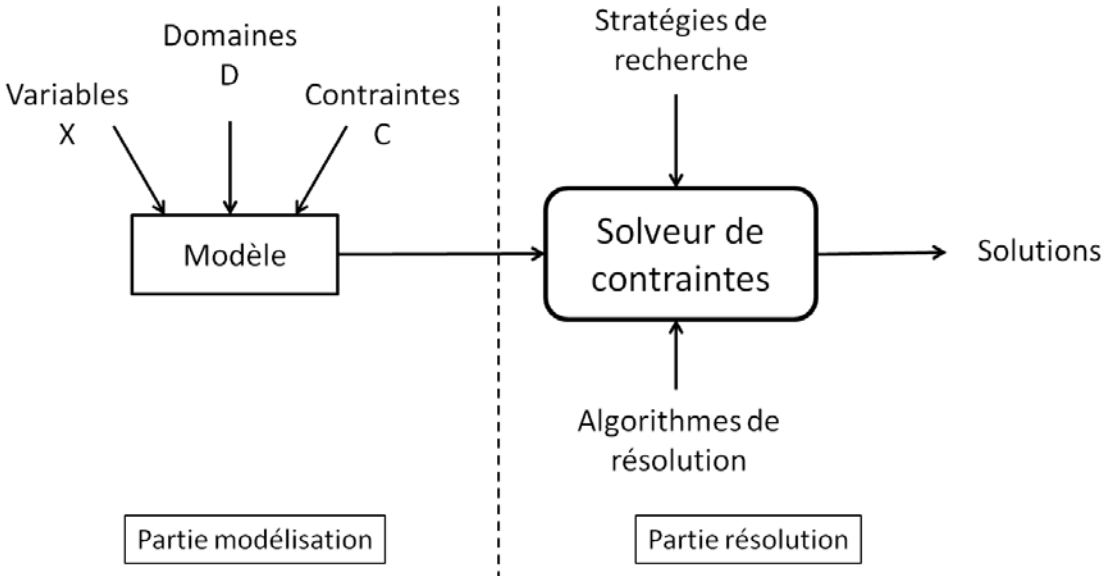

FIGURE 3.1 : Modélisation et résolution des problèmes de satisfaction de contraintes

- les solutions réalisables : celles qui respectent les contraintes exprimées, dans ce cas le résultat peut être une seule solution (la première solution trouvée) ou toutes les solutions possibles ;
- les solutions optimales : celles qui représentent le résultat d'une minimisation (ou maximisation) d'une fonction de coût.

Dans la suite de ce chapitre, nous introduisons les problèmes de satisfaction de contraintes avec ou sans optimisation. Nous présenterons par la suite les méthodes de résolution. Nous avons choisi de décrire quatre méthodes : le filtrage des variables, la propagation des contraintes, le *Backtracking* et la recherche locale. Nous présenterons les stratégies de recherche adoptées pour résoudre efficacement un problème. Une section est dédiée ensuite aux solveurs de contraintes existants. Enfin, ce chapitre se termine par un exemple qui permet d'illustrer le fonctionnement de la PPC.

3.2 Problèmes de satisfaction de contraintes

Un problème de satisfaction de contraintes se définit par le triplet suivant (X, D, C) :

- un ensemble de variables $X = \{X_1, X_2 \dots X_n\}$;
- un ensemble de domaines des variables $D = \{D(X_1), D(X_2) \dots D(X_n)\}$, où $D(X)$ peut être un ensemble fini de valeurs possibles que peut prendre la variable X ou un ensemble de valeurs fournies par un intervalle ou une union d'intervalles ;

- un ensemble de contraintes $C = \{C_1, C_2 \dots C_m\}$, où C_i est une contrainte définie par un sous-ensemble des variables sur lesquelles elle porte et exprime une propriété qui doit être satisfaite par les variables correspondantes. C_i est donc une relation définie sur un ensemble de variables.

Dans la programmation par contraintes, une solution admissible pour un problème est une affectation à chaque variable d'une valeur de son domaine de recherche, de telle sorte que les contraintes exprimées sur ces variables soient satisfaites. Un solveur est alors utilisé pour trouver une ou plusieurs solutions au problème ainsi modélisé.

Exemple 1 : soit le problème suivant :

Les inconnues sont les valeurs des variables x, y et z. Chaque variable peut prendre une valeur entre 1 et 10. La valeur de x doit être supérieure à la valeur de y et la somme des trois valeurs de x, y et z doit être égale à 12.

Pour modéliser ce problème en tant que problème de satisfaction de contraintes, nous devons définir les variables du PSC, leurs domaines de définition et les contraintes exprimées dessus :

- variables : x, y et z trois variables entières ;
- domaines de définition : $x \in [1, 10]$, $y \in [1, 10]$ et $z \in [1, 10]$;
- contraintes : $C_1 : x > y$, $C_2 : x + y + z = 12$;

Une solution possible consiste à affecter les valeurs suivantes aux variables x, y et z :
$x \leftarrow 4$
$y \leftarrow 2$
$z \leftarrow 6$

3.3 Problèmes de satisfaction de contraintes avec optimisation

Une autre caractéristique importante et intéressante de la programmation par contrainte est la possibilité d'exprimer non seulement des contraintes dures à satisfaire, mais aussi une fonction de coût à optimiser. La PPC donne la l'opportunité d'optimiser une variable (ou un objectif) en maximisant ou en minimisant une fonction de coût.

Exemple 2 : reprenons le même problème définit dans l'exemple 1 de la section précédente, en ajoutant une contrainte d'optimisation qui consiste à maximiser la valeur de la variable z. Le problème serait ainsi :

- variables : x, y et z trois variables entières ;

- domaines de définition : $x \in [1,10]$, $y \in [1,10]$ et $z \in [1,10]$;

- contraintes : $C_1 : x > y$, $C_2 : x + y + z = 12$;

- fonction à maximiser : $f(x,y,z) = z$.

La solution optimale pour ce problème est la suivante :
$x = 2$
$y = 1$
$z = 9$

Pour certains problèmes, il peut y avoir plusieurs objectifs dont on cherche à s'approcher. Or, une seule fonction peut être maximisée ou minimisée. Dans ce cas, il faut combiner ces objectifs.

Soient f la fonction à maximiser par le solveur et $o_1...o_k$ les k objectifs avec $\alpha_1...\alpha_k$ les poids respectifs associés à ces objectifs [1]. Les objectifs sont ainsi combinés dans une unique fonction de coût f, de la façon suivante :

$$f = \sum_{i=1}^{k} \alpha_i \times o_i \tag{3.1}$$

Le solveur utilise d'abord une première recherche pour trouver une solution (Sol_0) satisfaisant les contraintes définies. Dans une deuxième étape, il ajoute une nouvelle contrainte sous la forme $f(A) < f(Sol_0)$ (resp. $f(A) > f(Sol_0)$) dans le cas f doit être minimisée (respectivement maximisée). Ici, A désigne la dernière solution trouvée. Ce processus est répété pour chaque nouvelle solution rencontrée.

La programmation par contraintes n'impose pas d'algorithme particulier pour résoudre les problèmes de satisfaction de contraintes. Plusieurs méthodes de recherche de solutions sont possibles. La section suivante présente quelques-unes de ces méthodes.

3.4 Méthodes de résolution des PSC

Parmi les approches de résolution des problèmes de satisfaction de contraintes, on peut citer le filtrage des variables, la propagation de contraintes, le *backtracking* et la recherche locale. Ces approches sont décrites dans les sections suivantes.

3.4.1 Filtrage des variables

Dans la programmation par contraintes, un problème de satisfaction de contraintes est considéré comme un ensemble de sous-problèmes pour lesquelles des méthodes effi-

[1] Si un objectif est une valeur à minimiser, alors il suffit de lui donner un poids négatif.

caces de résolution sont associées. Chaque contrainte du modèle représente un sous-problème et, à chaque contrainte, un algorithme de filtrage est utilisé. Le rôle d'un algorithme de filtrage est de supprimer les valeurs impossibles des domaines des variables sur lesquelles porte la contrainte, c'est-à-dire, les valeurs qui n'appartiennent à aucune solution du sous-problème et pour lesquelles il n'est pas possible de satisfaire la contrainte en question. La suppression de ces valeurs est effectuée en tenant compte des valeurs possibles des autres domaines de variables impliquées dans la contrainte.

Par exemple, étant donné x et y deux variables entières, pour la contrainte $(x > y)$ avec $D(x) = [0, 20]$ et $D(y) = [10, 30]$, l'algorithme de filtrage associé à cette contrainte modifiera les domaines de x et de y en supprimant les valeurs de 0 à 10 de $D(x)$ et les valeurs de 20 à 30 de $D(y)$. Ces valeurs sont supprimées car il est impossible de les affecter aux variables (à titre d'exemple, c'est impossible d'affecter la valeur 8 à la variable x car la contrainte exprimée impose que x doit être supérieure à y et la plus petite valeur que y peut prendre est 10). Par conséquent, les domaines des variables x et y sont modifiés comme suit : $D(x) = [11, 20]$ et $D(y) = [10, 19]$. Ce mécanisme de filtrage consiste donc à réduire les domaines de recherches des différentes variables.

3.4.2 Propagation de contraintes

Après chaque modification du domaine de recherche d'une variable (apportée par un algorithme de filtrage associé à une contrainte), les algorithmes de filtrage des contraintes faisant intervenir cette variable sont appliquées. Le but est d'apporter éventuellement d'autres suppression et de réduire d'autres domaines de recherche de variables pouvant être affectés par cette modification. On dit alors qu'une modification a été propagée. Ce mécanisme, appelé propagation de contraintes, est répété jusqu'à ce que plus aucune modification ne puisse se produire. Si la propagation entraîne une contradiction (que des valeurs impossibles dans les domaines de recherche des variables), le moteur de propagation arrête le processus.

Les techniques de propagation de contraintes sont des méthodes utilisées pour transformer le problème de satisfaction de contraintes en un autre problème qui soit équivalent mais qui est plus simple à résoudre. Cela consiste à décomposer le problème initial en un ensemble de sous-problèmes. Les méthodes utilisées dans ce type de résolution imposent la cohérence locale au sein de chaque sous-problème. Cette idée permet de trouver une solution au problème en moins de temps.

Ces algorithmes de propagation sont utilisés pour garantir la résolution du problème, c'est-à-dire trouver une solution s'il en existe, ou prouver que le problème n'est

pas satisfiable, c'est-à-dire aucune solution ne peut satisfaire les contraintes exprimées.

3.4.3 Le *Backtracking*

Le « *Backtracking* » (appelé aussi retour sur trace) est actuellement l'algorithme le plus largement utilisé dans la résolution des problèmes de satisfaction de contraintes. Il est basé sur la construction d'un arbre de recherche (figure 3.2) où chaque niveau de l'arbre correspond à une variable, chaque nœud par niveau représente une valeur possible de la variable et les branches d'un nœud représentent des choix alternatifs qui pourraient être examinés afin de trouver une solution et qui doivent respecter les contraintes exprimées. Les contraintes sont utilisées ici pour tailler les sous-arbres ne contenant pas de solutions. L'approche de *Backtracking* consiste à effectue un parcours en profondeur d'un arbre de recherche.

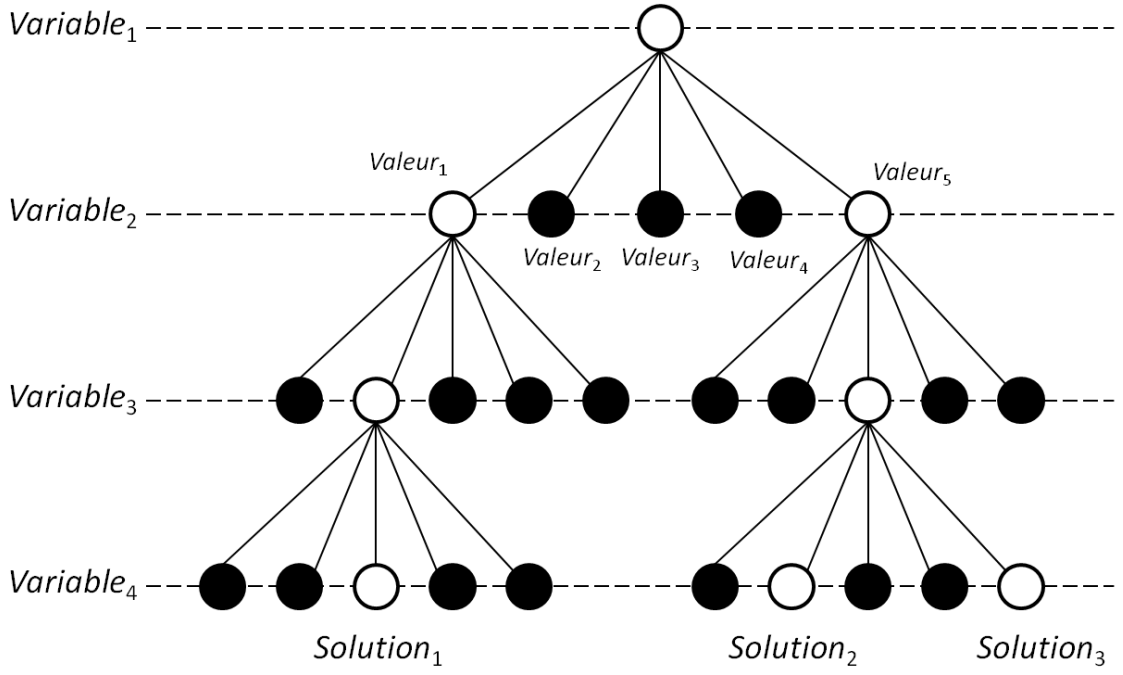

FIGURE 3.2 : Arbre de recherche pour réaliser le *backtracking*.

Le *Backtracking* est un algorithme récursif qui commence de la racine de l'arbre et descend progressivement d'un niveau. Cet algorithme s'exécute en continu jusqu'à trouver une feuille. Une fois que cela s'est produit, l'algorithme revient au nœud parent et le descendant suivant, s'il existe, est choisi. L'algorithme continue à s'exécuter jusqu'à ce que le contrôle revienne au niveau du nœud racine de l'arbre et tous les nœud descendants ont été visités et vérifiés. Les feuilles qui donnent une seule solution ont généralement des domaines qui sont des ensembles uniques. Ce qui est intéressant dans la méthode

de *backtracking*, c'est que si l'on ne souhaite trouver qu'une seule solution, le programme s'arrête après la première feuille générant une solution réalisable. Par contre, si on souhaite trouver toutes les solutions possibles, le programme parcourt l'arbre tout entier. Notant que l'algorithme de *backtracking* ne vérifie que les contraintes impliquant la variable courante (nœud courant) et les variables précédentes (nœuds précédents).

Les algorithmes de *backtracking* sont généralement complets [2]. Ils sont utilisés pour garantir qu'une solution sera trouvée s'il en existe, ou pour montrer qu'un PSC n'admet pas de solution.

Nous allons dans la dernière section de ce chapitre appliquer l'algorithme de *backtracking* sur un exemple concret pour bien illustrer son principe et son fonctionnement.

3.4.4 La recherche locale

Les techniques de recherche locale sont des algorithmes de satisfiabilité incomplets [3]. Ces algorithmes peuvent aboutir à l'une de ces deux possibilités : trouver une solution à un problème ou ne pas trouver une solution, même s'il en existe. Le processus consiste à améliorer itérativement une assignation complète des variables. Une assignation complète est l'affectation de valeurs à toutes les variables du modèle. À chaque étape, les valeurs de certaines variables sont changées afin d'augmenter le nombre de contraintes satisfaites par cette assignation. C'est un algorithme qui ne cherche pas à tester exhaustivement toutes les instanciations possibles.

3.5 Stratégies de résolution

Le mécanisme de recherche de solutions a pour but de trouver une solution, éventuellement optimale (maximisant ou minimisant une fonction de coût). Il met en œuvre différents moyens qui vont permettre au solveur d'atteindre des solutions. Parmi ces moyens nous pouvons citer les stratégies de choix de variables et de valeurs. Ces stratégies, souvent appelées stratégies de branchement, se présentent sous la forme d'heuristiques définissant les critères qui vont permettre de déterminer la prochaine variable qui sera instanciée ainsi que la valeur qui lui sera attribuée.

Un algorithme de recherche arborescente nécessite que l'ordre dans lequel les variables sont examinées soit précisé. L'utilisation des heuristiques de sélection de variables peut considérablement affecter l'efficacité des algorithmes de résolution des pro-

[2] Un algorithme complet est un algorithme qui donne toujours une réponse (si assez de temps et d'espace)

[3] Un algorithme incomplet est un algorithme qui peut échouer à retourner une réponse. Il peut trouver une solution à un problème, mais il peut échouer même si le problème est satisfiable

blèmes à contraintes. La sélection des variables pourra suivre un ordre statique, dans lequel l'ordre des variables est spécifié avant le début de la recherche et n'est pas modifié par la suite. Elle peut également suivre un ordre dynamique, dans lequel le choix de la prochaine variable est précisé en tenant compte de l'état actuel de la recherche.

Les heuristiques statiques de sélection de variables gardent le même ordre tout au long de la recherche, en utilisant uniquement des informations structurelles sur l'état initial de la recherche. Les heuristiques dynamiques du choix de variables sont considérablement plus efficaces. Ces heuristiques sont dynamiques parce que le choix des variables prend en compte des informations sur l'état actuel du problème à chaque instant de la recherche. La stratégie « Dom » [HE80] était la première heuristique dynamique bien connue et utilisée dans la PPC. Les auteurs admettent que les heuristiques qui minimisent le branchement en profondeur sont les plus efficaces.

Après avoir sélectionné la variable à laquelle une valeur va être assignée, l'algorithme de recherche de solutions devra décider comment choisir l'ordre dans lequel les valeurs doivent être attribuées aux différentes variables, à moins que des valeurs doivent être attribuées tout simplement dans leur ordre d'apparition dans le domaine de recherche. Les heuristiques de sélection de valeurs peuvent influencer le nombre de nœuds explorés, le nombre de solutions retournées en T minutes et par conséquent le temps d'exécution total de l'algorithme.

3.6 Solveurs existants

Les solveurs de contraintes sont utilisés pour résoudre des problèmes de satisfaction et d'optimisation de contraintes. Plusieurs solveurs de problèmes de satisfaction de contraintes ont été développés. Les plus populaires sont : le solveur *ILOG* (qui a été racheté par IBM en 2008), *OR-TOOLS* (dévelopé par Google), le solveur *Microsoft Solver Foundation* (développé par Microsoft) et CHOCO (développé par l'école des mines de Nantes et financé par Bouygues SA et Amadeus SA).

Dans notre travail, nous avons utilisé le solveur CHOCO [JRL08]. CHOCO est une bibliothèque java fondée sur un mécanisme de propagation basé sur les événements avec des structures « backtrackable ». C'est un logiciel open-source hébergé par GitHub (pour la version 3) et par sourceforge.net (pour les versions antérieures). Il est distribué sous licence BSD permettant une utilisation académique aussi bien qu'une utilisation commerciale. CHOCO est un solveur de contraintes facile à utiliser, optimisé et extensible. Les contributions des chercheurs académiques peuvent potentiellement enrichir le solveur CHOCO en fournissant des améliorations à court et à long terme.

Les différents solveurs de contraintes participent à des compétitions annuelles [LRvD10]. Ils sont alors soumis à des tests qui évaluent leurs performances. Les tests concernent le temps de réponse des solveurs, les fonctionnalités offertes (contraintes, stratégies...), les types de problèmes traités (allocation, concurrence...), la mesure de l'extensibilité, etc. D'après *MiniZinc challenge*, la compétition mondiale des solveurs de contraintes, organisée par le centre de recherche NICTA (*National Information and Communications Technology Australia*) en 2013 et en 2014, CHOCO est classé parmi les solveurs les plus rapides. Les organisateurs ont mis 30 solveurs en compétition et ils les ont évalué sur 20 problèmes différents. Des scores ont été calculés selon la qualité de la réponse et le temps de réponse pour chacun des 20 problèmes à résoudre. Dans le classement final, CHOCO a obtenu deux médailles d'argent et trois médailles de bronze. De plus, CHOCO a prouvé qu'il est le solveur Java le plus performant. Les compétitions et les résultats obtenus sont disponibles sur ces sites : http://www.minizinc.org/challenge2013/results2013.html et http://www.minizinc.org/challenge2014/results2014.html.

Selon le solveur CHOCO, définir une stratégie de recherche est très important car une stratégie de recherche bien adaptée peut réduire le nombre de nœuds développés, le nombre de « *backtracks* » et par conséquent le temps que le solveur prendra pour trouver des solutions au problème. Pour cela, CHOCO propose des stratégies prédéfinies [4] et donne la possibilité aux utilisateurs d'implémenter leurs propres stratégies qui s'adaptent à leurs modèles et à leurs besoins. CHOCO est également un solveur extensible puisqu'il offre aux utilisateurs la possibilité d'ajouter leurs propres contraintes et d'implémenter leurs propres algorithmes de filtrages qui correspondent à leurs besoins sans utiliser les contraintes de base proposés par les solveurs.

3.7 Exemple des N-Reines

Pour illustrer le fonctionnement de la PPC et montrer la multitude des modèles possibles pour un problème, ainsi que les critères de choix d'un modèle par rapport à un autre, nous avons choisi de présenter le problème classique des N-Reines.

Le probleme des N-Reines consiste à placer n reines du jeu d'échec sur un damier de taille $n \times n$, de telle sorte qu'aucune reine ne puisse en capturer une autre : une reine peut en capturer une autre si elles se trouvent sur une même diagonale, une même ligne ou une même colonne de l'échiquier. Le principe du jeu est illustré par la figure 3.3.

[4] Documentation de CHOCO. https://github.com/chocoteam/choco3/blob/master/user_guide.pdf

Plusieurs modélisations peuvent être proposées.

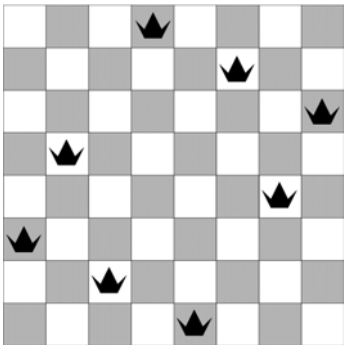

FIGURE 3.3 : Exemple de solutions au problème des n reines.

3.7.1 Premier modèle

Un premier modèle consiste à associer deux variables entières L_i et C_i à chaque reine i. L_i et C_i correspondent respectivement à la ligne et à la colonne sur laquelle la reine est positionnée. Le problème s'écrit alors comme suit :

- variables : $X = \{L_1, ..., L_n, C_1, ..., C_n\}$;
- domaines de définition : $D(L_1) = ... = D(L_n) = D(C_1) = ... = D(C_n) = \{1..n\}$;
- contraintes :

  Les reines doivent être sur des lignes différentes :

  $\rightarrow Clig = \{L_i \neq L_j / i \in \{1..n\}, j \in \{1..n\}, i \neq j\}$

  Les reines doivent être sur des colonnes différentes :

  $\rightarrow Ccol = \{C_i \neq C_j / i \in \{1..n\}, j \in \{1..n\}, i \neq j\}$

  Les reines doivent être sur des diagonales ascendantes différentes :

  $\rightarrow Cdm = \{C_i + L_i \neq C_j + L_j / i \in \{1..n\}, j \in \{1..n\}, i \neq j\}$

  Les reines doivent être sur des diagonales descendantes différentes :

  $\rightarrow Cdd = \{C_i - L_i \neq C_j - L_j / i \in \{1..n\}, j \in \{1..n\}, i \neq j\}$

Cette modélisation est basique et représente l'aspect naturel et intuitif de la solution. Elle présente un nombre élevé de variables. Une amélioration de ce modèle est possible en réduisant le nombre de variables et en enlevant les informations superflues.

3.7.2 Deuxième modèle

Dans le cas où on part de l'évidence qu'il n'y aura qu'une seule reine sur chaque colonne de l'échiquier, le problème peut se résumer à déterminer sur quelle ligne se positionne la reine placée sur la colonne i. Un deuxième modèle consiste donc à associer une variable X_i à chaque colonne i. La variable X_i désigne le numéro de la ligne sur laquelle la reine de la colonne i est placée. Le problème s'écrit alors :

- variables : $X = \{X_i / i \in \{1..n\}\}$;
- domaines de définition : $D(X_1) = ... = D(X_n) = \{1..n\}$;
- contraintes :

  Les reines doivent être sur des lignes différentes :

  $\rightarrow Ccol$: Contrainte satisfaite implicitement par le modèle.

  Les reines doivent être sur des colonnes différentes :

  $\rightarrow Clig = \{X_i \neq X_j / i \in \{1..n\}, j \in \{1..n\}, i \neq j\}$

  Les reines doivent être sur des diagonales ascendantes différentes :

  $\rightarrow Cdm = \{X_i + i \neq X_j + j / i \in \{1..n\}, j \in \{1..n\}, i \neq j\}$

  Les reines doivent être sur des diagonales descendantes différentes :

  $\rightarrow Cdd = \{X_i - i \neq X_j - j / i \in \{1..n\}, j \in \{1..n\}, i \neq j\}$

Un exemple d'un arbre de recherche construit par l'algorithme de *Backtracking* est illustré dans la figure 3.4. L'algorithme est appliqué sur l'exemple de 4 reines en utilisant le deuxième modèle. À chaque niveau de l'arbre, une variable du modèle est concernée par l'affectation d'une valeur tout en respectant les contraintes exprimées. Si le solveur trouve une valeur possible, l'algorithme passe au niveau suivant (variable suivante). Si aucune valeur n'est possible, l'algorithme fait un retour sur trace et parcourt de nouveau l'espace de recherche. Une solution est trouvée si tous les niveaux de l'arbre sont parcourus et toutes les variables sont instanciées.

3.7.3 Troisième modèle

Contrairement aux deux premiers modèles où les variables désignent les positions des reines sur l'échiquier, les variables du troisième modèle désignent les états des cases de l'échiquier. Une variable binaire est associée à chacune des $n \times n$ cases. X_{ij} désigne la variable associée à la case située à la ligne i et à la colonne j. Elle peut prendre la valeur

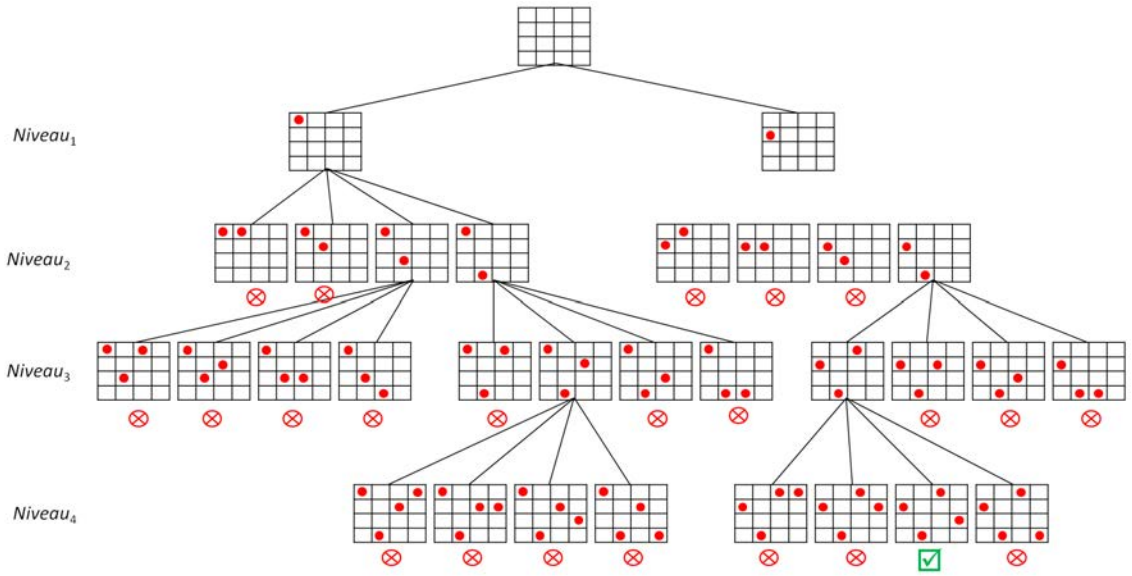

FIGURE 3.4 : *Backtrack* sur l'exemple des 4 reines.

0 si la case est vide ou la valeur 1 si la case est occupée par une reine. Le problème s'écrit alors :

- variables : $X = \{X_{ij}\}, i \in \{1..n\} etj \in \{1..n\}$;
- domaines de définition : $D(X_{ij}) = \{0,1\}, i \in \{1..n\}, j \in \{1..n\}$;
- contraintes :

  Il y a une reine par ligne :

  $\rightarrow Clig = \left\{\sum_{j=1}^{n} X_{ij} = 1/i \in 1..n\right\}$

  Il y a une reine par colonne :

  $\rightarrow Ccol = \left\{\sum_{i=1}^{n} X_{ij} = 1/j \in 1..n\right\}$

  Les reines doivent être sur des diagonales ascendantes différentes :

  $\rightarrow Cdm = \{i + j = k + l \Longrightarrow X_{ij} + X_{kl} \leq 1/i, j, k, l \in \{1..n\}\}$

  Les reines doivent être sur des diagonales descendantes différentes :

  $\rightarrow Cdd = \{i - j = k - l \Longrightarrow X_{ij} + X_{kl} \leq 1/i, j, k, l \in \{1..n\}\}$

Cette modélisation permet de réduire considérablement l'espace de recherche puisque le domaine de définition des variables est binaire (0 ou 1). Cela permet au solveur d'accélérer la recherche de solutions.

3.7.4 Bilan

Nous avons présenté à l'aide de cet exemple, trois modèles possibles pour résoudre le problème des N-reines. Nous avons montré la différence entre ces modèles. La programmation par contraintes est un outil qui permet de résoudre des problèmes de différentes façons selon la modélisation adoptée. Nous avons montré à travers cet exemple qu'une résolution fiable du problème ne peut être atteinte que si le modèle est bien étudié. Une bonne résolution du problème est avant tout une bonne modélisation permettant une bonne formulation des contraintes.

3.8 Conclusion

Dans ce chapitre, nous avons introduit les principes de base de la programmation par contraintes que nous avons utilisée pour résoudre le problème de création automatique de résumés de vidéos. Nous avons décrit des méthodes de résolutions ainsi que les stratégies de recherche utilisées par la programmation par contraintes. Nous avons choisi l'exemple des N-reines pour décrire la multitude des modèles possibles pour un problème donné.

L'utilisation de la programmation par contraintes a un avantage évident : nous pouvons nous concentrer sur le cœur de notre problème, c'est-à-dire sur la forme voulue pour le résumé plutôt que sur la façon de l'obtenir. Dans ce chapitre, nous nous sommes intéressés à cette deuxième partie, nous allons maintenant nous intéresser, dans la deuxième partie de la thèse, à la manière dont on peut exprimer un problème de génération de résumés vidéo selon ce paradigme de programmation.

Deuxième partie

Contributions de la thèse

Chapitre 4

Modélisation basée sur la segmentation en plans

Sommaire

4.1 Introduction

Le principe de fonctionnement général de notre approche est décrit par le schéma de la figure 4.1. Un ensemble d'attributs de bas et de haut niveau est calculé à partir de la

vidéo d'entrée. Ces attributs sont utilisés pour décrire le contenu audiovisuel (couleur, mots-clés, transcription de la parole, volume de l'audio, etc.). La manière dont le résumé doit être créé est exprimée par un expert sous forme de contraintes à satisfaire et d'une fonction de coût à optimiser. Un exemple de contraintes exprimées par un expert est d'interdire l'interruption du flux de parole d'un commentateur. S'assurer qu'un résumé couvre les différentes parties d'une vidéo en mettant l'accent sur la partie finale est un autre exemple de contraintes pouvant être exprimées par un expert. Ce type de contraintes est exprimé hors ligne. Un autre ensemble de contraintes peut également être exprimé par un ensemble de paramètres définis par l'utilisateur final. Un exemple typique de ce deuxième type de contraintes concerne la durée souhaitée du résumé. Ce type de contraintes est exprimé, en ligne, sous forme de préférences utilisateurs.

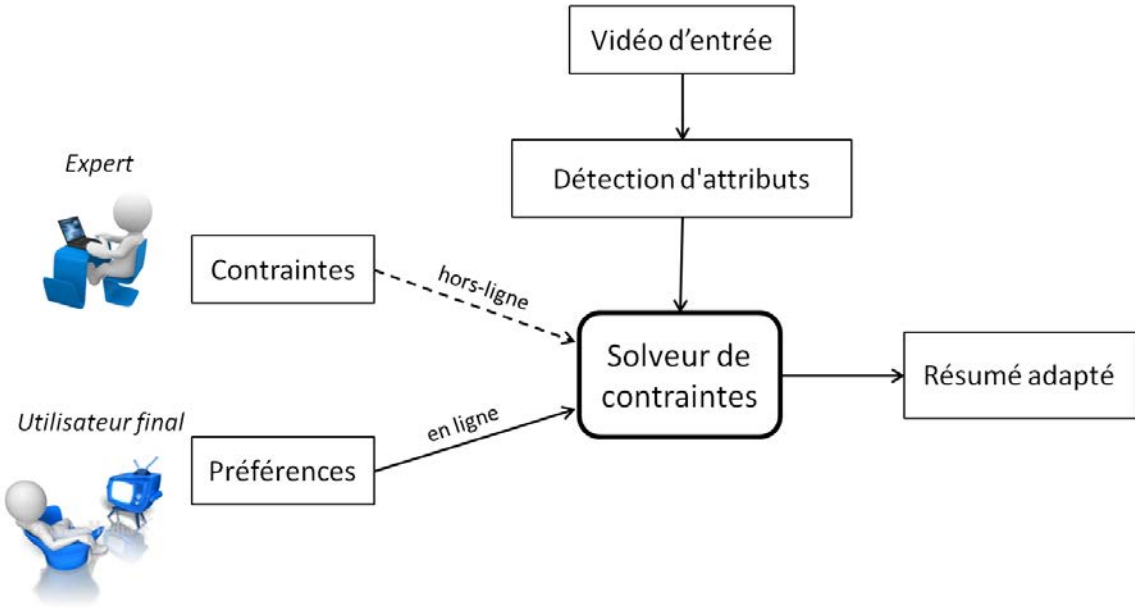

FIGURE 4.1 : Schéma du fonctionnement général de notre méthode.

Dans ce contexte, le résumé est le résultat de la résolution d'un problème de satisfaction de contraintes. Comme expliqué précédemment, la solution d'un tel problème n'est pas nécessairement unique. Un ensemble de résumés satisfaisant la totalité des contraintes exprimées peut être retourné par le solveur de contraintes.

Cette approche est suffisamment flexible pour pouvoir définir différents types de contraintes liées au contenu vidéo, mais aussi aux préférences de l'utilisateur. L'expert (ou encore l'utilisateur final) peut ajouter, à tout moment, une nouvelle contrainte ou modifier une contrainte existante sans avoir à revoir l'ensemble du modèle ou à modifier aucun paramètre interne.

La programmation par contraintes permet d'exprimer facilement la forme attendue du résumé : durée, événements à sélectionner, etc. Les contraintes sont exprimées sur des attributs détectés à partir de la vidéo d'entrée.

La création de résumés vidéo nécessite une segmentation temporelle de la vidéo d'entrée. Pour cela, nous avons proposé, dans un premier temps, de modéliser le problème en se basant sur une segmentation de la vidéo en plans. Le plan est l'unité de base naturelle à laquelle on pense quand on parle de la segmentation vidéo. De plus, c'est ce qui est utilisé majoritairement dans les travaux qui traitent de l'analyse et de l'indexation vidéo. La segmentation en plans a fait l'objet de nombreux travaux tel que [BR96] et [Hano2]. Un plan est défini comme un ensemble d'images qui assure une continuité visuelle qui n'implique pas l'audio. Il est composé d'images qui représentent la même scène physique, désigne une opération unique d'une même caméra sans interruption, ou contient un événement ou une action distincte.

Un autre problème de la segmentation en plans est qu'une image d'une vidéo n'est pas forcément issue de la prise unique d'une caméra, mais elle peut être le résultat du montage de plusieurs flux pris par plusieurs caméras dont les plans ne sont pas synchrones ou de nombreux effets obtenus numériquement. Comme dans de nombreux journaux télévisés par exemple, l'image présente à la fois une vue sur le présentateur, des bandeaux qui défilent, une image ou une vidéo incrustée, etc.

Dans ce chapitre nous présentons l'étape de prétraitement qui consiste à analyser finement le contenu vidéo et à détecter les attributs de bas niveau sur lesquels les contraintes sont exprimées. Nous présentons par la suite deux modèles basés sur la segmentation en plans pour la création automatique de résumés. Nous décrivons les deux modèles, les expérimentations, les avantages et les limites de chacun. Nous terminons par une conclusion.

4.2 Prétraitement : analyse de la vidéo et détection d'attributs de bas niveau

Dans notre approche, une étape de prétraitement de la vidéo d'entrée est nécessaire pour la création automatique du résumé. Par définition, une vidéo est une évolution spatio-temporelle de différents médias (vidéo, audio et texte). Une analyse fine de la vidéo est nécessaire pour extraire un ensemble d'attributs de bas niveau dans le but d'exprimer des contraintes liées au contenu de la vidéo. Ces attributs peuvent donc être extraites à partir de l'information visuelle (visage, l'intensité de mouvement, couleur...), de l'information sonore (musique, applaudissements, silence...) ou de l'information textuelle

(transcription de la parole, texte incrusté, sous-titres).

Afin de pouvoir détecter des attributs de bas niveau sur lesquelles porteront des contraintes, nous avons utilisé les algorithmes suivants pour la détection d'attributs :

- le descripteur de la couleur dominante : nous avons utilisé le descripteur (*DCD*) de la norme MPEG-7 [YCK+08] qui permet de détecter 8 couleurs dominantes au maximum. Ce descripteur fournit les couleurs dominantes les plus représentatives présentes dans une image ainsi que leurs propriétés statistiques comme la distribution et la variance. La méthode de détection des couleurs dominantes utilise l'algorithme de *Lloyd-Max* décrit dans [GG12] afin de regrouper les valeurs de couleur des pixels ;

- détection des applaudissements, de musique et de silence : nous avons utilisé une méthode similaire à celle décrite dans [SS97]. La méthode permet de segmenter un contenu audiovisuel en cinq classes : parole, musique, applaudissement, silence et autre. Ces classes sont présentées sous la forme de segments labellisés sans recouvrements. La classification sonore s'appuie sur l'extraction de paramètres de bas-niveaux, le plus souvent fréquentiels, qui servent à discriminer entre plusieurs types de sons. Les paramètres sont extraits toutes les trames de 10 ms. L'algorithme de classification est basé sur la modélisation des paramètres à l'aide de GMMs (*Gaussian Mixture Models*) [Rey09]. La solution proposée repose sur l'utilisation de quatre types de paramètres : modulation d'énergie à 4 Hz, MFCC, flux spectral et chroma. Pour une trame donnée et pour chaque classe de son, l'algorithme estime la probabilité qu'un des paramètres prenne la valeur pour un signal appartenant à la classe. Pour cela, cette probabilité est modélisée par des mélanges de gaussiennes (GMM). Pour chaque trame, la décision est prise sur la classe la plus probable. Pour chaque classe, 4 jeux de paramètres sont utilisés, ce qui conduit à apprendre 4 modèles par classe. L'apprentissage est effectué de façon indépendante pour chaque cas à l'aide de données d'exemples ;

- la segmentation de paroles : cette méthode permet d'annoter le signal audio en des segments *parole/non-parole*. La méthode consiste à découper le signal audio en trames x (typiquement, 20 ou 30 ms), sur lesquelles des coefficients acoustiques sont calculés (les coefficients cepstraux). Chacune des trames x_t est classifiée en parole si $P(x_t \ / \ Parole) > P(x_t \ / \ NonParole)$, où les probabilités $P(x_t \ / \ Parole)$ et $P(x_t \ / \ NonParole)$ sont estimées par un mélange de gaussiennes (pour chacune des classes Parole et NonParole). Pour un signal audio donné, une suite de labels *parole* et *non-parole* est donc obtenue. Cette suite de labels est ensuite lissée, en

considérant des contraintes minimales de durée de segment de *parole* et de segment de *non-parole* ;

- la détection et le suivi des visages : nous avons utilisé la méthode décrite dans [MG+07] pour la détection et le suivi des visages dans une vidéo. La méthode se base sur le détecteur de visages CFF (*Convolutional Face Finder*) [GD04]. Cette approche neuronale se base sur un réseau de neurones convolutionnel entraîné à effectuer une classification binaire (visage / non visage) sur un corpus d'images enrichi avec du *Bootstrapping* (en injectant dans le corpus, de manière itérative, des faux positifs). Le suivi est ensuite effectué de la manière suivante : le CFF est appliqué sur l'image complète soit sur un début de segment (par exemple quand un changement de plan est détecté), soit sur une image toutes les deux secondes. Sur toutes les autres images de la vidéo, la détection est effectuée sur une petite zone définie autour de chaque visage détecté. Ceci permet d'avoir des performances supérieures au temps réel, tout en préservant la robustesse ;

- détection des segments audio ayant un volume élevé ou présentant un changement brusque de volume : nous avons utilisé une méthode de détection des pics audio pour pouvoir repérer les événements intéressants ;

- détection des publicités : cette méthode utilise la détection des séparateurs qui annoncent le début et la fin de la publicité.

Ces attributs sont représentés sous forme de l'ensemble des segments où l'attribut en question est présent, comme indiqué dans la figure 4.2. Ainsi, nous appelons segment de l'attribut $Attribut_k$ une partie de la vidéo comprise entre deux limites de temps (début et fin). Ces segments sont représentés par deux tableaux qui désignent le début et la fin de chacun :

$$Attribut_k \begin{cases} Debut\_Attribut_k = \{deb\_Attribut_k[i],\ i = 1..m\} \\ Fin\_Attribut_k = \{fin\_Attribut_k[i],\ i = 1..m\} \end{cases} \tag{4.1}$$

où m est le nombre de segments de l'attribut en question ($Attribut_k$) détectés dans la vidéo d'entrée, $deb\_Attribut_k[i]$ (resp. $fin\_Attribut_k[i]$) est le début (resp. la fin) du segment numéro i de l'attribut $Attribut_k$.

Tout attribut représentable sous forme d'un ensemble de segments est utilisable dans notre modèle, quelle que soit sa nature, il suffit de disposer de l'algorithme de détection correspondant. Ce que nous avons présenté par l'équation 4.1, n'est qu'un exemple de ce que l'on peut utiliser. Jusque-là, nous avons représenté à travers un segment la présence

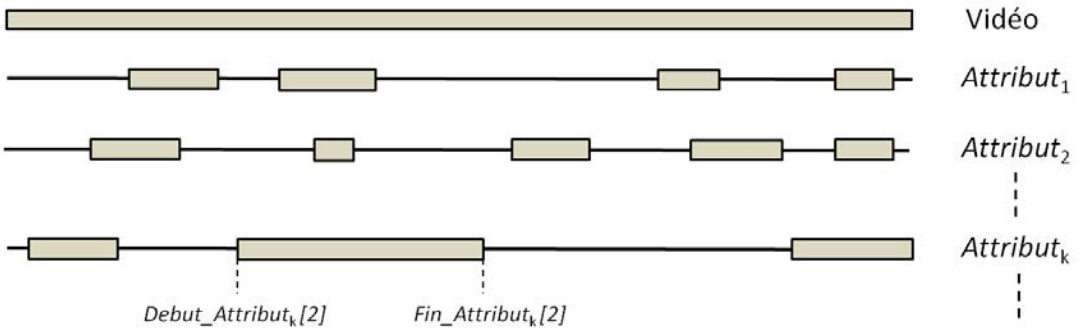

FIGURE 4.2 : Représentation des attributs détectés à partir de la vidéo.

(ou non) d'un attribut. En effet, les segments peuvent être valués. La valeur peut indiquer le nombre de visages présents, la taille du visage, le nombre de locuteurs, etc. Les plans eux mêmes entrent dans ce formalisme, chacun d'eux étant valué par sa durée.

Comme nous le verrons plus tard, cette présentation des attributs est très pratique. Elle facilite la définition des contraintes à l'aide d'opérateurs simples comme l'intersection par exemple.

4.3 Premier modèle

Étant donné qu'une vidéo est un ensemble de plans contigus, nous avons proposé un premier modèle basé sur la sélection de plans à inclure dans le résumé final. La première étape consiste à segmenter la vidéo en un ensemble de plans. Le processus de création de résumés consiste alors à retenir ou à éliminer des plans entiers. Le résumé final est l'assemblage des plans sélectionnés. Ce modèle repose donc pleinement sur la segmentation en plans. La figure 4.3 illustre le principe de ce modèle. Un plan est, soit entièrement sélectionné et inclus dans le résumé, soit rejeté.

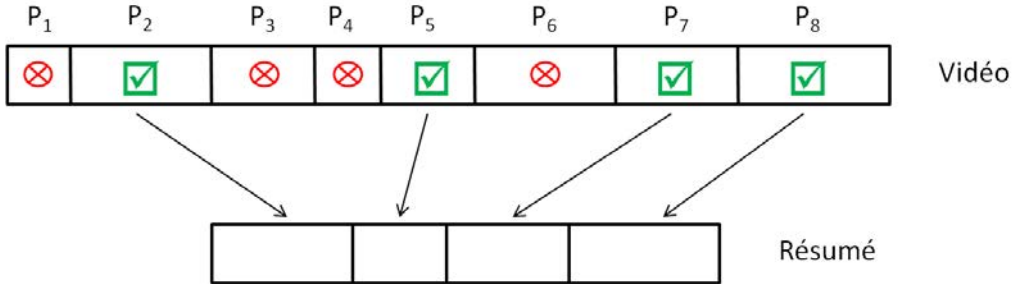

FIGURE 4.3 : Principe du premier modèle. Les plans P_2, P_5, P_7 et P_8 sont sélectionnés et concaténés pour construire le résumé.

Dans la section suivante, nous allons présenter un exemple de modèle qui ne peut aboutir et qui n'a donc pas été retenu par la suite. Le but est de mettre en exergue les difficultés rencontrées et de montrer que la modélisation n'est pas une tâche évidente.

4.3.1 Préliminaire : une première tentative de modélisation

Vu que notre démarche est basée sur la sélection de plans, la toute première idée était de créer un tableau de données *duree_plan* de taille p (p est le nombre de plans dans la vidéo d'entrée) où la case i du tableau *duree_plan* contient la durée du plan d'indice i. La tâche du solveur consiste à sélectionner les valeurs (les cases du tableau) qui respectent la contrainte de durée, sélectionnant ainsi n plans à inclure dans le résumé. Le résumé est, alors, la concaténation de ces n plans sélectionnés. Pour cela, le nombre de variables du modèle proposé est égal au nombre d'extraits (sous forme de plans dans ce cas) qui sont sélectionnés à partir de la vidéo d'origine et qui sont inclus dans le résumé final (ce nombre est n).

Le modèle est donc définit par :

$$Extraits = \{extrait_i,\ i = 1..n\} \tag{4.2}$$

où le domaine de définition de chaque $extrait_i$ est défini par les valeurs du tableau *duree_plan*, autrement dit, une parmi les valeurs de *duree_plan* est attribuée à chaque variable $extrait_i$:

$$extrait_i \in duree\_plan \tag{4.3}$$

La première contrainte à imposer sur le résumé à générer est sa durée. La contrainte de durée peut être exprimée de la façon suivante :

$$\sum_{i=1}^{n} extrait_i = Duree \tag{4.4}$$

Cette façon avec laquelle le problème est modélisé n'est pas la plus appropriée car elle pose quelques ennuis en liaison avec les contraintes qui peuvent être exprimées et l'ensemble de solutions calculées. Pour bien comprendre le contexte et les ennuis rencontrés, nous proposons d'appliquer le modèle sur un autre problème équivalent en miniature : étant donnée un tableau T de p entiers, retourner les n entiers ayant une somme égale à S. Avec $T = [5,3,8,5,1,2,9,4]$, $n = 3$ et $S = 6$, l'ensemble de solutions possibles sera :

6 = 1 + 1 + 4 || 6 = 1 + 2 + 3 || 6 = 1 + 3 + 2 || 6 = 1 + 4 + 1 || 6 = 2 + 1 + 3

6 = 2 + 2 + 2 || 6 = 2 + 3 + 1 || 6 = 3 + 1 + 2 || 6 = 3 + 2 + 1 || 6 = 4 + 1 + 1

Cet exemple montre clairement qu'un plan peut être sélectionné plusieurs fois dans le même résumé (exemple : la solution $6 = 2 + 2 + 2$ où le sixième plan est inclus trois fois dans le même résumé), ce qui contredit avec la définition d'un **résumé**. Une solution qui pourrait résoudre une partie de ce problème, est l'utilisation de la contrainte *AllDifferent*. C'est une contrainte utilisée, directement ou indirectement, dans la plupart des problèmes en pratique. *AllDifferent* est une contrainte globale qui impose à toutes les variables du modèle de prendre des valeurs deux à deux différentes. Elle est implémentée dans plusieurs solveurs de contraintes (dont *Choco Solver*). En l'intégrant dans notre modèle, l'ensemble de solutions possibles est réduit :

6 = 1 + 2 + 3 || 6 = 1 + 3 + 2 || 6 = 2 + 1 + 3 || 6 = 2 + 3 + 1 || 6 = 3 + 1 + 2

6 = 3 + 2 + 1

En utilisant la contrainte *AllDifferent* sur l'ensemble de variables du modèle, une valeur de durée ne peut apparaître qu'une seule fois dans le résumé, un plan ne peut donc être sélectionné qu'une seule fois. Cependant, deux plans ayant la même durée ne peuvent pas être sélectionnés simultanément et ils sont considérés en tant qu'un seul.

Le problème majeur de ce modèle est l'impossibilité de récupérer l'indice i du plan sélectionné. Cela engendre l'impossibilité d'exprimer d'autres contraintes telles que la répartition des plans, la représentativité du résumé ou encore des contraintes sur le voisinage des différents plans. La localisation du plan sélectionné est impossible avec cette façon de modéliser le problème. Nous avons donc pensé à une modélisation qui nous aide à garder la traçabilité des plans sélectionnés, savoir de quelle partie de la vidéo ils proviennent et pouvoir exprimer des contraintes, principalement, de voisinage.

Nous avons présenté ce premier exemple de modèle du problème pour bien comprendre le problème de modélisation et pour acquérir une idée claire des difficultés que l'on peut rencontrer. Dans la section suivante, nous allons présenter le modèle que nous avons adopté pour le problème de création de résumés vidéo en se basant sur la sélection de plans.

4.3.2 Premier modèle retenu : modèle #1

Rappelons que le principe de ce premier modèle est la sélection de plans à inclure dans le résumé. Chaque plan est représenté par une variable binaire auquelle la valeur 1 est attribuée si le plan correspondant est à retenir dans le résumé, et la valeur 0 sinon. Notre modèle est défini donc avec p variables binaires, p est le nombre total de plans dans la vidéo d'entrée. Contrairement à l'exemple de modèle décrit précédemment, nous créons,

dans le modèle #1, autant de variables que de plans. Cela nous permettra d'élargir la liste de contraintes possibles (voisinage, représentativité...) en accédant directement à l'indice de la variable en question.

Les variables de notre modèle sont :

$$Extraits = \{extrait_i,\ i = 1..p\} \tag{4.5}$$

où le domaine de définition de chaque $extrait_i$ est $\{0, 1\}$. L'attribution de la valeur 1 ou 0 aux variables *Extraits* s'effectue en tenant en compte des contraintes à satisfaire.

Avec ce modèle, nous n'utilisons pas les segments d'attributs tel qu'ils sont définit par la formule 4.1. Une étape de projection des segments d'attributs sur les plans est nécessaire pour pouvoir exprimer et appliquer des contraintes sur ces attributs. La figure 4.4 illustre ce prétraitement supplémentaire appliqué sur les segments d'attributs.

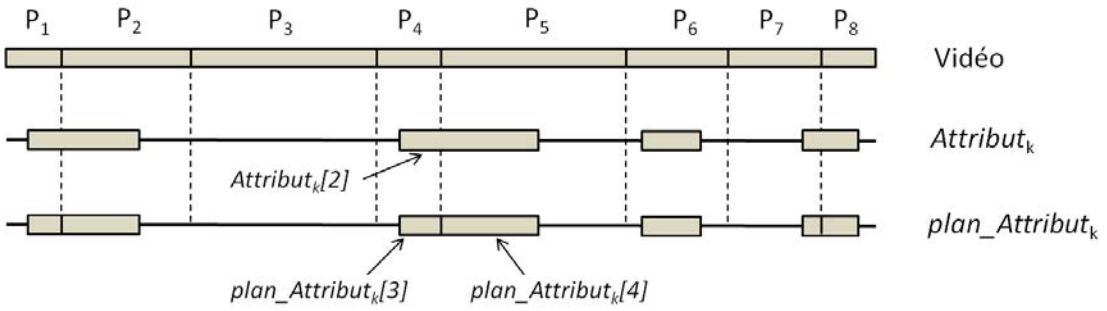

FIGURE 4.4 : Projection des m segments d'attributs sur les p plans et création de l nouveaux segments d'attributs.

Le résultat de ce prétraitement sur les m segments d'attributs extraits, est un nouvel ensemble de l segments, tel que $l \geq m$ où m est le nombre de segments dans la ligne $Attribut_k$ et l le nombre de segments de la ligne $plan\_Attribut_k$. Par conséquent, le segment $plan\_Attribut_k\,[i]$ indique que la présence de l'attribut k dans le plan i ainsi que ses limites de début et de fin. Par exemple, le segment $plan\_Attribut_k\,[3]$ et $plan\_Attribut_k\,[4]$ ont été générés par l'intersection des plans P_4 et P_5 avec le segment d'attribut $Attribut_k\,[2]$.

4.3.3 Modèle #1 : formulation de contraintes

Avec ce modèle, les contraintes peuvent être exprimées en utilisant des opérations de base (tel que la somme, produit, comparaison et les opérateurs logiques) entre les variables *Extraits* et les différents attributs détectés à partir de la vidéo.

Ce modèle nous offre une grande facilité d'expression et de formulation de contraintes. L'aspect binaire des variables a une influence directe sur les contraintes et sur la façon dont elles sont exprimées. Par contre, les contraintes formulées peuvent, comme on le verra plus tard, entraîner un risque de tomber dans les domaines impossibles où aucune solution n'est possible pour le problème (voir expérience 1 de la section 4.3.4). Cela est dû aux domaines de recherche réduits.

4.3.3.1 Durée du résumé

Pour exprimer une contrainte sur la durée totale du résumé à créer, nous avons utilisé le produit scalaire des deux tableaux *Extraits* et *duree_plan* (*duree_plan* étant un tableau contenant les durées des plans) de la façon suivante :

$$dmin \leq \sum_{i=1}^{p} duree\_plan_i * extrait_i \leq dmax \tag{4.6}$$

où *dmin* (respectivement *dmax*) est la durée minimale (respectivement maximale) autorisée pour le résumé souhaitée, $duree\_plan_i$ est la durée du plan i. Le produit scalaire est réalisé en utilisant une contrainte globale qui est généralement prédéfinie dans les solveurs de contraintes[1]. En pratique, il est beaucoup plus efficace d'utiliser la contrainte globale prédéfinie que de parcourir les deux tableaux et d'utiliser à chaque itération les opérateurs *somme* et *multiplication*.

4.3.3.2 Durée minimale d'un extrait

Pour éviter d'avoir un résumé trop fragmenté, nous pouvons poser une limite sur la durée des plans à intégrer dans le résumé. Une contrainte sur la durée minimale d'un plan peut donc être exprimée : sélectionner seulement les plans ayant une durée minimale *duree_plan_min*. Cette contrainte assure la bonne compréhension des extraits sélectionnés puisque l'œil humain a besoin d'un minimum de temps pour percevoir un objet. Cela s'exprime de la façon suivante :

$$(extrait_i * duree\_plan_i) \geq (extrait_i * duree\_plan\_min) \tag{4.7}$$

Si le plan i est sélectionné ($extrait_i = 1$), la contrainte n'est vérifiée que si $duree\_plan_i$ (la durée du plan i) est supérieure ou égale à la durée minimale autorisée *duree_plan_min*. Si le plan i n'est pas sélectionné ($extrait_i = 0$), la contrainte est toujours vérifiée même si le plan i est trop court.

[1] Dans CHOCO, le produit scalaire s'effectue en utilisant la contrainte globale *Scalar*.

4.3.3.3 Attribut non souhaité

Dans le résumé, nous pourrions être amenés à ne pas vouloir intégrer des plans qui présentent un attribut particulier. Par exemple, on ne veut pas avoir dans un résumé de match de foot, des plans qui présentent les spectateurs. Il s'agit d'attributs qui sont non souhaités. Une façon de répondre à ce besoin, c'est de poser une contrainte qui interdit qu'un attribut donné ($Attribut_k$) soit présent dans un extrait du résumé. Cette contrainte peut être exprimée comme suit :

$$\sum_{i=1}^{p}(plan\_Attribut_k\,[i] * extrait_i) = 0 \tag{4.8}$$

Cette contrainte assure que si le plan est sélectionné ($extrait_i = 1$), l'attribut $Attribut_k$ ne doit pas y être présent. Ce qui garantit que la somme au final soit nulle.

4.3.3.4 Présence d'un attribut et voisinage

A contrario, nous pourrions être amenés à imposer qu'un attribut $Attribut_k$ doive, obligatoirement, être présent dans les extraits qui constitueront le résumé final. Cette contrainte est assurée par la règle suivante :

$C : \forall i, i \in [1..n]$ si le plan est sélectionné ($extrait_i = 1$) alors l'attribut $Attribut_k$ est présent ($plan\_Attribut_k\,[i] = 1$) :

$extrait_i$	$plan\_Attribut_k\,[i]$	$C : extrait_i \Rightarrow plan\_Attribut_k\,[i]$
1	1	1
1	0	0
0	1	1
0	0	1

TABLE 4.1 : Contrainte de présence d'un attribut k dans chaque plan sélectionné.

Jusque-là, nous avons raisonné avec des contraintes exprimées sur l'extrait lui-même. On pourrait être amenés aussi à exprimer des contraintes sur le voisinage de l'extrait. Nous pouvons exprimer la possibilité de garantir que les plans qui succèdent les plans sélectionnés contiennent un attribut $Attribut_k$:

$$\forall i, i \in [1..p-1], extrait_i \Rightarrow plan\_Attribut_k\,[i+1] \tag{4.9}$$

Par exemple, si on part de l'évidence que les parties qui précèdent les applaudisse-

ments sont souvent les plus intéressantes, la contrainte de voisinage est exprimée ainsi :

$$\forall i, i \in [1..p-1], extrait_i \Rightarrow plan\_Applaudissement\,[i+1] \tag{4.10}$$

où *plan_Applaudissement* est la projection de l'attribut *Applaudissement* sur le vecteur *plan*.

4.3.4 Modèle #1 : expérimentations et évaluation

Dans cette section, nous allons présenter des expériences pour évaluer uniquement l'efficacité du modèle proposé. La qualité des résumés résultants ne sera pas étudiée.

Nous avons évalué notre modèle en ajoutant une nouvelle contrainte (*CP*) appliquée sur l'attribut de la parole. La contrainte consiste à ne pas autoriser la coupure des groupes de souffle des personnes qui parlent dans les vidéos (ne pas couper la parole). Cette contrainte n'est pas définie dans la section 4.3.3 car ce n'est pas possible, avec notre modèle actuel, de l'exprimer au niveau de la formulation de contraintes du modèle sans passer par la création de variables intermédiaires qui surchargent et alourdissent le modèle. Une méthode déconseillée de formulation de la contrainte *CP* est donc :

- Créer p variables binaires : $Vars = \{v_i,\ i = 1..p\}$
où le domaine de définition de v_i est : $v_i \in \{0\ ,\ 1\}$
- v_i = 1 si chacune des deux frontières du plan i ne coupe pas un segment de *parole* (plan sélectionnable)
- v_i = 0 si une des deux frontières du plan i coupe à un segment de *parole* (plan non sélectionnable)
- $CP : extrait_i \Rightarrow v_i$

Notons qu'en programmation par contraintes, les contraintes du modèle ne peuvent être exprimées que sur, **au moins**, une variable du modèle (une variable de décision). Étant donné que la contrainte *CP* désigne que : « les **plans** qui peuvent être sélectionnés sont les plans dont les bordures ne correspondent pas à des segments de **paroles** », elle ne peut être exprimée qu'en utilisant des variables intermédiaires car nous ne disposons que de deux variables ordinaires de données (*parole* et *plan*) sur lesquelles la contrainte ne peut pas être exprimée. Aucune variable de *Extraits* n'est utilisée pour exprimer la contrainte *CP*, par conséquent, la contrainte n'implique aucune variable du modèle. Prenons l'exemple de la contrainte « supérieur », elle ne peut être exprimée que sous l'une de ces trois formes :

- *superieur (var, val)* : la valeur attribuée à la variable *var* doit être supérieure à la

valeur *val* ;

- *superieur (var1, var2)* : la valeur attribuée à la variable *var*1 doit être supérieure à la valeur attribuée à la variable *var*2 ;

- *superieur (val, var)* : la valeur *val* doit être supérieure à la valeur attribuée à la variable *var*.

Pour évaluer le modèle #1, nous avons utilisé un premier ensemble de données de test $Ensemble_1 = \{J_1, J_2, J_3, J_4, J_5\}$ composé de cinq vidéos d'une émission de jeu télévisé « Des chiffres et des lettres »[2]. La durée totale de l'ensemble de vidéos est de 2 heures 57 minutes.

Nous avons testé le modèle #1, dans un premier temps, sur la vidéo J_1 avec et sans l'utilisation de la contrainte *CP*. Dans une deuxième expérience, nous avons appliqué le modèle #1 sur le reste des vidéos de l'ensemble $Ensemble_1$, en utilisant toutes les contraintes (y compris la contrainte *CP*). Nous avons calculé à chaque fois la durée d'exécution qui correspond à la durée totale que prend le solveur pour calculer et retourner toutes les solution possibles.

Modèle	Nombre de solutions	Durée d'exécution
Sans la contrainte CP	3295	31 secondes
Avec la contrainte CP	0	1 seconde

TABLE 4.2 : Effet de la contrainte *CP* sur le modèle #1.

Durée du match	Nombre de solutions	Durée d'exécution
J_1 : 26 minutes	0	1 seconde
J_2 : 39 minutes	529	4 secondes
J_3 : 48 minutes	0	1 secondes
J_4 : 32 minutes	0	1 secondes
J_5 : 32 minutes	317	3 secondes

TABLE 4.3 : Temps d'exécution et nombre de solutions retournées avec modèle #1.

Les tableaux 4.2 et 4.3 montrent qu'en ajoutant la contrainte *CP*, le problème est devenu trop contraint vu que les domaines de recherche des variables sont restreints (variables binaires : seulement deux valeurs 0 et 1). Par conséquent, aucune solution ne résoud le problème. Nous pouvons conclure que le modèle est trop vulnérable à l'ajout de contraintes.

[2]Des chiffres et des lettres est un jeu télévisé français diffusé sur France 3 reposant sur les compétences en calcul et sur la connaissance du vocabulaire des candidats.

Par contre, dans le cas où la vidéo présente des solutions même dans la présence de la contrainte *CP*, le modèle branche directement sur une première solution trouvée (temps de réponse rapide) et l'exploration de tous les nœuds de l'arbre est très rapide (temps d'exécution très court). Cela montre que le modèle #1 est très efficace et rapide. L'exploration de la totalité de l'espace de recherche est possible en un temps très raisonnable et la résolution du problème est presque immédiate (Ce qui n'est pas forcément les cas avec d'autres modèles possibles, comme on le verra plus tard).

4.3.5 Modèle #1 : limites du modèle

Un des avantages de ce modèle est son espace de recherche très réduit ce qui permet une exploration complète des domaines de recherche des différentes variables du modèle en un temps raisonnable. La recherche de solution par le solveur est par conséquent très efficace. Ce modèle peut être suffisant pour certaines applications et dans certains contextes tel que la sélection de plans ou de scènes intéressants d'une vidéo sur lesquels s'applique le processus de création automatique de résumés dans le but d'accélérer la tâche.

Bien que le modèle soit très simple à mettre en place et offre une facilité d'expression de contraintes en utilisant des opérateurs logiques et arithmétiques, son inconvénient majeur réside dans le fait qu'on ne peut pas exprimer toutes les contraintes, par exemple celle exprimée sur le nombre ou la durée des plans sélectionnés qui sont juxtaposés dans la vidéo d'entrée. Cette contrainte permet d'imposer des règles sur la répartition des plans sélectionnés sur la totalité de la vidéo d'entrée ou encore sur la durée totale maximale (ou minimale) des plans juxtaposés sélectionnés ensemble.

D'autre part, il est extrêmement dépendant des bordures des plans et ne profite pas pleinement des points forts de la programmation par contraintes car la sélection des plans est simplement basé sur une décision binaire. Dans le cas d'un plan très long où seulement une courte partie est intéressante, ce modèle peut être utilisé mais les résumés risquent d'être de mauvaise qualité.

4.4 Deuxième modèle : modèle #2

Pour palier aux limitations du modèle #1, nous proposons un second modèle dans lequel le résumé est composé d'un ensemble d'extraits sous forme de plans ou de parties de plan. La figure 4.5 illustre un cas d'usage. Chaque plan peut être soit non sélectionné (pas d'extrait à partir du deuxième plan), soit entièrement sélectionné ($extrait_4$ à partir du quatrième plan), soit partiellement sélectionné ($extrait_1$ et $extrait_3$).

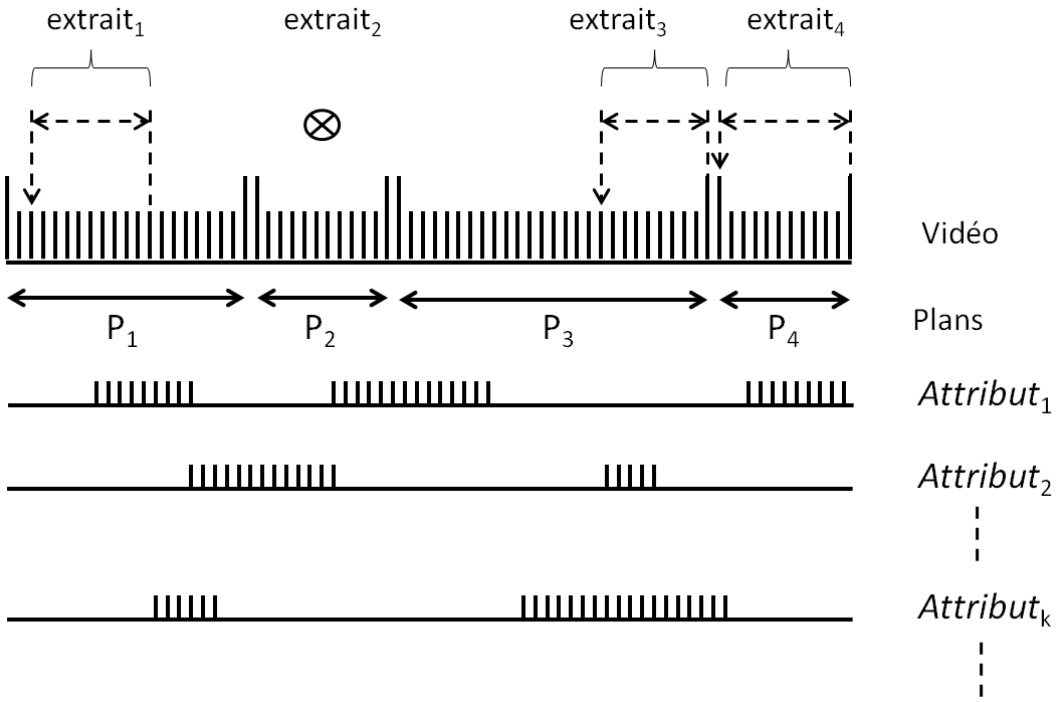

FIGURE 4.5 : Deuxième modélisation : représentation des attributs, de la segmentation en plans et des extraits sélectionnés pour former le résumé.

4.4.1 Modèle #2 : modélisation du problème

Comme dans le premier modèle, le résumé généré avec ce modèle est constitué d'un ensemble d'extraits sélectionnés à partir de la vidéo d'entrée. La particularité est que l'extrait peut être un sous-plan (une partie du plan) représenté par un ensemble d'images consécutives appartenant au plan. Nous avons fait en sorte, à travers le modèle #2, d'autoriser la sélection d'un extrait au sein du plan afin de pouvoir cibler uniquement les parties intéressantes. D'autre part, il est inefficace, du point de vue de la qualité du résumé, de sélectionner un plan entier très long et contenant des informations redondantes. Cela permet à notre modèle #2 d'être capable à cibler uniquement les parties intéressantes des plans, par rapport à une application et à un type de vidéo particulier..

Dans ce modèle, nous définissons un extrait par un couple de variables : l'index de sa première image (image de début de l'extrait) et le nombre d'images qu'il contient (le nombre d'images désigne la durée dans le modèle #2). Nous proposons donc d'attribuer deux variables ($fdebut_i$, $fnbr_i$) pour chaque plan ($plan_i$) : $fdebut_i$ désigne la première image de l'extrait $extrait_i$ sélectionné à partir de $plan_i$ et $fnbr_i$ désigne le nombre d'images de l'extrait d'indice i. Si aucun extrait n'est sélectionné à partir d'un plan i, $fnbr_i$ est égal à 0.

Les variables de notre modèle #2 sont ainsi :

$$Extraits \begin{cases} FDebut = \{fdebut_i,\ i = 1..p\} \\ FNbr = \{fnbr_i,\ i = 1..p\} \end{cases} \tag{4.11}$$

où p est le nombre total de plans et les domaines de $fdebut_i$ et $fnbr_i$ sont :

$$\begin{aligned} fdebut_i &\in [debut\_plan_i\ ..\ fin\_plan_i] \\ fnbr_i &\in [0\ ..\ duree\_plan_i] \end{aligned} \tag{4.12}$$

Ici $debut\_plan_i$ (respectivement fin_plan_i) est l'indice de la première image (respectivement la dernière image) du plan $plan_i$ et $duree\_plan_i$ est le nombre d'images du plan $plan_i$.

Afin de réduire le coût de calcul du processus de création de résumés, nous avons proposé de quantifier les domaines de recherche des variables $fdebut_i$ et $fnbr_i$ (par exemple un pas de quantification de 12 images, équivalent à une demi seconde). Le nombre de valeurs possibles pour les variables est réduit alors que le résumé est, visuellement, presque non modifié du point de vue de l'utilisateur.

4.4.2 Modèle #2 : formulation de contraintes

Plusieurs contraintes peuvent être combinées pour générer un résumé vidéo. Nous proposons de les classer en cinq types de contraintes. Ils sont décrits dans les paragraphes qui suivent.

4.4.2.1 Les contraintes de modélisation

Contrairement au modèle #1 proposé dans la section 4.3 qui est un modèle simple et cohérent en soi, ce deuxième modèle nécessite des contraintes qui portent sur des variables du modèle afin d'assurer notamment sa cohérence. En d'autres termes, il s'agit de garantir par des contraintes que les variables prennent des valeurs cohérentes. Dans notre cas, nous définissons deux contraintes :

$$fdebut_i + fnbr_i - 1 \ \leq\ fin\_plan_i \tag{4.13}$$

$$fnbr_i = 0 \implies fdebut_i = debut\_plan_i \tag{4.14}$$

où $debut\_plan_i$ (respectivement fin_plan_i) est le début (respectivement la fin) du plan $plan_i$. La contrainte (4.13) est utilisée pour s'assurer que l'extrait sélectionné ne dépasse

pas les limites du plan correspondant. Quant à la contrainte (4.14), elle permet au solveur d'éviter de considérer inutilement des valeurs de variables lors de l'exploration de l'espace de recherche : le solveur n'a pas besoin d'agir sur $fdebut_i$ si le plan n'est pas retenu et n'inclut pas un extrait sélectionné, autrement dit, si $fnbr_i = 0$.

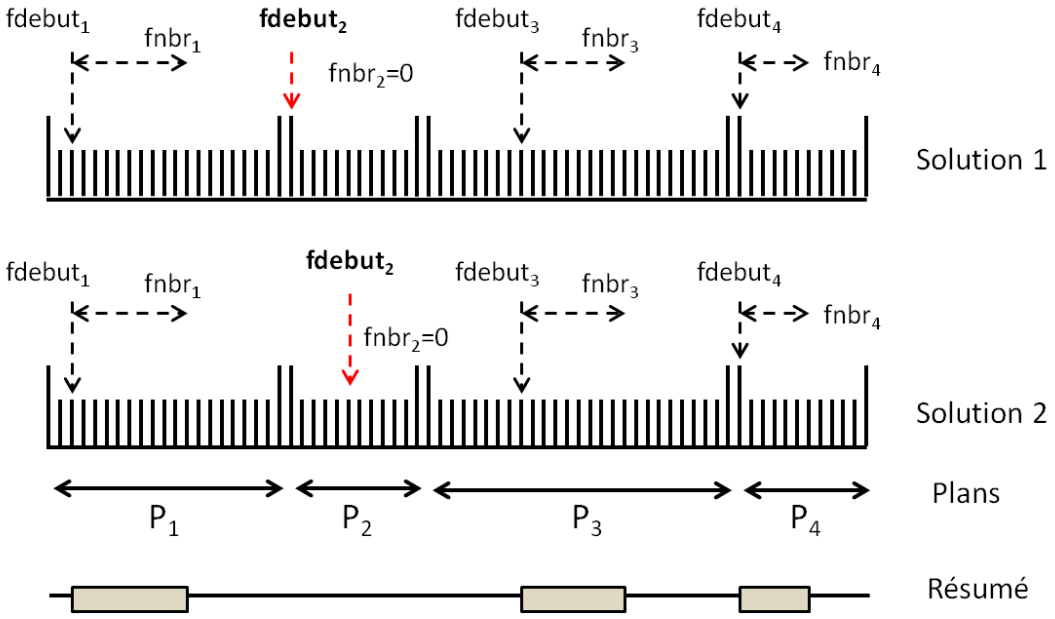

FIGURE 4.6 : Cas de non-utilisation de la contrainte (4.14 : exemple de résumés identiques retournés par le solveur mais considérés comme deux solutions différentes ($fdebut_2$ étant différentes).

La figure 4.6 illustre un exemple de deux solutions d'un même résumé, retournées par le solveur et considérées différentes en l'absence de la contrainte (4.14). En utilisant la contrainte (4.14), le solveur retourne une unique solution où $fdebut_2$ est positionnée au début du plan 2. Comme nous le démontrerons par les expériences présentées dans la section 4.4.4, cette contrainte a un impact significatif sur l'efficacité du solveur, c'est-à-dire sur son temps de réponse.

4.4.2.2 Les contraintes globales

Celles-ci s'appliquent sur l'ensemble du résumé. Un exemple de ce type de contraintes est la couverture temporelle et la répartition des extraits sur la totalité de la vidéo, c'est-à-dire la capacité à représenter les différentes parties de la vidéo dans le résumé.

Un autre exemple de contrainte globale est la durée du résumé. C'est l'un des critères les plus importants qu'un utilisateur exprime habituellement. Cette contrainte est définie

dans l'équation (4.15) :

$$dmin \leq \sum_{i=1}^{p} fnbr_i \leq dmax \tag{4.15}$$

où $dmin$ et $dmax$ désignent la durée maximale et la durée minimale du résumé. La spécification d'un intervalle pour la durée souhaitée du résumé au lieu d'une valeur unique, est importante. Cela donne au solveur la liberté nécessaire pour explorer des solutions voisines de la durée souhaitée et augmente les chances de trouver rapidement une solution réalisable. La section 4.4.4 présente une étude expérimentale de ce point.

4.4.2.3 Les contraintes d'élagage

Ce type de contraintes est utilisé pour éliminer les parties de la vidéo qui ne respectent pas un ensemble de critères. Elles peuvent par exemple être liées à un attribut. Cela est exprimé par la contrainte (4.16) où $plan\_Attribut_k[i]$ correspond à la présence de l'attribut $Attribut_k$ dans la i-ème plan. Cela peut aussi être lié à la longueur de l'extrait sélectionné (4.17) où $\vee$ désigne l'opérateur logique *ou* et $duree\_extrait\_min$ est un seuil présentant la durée minimale d'un extrait pouvant être inclus dans le résumé.

$$(fnbr_i \times plan\_Attribut_k[i]) = 0 \tag{4.16}$$

$$(fnbr_i \geq duree\_extrait\_min) \vee (fnbr_i = 0) \tag{4.17}$$

La contrainte (4.16) signifie que si $fnbr_i > 0$, c'est-à-dire qu'un extrait est sélectionné à partir du plan $plan_i$, l'attribut $Attribut_k$ ne doit pas être présent dans ce plan ($plan\_Attribut_k[i] = 0$). Quant à la contrainte (4.17), elle assure que chaque extrait sélectionné a une durée minimale de $duree\_extrait\_min$.

4.4.2.4 Les contraintes de voisinage

Des contraintes peuvent également être exprimées sur les voisinages des extraits. Par exemple, dans les vidéos de sport, les événements importants sont généralement suivis d'applaudissements. L'équation (4.18) illustre un exemple de telles contraintes.

$$(fnbr_i \neq 0) \implies (plan\_Attribut_k[i+1] \neq 0) \tag{4.18}$$

Ici, pour qu'un extrait soit sélectionné à partir du plan $plan_i$, l'attribut $Attribut_k$ doit être présent dans le plan qui le suit, $plan_{i+1}$.

4.4.2.5 Ajout de fonctions de coût à optimiser

Comme mentionné précédemment, la programmation par contraintes donne la possibilité de définir une fonction à optimiser. Ainsi, nous laissons le solveur choisir entre différentes solutions et favoriser certaines d'entre elles en maximisant ou en minimisant un critère donné. Ce critère peut être par exemple le nombre d'images qui contiennent un attribut spécifique. Il peut également être le nombre d'images entre la fin de l'extrait et la fin du plan. Demander au solveur de minimiser ce nombre revient à demander de sélectionner, de préférence, des extraits vers la fin des plans.

Une fonction de coût est associé pour chaque critère à optimiser. La fonction de coût g_{opt} globale peut être décrite sous la forme suivante :

$$g_{opt} = \sum_i \alpha_i . Opt_i \tag{4.19}$$

où Opt_i représente un coût à optimiser et α_i est un poids qui lui est associé. Un poids peut être positif ou négatif selon le type d'optimisation (minimisation ou maximisation). C'est un paramètre de haut niveau qui reflète l'importance du critère correspondant et qui n'est pas toujours évident ni facile d'ajuster.

4.4.3 Modèle #2 : stratégie d'évaluation

Un des problèmes évoqués lors de l'évaluation des approches de création automatique de résumés vidéo est la difficulté de comparer l'approche proposée avec celles déjà existantes. Le domaine est trop large et chaque solution est évaluée sur un type différent de vidéos en considérant des critères différents. Par ailleurs, et comme expliqué précédemment, il n'existe pas de bases vidéos publiques et communes pouvant servir à comparer différentes solutions. Cela rend toute étude comparative impossible à faire.

Nous avons choisi de réaliser les expérimentations sur des vidéos de matchs de tennis. Plusieurs raisons nous ont conduit à ce choix. D'abord, les matchs de tennis sont bien structurés. Ensuite, les parties non intéressantes peuvent être facilement annotées. Enfin, des résumés éditoriaux des matchs importants sont généralement disponibles et peuvent être utilisés pour l'évaluation. Un résumé éditorial est un résumé créé manuellement par des experts de la télévision.

Dans ce contexte, les résumés éditoriaux (RE) peuvent être utilisés pour dériver des mesures objectives de la qualité des résumés générés. En l'occurrence, nous avons proposé de calculer l'intersection entre les résumés éditoriaux et les résumés générés automatiquement. Comme critère de mesure, nous avons retenu le rapport entre cette

intersection et la durée des résumés générés automatiquement, que nous avons noté « $\bigcap$ *avec RE* ». En d'autres termes, nous avons choisi de calculer le pourcentage de l'intersection entre les deux résumés par rapport à la durée du résumé créé automatiquement. Bien évidemment, l'objectif n'est pas d'avoir une parfaite correspondance entre les deux résumés. Notons que deux résumés éditoriaux du même match, générés par deux journalistes différents sont généralement différents. L'idée étant d'utiliser cette mesure pour effectuer une évaluation relative des différents résumés générés automatiquement par rapport aux résumés éditoriaux.

Nous avons proposé d'utiliser une seconde mesure qui s'appuie sur les parties *non intéressantes* de la vidéo (*PNI*). Nous avons demandé à des volontaires d'annoter toutes les parties de la vidéo du match qu'ils considèrent comme non intéressantes et qui ne devraient pas être présentes dans un résumé. L'évaluation consiste ensuite à mesurer la capacité de la méthode de création de résumés d'écarter ces parties. Cela est mesuré à travers le calcul de l'intersection entre les partie non intéressantes et les résumés générés automatiquement.

Afin de pouvoir évaluer notre méthode et montrer ses avantages, nous avons proposé et utilisé trois méthodes basiques (*baseline*) pour la création de résumés automatiques :

- une sélection purement aléatoire de plans (RS) ;
- une sélection purement aléatoire de plans de jeu (RG) ;
- une sélection uniforme et périodique de plans de jeu (UG) : la sélection prend en considération la durée totale de la vidéo et la durée souhaitée du résumé.

Sachant que pour RG et UG, un plan de jeu est un plan qui correspond à un angle de prise de vue qui couvre l'ensemble du terrain de tennis et qui montre un échange de balle entre les joueurs.

4.4.4 Modèle #2 : expérimentations et évaluation

Dans cette section, nous proposons d'effectuer tout d'abord une évaluation basée sur une description du résultat pour montrer entre autres la souplesse de notre solution. Nous montrons en particulier que nous pouvons facilement ajouter de nouvelles contraintes afin de modifier le résumé. Nous étudions aussi l'impact des contraintes de modélisation sur l'efficacité de l'approche. Dans une deuxième partie, nous nous intéressons à l'évaluation de la qualité des résumés générés.

En plus des contraintes de modélisation (4.13) et (4.14), nous avons défini un ensemble de contraintes pour créer le résumé. Nous avons défini ces règles selon nos points de vue. Ces contraintes sont les suivantes :

- la durée du résumé : la durée d'un résumé doit être dans un intervalle, donné comme un paramètre d'entrée qui peut être choisi par l'utilisateur final ;

- les segments de parole ne doivent pas être coupés. L'idée est d'éviter, dans le résumé, de couper une personne qui parle ;

- chaque extrait sélectionné doit contenir des applaudissements, ou doit avoir, comme adjacent, un extrait sélectionné qui contient des applaudissements. Cette contrainte assure que chaque bloc d'extraits (un ensemble d'extraits sélectionnés adjacents) dans le résumé, contienne obligatoirement des applaudissements. Ici, les applaudissements indiquent généralement que quelque chose d'intéressant vient de se produire ;

- pour qu'un extrait soit retenu, il faut que sa durée dépasse une durée minimale pour éviter les extraits courts isolés. Le seuil de durée minimum a été fixé à quelques secondes pour qu'il soit compréhensible. Le même principe décrit précédemment est appliqué pour cette contrainte : chaque extrait sélectionné doit avoir une durée minimale ou doit être adjacent à un autre extrait sélectionné ;

- afin d'augmenter le nombre d'extraits intéressants, nous avons proposé de maximiser globalement la présence des applaudissements dans le résumé ;

- la somme des durées des plans de jeu est maximisée. Les plans de jeu sont détectés en utilisant le descripteur de la couleur dominante (DCD) présenté dans la section 4.2. Nous avons calculé le pourcentage de la couleur dominante pour chaque image de la vidéo et nous avons sélectionné les plans composés d'une majorité d'images contenant une couleur dominante (qui correspond à la couleur du terrain) avec un pourcentage d'au moins 20%. Ce seuil a été fixé expérimentalement et il a été appliqué sur une base données contenant des matchs de tennis de tournois différents avec couleurs de terrain et prises de caméras différentes. Les résultats de détection des plans de jeu sont fiables ;

- la présence de visages dans les extraits sélectionnés est minimisée. L'idée est de minimiser le nombre d'images où un visage d'une hauteur minimale de 20 pixels est détecté. Les parties qui montrent des visages en gros plan sont souvent moins

intéressants que celles qui montrent le jeu et qui ne contiennent pas de visages de cette taille ;

- dans le résumé, il est important d'intégrer la séquence de jeu qui montre la balle de match. Ceci est identifié comme étant le dernier plan de jeu dans la vidéo.

Les contraintes d'optimisation ont été définies par une fonction de coût où le même poids a été attribué à chaque attribut.

4.4.4.1 Description de l'ensemble de données de test

Pour les expérimentations avec ce deuxième modèle, nous avons utilisé un deuxième ensemble de données de test contenant 4 matchs de tennis du tournois de Roland Garros 2013. Il est désigné par $Ensemble_2 = \{M_1, M_2, M_3, M_4\}$. La durée totale des vidéos de cet ensemble est de 8 heures et 22 minutes.

Les deux premiers matchs possèdent des résumés éditoriaux d'une durée de 20 minutes chacun. Pour les deux autres matchs (M_3 et M_4), nous avons utilisé une annotation manuelle des parties non intéressantes. Cette annotation a été réalisée par des volontaires indépendants. Ils s'agit de collègues qui n'ont pas participé à nos travaux de recherche et qui sont plutôt des fans de tennis.

4.4.4.2 Analyse de performance du modèle #2

La formulation de toutes les contraintes et leur mise en œuvre avec Choco était directe et c'est l'un des principaux points forts de l'approche. Le modèle proposé est très convenable et nous a permis d'exprimer toutes les contraintes auxquelles nous avons pensé.

Afin d'évaluer l'efficacité de notre modèle, nous avons d'abord étudié la contrainte de modélisation 4.14. Nous rappelons que cette contrainte a pour but d'éviter l'exploration des possibilités inutiles par le solveur : si $fnbr_i = 0$, cela signifie que le plan $plan_i$ n'est pas sélectionné et qu'il n'est pas nécessaire d'agir sur l'autre variable associée, $fdebut_i$.

Pour cela, nous avons généré deux résumés du match M_2, avec et sans tenir compte de la contrainte (4.14). La durée souhaitée a été réglée entre 4 et 5 min. Les résultats obtenus sont décrits dans le tableau 4.4.

Ce tableau montre clairement que la contrainte (4.14) a un impact direct sur l'efficacité du solveur. Cette contrainte permet au solveur de retourner, beaucoup plus rapidement, un nombre plus élevé de solutions. De plus, en terme d'intersection avec le résumé éditorial, ces solutions sont de meilleure qualité. Après 2 min, et avec la contrainte (4.14),

	Sans la contrainte 4.14		avec la contrainte 4.14	
	$\bigcap$ *avec RE*	# de solutions	$\bigcap$ *avec RE*	# de solutions
Après 2 min	17%	12	20%	37
Après 30 min	20%	54	23%	441

TABLE 4.4 : Resultats sur M_2. Impact de la contrainte 4.14.

le solveur renvoie 37 solutions, dont la dernière solution retournée inclut 20% du résumé éditorial. Pour obtenir le même résultat sans l'utilisation de la contrainte 4.14, il faut attendre 30 minutes.

Nous avons également étudié l'influence de la contrainte de durée 4.15 sur la qualité des résumés de M_2. Trois cas ont été considérés : (1) en spécifiant une durée fixe, (2) en spécifiant un intervalle de 1 minute, (3) en précisant uniquement une durée minimale. Les résultats obtenus sont présentés dans le tableau 4.5. Ces résultats sont obtenus après 1 heure d'exécution. Ils montrent que le meilleur compromis est obtenu lors de la spécification d'un intervalle. Cela donne la liberté au solveur d'explorer plusieurs solutions, tout en restant dans le voisinage de la durée souhaitée. Par contre, en terme d'intersection avec le résumé éditorial, le meilleur résumé est obtenu en laissant plus de liberté au solveur.

Durée souhaitée (D)	$\bigcap$ *avec RE*	Durée du résumé
$D = 5\ min$	21%	5 min
$5\ min \geq D \geq 4\ min$	27%	4 min 51 s
$D \geq 5\ min$	28%	6 min 51 s

TABLE 4.5 : Resultats sur M_2. Impact de la contrainte de durée.

Nous avons effectué une dernière expérience sur la variation de la qualité du résumé par rapport au temps d'exécution laissé au solveur. En utilisant le match M_2, encore une autre fois, nous avons demandé au solveur de créer un résumé d'une durée comprise entre 4 et 5 minutes, et nous avons analysé les solutions obtenues après 20 s, 2 min, 30 min... d'exécution. Les résultats obtenus sont représentés dans la Figure 4.7. Une proportion de 17% du résumé obtenu après 20 secondes du lancement de la recherche de solutions, viennent du résumé éditorial. Ce ratio augmente en fonction du temps passé par le solveur à explorer l'espace de recherche. Il atteint 33% au bout de 3 heures.

Jusque là, nous avons fourni au solveur des contraintes d'optimisation sans spécifier aucun seuil. Toutefois, compte tenu du type des vidéos d'entrée, si l'on considère qu'au moins la moitié du résumé doit être des plans de jeu et 5% devraient présenter des applaudissements, nous pouvons fournir au solveur ces seuils afin de l'orienter et lui éviter d'explorer des possibilités inutiles. Le tableau 4.6 montre qu'après l'application de

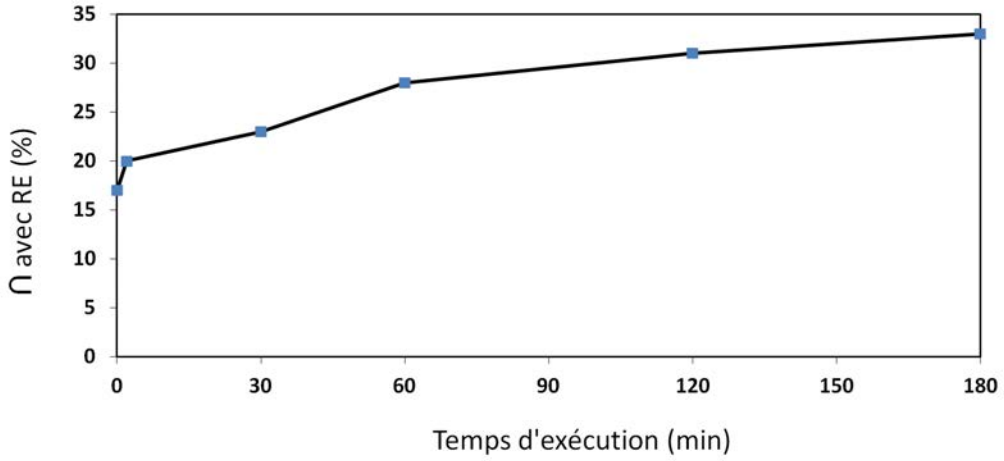

FIGURE 4.7 : Qualité du résumé en fonction du temps passé par le solveur.

ces deux seuils, nous obtenons un résumé dont le rapport d'intersection avec le résumé éditorial est égal à 30% en seulement 2 minutes.

	Sans utilisation de seuils	En utilisant des seuils
Temps d'exécution	180 min	2 min
$\bigcap$ *avec RE*	30%	30%

TABLE 4.6 : Résultats sur M_2. Impact de l'utilisation de seuils.

Cela montre que la manière avec laquelle le modèle est utilisé, a un impact important sur son efficacité. L'ajout de ces deux seuils simples et intuitifs permet de réduire considérablement le temps de réponse du solveur. Nous les avons donc utilisés pour l'ensemble des expériences qui sont présentés dans le reste de ce chapitre.

4.4.4.3 Étude sur la flexibilité du modèle #2

Un autre critère important des méthodes de création automatique de résumés est leur flexibilité, c'est-à-dire la capacité d'adapter facilement le résumé résultant.

Pour étudier ce critère, nous nous sommes intéressés à la répartition des extraits dans les deux résumés éditoriaux de M_1 et de M_2. L'analyse de cette répartition nous a permis de remarquer que pour M_2, un pourcentage de 86,35% du résumé éditorial vient de la deuxième moitié du match. Si ceci doit être pris en compte lors de la génération du résumé, nous n'avons qu'à ajouter une contrainte sur la répartition des extraits (CR) qui maximise la différence entre la durée totale des extraits sélectionnés provenant de la deuxième moitié, et la durée totale des extraits sélectionnés provenant de la première moitié du match. Nous avons généré deux résumés de M_2 de durées de 4 à 5 min, avec

et sans l'utilisation de la contrainte CR. La Figure 4.8 présente la répartition des extraits de ces deux résumés. En outre, en utilisant cette contrainte, la qualité du résumé généré par rapport au résumé éditorial a été augmentée de 30% à 42%.

FIGURE 4.8 : Impact de la contrainte CR sur la répartition des segments (en noir) dans les résumés de M_2.

Comme cette propriété de répartition n'est pas adoptée par tous les résumés éditoriaux des matchs de tennis, nous ne l'avons pas incluse dans l'ensemble des contraintes.

L'objectif de cette expérimentation est de montrer qu'il est possible, avec notre méthode, de prendre en compte une règle ou une propriété supplémentaire ou même de modifier la structure des résumés, sans modifier aucune autre contrainte ni toucher au modèle.

4.4.4.4 Évaluation de la qualité des résumés

Pour évaluer notre modèle #2, nous l'avons comparé avec les trois méthodes basiques (RS, RG et UG) décrites précédemment. Les résultats obtenus en appliquant notre modèle sur les matchs M_1 et M_2 sont présentés dans le tableau 4.7. Quatre résumés de durées différentes (4-5 min, 9-10 min, 14-15 min et 19-20 min) ont été générés pour chaque match. L'évaluation porte sur leur intersection avec les résumés éditoriaux. Quant aux matchs M_3 et M_4, l'évaluation a été effectuée par rapport à la proportion de segments non-intéressants dans les résumés générés. Les résultats obtenus figurent dans le tableau 4.8. Lors de la génération de résumés, nous avons arrêté le solveur après 2 minutes et nous avons retenu la dernière solution trouvée.

Les deux tableaux 4.7 et 4.8 montrent que notre solution surpasse largement les méthodes basiques. Entre 21% et 43% de nos résumés sont présents dans les résumés éditoriaux. Nous pouvons également remarquer que notre solution permet d'obtenir les meilleurs résultats en écartant les parties non intéressantes. Celles-ci représentent dans le pire des cas 22% de la durée de notre résumé. Dans le tableau 4.8, il y a des valeurs manquantes pour RG et UG qui sont des résumés basiques ne contenant que des plans de jeu. La vidéo M_4 correspond à un match court qui ne contient pas assez de plans de

	M_1				M_2			
Durée	4-5 min	9-10 min	14-15 min	19-20 min	4-5 min	9-10 min	14-15 min	19-20 min
RS	1%	6%	9%	9%	9%	14%	16%	14%
RG	8%	15%	10%	13%	15%	19%	25%	21%
UG	8%	16%	9%	12%	18%	18%	25%	22%
Notre solution	43%	32%	28%	21%	30%	28%	32%	33%

TABLE 4.7 : Résultats sur M_1 et M_2. Évaluation par rapport au résumé éditorial (RE).

	M_3				M_4			
Durée	4-5 min	9-10 min	14-15 min	19-20 min	4-5 min	9-10 min	14-15 min	19-20 min
RS	41%	50%	44%	40%	32%	46%	32%	33%
RG	16%	28%	23%	22%	21%	23%	-	-
UG	31%	27%	23%	22%	23%	27%	-	-
Notre solution	15%	11%	13%	9%	10%	8%	15%	22%

TABLE 4.8 : Résultats sur M_3 et M_4. Proportion de segments non intéressants dans les résumés.

jeu pour atteindre la durée souhaitée (14-15 min et 19-20 min).

4.4.4.5 Impact de la fiabilité de la détection des attributs

Comme nous l'avons déjà décrit, certaines contraintes sont exprimées sur des attributs de bas niveau détectés automatiquement à partir de la vidéo. Ces attributs présentent inévitablement des erreurs, même si les techniques de détection d'attributs ont connu un progrès significatif durant ces dernières années. Le but de cette expérience est d'évaluer l'impact des erreurs introduites durant l'étape de détection des attributs sur la qualité des résumés générés.

Pour cela, nous avons sélectionné deux matchs de tennis (M_2 et M_3) à partir de l'ensemble $Ensemble_2$ (voir section 4.4.4.1) et nous avons corrigé manuellement les attributs détectés pour ces matchs. Ces matchs ont été choisis de telle sorte que nous pouvons utiliser les différents critères d'évaluation que nous avons proposés : l'intersection avec un résumé éditorial de 20 min pour M_2 et l'intersection avec les parties non intéressantes pour M_3. Les attributs relatifs aux visages et à la couleur dominante sont très fiables. Ils n'ont nécessité qu'une seule vérification en un seul passage. Les attributs d'applaudissements et de la parole ont cependant été complètement vérifiés et corrigés. Les résultats obtenus en utilisant les attributs corrigés pour M_2 et M_3 sont décrits dans le tableau 4.9. Nous avons comparé les résultats avant et après la correction des attributs en utilisant le modèle #2.

Durée	5 min		10 min		15 min		20 min	
Correction	avant	après	avant	après	avant	après	avant	après
M_2	30%	48%	28%	39%	32%	31%	33%	40%
M_3	15%	6%	11%	7%	13%	9%	9%	8%

TABLE 4.9 : Impact de la correction de la détection des attributs sur l'intersection avec RE pour M_2 et sur l'intersection avec PNI pour M_3 en utilisant le modèle #2.

Dans l'ensemble, et indépendamment des critères d'évaluation, la correction des attributs permet à notre méthode de générer des résumés de meilleure qualité. En particulier, les limites des segments de parole n'étaient pas souvent positionnées précisément dans la vidéo. La correction de ces limites a permis donc de ne pas interrompre les commentateurs dans les résumés de M_2 et M_3. Ce genre d'erreurs a un impact négatif direct sur le résumé généré. Nous avons exprimé de fortes contraintes sur la parole dans les extraits à sélectionner. Nous avons également imposé que chaque extrait sélectionné doit contenir des applaudissements. Ainsi, une légère modification des limites des segments de ces deux attributs a nécessairement un impact sur l'espace de recherche sur lequel agit le solveur et par conséquent sur le résumé généré. Cette expérience montre aussi que nos contraintes sont très pertinentes puisque la « qualité » s'est considérablement améliorée après la correction des attributs.

4.4.5 Modèle #2 : limites du modèle

Ce modèle est très flexible et permet d'exprimer une large variété de contraintes. Cependant, il reste toujours très dépendant des frontières des plans.

Avec ce modèle, il est par exemple difficile de définir des contraintes permettant de sélectionner un extrait qui s'étale sur deux plans ou plus. Il est difficile aussi d'imposer un nombre précis de plans (ou un nombre minimal de plans) qui doivent être sélectionnés sans interruption pour qu'un événement pertinent et s'étalant sur plusieurs plans soit entièrement sélectionné. Il est difficile, par conséquent, d'imposer une durée minimale sur une succession de plans sélectionnés afin de rendre la sélection plus lisible et compréhensible.

D'autre part, la segmentation est appliquée sur le signal visuel et ne correspond pas forcément à la segmentation qui serait créée si l'on considère d'autres attributs de bas niveau liés au signal audio par exemple.

Ce modèle présente ainsi quelques limites qui nous ont amené à réfléchir à un modèle alternatif s'affranchissant complètement de la segmentation en plans. C'est l'objet du chapitre suivant.

4.5 Conclusion

Dans ce chapitre, nous avons proposé une nouvelle solution pour le problème de création automatique de résumés vidéo, basée sur la programmation par contraintes. L'avantage principal de notre solution est sa capacité à séparer clairement les règles de production de l'algorithme de génération de résumés. Elle permet également aux utilisateurs d'exprimer facilement des contraintes de haut niveau et des préférences grâce aux deux modèles proposés. Dans le chapitre suivant nous proposerons un troisième modèle qui ne dépend pas des frontières des plans.

Chapitre 5

Modélisation sans segmentation

Sommaire

5.1 Introduction

Pour éviter toutes les limitations liées à une segmentation prédéfinie, nous proposons un troisième modèle avec lequel le résumé final est un ensemble d'extraits, un extrait étant un segment de la vidéo d'entrée qui ne dépend d'aucune segmentation prédéfinie. Ce troisième modèle est appliqué sur la totalité de la vidéo en utilisant des contraintes globales. La figure 5.1 illustre le passage du modèle #2 qui dépend des frontières des plans au modèle #3 qui ne dépend d'aucune segmentation de la vidéo d'entrée.

Dans notre modèle #3, nous reprenons presque le même schéma que celui présenté dans la figure 4.1. Un module de prétraitement vidéo est utilisé pour l'extraction des attributs de bas niveau. Ce dernier est appliqué sur les signaux visuel et audio de la vidéo d'entrée. Un ensemble de contraintes est exprimé sur les attributs appropriés extraits en fonction du type de la vidéo d'entrée. Associé à un ensemble supplémentaire

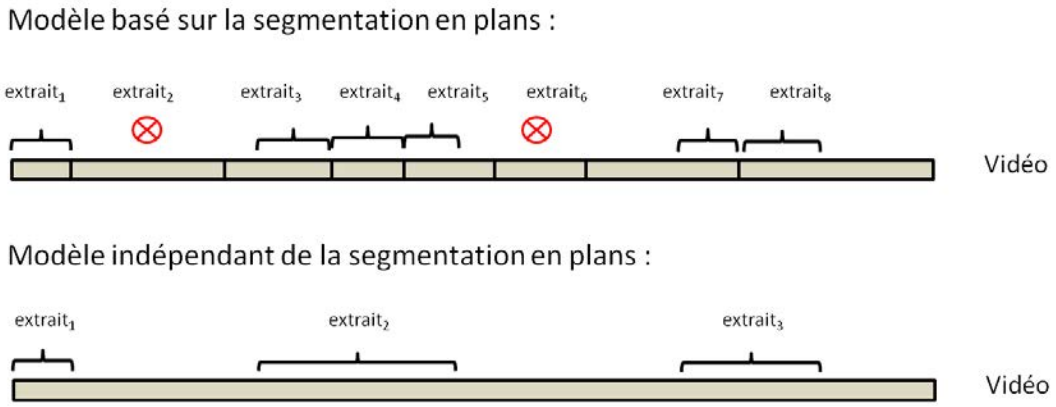

FIGURE 5.1 : Passage d'une modélisation basée sur les plans à une modélisation indépendante des frontières des plans.

de contraintes exprimant les préférences des utilisateurs, le solveur crée un résumé vidéo qui satisfait toutes les contraintes considérées.

5.2 Troisième modèle : modèle #3

5.2.1 Modèle #3 : modélisation

Le résumé est un ensemble de n extraits vidéo. n est un nombre prédéfini d'extraits à sélectionner à partir de la totalité de la vidéo. Nous avons choisi d'extraire un nombre fixe d'extraits afin de simplifier la modélisation du problème en utilisant la PPC. Dans la pratique, nous ajustons le modèle avec différentes valeurs de n et nous optons pour la valeur qui génère la meilleure solution.

Chaque extrait est défini par trois composantes : un instant de début, un instant de fin et une durée. Bien évidemment, la troisième composante peut être calculée en utilisant les deux autres, mais pour des raisons d'efficacité, il est apparu préférable de créer trois variables pour chaque extrait plutôt que de calculer une valeur associée à chaque fois que nous en avons besoin. Dans la programmation par contraintes, le calcul d'une grandeur est assuré par une contrainte et une variable supplémentaires. Dans l'absence de la troisième variable (durée), le solveur est obligé de la calculer à chaque fois à l'aide d'une contrainte et d'une variable de plus.

La figure 5.2 illustre un cas d'usage de sélection d'extraits en utilisant le modèle #3. Tous les extraits sélectionnés et assemblés forment le résumé final et les variables de

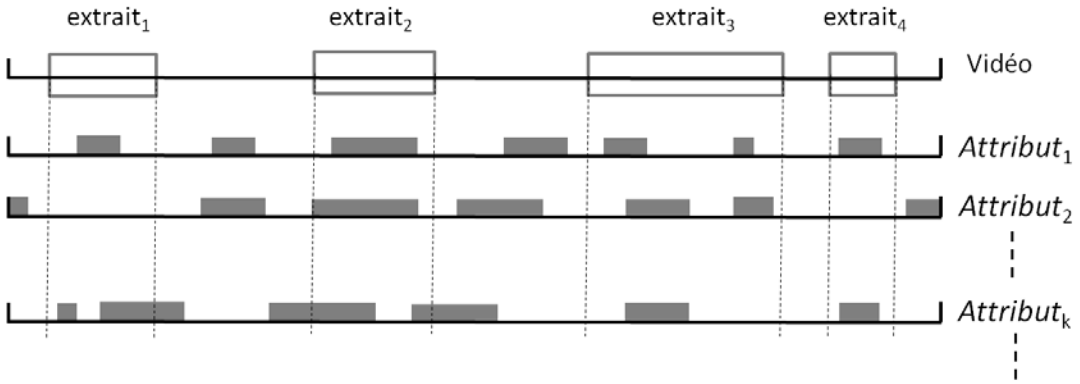

FIGURE 5.2 : Troisième modélisation : sélection de n extraits en tenant compte des attributs et indépendamment de toute segmentation.

notre modèle sont définies comme suit :

$$Extraits \begin{cases} Debut\_extrait = \{debut\_extrait_i,\ i = 1..n\} \\ Fin\_extrait = \{fin\_extrait_i,\ i = 1..n\} \\ Duree\_extrait = \{duree\_extrait_i,\ i = 1..n\} \end{cases} \tag{5.1}$$

où les domaines de $debut\_extrait_i$, $fin\_extrait_i$ et $duree\_extrait_i$ sont :

$$\begin{aligned} debut\_extrait_i &\in [debut\_video\ ..\ fin\_video - duree\_extrait\_min] \\ fin\_extrait_i &\in [debut\_video + duree\_extrait\_min\ ..\ fin\_video] \\ duree\_extrait_i &\in [duree\_extrait\_min\ ..\ duree\_extrait\_max] \end{aligned} \tag{5.2}$$

Dans ce modèle, toutes les variables *Debut_extrait* ont le même domaine initial (de même que les variables *Fin_extrait* et les variables de durée *Duree_extrait*). Cela engendre un large domaine de recherche pour chaque variable. Cependant, en comparaison avec le modèle #2, ceci permet de surmonter les limitations provenant de la segmentation en plans et rend le modèle plus souple et plus intuitif.

Les contraintes de durée sont exprimées au niveau de la durée de chaque extrait sélectionné. Elles peuvent être définies directement sur les domaines des variables, tel que défini dans l'équation 5.2, ou par les deux contraintes suivantes :

$$duree\_extrait_i \geq duree\_extrait\_min \tag{5.3}$$

$$duree\_extrait_i \leq duree\_extrait\_max \tag{5.4}$$

où $duree\_extrait\_min$ est la durée minimale d'un extrait et $duree\_extrait\_max$ est la durée

maximale qu'un extrait ne doit pas dépasser.

Afin d'assurer la cohérence de notre modèle (i.e. des valeurs cohérentes sont attribuées aux différentes variables) et d'optimiser l'espace de recherche (réduire l'espace de recherche en supprimant les valeurs impossibles), les contraintes de modélisation suivantes sont exprimées :

5.2.1.1 La dépendance intra-extrait

Les valeurs attribuées au début, à la fin et à la durée de chaque extrait doivent être cohérentes :

$$fin\_extrait_i = debut\_extrait_i + duree\_extrait_i \tag{5.5}$$

5.2.1.2 La dépendance inter-extraits

Notre modèle gère la relation des différents extraits entre eux. Afin de réduire l'espace de recherche du solveur, nous pouvons choisir soit d'ordonner les extraits ou bien, tout simplement, de faire en sorte que les extraits ne se chevauchent pas.

Ordonnancement des extraits : $extrait_i$ devrait précéder $extrait_{i+1}$. En d'autres termes, $extrait_{i+1}$ commence après la fin de $extrait_i$. Ceci peut être exprimé simplement par l'inégalité suivante :

$$fin\_extrait_i < debut\_extrait_{i+1} \tag{5.6}$$

Non chevauchement des extraits : ici, l'ordre entre les extraits n'est pas pris en considération et n'intervient pas dans l'affectation de valeurs au différentes variables. Cependant, il n'y a pas d'intersection entre les différents extraits qui doivent être totalement distincts. Il ne doit pas y avoir de chevauchement entre les extraits deux à deux. Dans ce cas, la contrainte suivante peut être utilisée :

$$(debut\_extrait_i > fin\_extrait_j) \vee (fin\_extrait_i < debut\_extrait_j) \tag{5.7}$$

Les contraintes de modélisation sont nécessaires que ce soit avec un modèle à deux variables (*Debut_extrait* et *Fin_extrait*) ou un modèle à trois variables (tel qu'il est le cas actuellement : *Debut_extrait*, *Fin_extrait* et *Duree_extrait*). Dans le premier cas, la

contrainte de dépendance intra-extrait serait définie comme suit :

$$debut\_extrait_i < fin\_extrait_i \tag{5.8}$$

5.2.2 Modèle #3 : formulation de contraintes

Notre modèle s'appuie principalement sur la comparaison d'un ensemble de segments d'attributs et un ensemble d'extraits (comme le montre la figure 5.2). Pour assurer une flexibilité et une capacité à exprimer différents types de contraintes, notre modèle doit être capable d'exprimer différentes opérations entre ces deux ensembles. Les relations d'Allen couvrent bien ces opérations. Le modèle sera capable d'exprimer n'importe quelle contrainte souhaitée par un utilisateur, s'il est capable de réaliser les relations d'Allen. Ceci est décrit dans cette section.

Étant donné que la modélisation que nous avons proposée pour résoudre le problème de création automatique de résumés vidéo est une modélisation temporelle, nous proposons d'utiliser l'algèbre Allen pour représenter les différentes relations entre nos composantes du modèle (extraits et segments d'attributs). L'algèbre Allen [All83] est une logique temporelle qui définit toutes les relations possibles entre des intervalles de temps et fournit une table de composition. Par conséquent, il décrit un ensemble de 13 relations atomiques conjonctives ou disjonctives caractérisant une situation temporelle entre deux intervalles précis (figure 5.3). Le problème de satisfaction pour l'algèbre d'Allen est un problème NP-complet.

Concrètement, avec notre modèle et en s'appuyant sur l'algèbre d'Allen, nous sommes en mesure d'exprimer toute contrainte possible par la manipulation de la relation entre un ensemble d'extraits d'une part et un ensemble de segments d'attributs d'autre part. Ici, les contraintes utilisées pour créer le résumé peuvent être élémentaires ou complexes. Les contraintes élémentaires ne nécessitent qu'une seule relation d'Allen pour être exprimée, un exemple de ce type de contraintes, les extraits doivent nécessairement contenir un segment d'un attribut donné ($extrait_i\ Di\ Attribut_k\,[j]$), un attribut doit être présent pendant un extrait ($extrait_i\ D\ Attribut_k\,[j]$) ou aussi, les extraits doivent commencer par un attribut donné ($extrait_i\ Si\ Attribut_k\,[j]$). Quant aux contraintes complexes, elles peuvent être exprimées en utilisant une combinaison de plusieurs relations d'Allen. La contrainte « Les extraits ne doivent couper aucun segment de parole » est un exemple d'une contrainte complexe qui dérive d'une combinaison de relations d'Allen. Cette contrainte peut être écrite en utilisant six relations d'Allen (P, Pi, Di, Si, Fi, E) :

$$\forall i, j,\ (extrait_i\ P\ Attribut_k\,[j]) \vee (extrait_i\ Pi\ Attribut_k\,[j]) \vee (extrait_i\ Di\ Attribut_k\,[j]) \vee$$

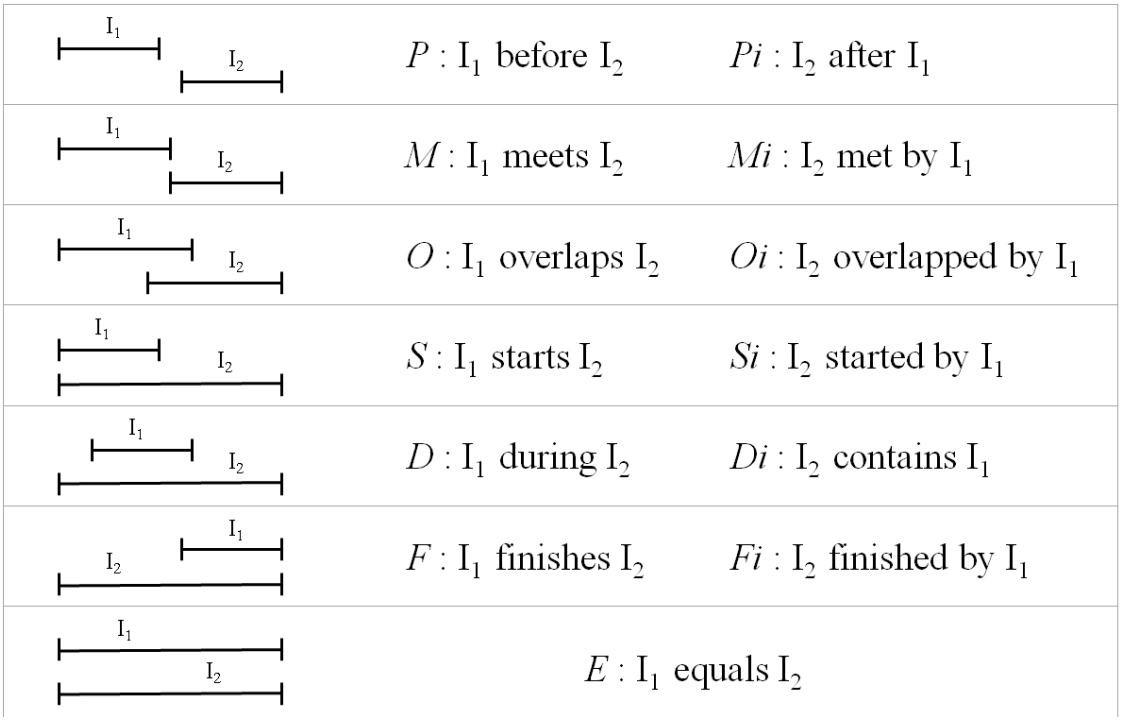

FIGURE 5.3 : Algèbre d'Allen : illustration des différentes relations possibles entre deux intervalles.

$(extrait_i\ Si\ Attribut_k\,[j]) \vee (extrait_i\ Fi\ Attribut_k\,[j]) \vee (extrait_i\ E\ Attribut_k\,[j])$.

À l'exception des relations « after » et « before » qui assurent la disjonction entre les extraits et les segments d'attributs, tout le reste des relations d'Allen nécessite le calcul d'une intersection entre elles. Par conséquent, une information supplémentaire peut être calculée et utilisée pour exprimer d'autres contraintes. Par exemple, nous pouvons calculer la taille de l'intersection entre les extraits et les segments d'un attribut spécifique. Cela donne la quantité de cet attribut dans le résumé final. Ceci est défini par $Qte\_attribut_k$.

En utilisant notre troisième modèle et l'algèbre d'Allen, on distingue deux types différents de contraintes qui peuvent être exprimées en plus des fonctions à optimiser :

5.2.2.1 Les contraintes liées au contenu vidéo

Ces contraintes prennent en compte le contenu audiovisuel du résumé final. Pour notre cas d'utilisation concernant le tennis (introduit dans la section 4.4.3 du chapitre précédent), nous utilisons les relations d'Allen déjà définies dans la figure 5.3 pour formuler les contraintes suivantes :

- chaque extrait contient (*contains*) au moins un segment de jeu (échange complet de

la balle entre les joueurs) ;

- chaque extrait contient (*contains*), recouvre (*overlaps*) ou est recouvert par (*overlapped by*) un segment d'applaudissements ;

- chaque extrait ne recouvre (*overlaps*) et n'est recouvert par (*overlapped by*) aucun segment de parole. Il n'est inclus (*during*) dans aucun segment de parole ;

- chaque extrait ne recouvre (*overlaps*) aucun segment de jeu (l'extrait peut être recouvert par (*overlapped by*) un segment de jeu) ;

- l'extrait vient avant (*before*) ou après (*after*) un segment de publicité (disjoint) ;

- le dernier extrait contient (*contains*) la balle de match ;

5.2.2.2 Les contraintes globales

Ces contraintes concernent le résumé dans sa totalité. Elles sont détaillées comme suit :

Durée du résumé : la contrainte de durée du résumé peut être formulée comme suit :

$$dmin \leq \sum_{i=1}^{n} duree\_extrait_i \leq dmax \tag{5.9}$$

où $dmin$ et $dmax$ sont la durée minimale et maximale d'un résumé.

Seuil de présence d'un attribut : selon le type de la vidéo et des préférences de l'utilisateur, le résumé final doit contenir une quantité minimale d'un attribut donné. Par exemple, il est intéressant de regarder les échanges de balle entre les joueurs pendant une période intéressante d'un résumé d'un match de tennis. Pour les programmes musicaux de télévision, le résumé favorise la musique par rapport à d'autres attributs. Par conséquent, la quantité d'un attribut doit dépasser un seuil défini, cela peut simplement être exprimé en utilisant cette contrainte :

$$Qte\_attribut_k \geq seuil \tag{5.10}$$

5.2.2.3 Les fonctions de coût à optimiser

En utilisant des fonctions à optimiser, le solveur crée un résumé qui maximise ou minimise la quantité d'un attribut donnée dans l'ensemble des solutions possibles. Par exemple, il serait intéressant de maximiser la quantité d'applaudissements dans notre

résumé afin d'inclure les parties les plus intéressantes d'une vidéo. La fonction de coût g_{opt} suivante peut être utilisée pour formuler les contraintes d'optimisation :

$$g_{opt} = \sum_k \alpha_k . Qte\_attribut_k \tag{5.11}$$

Les poids α_k correspondent à l'importance attribuée à chaque attribut à optimiser. Le poids peut être positif ou négatif selon le type d'optimisation (maximisation ou minimisation). En utilisant cette fonction de coût, le solveur crée un résumé qui maximise ou minimise la quantité d'un (ou plusieurs) attribut(s) donné(s).

5.2.3 Modèle #3 : implémentation des contraintes

L'expression de toutes les contraintes décrites ci-dessus en utilisant des contraintes standards proposées par les différents solveurs n'est pas une tâche facile. Les contraintes standards proposées s'appuient sur des opérateurs arithmétiques (par exemple, la différence, la somme, le maximum) et des opérateurs logiques (par exemple, la conjonction, la disjonction). Elles peuvent aussi être des contraintes globales prédéfinies (par exemple *alldifferent*, *cumulative*). L'ensemble de ces contraintes est décrit en détail dans le catalogue de contraintes [BCR05] [1]. Ces contraintes standards prédéfinies peuvent être utilisées d'une manière simple pour mettre en œuvre le modèle #1 et le modèle #2.

Quant au modèle #3, l'utilisation de ces contraintes standards uniquement pour exprimer nos contraintes globales n'est pas suffisante. Ces dernières s'expriment sur deux ensembles de segments et non pas sur deux segments isolés. Cela pose des problèmes de mémoire causés par le grand nombre de contraintes standards utilisées, et par conséquent, le grand nombre de variables intermédiaires générées par le solveur au cours la résolution du problème. Ce nombre dépend de deux paramètres : (1) le nombre de segments d'attributs extraits à partir de la vidéo (m segments d'attributs) et (2) le nombre d'extraits à sélectionner et à inclure dans le résumé final (n extraits). Dans ce contexte, deux boucles imbriquées sont nécessaires pour définir une contrainte globale. Le solveur crée une variable intermédiaire pour chaque contrainte de base utilisée à chaque itération de la boucle. Cela conduit à des complexités élevées en temps de réponse et en mémoire à cause de l'algorithme quadratique employé.

L'utilisation des contraintes standards pour la mise en place du modèle #3 et pour la formulation des règles de création de résumés, est donc difficile et très coûteuse car une contrainte standard ne s'applique que sur deux intervalles isolés. Cependant, notre modèle nécessite des opérations ensemblistes globales, appliquées sur deux ensembles

[1] Catalogue des contraintes prédéfinies. http://sofdem.github.io/gccat/

d'intervalles. Nous citons ici un exemple d'une contrainte globale qui illustre ce cas : $Ne\_pas\_couper(Extraits, Attribut_k)$ est une contrainte globale appliquée sur deux ensembles d'intervalles $Extraits$ et $Attribut_k$.

Pour une seule itération, cette contrainte globale nécessite 7 contraintes standards pour être exprimée sur seulement deux segments. Notons que chaque contrainte standard possède son propre algorithme de filtrage prédéfini :

$\forall i, j,\ (deb\_extrait_i \geq fin\_Attribut_k[j])$ or $(fin\_extrait_i \leq debut\_Attribut_k[j])$ or $((deb\_extrait_i \leq deb\_Attribut_k[j])\ and\ (fin\_extrait_i \geq fin\_Attribut_k[j]))$
où $deb\_extrait_i$ (resp. $fin\_extrait_i$) est le début (resp. la fin) d'un extrait i et $debut\_Attribut_k[j]$ (resp. $fin\_Attribut_k[j]$) est le début (resp. la fin) du segment $Attribut_k[j]$.

En utilisant cette formulation et en se basant sur ce qui précède, le solveur doit créer $m * n * 7$ variables intermédiaires, ce qui alourdit considérablement la résolution du problème. Ici, m est le nombre de segments de l'attribut $Attribut_k$ et n est le nombre d'extraits.

Pour faire face à ces problèmes, nous avons défini et implémenté nos propres contraintes globales dans CHOCO avec, pour chacune d'elles, un algorithme de filtrage spécifique. Ces nouvelles contraintes définissent des opérations ensemblistes appliquées sur deux ensembles d'intervalles. Une opération ensembliste peut avoir trois formes :

One-to-all **:** définit la relation entre un extrait donné et un ensemble de segments d'un attribut donné. Cela permet par exemple d'exprimer la contrainte suivante : le premier extrait dans le résumé doit commencer par un segment de musique ;

All-to-all **:** exprime la relation entre les extraits et les segments d'un attribut donné. Cela permet par exemple d'exprimer la contrainte suivante : tous les extraits doivent contenir au moins un segment d'applaudissement ;

One-to-one **:** définit la relation entre un extrait donnée et un segment d'un attribut spécifique. Cela permet par exemple d'exprimer la contrainte suivante : le dernier extrait dans le résumé doit contenir la balle du match.

Le solveur de contraintes CHOCO offre la possibilité d'implémenter facilement de nouvelles contraintes. Un utilisateur de CHOCO peut ainsi rajouter de nouvelles contraintes en spécifiant les trois étapes suivantes :

1. un algorithme de filtrage adapté ayant pour but de réduire les domaines de recherche des différentes variables impliquées dans l'expression de la contrainte ;

2. une condition d'arrêt indiquant que l'exploration en cours a échoué et qu'un retour sur trace (*backtracking*) est indispensable ;

3. une condition de satisfaction de la contrainte. C'est une dernière étape de validation des valeurs attribuées aux variables. Cette condition est utilisée pour éviter toute contradiction.

Avant de décrire nos contraintes globales, notons que pour réduire les domaines de recherche des variables, en PPC, trois types d'actions agissant sur les domaines de recherche des variables peuvent être réalisés :

1. l'ajustement de bornes : dans le cas d'un ajustement de la borne inférieure, il s'agit de supprimer toutes les valeurs qui sont inférieures à une certaine valeur et dans le cas d'un ajustement de la borne supérieure, il s'agit de supprimer toutes les valeurs qui sont supérieures à une certaine valeur ;

2. la suppression de valeur(s) : supprimer une (ou plusieurs) valeurs à partir d'un domaine de recherche d'une variable ;

3. l'instanciation : supprimer toutes les valeurs, excepté une, à partir d'un domaine de recherche d'une variable.

Dans les sections qui suivent, nous allons décrire les quatre nouvelles contraintes que nous avons définies, leurs fonctionnements et l'implémentation de leurs algorithmes de filtrage.

5.2.3.1 La contrainte « contient »

C'est une contrainte de type *All-to-all* qui exprime le fait que chaque extrait doit contenir un seul segment d'un attribut donné. Ici, un ajustement de la borne inférieure et de la borne supérieure de chaque domaine de recherche des différentes variables de décision est appliquée. Comme présenté dans l'équation 5.1 (page 77), un extrait est défini par deux variables $debut\_extrait_i$ et $fin\_extrait_i$. Notre nouvelle contrainte globale a pour but de filtrer des valeurs à partir des domaines de recherche de chacune de ces deux variables en ajustant leurs bornes inférieures et supérieures (et cela, pour les n extraits).

L'algorithme de filtrage est illustré par la figure 5.4. La borne supérieure de la variable de début de l'extrait i ($debut\_extrait_i$) est ajustée au début du segment d'attribut $Attribut_k$ [2], et la borne inférieure de la variable de fin de l'extrait i ($fin\_extrait_i$) est

ajustée à la fin du segment d'attribut $Attribut_k[0]$. En d'autres termes, les valeurs impossibles qui ne peuvent pas être attribuées à $debut\_extrait_i$ et à $fin\_extrait_i$ sont supprimées. Ça correspond à la partie hachurée dans la figure.

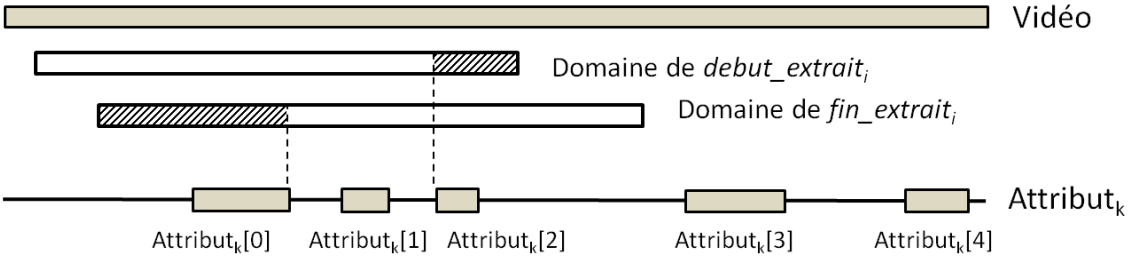

FIGURE 5.4 : Algorithme de filtrage de la contrainte « contient »

La condition d'arrêt de cet algorithme est la suivante : « il existe un extrait qui ne contient aucun segment de l'attribut en question ». Autrement dit, il existe un extrait $extrait_l$ tel que, entre la borne inférieure du domaine de recherche de $debut\_extrait_l$ et la borne supérieure du domaine de recherche de $fin\_extrait_l$, aucun segment de l'attribut n'existe.

Une fois que la propagation des contraintes est terminée et le filtrage est appliqué, les variables sont instanciées en satisfaisant l'ensemble de contraintes exprimées. La condition de satisfaction à vérifier afin de valider l'assignation des valeurs est :

$\forall i,\ \exists j,\ (debut\_extrait_i < debut\_Attribut\,[j]) \wedge (fin\_extrait_i > fin\_Attribut\,[j])$.

Une spécialisation de la contrainte « contient » peut être mise en œuvre pour assurer une contrainte de type *One-to-all* : spécifiquement l'extrait $extrait_l$ doit contenir un segment (n'importe lequel) d'un attribut donné.

5.2.3.2 La contrainte « ne pas couper »

Cette contrainte est également de type *All-to-all*. Elle est exprimée sur les variables afin d'interdire qu'un extrait coupe les segments d'un attribut donné. En d'autres termes, les extraits doivent commencer soit avant le début d'un segment d'attribut ou après sa fin et se terminer soit avant le début d'un segment d'attribut ou après sa fin. Cela peut s'exprimer à l'aide de la condition de satisfaction suivante :

$\forall i, j,\ ((debut\_extrait_i < debut\_Attribut\,[j]) \vee (debut\_extrait_i > fin\_Attribut\,[j])) \wedge ((fin\_extrait_i < debut\_Attribut\,[j]) \vee (fin\_extrait_i > fin\_Attribut\,[j]))$

L'algorithme de filtrage de cette contrainte est illustré par la figure 5.5. Pour le mettre en œuvre, un autre type d'action sur les domaines de recherche est utilisé. Il s'agit de la suppression de valeurs impossibles à partir du domaine de recherche de chaque variable

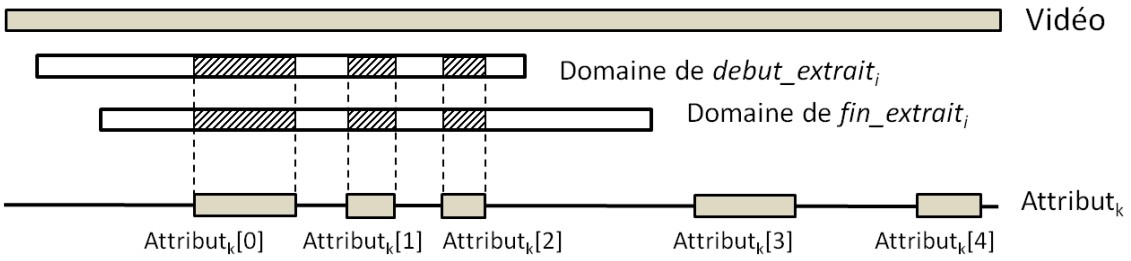

FIGURE 5.5 : Algorithme de filtrage de la contrainte « ne pas couper »

de décision. En effet, les valeurs appartenant aux segments de l'attribut en question (parties hachurées dans la figure 5.5) sont éliminées des domaines de recherche des différentes variables.

5.2.3.3 La contrainte « quantité »

Pour calculer la quantité de l'intersection entre l'ensemble des extraits d'un côté et un ensemble de segments d'un certain attribut $Attribut_k$ d'un autre côté, une nouvelle contrainte globale est requise. La particularité de cette contrainte est qu'elle retourne une valeur de sortie. Une variable supplémentaire $Qte\_attribut_k$ ayant un domaine initial $[0..dmax]$ est donc créée. La variable $dmax$ correspond à la durée maximale du résumé. La valeur de retour de cette contrainte est ainsi égale à la quantité de l'attribut $Attribut_k$ dans le résumé final. Cette contrainte peut être implémentée de deux façons différentes. Nous allons décrire dans les sections suivantes, les deux algorithmes de filtrage que nous avons proposés :

5.2.3.3.1 Premier algorithme de filtrage : Une première façon d'implémenter la contrainte « quantité » consiste à utiliser le troisième type d'action sur les domaines de recherche des variables (décrit dans la page 84) qui est l'instanciation de la variable supplémentaire $Qte\_attribut_k$. Celle-ci est calculée en utilisant l'équation suivante :

$$\forall i, j, \sum max(0, min(fin\_extrait_i, fin\_Attribut_k[j]) - max(deb\_extrait_i, deb\_Attribut_k[j]))$$

L'instanciation de la variable $Qte\_attribut_k$ n'est effectuée que si l'assignation de valeurs à toutes les variables $Extraits$ (voir 5.1) est terminée. Cette contrainte n'est donc réveillée que si toutes les variables sont instanciées. Ce détail est très important pour éviter le filtrage inutile et pour avoir la valeur exacte de la quantité recherchée liée à l'assignation courante. Les parties pointillées dans la figure 5.6 représentent les parties en commun entre les extraits et les segments de l'attribut en question qui représentent l'intersection entre les deux ensembles. La somme de ces quantités représente le temps

de présence de l'attribut $Attribut_k$ dans le résumé et la valeur calculée est affectée à la variable $Qte\_attribut_k$ à la fin de la propagation de la contrainte.

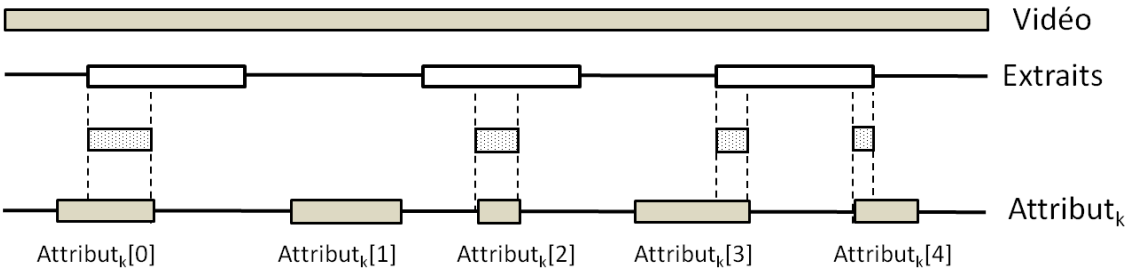

FIGURE 5.6 : Un premier algorithme de filtrage de la contrainte « quantité ».

Le filtrage est effectué, par instanciation, uniquement sur la variable supplémentaire $Qte\_attribut_k$. Vu que cette contrainte ne traite pas les domaines de recherche des variables du modèle (*Extraits*), elle est implémentée sans condition d'arrêt et ni condition de satisfaction.

5.2.3.3.2 Deuxième algorithme de filtrage : Ici, l'algorithme de filtrage est basé sur le premier type d'action sur les domaines de recherche des variables : l'ajustement de bornes.

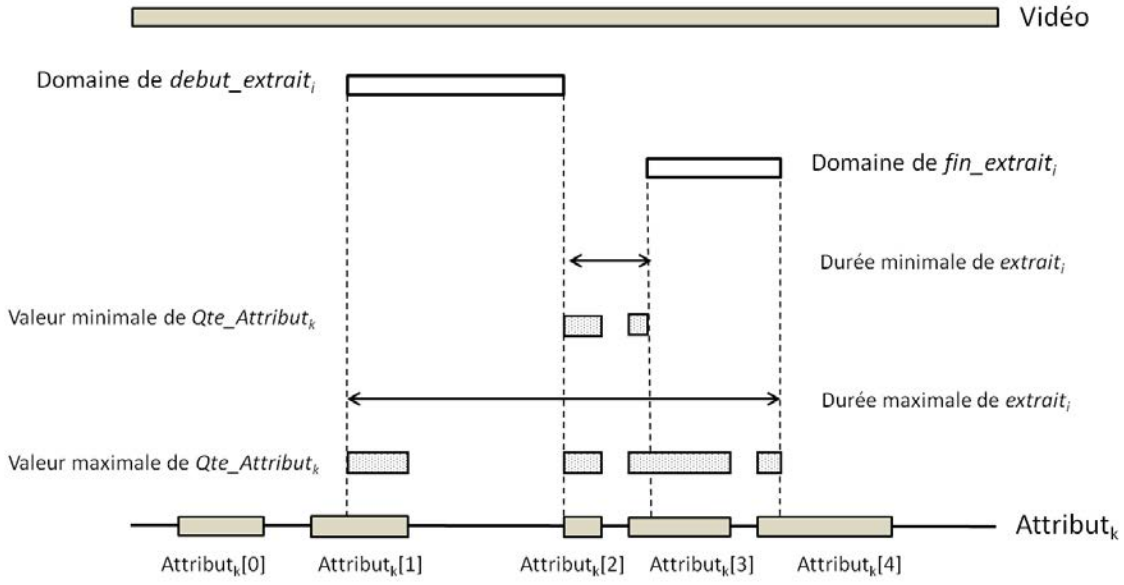

FIGURE 5.7 : Un deuxième algorithme de filtrage de la contrainte « quantité ». Illustration sur un seul extrait ($extrait_i$).

La figure 5.7 décrit l'algorithme de filtrage appliqué sur l'extrait $extrait_i$ afin de mon-

trer son principe. La borne inférieure de la variable de sortie $Qte\_attribut_k$ est ajustée à la quantité minimale que peut prendre la variable. La borne supérieure de la variable de sortie $Qte\_attribut_k$ est ajustée à la valeur maximale de la quantité de l'intersection. En effet, la valeur de la variable $Qte\_attribut_k$ ne peut, en aucun cas, dépasser la valeur de l'intersection entre les segments de l'attribut $Attribut_k$ et les extraits ayant la plus longue durée possible. De même, la valeur de la variable $Qte\_attribut_k$ ne peut, en aucun cas, être inférieure à la valeur de l'intersection entre les segments de l'attribut $Attribut_k$ et les extraits ayant la plus courte durée possible. Chaque ajustement d'un domaine de recherche d'une variable $extrait_i$ par le mécanisme de propagation de contraintes, entraînera une réduction du domaine de recherche de la variable $Qte\_attribut_k$.

Dans ce cas, une condition de satisfaction est indispensable pour éviter toute contradiction après l'assignation des valeurs (notamment pour la variable $Qte\_attribut_k$).

5.2.3.3.3 Bilan de la contrainte « quantité » : Deux façons d'implémenter la contrainte « quantité », avec deux algorithmes de filtrage différents, sont possibles. Nous avons testé et retenu la deuxième façon car elle est plus rapide mais nous avons quand même présenté le premier algorithme de filtrage pour montrer la multitude d'algorithmes de filtrage possibles pour une seule contrainte.

5.2.3.4 La contrainte « précédence »

La contrainte globale « précédence » assure l'ordonnancement des extraits entre eux. Cette contrainte peut à la fois être exprimée avec une contrainte globale (comme nous allons le détailler ci-dessous) ou bien par une contrainte utilisant les opérateurs standards fournis par CHOCO (voir équation 5.6, page 78). Elle s'applique sur deux extraits qui se suivent ($extrait_i$ et $extrait_{i+1}$)

La figure 5.8 illustre le fonctionnement de l'algorithme de filtrage de cette contrainte. Tout comme les contraintes globales précédentes, le but est de filtrer le maximum de valeurs impossibles, et cela, en évitant d'éliminer d'éventuelles valeurs possibles.

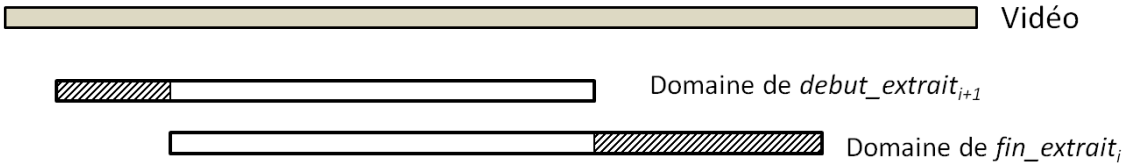

FIGURE 5.8 : Algorithme de filtrage de la contrainte « précédence ».

L'algorithme de filtrage procède comme suit :

- la borne supérieure du domaine de recherche de la variable de fin de l'extrait i ($fin\_extrait_i$) est ajustée à la borne supérieure du domaine de recherche de la variable de début de l'extrait $i+1$ ($debut\_extrait_{i+1}$) ;
- la borne inférieure du domaine de recherche de la variable de début de l'extrait $i+1$ ($debut\_extrait_{i+1}$) est ajustée à la borne inférieure du domaine de recherche de la variable de fin de l'extrait i ($fin\_extrait_i$).

La condition de satisfaction utilisée pour valider l'assignation de valeurs est :
$\forall i, debut\_extrait_{i+1} > fin\_extrait_i$

5.2.4 Modèle #3 : les différentes stratégies de recherche

Les stratégies de recherche permettent de définir une manière spécifique de parcourir l'espace de recherche. Elles déterminent comment construire l'arbre de recherche et dans quel ordre parcourir ses nœuds (l'arbre de recherche est décrit dans la figure 3.2, page 37). Nous distinguons entre les stratégies de sélection de variables (choix de la variable à instancier) et les stratégies de sélection de valeurs (choix de la valeur à affecter à la variable choisie).

Nous avons utilisé et expérimenté un ensemble de stratégies de recherche présentées dans les sections suivantes :

5.2.4.1 Recherche aléatoire

Cette stratégie sélectionne aléatoirement une variable à laquelle est attribuée une valeur aléatoirement sélectionnée à partir de son domaine de recherche.

5.2.4.2 Recherche incrémentale

La sélection des variables et des valeurs suit l'ordre d'apparition de celles-ci. Autrement dit, suivant cette stratégie, le solveur attribut à la première variable non-instanciée, la borne inférieure de son domaine de recherche. Une variable non-instanciée est une variable à laquelle le solveur n'a pas encore affecté une valeur.

5.2.4.3 Stratégie de recherche par activité

C'est une heuristique de choix de variables. L'idée derrière cette stratégie est d'utiliser le critère de mesure de l'activité des variables lors de la propagation pour guider la recherche. Ainsi, pour chaque décision prise au niveau d'un nœud de l'arbre, l'activité d'une variable est augmentée si un ajustement lié à cette variable se produit. Au début,

les décisions sont prises aléatoirement, et après un certain nombre d'essais, le choix du branchement est effectué sur la variable ayant le score d'activité le plus élevé. En d'autres termes, la recherche par activité favorise les variables dont le domaine de recherche est fréquemment réduit par le mécanisme de propagation des contraintes.

5.2.4.4 La stratégie *DomOverWDeg*

C'est aussi une heuristique de sélection de variables. La stratégie sélectionne la variable non-instantiée ayant le ratio r le plus petit :

$$r = \frac{dom}{w * deg} \tag{5.12}$$

Pour chaque variable et pour chaque itération durant le mécanisme de propagation de contraintes, trois composantes sont prises en compte par la stratégie et interviennent dans la sélection de variables :

- dom : la taille courante du domaine de recherche ;
- w : le nombre de contraintes non-instanciées qui sont concernées par la variable courante ;
- deg la somme du nombre d'échecs causées par chaque contrainte (parmi les w contraintes) depuis le début de l'exploration et de la recherche de solutions. Autrement dit, à chaque fois qu'un échec est rencontré, le degré des variables impliquées dans la contrainte non satisfaite est incrémenté d'un point. Le degré des variables est initialisé à 1.

5.2.5 Modèle #3 : expérimentations et évaluation

Pour ce modèle, en plus des contraintes de modélisation (5.5), (5.6) et (5.7), nous avons exprimé l'ensemble de contraintes définies dans la section 5.2.2.

- la durée du résumé doit appartenir à un intervalle donné en paramètres : $[dmin..dmax]$;
- chaque extrait contient au moins un segment entier de jeu (tout un segment de jeu) ;
- chaque extrait contient des applaudissements ;
- les extraits ne coupent pas la parole ;

- les extraits ne coupent pas les segments de jeu ;
- le résumé de contient pas de publicité ;
- la balle de match doit être présente dans le résumé ;
- la quantité minimale des applaudissements dans le résumé final doit être égale à un certain seuil (fixé empiriquement) ;
- la quantité de jeu dans le résumé est maximisée.

5.2.5.1 Description de l'ensemble de données de test

En plus de l'ensemble $Ensemble_2$ décrit dans la section 4.4.4.1 page 68, nous avons utilisé un troisième ensemble de données de test, noté $Ensemble_3$. Cet ensemble est composé de 8 matchs de tennis du tournoi de Roland Garros 2014. La durée totale des matchs est de 20 heures et 30 minutes :

$Ensemble_3 = \{M_5, M_6, M_7, M_8, M_9, M_{10}, M_{11}, M_{12}\}$.

Chacun de ces matchs possède un résumé éditorial de durée égale à 3 minutes. Nous avons récupéré ces résumés éditoriaux à partir du site web officiel du tournois de Roland Garros.

5.2.5.2 Étude du modèle #3

Afin de comparer l'efficacité des deux modèles (modèle #2 et modèle #3), autrement dit, leur capacité à générer rapidement un résumé, nous avons utilisé les matchs M_1 et M_2 de l'ensemble $Ensemble_2$. Pour chaque modèle, nous avons mesuré le temps de la première solution réalisable calculée et retournée par le solveur. Les résultats sont présentés dans le tableau 5.1, ils montrent que le modèle #3 permet au solveur d'être beaucoup plus rapide, et par conséquent, permet une résolution plus efficace du problème.

	1re solution retournée avec modèle #2	1re solution retournée avec modèle #3
M1	1 minute et 27 secondes	1 seconde
M2	1 minute et 48 secondes	2 secondes

TABLE 5.1 : Modèle #2 vs. Modèle #3. Comparaison des temps pris par le solveur pour retourner une première solution.

5.2.5.3 Évaluation de la qualité des résumés

Dans cette section, nous avons utilisé le même protocole d'évaluation que celui utilisé dans 4.4.4. Nous avons également réutilisé les données de test de l'ensemble $Ensemble_2$

(voir 4.4.4.1, 68) pour évaluer la qualité des résumés générés et pour pouvoir comparer les deux modèles.

Deux critères d'évaluation ont été utilisés :

1. sur les deux premiers matchs : l'intersection avec des résumés éditoriaux qui reflète la capacité de notre solution à sélectionner les mêmes parties que celles qui composent le résumé éditorial. Les résultats sont détaillés dans le tableau 5.2 ;

2. sur les deux derniers matchs : la proportion des parties non intéressantes dans les résumés générés par notre solution, ce qui permet d'évaluer la capacité de notre solution à éliminer les parties non intéressantes. Les résultats sont indiqués dans le tableau 5.3.

Durée	5 min		10 min		15 min		20 min	
modèle	modèle #2	modèle #3	modèle #2	modèle #3	modèle #2	modèle #3	modèle #2	modèle #3
M_1	43%	51%	32%	43%	28%	37%	21%	41%
M_2	30%	56%	28%	41%	32%	60%	33%	59%

TABLE 5.2 : Étude comparative entre les résultats du modèle #2 et ceux du modèle #3 : évaluation par rapport aux résumés éditoriaux.

Durée	5 min		10 min		15 min		20 min	
modèle	modèle #2	modèle #3	modèle #2	modèle #3	modèle #2	modèle #3	modèle #2	modèle #3
M_3	15%	15%	11%	10%	13%	11%	9%	7%
M_4	10%	8%	8%	8%	15%	11%	22%	11%

TABLE 5.3 : Étude comparative entre les résultats du modèle#2 et ceux du modèle#3 : évaluation par rapport aux parties non intéressantes.

Les deux tableaux révèlent clairement que le troisième modèle fournit une bien meilleure qualité de résumés vidéo, au sens de critères d'évaluation considérés, que le deuxième modèle.

Pour mieux évaluer la qualité des résumés générés avec notre troisième modèle, Nous l'avons appliqué sur $Ensemble_3$ (voir 5.2.5.1) composé de 8 matchs de tennis avec un résumé éditorial de 3 minutes pour chacun. Tous les résumés sont générés en une minute, c'est à dire le solveur est interrompu après une minute et la dernière solution trouvée est retournée. Les résumés générés sont évalués en utilisant le critère de l'intersection avec les résumés éditoriaux. Le tableau 5.4 décrit les résultats obtenus.

	M_5	M_6	M_7	M_8	M_9	M_{10}	M_{11}	M_{12}
$\bigcap$ *avec RE*	24%	45%	21%	17%	33%	21%	41%	18%
# d'extraits	7	9	9	7	9	8	9	9

TABLE 5.4 : $Ensemble_3$ - évaluation des résumés générés avec le modèle #3 par rapport aux résumés éditoriaux.

Même si les résumés éditoriaux, dans ce cas, sont très courts (environ 3 minutes) et les résumés générés automatiquement par notre méthode sont presque de la même durée (entre 2 minutes 45 secondes et 3 minutes), le taux d'intersection entre les deux résumés est relativement élevé. La sélection des parties intéressantes dont la durée totale est d'environ 3 minutes d'une vidéo dont la durée moyenne est de 2 heures et demi est très exigeant. Avoir une intersection significative (27,5% en moyenne) entre un résumé généré par notre méthode et un résumé éditorial montre que notre modèle et les contraintes que nous avons définies sont efficaces pour cibler les parties intéressantes de la vidéo.

Le choix du nombre d'extraits n qui constitueront le résumé final dépendent de plusieurs critères.

- La durée souhaitée du résumé final, comprise entre une durée minimale (*dmin*) et une durée maximale (*dmax*) (voir l'équation 5.13).
- La durée minimale et maximale autorisée pour un extrait (*duree_extrait_min* et *duree_extrait_max*) (voir l'équation 5.13).
- Des expérimentations effectuées sur le temps de réponse du solveur : certains intervalles de valeurs possibles de n sont écartés expérimentalement. Le solveur passe un temps énorme à explorer l'arbre de recherche sans trouver rapidement une solution réalisable à cause du nombre très élevé (ou très faible) d'extraits demandés.
- Des expérimentations liées à la qualité des résumés résultants : en fonction de la durée souhaitée du résumé, le nombre d'extraits par résumé peut être limité. (Trop d'extraits par résumé n'est pas agréable à regarder car il y aura beaucoup de changement, peu d'extraits non plus car il y aura une stabilité ennuyante).

$$\frac{dmax}{duree\_extrait\_max} < M < \frac{dmax}{duree\_extrait\_min} \tag{5.13}$$

5.2.5.4 Impact de la fiabilité de la détection des attributs

Nous avons repris la même expérience décrite dans 4.4.4.5 en utilisant cette fois le modèle #3. Nous rappelons que le but de cette expérience est d'étudier l'influence de la

détection des attributs de bas niveau sur la forme finale des résumés. Pour cela, nous avons comparé les résumés générés avant et après la correction de la détection de ces attributs. Les résultats obtenus sur les matchs M_2 et M_3 de l'ensemble $Ensemble_2$ sont présenté dans le tableau 5.5.

Durée	5 min		10 min		15 min		20 min	
Correction	avant	après	avant	après	avant	après	avant	après
M_2	56%	66%	41%	68%	60%	62%	59%	67%
M_3	15%	4%	10%	6%	11%	5%	7%	6%

TABLE 5.5 : Impact de la correction de la détection des attributs sur l'intersection avec RE pour M_2 et sur l'intersection avec PNI pour M_3 en utilisant le modèle #3.

Nous avons également choisi le match M_{10} de l'ensemble $Ensemble_3$ (voir section 5.2.5.1). Nous avons généré un résumé avec une durée d'environ 3 min pour lequel nous avons calculé l'intersection avec un résumé éditorial de 3 min. L'intersection obtenue avec le résumé éditorial est de 30% (cette intersection était de 21% avant la correction des attributs).

5.3 Conclusion

Dans ce chapitre, nous avons proposé une nouvelle modélisation pour résoudre le problème de création automatique de résumés vidéo en utilisant la programmation par contraintes. Nous avons proposé un modèle qui s'affranchit de toute segmentation en plans de la vidéo, ce qui a amélioré la flexibilité de notre modèle et l'expressivité des contraintes formulées. À la suite de nos expériences, nous concluons qu'une bonne modélisation de notre problème de création automatique de résumés vidéo peut considérablement améliorer la qualité du résumé généré et même l'efficacité du solveur de contraintes. Nous avons également implémenté de nouvelles contraintes avec leurs propres algorithmes de filtrage.

Chapitre 6

Évaluation : Tests utilisateurs

Sommaire

6.1 Introduction

Comme présenté dans la section 2.7, les tests utilisateurs représentent l'une des méthodes d'évaluation de la qualité des résumés générés automatiquement.

Les tests utilisateurs semblent être à première vue la meilleure méthode d'évaluation. Cependant, dans la pratique, cette méthode nécessite de définir un protocole rigoureux afin de sélectionner un nombre important d'utilisateurs, d'établir des mesures d'évaluation pertinentes et de choisir le cas d'utilisation approprié. En outre, les résultats doivent être soigneusement et formellement analysés. Cela se fait rarement dans la pratique. En particulier, le nombre d'évaluateurs est très limité (par exemple 5 évaluateurs dans [SC01], 10 évaluateurs dans [dAdLdA08], et 23 évaluateurs dans [San07]).

La dernière expérience que nous avons effectuée afin d'évaluer la qualité des résumés générés est l'évaluation par des utilisateurs à large échelle. Nous avons utilisé de

nouveau le cas d'usage des vidéos de tennis et nous nous sommes concentrés sur le troisième modèle de notre méthode. Les expériences présentées dans le chapitre 5 montrent que ce modèle est le plus efficace et le plus performant.

Le but de cette campagne est de pouvoir se positionner entre les résumés éditoriaux qui sont censés être parfaits et les résumés basiques qui ne sont pas bien créés et ne sont pas agréables à voir. Un autre but de cette campagne est de considérer uniquement les résumés créés avec notre méthode et de pouvoir statistiquement comparer les résumés créés avec différentes stratégies de recherche et en sélectionnant différents nombres d'extraits.

Dans ce chapitre nous introduisons le protocole d'évaluation que nous avons adopté. Nous analysons par la suite les résultats de la campagne de tests utilisateurs menée pour évaluer la qualité des résumés générés automatiquement. Une étude statistique est effectuée pour analyser les courbes résultantes.

6.2 Description du protocole

L'évaluation a été effectuée en ligne à travers un site web, où l'évaluateur regarde un résumé et lui attribue un score. Le score est une valeur entière qui varie entre 1 et 10. Chaque évaluateur peut évaluer jusqu'à 40 résumés. Il/elle a également la possibilité de laisser un commentaire/impression concernant la qualité de chaque résumé.

L'ensemble $Ensemble_3$ qui contient 8 matchs de tennis a été utilisé pour cette campagne de tests utilisateurs. Pour chaque match, 20 résumés ont été générés :

- un résumé éditorial récupéré à partir du site officiel du tournoi Roland Garros. Le résumé éditorial est la concaténation d'un ensemble d'extraits audiovisuels avec la bande son originale associée à chaque extrait ;
- 16 résumés générés automatiquement avec notre méthode :
  - avec différents nombre d'extraits (8, 10, 12, 14) ;
  - avec différentes stratégies de résolution (les quatre stratégies présentées dans la section 5.2.4) ;
- 3 résumés générés en utilisant des méthodes simples et basiques :
  - sélection aléatoire de plans ;
  - sélection aléatoire de plans de jeu (échanges complet de la balle) ;

– sélection uniforme d'extraits d'une durée de 10 secondes chaque période de temps t (calculée en fonction de la durée du match dm et la durée du résumé dr) : $t = (dm * 10)/dr$.

Avant de commencer l'évaluation, nous introduisons la tâche aux évaluateurs à travers un cas d'utilisation simple : l'évaluateur est un utilisateur qui a raté un match de tennis, il/elle peut avoir une idée sur le score final du match et il/elle dispose seulement de 3 minutes pour regarder un résumé du match. Il/elle le regarde de façon linéaire, comme si c'était un service offert à la télévision, et donne une note au résumé. Nous n'avons fourni aucune explication sur les techniques évaluées ni sur les critères d'évaluation. Les évaluateurs ont effectué cette tâche d'évaluation chez eux.

L'ordre de présentation des résumés aux évaluateurs est aléatoire. Cependant, en raison du nombre de résumés éditoriaux relativement faible (par rapport à ceux générés automatiquement), nous les avons dupliqués. L'idée est d'augmenter leurs chances d'être évalué.

Cette campagne s'est étalée sur cinq semaines et a impliqué 61 évaluateurs. À la fin, nous avons recueilli 1096 évaluations. Chaque résumé a été évalué en moyenne 6,85 fois et chaque évaluateur a évalué 18 résumés en moyenne.

Les évaluations recueillies ont été analysées et les résultats sont décrits dans les sections suivantes.

6.3 Analyse des résultats : étude comparative

Les notes collectées sont présentées dans la figure 6.1. Le tableau 6.1 donne pour chaque type de résumé, le nombre total d'évaluations collectées, la moyenne et l'écart type des notes obtenues. L'écart type des notes attribuées aux résumés éditoriaux est le plus bas, ce qui signifie que les évaluateurs sont assez d'accord sur les notes de cette catégorie de résumés. Nous pouvons également remarquer que les scores obtenus par les méthodes basiques sont très faibles et se démarquent clairement de ceux obtenus par les deux autres catégories de résumés. Cependant, la différence entre les notes des résumés éditoriaux et celles des résumés générés avec notre méthode est très faible et les deux distributions de notes se chevauchent largement. Néanmoins, afin d'étudier formellement cette différence entre les notes, nous proposons d'analyser la distribution des notes en utilisant des tests statistiques [MGo8].

Nous avons, tout d'abord, vérifié si nous pouvons considérer que les deux distributions de notes peuvent être approximée par une distribution normale. Pour cela, nous

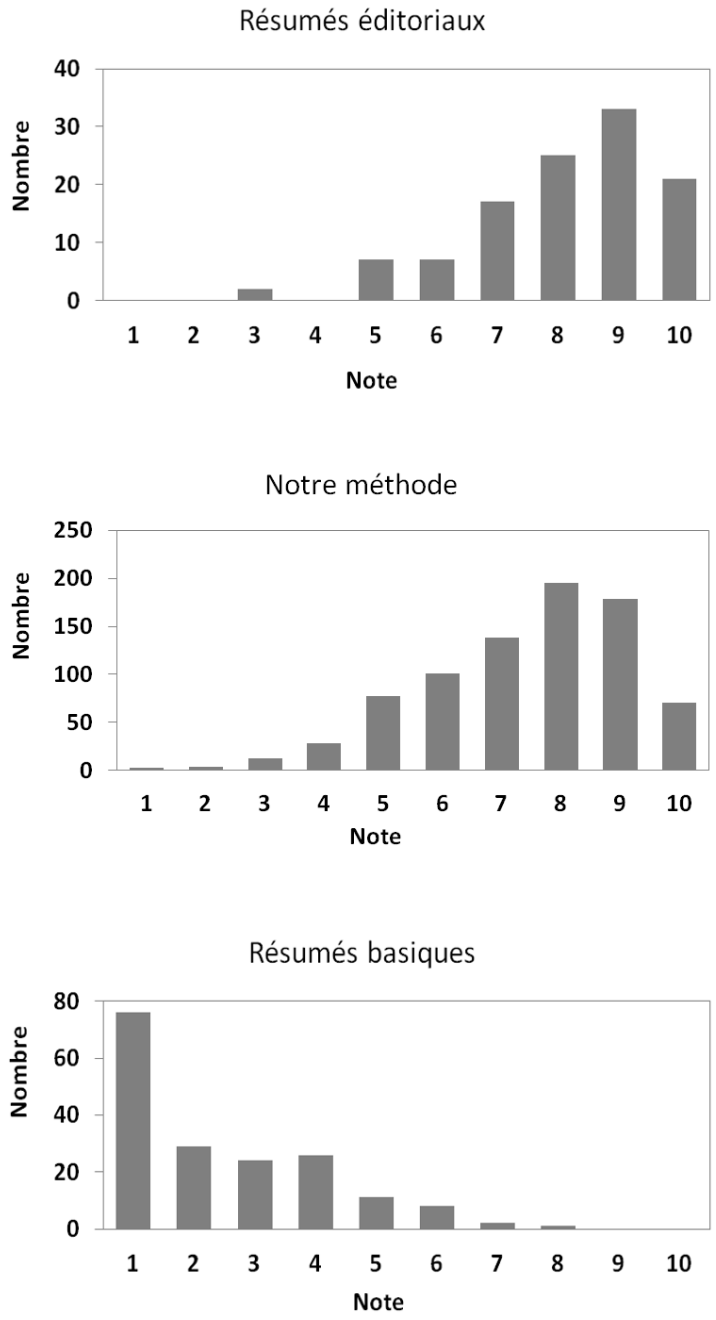

FIGURE 6.1 : Notes obtenues par les trois types de résumés.

avons utilisé le test de Kolmogorov-Smirnov (KS) avec l'hypothèse nulle que l'histogramme suit une loi normale standard N(0,1). Avant d'effectuer les tests de KS, les histogrammes de scores ont été normalisés pour avoir une moyenne nulle et un écart type de 1. L'application du test KS sur l'histogramme de notes issue de notre méthode donne un *p-value* $= 2,07\ e^{-23}$ et un *p-value* $= 0,0003$ pour l'histogramme de notes attri-

buées aux résumés éditoriaux. Par conséquent, l'hypothèse nulle peut être rejetée avec une confiance de 95% pour les deux histogrammes de notes.

	Nombre d'évaluations	Moyenne (μ)	Écart-type (σ)
Résumés éditoriaux	112	8,12	1,56
Notre méthode	807	7,42	1,75
Méthodes basiques	177	1,65	2,46

TABLE 6.1 : Nombre d'évaluations, moyenne et écart type des notes obtenues par les trois types de résumés.

En raison du fait que les données d'entrée (les deux histogrammes de notes) ne suivent pas une loi normale, nous ne pouvons pas utiliser un *T-test* de Student pour comparer leurs moyennes. Pour cela, nous avons utilisé un *U-test* de Mann-Whitney pour comparer les médianes des deux histogrammes de notes. L'hypothèse nulle du *U-test* de Mann-Whitney est que les médianes des deux histogrammes de score soient égales. Nous avons obtenu un *p-value* $= 3,28\ e^{-05}$. Par conséquent, l'hypothèse nulle peut être rejetée avec 95% de confiance. En d'autres termes, les médianes des deux histogrammes de score ne sont pas égales (d'un point de vue statistique). Cela signifie que les évaluateurs considèrent que les résumés éditoriaux sont légèrement meilleurs que ceux générés par notre méthode.

Afin de mieux observer la différence entre les évaluations attribuées aux différents types de résumés, nous avons utilisé l'outil *dfittool* de matlab pour lisser les courbes et les normaliser. La figure 6.2 montre que même si les résumés éditoriaux sont un peu mieux notés que les résumés générés par notre méthode automatique, la différence entre les deux courbes reste marginale.

6.4 Analyse des résultats : l'impact du nombre d'extraits

Afin d'étudier l'impact du nombre d'extraits dans les résumés générés avec le troisième modèle, nous avons analysé les histogrammes de notes en se basant sur le nombre d'extraits par résumé. Nous rappelons que nous avons varié ce paramètre sur les quatre valeurs suivantes : 8, 10, 12 et 14. Les résultats sont présentés dans la figure 6.3 et le tableau 6.2.

Les histogrammes de notes obtenues par les différents résumés sont très semblables et il semble que ce paramètre n'a aucune influence sur la qualité des résumés du point de vue de l'utilisateur. Nous avons toutefois effectué un test statistique pour vérifier si les médianes des quatre distributions sont identiques. Les résultats obtenus montrent que seule la médiane des notes attribuées aux résumés ayant 8 extraits est différente de

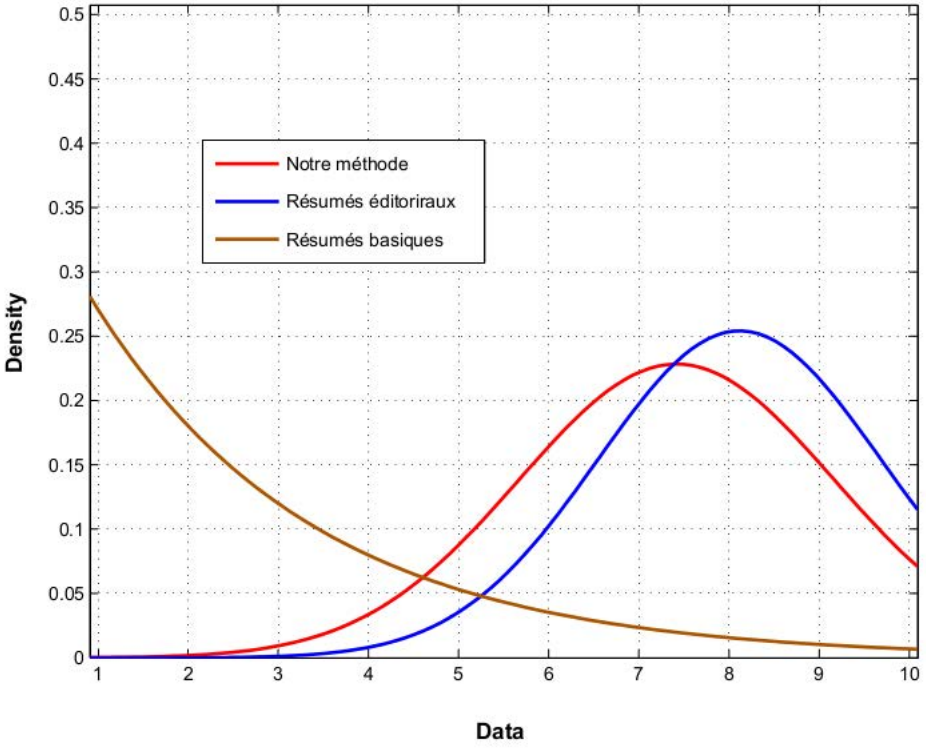

FIGURE 6.2 : Courbes lissées et normalisées en utilisant *dfittool*.

celles des résumés ayant 10, 12 et 14 extraits. Par conséquent, les résumés comprenant plus d'extraits sont susceptibles d'être mieux appréciés par les utilisateurs que ceux créés avec moins d'extraits. Avoir plusieurs extraits dans le résumé augmente la chance de correspondre aux attentes des utilisateurs et de fournir une meilleure couverture du contenu de la vidéo d'entrée.

Nombre d'extraits	Nombre d'évaluations	Moyenne (μ)	Écart-type (σ)
8	201	7,09	1,78
10	218	7,48	1,74
12	204	7,48	1,8
14	184	7,64	1,61

TABLE 6.2 : Nombre d'évaluations, moyenne et écart type des notes obtenues par des résumés ayant différents nombres d'extraits.

6.5 Analyse des résultats : l'impact de la stratégie de résolution

Pour créer les résumés avec le modèle #3 de notre méthode, nous avons utilisé quatre stratégies de résolution différentes. Les notes obtenues par stratégie sont présentées dans

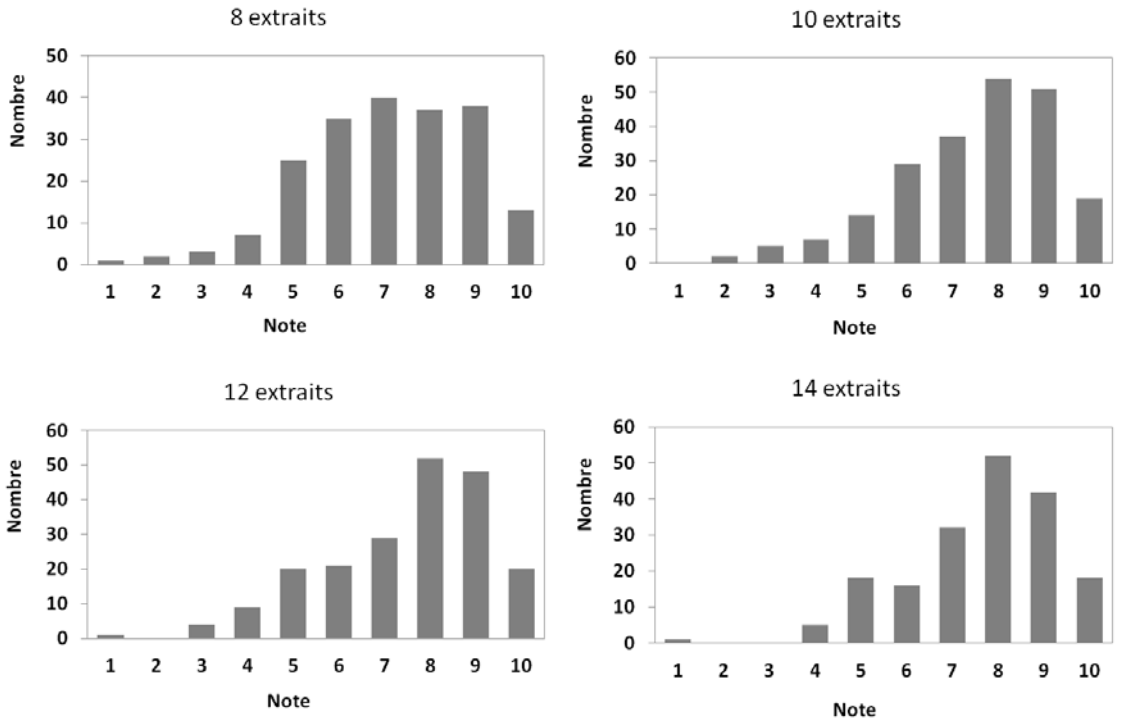

FIGURE 6.3 : Notes Vs. nombre d'extraits dans les résumés générés par notre méthode.

la figure 6.4 et le tableau 6.3.

Aussi bien les figures que les tests statistiques, montrent que la stratégie de recherche n'a aucun impact sur la qualité des résumés générés, du point de vue de l'utilisateur.

Stratégie	Nombre d'évaluations	Moyenne (μ)	Écart-type (σ)
Stratégie 1	232	7,55	1,59
Stratégie 2	202	7,28	2,01
Stratégie 3	204	7,42	1,7
Stratégie 4	169	7,41	1,64

TABLE 6.3 : Nombre d'évaluations, moyenne et écart type des notes obtenues par des résumés ayant différents nombres d'extraits.

6.6 Analyse des résultats : étude sur l'évolution temporelle des évaluations

Une évolution stable au cours du temps des évaluations des utilisateurs augmente le degré de confiance dans cette étude. Nous avons étudié l'évolution temporelle et la stabilité des évaluations au cours du temps. Il s'agit d'étudier la différence entre les évaluations effectuées au début de la campagne d'évaluation et celles effectuées à la fin

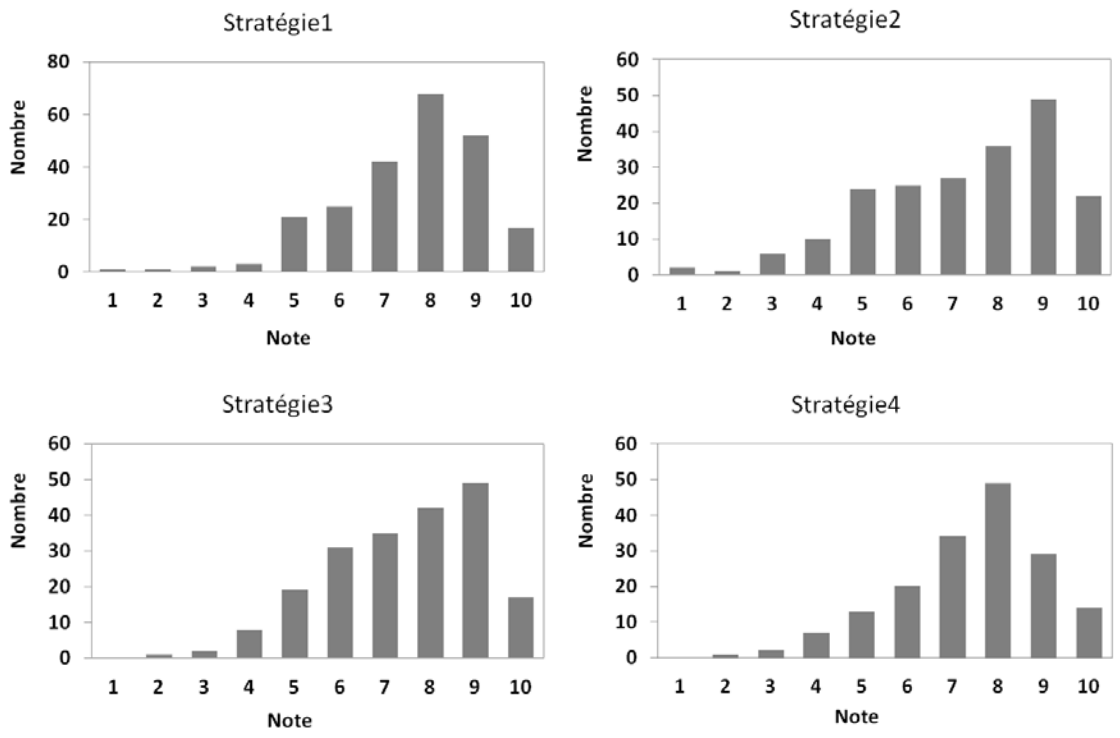

FIGURE 6.4 : Notes vs. stratégies de résolution dans les résumés générés par notre méthode.

de la campagne et de prouver que l'évolution est approximativement stable.

Nous avons utilisé l'outil *cftool* de Matlab qui permet de lisser (*smooth*) une courbe et nous avons tracé les trois courbes des figures 6.5, 6.6 et 6.7.

Le tableau 6.4 décrit l'évolution au niveau du critère de la moyenne.

	33% des évaluations	66% des évaluations	100% des évaluations
Résumés éditoriaux	8,08	8,11	8,12
Résumés de notre méthode	6,91	7,13	7,42
Résumés basiques	1,38	2,36	2,46

TABLE 6.4 : Évolution temporelle de la moyenne des évaluations par type de résumé.

Les courbes montrent que les évaluations effectuées par les évaluateurs ont suivi une évolution temporelle stable. Ceci représente un indicateur sur le degrés de confiance de ces évaluations. Les figures 6.5, 6.6 et 6.7 montrent que, pour chaque type de résumé, les trois courbes (33%, 66% et 100%) sont visuellement semblables.

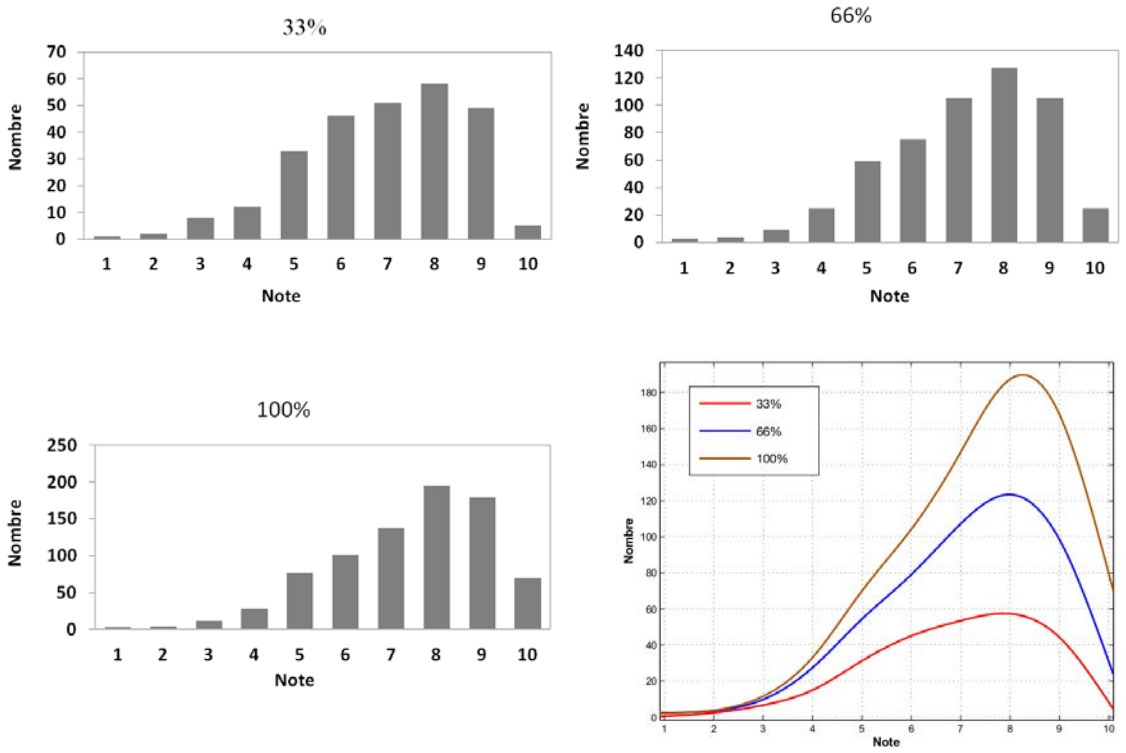

FIGURE 6.5 : Évolution temporelle des évaluations des résumés de notre méthode.

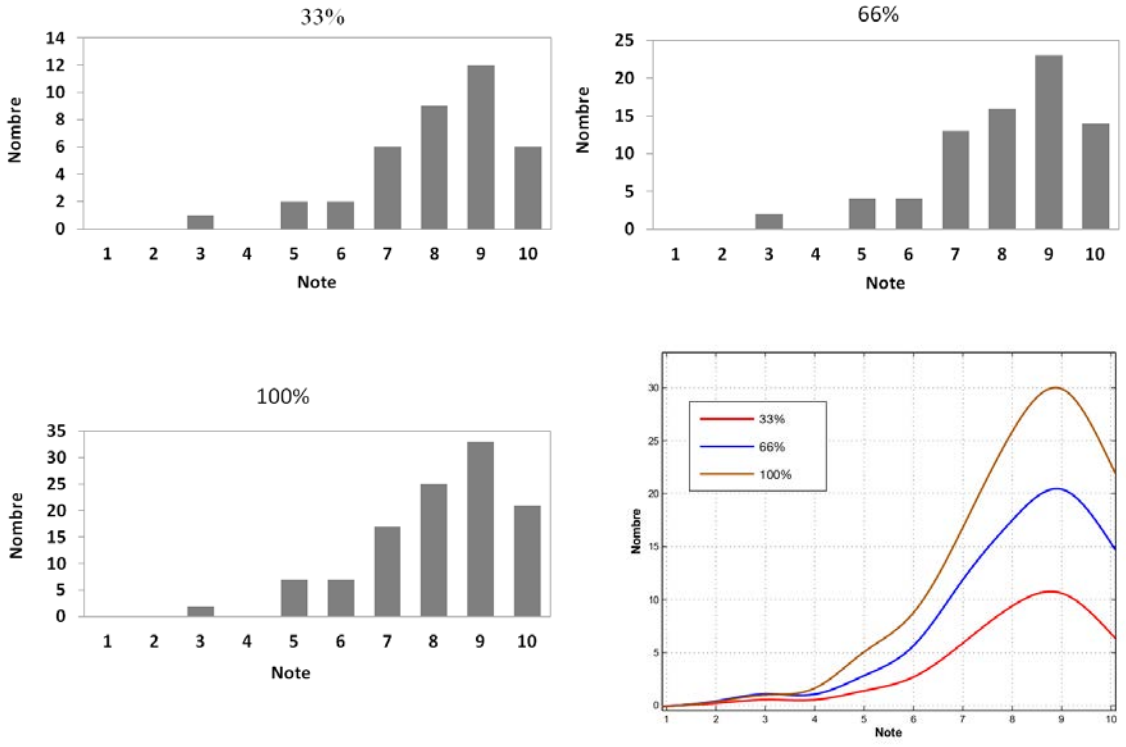

FIGURE 6.6 : Évolution temporelle des évaluations des résumés des résumés éditoriaux.

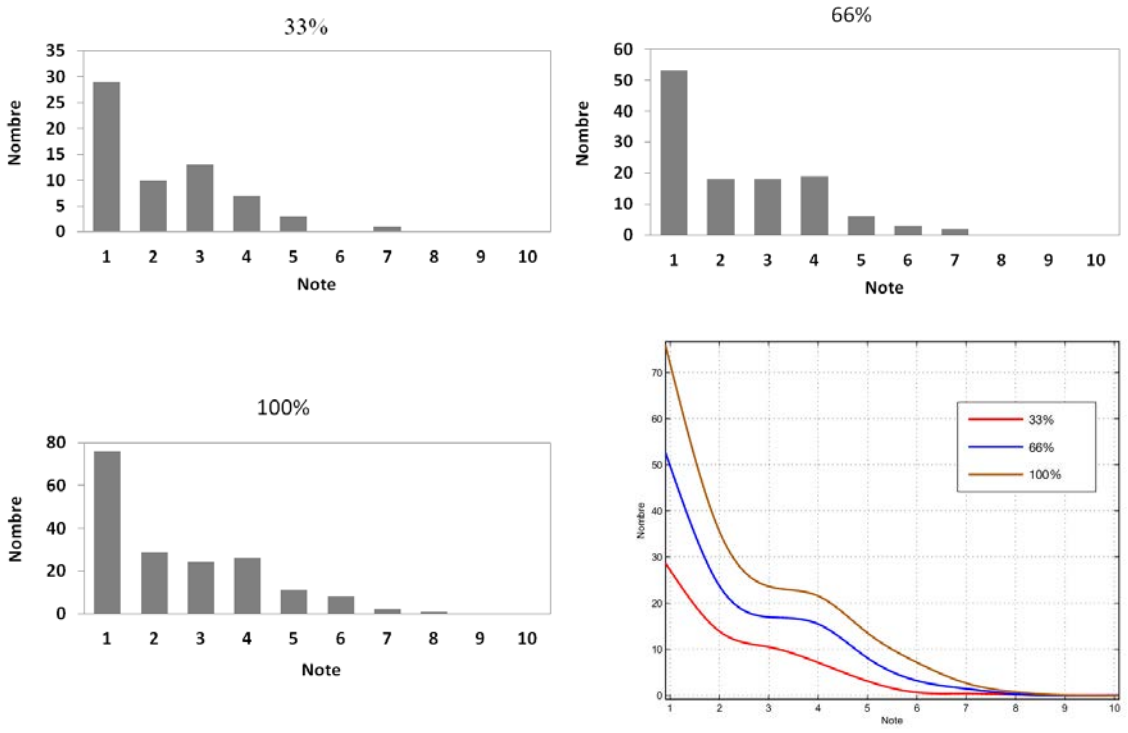

FIGURE 6.7 : Évolution temporelle des évaluations des résumés des méthodes basiques.

6.7 Analyse des résultats : analyse des commentaires des évaluateurs

Pour analyser les commentaires des évaluateurs, nous avons utilisé l'outil *TagCrowd* [RB08]. TagCrowd est un outil permettant de visualiser la fréquence des mots dans un texte en créant ce qui est populairement connu par un **nuage de mots**. Cet outil permet de faire une analyse d'un texte basée sur la fréquence d'usage de mot-clés. Il est basé sur l'algorithme TF-IDF [SJ72].

Pour chaque groupe de résumés (basiques, automatiques et éditoriaux) nous avons collecté tous les commentaires et, en utilisant l'outil TagCrowd, nous avons identifié les mots les plus utilisés par chacun des groupes de commentaires. L'algorithme adopté par cet outil élimine tout d'abord les mots vides (et, de, le, un...) et, selon le nombre d'occurrences de chacun des mots clés, il fournit le *Top 50* des mots clés sous forme d'image informative où les mots clés les plus foncés correspondent aux mots clés les plus utilisé. Les résultats obtenus sont présentés dans les figures 6.8, 6.9 et 6.10.

La figure 6.8 représente le nuage de mots lié aux commentaires des résumés éditoriaux. La figure indique qu'il n'y a pas de mot spécifique fréquemment répété par les évaluateurs et qui donne un sens à l'évaluation (à l'exception du mot « match », sa pré-

FIGURE 6.8 : Nuage de mots à partir des commentaires des résumés éditoriaux

sence étant évidente). Nous remarquons la présence de mots liés à la bonne qualité des résumés générés, comme « apprécie », « aime », « équilibré »...

FIGURE 6.9 : Nuage de mots à partir des commentaires des résumés de notre méthode.

La figure 6.9 représente le nuage de mots lié aux commentaires des résumés générés par notre méthode automatique. Nous remarquons la présence des mots « coupe », « coupure », « commentaires », « commentateur »... Ces mots font référence aux coupures au niveau de la parole qui sont critiquées par les évaluateurs. La principale origine de ces coupures est liée aux erreurs générées par les algorithmes automatiques de détection de la parole. Le processus de création des résumés avec notre méthode exige la non-coupure des phrases des commentateurs, mais sur le plan pratique, la détection n'est pas parfaite. D'autre part, le mot « bon » est clairement présenté dans le nuage de mots, ce qui reflète la satisfaction d'un nombre considérable d'évaluateurs par les résumés créés avec notre méthode.

FIGURE 6.10 : Nuage de mots à partir des commentaires des résumés basiques

La figure 6.9 représente le nuage de mots lié aux commentaires des résumés basiques. La présence des mots « incohérent », « mauvais », « court »... montrent que les résumés n'ont pas du tout plu aux évaluateurs. Le nuage est très hétérogène. Beaucoup de mots sont présents en couleur foncée. De plus, les évaluateurs ont utilisé en masse une grande variété de mots, ce qui reflète l'aspect aléatoire des résumés évalués.

6.8 Conclusion

Dans ce chapitre, nous avons présenté les résultats d'une étude statistique effectuée sur une expérience de tests utilisateurs que nous avons mené afin d'évaluer notre méthode automatique ainsi que les résumés générés. Cette étude montre que les résumés générés automatiquement, pour l'application du tennis, sont des *bon résumés* de point de vue statistique. Mener une telle campagne de tests utilisateurs n'est pas une tâche facile (difficultés de recrutement d'évaluateurs, longue attente pour atteindre un nombre élevé d'évaluateurs...).

Chapitre 7

Conclusion générale

Sommaire

Ce dernier chapitre de conclusion dresse un bilan des travaux effectués dans le cadre de cette thèse. Nous rappelons dans un premier temps les principales contributions de ces travaux. Nous discutons les limitations des approches proposées, ainsi que les améliorations possibles. Nous présentons ensuite quelques pistes de travaux futurs et des perspectives de recherche, ainsi que la liste des publications associées à ces travaux de thèse.

7.1 Récapitulatif des contributions

Dans ce travail, nous avons proposé une nouvelle approche pour la création automatique de résumés audiovisuels de vidéos. Notre méthode est basée sur la programmation par contraintes. Les règles de production des résumés et les préférences des utilisateurs sont exprimées sous la forme d'un ensemble de contraintes à satisfaire. Le principal avantage de notre solution est sa capacité à séparer clairement les règles de production de l'algorithme de génération de résumés. Cela permet de facilement adapter le résumé, en ajoutant ou en modifiant une (ou plusieurs) règle(s), sans avoir à modifier l'algorithme de génération du résumé.

Nous avons proposé trois modèles et nous avons étudié les performances et l'efficacité de l'approche sur une application réelle : le résumé des vidéos de matchs de tennis. Des expériences ont été effectuées sur les trois modèles proposés pour mesurer leurs performances, évaluer la qualité des résumés générés et pour les comparer entre eux.

Nous avons également mené une expérience de tests utilisateurs à large échelle. La compagne de tests a impliqué 61 évaluateurs qui ont effectué 1096 évaluations. Dans le cadre de cette compagne, les résumés générés ont été comparés à des résumés éditoriaux. Les notes obtenues sont très semblables pour les deux types de résumés, même si les tests statistiques montrent que les résumés éditoriaux sont perçus comme étant meilleurs par les évaluateurs.

Les différents modèles proposés ainsi que les expérimentations réalisées ont permis de valider l'essentiel des objectifs que nous nous sommes fixés, en particulier en ce qui concerne la flexibilité et l'adaptabilité. Ces modèles présentent néanmoins quelques limitations. Le modèle #1 et le modèle #2 sont des modèles efficaces et simples à mettre en œuvre, mais en contrepartie, ils sont très dépendants des frontières des plans et ne sont pas suffisamment flexible pour pouvoir exprimer tous types de contraintes. Quant au modèle #3, c'est un modèle très flexible permettant d'exprimer une plus large variété de contraintes sur les résumés. Sa mise en place ainsi que la formulation de l'ensemble des règles de production du résumé a cependant nécessité l'implémentation de nouvelles contraintes avec leurs algorithmes de filtrage au niveau du solveur. Nous sommes donc passés d'une simple utilisation d'un solveur de programmation par contraintes, à la contribution à l'amélioration du solveur lui-même. Cela requiert des connaissances approfondies sur le fonctionnement interne de ces solveurs, que nous avons dû acquérir.

Nos travaux ont permis d'alimenter et d'impulser plusieurs autres travaux. Notre modèle #2 a été par exemple utilisé, dans le cadre du stage de master [1] de M. Caio Corro [Cai14], pour la création de résumés textuels de vidéos. L'objectif de ce stage, intitulé « Résumé automatique de texte appliqué au résumé multi-modal de vidéos par programmation par contraintes », a été de créer des résumés textuels de vidéos en utilisant la transcription automatique de la parole. Les expériences menées ont montré que les résultats des évaluations des résumés textuels générés en utilisant le modèle #2 sont meilleurs que les résumés produits par les approches traditionnelles de traitement de la langue naturelle sous certaines conditions.

Dans le cadre de cette thèse, nous avons également collaboré avec des spécialistes de la communauté de la programmation par contraintes. Nous avons étudié les contraintes du modèle #2 avec M. Marc Christie, membre de l'équipe Mimetic [2] de l'IRISA de Rennes. Son expertise et sa contribution a été très utile lors de l'étape d'optimisation des contraintes. Étant donné qu'une contrainte peut être formulée de différentes façons,

[1] Le stage s'est déroulé en 2014, à l'équipe Texmex de l'IRISA, sous l'encadrement de M.Patrick Gros et Mme. Pascale Sébillot.

[2] Équipe Mimetic. http://www.irisa.fr/mimetic/

le but de l'optimisation des contraintes est de trouver la meilleure façon de formuler une contrainte afin de réduire le nombre de variables intermédiaires crées par le solveur et accélérer ainsi la recherche de solutions. Cette collaboration nous a permis d'acquérir une meilleure compréhension des tenants et aboutissants de la programmation par contraintes dans une application réelle et d'affiner nos modèles #1 et #2.

Lors de la phase d'implémentation du modèle #3, nous avons aussi collaboré avec l'équipe TASC [3] du laboratoire LINA de Nantes. Lors de la mise en œuvre du modèle #3, nous nous sommes heurtés à des difficultés liées aux limites des solveurs de contraintes et en particulier à celles de CHOCO. Les contraintes permettant de calculer l'intersection entre deux ensembles de segments temporels n'étant pas disponibles nativement dans ce solveur. Nous avons alors été amenés à proposer de nouvelles contraintes et implémenter les algorithmes de filtrage associés. La réflexion autour de ce point a été menée conjointement avec l'équipe TASC. Cela nous a permis d'implémenter nous même au sein de Choco de nouvelles contraintes, comme nous l'avons expliqué dans la section 5.2.3, en page 82. Cela a aussi permis de sensibiliser l'équipe TASC à des besoins en contraintes qui ne sont pas disponibles aujourd'hui dans Choco, et les a amené à travailler sur cette question. L'objectif est de définir une contrainte générique qui s'applique sur deux ensembles de segments (sous forme d'intervalles de temps). Le travail mené par l'équipe TASC s'est basé sur nos données et notre application et a permis d'apporter une amélioration au solveur Choco. Les résultats de ce travail ont fait l'objet de la publication d'un article par l'équipe TASC dans une conférence du domaine de la programmation par contraintes [DFP+15].

7.2 Perspectives

Les résultats de cette thèse ouvrent de nouvelles pistes de travail.

Amélioration du modèle #3. Dans le modèle #3, nous avons proposé de fixer *a priori* le paramètre n qui correspond au nombre d'extraits qui forment le résumé. Il s'agit de la principale limite de ce modèle. En pratique, cela nécessite de générer des résumés en variant n et de choisir la valeur de ce paramètre qui soit la plus *appropriée* pour un type de contenus particulier et pour un cas d'usage précis. Ceci peut parfois s'avérer contraignant, voire très difficile à réaliser en raison de la nécessité de définir des critères d'évaluation des résumés générés. Une solution consisterait à considérer ce paramètre comme une variable de décision que le solveur fixera en fonction de la vidéo à résumer

[3] Équipe TASC. https://www.lina.univ-nantes.fr/?-TASC-.html

et en fonction des contraintes. Ainsi, pour deux vidéos de mêmes types et pour un même ensemble de contraintes, le solveur peut générer deux résumés, chacun avec un nombre d'extraits différent. Une première tentative d'implémentation de cette solution nous a montré que l'efficacité de la solution se dégrade drastiquement. En d'autres termes, cela augmente significativement le temps de réponse.

Une solution possible serait de revoir la modélisation en utilisant les automates en programmation par contraintes [BCD+05]. De la même façon que les automates ordinaires, les automates dans un système de contraintes sont définis par un ensemble d'états, un ensemble de transitions, un alphabet, des états initiaux et des états finaux, comme illustré dans la figure 7.1. L'automate permet de reconnaître (ou pas) une séquence lue. Une séquence reconnue par un automate représente un mot du langage. Une séquence lue, par un automate en programmation par contraintes, représente une séquence de variables de décision. Ces variables peuvent représenter des segments contigus, de très courte durée, permettant de segmenter la vidéo d'entrée et de décider si le segment est à sélectionner ou non. Il serait intéressant donc d'essayer de modéliser notre problème de création de résumés en utilisant un ou plusieurs automates. La particularité liée à notre problème serait de pouvoir sélectionner, non pas toute la séquence qui doit être reconnue par l'automate, mais seulement une sous séquence qui représente le résumé (une partie de la vidéo). L'utilisation des automates dans un système de contraintes permettrait de faire disparaître les comparaisons des intervalles et, par conséquent, proposer un modèle plus efficace.

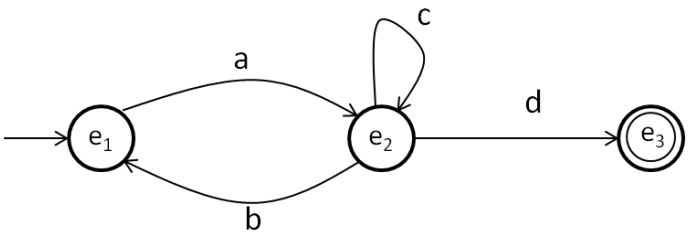

FIGURE 7.1 : Exemple illustratif d'un automate : $\{e_1, e_2, e_3\}$ est l'ensemble d'états, e_1 est l'état initial, e_2 est l'état final et $\{a, b, c, d\}$ est l'alphabet de l'automate.

Prise en compte de la structure des vidéos. Notre approche de création automatique de résumés de matchs de tennis n'a pris en compte aucune structuration temporelle du contenu vidéo. Nous l'avons appliqué directement sur la vidéo sans que le contenu de cette dernière ne soit finement modélisé. Nous avons, par exemple, détecté et inclus la balle de match d'une façon indirecte et implicite (en utilisant des attributs de bas

niveau n'ayant aucune lien avec la structure de la vidéo). Un match de tennis est, par contre, un contenu très bien structuré qui respecte un ensemble de règles du jeu. La figure 7.2 illustre la structuration d'un match de tennis. Plusieurs travaux ont proposé des méthodes d'indexation et de structuration automatiques de vidéos de matchs de tennis. On peut citer à titre d'exemple les travaux de thèse d'Ewa Kijak [Kijo3] et de Manolis Delakis [Delo6].

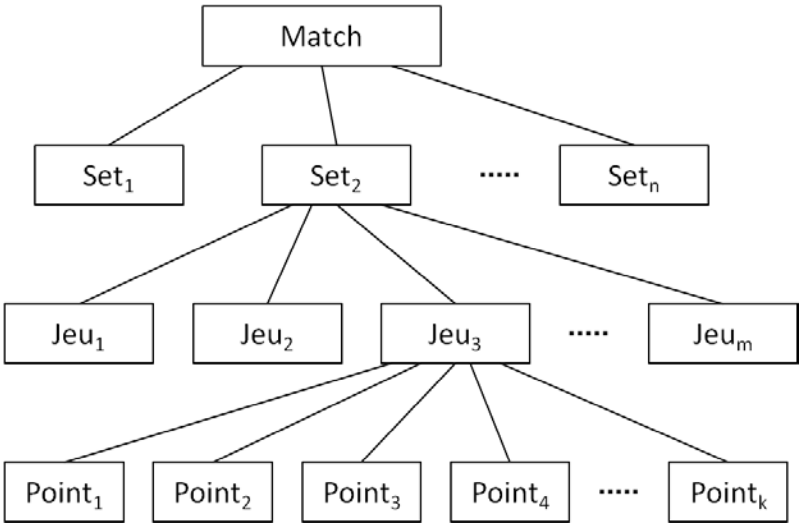

FIGURE 7.2 : Structure intrinsèque d'un match de tennis.

Il serait intéressant, dans le cadre d'une expérience à mener, d'étudier la possibilité d'appliquer nos modèles basés sur la programmation par contrainte sur un match de tennis structuré automatiquement. Autrement dit, étudier la complémentarité entre la formalisation des contraintes et la structuration de la vidéo. Il serait intéressant également d'aller plus loin en étudiant la possibilité de reformuler le problème différemment de façon à intégrer des contraintes qui prennent en compte la structure. Ici, un des problèmes difficiles à traiter concerne la formalisation des informations de structure dans la modélisation basée programmation par contrainte. En termes de contraintes, une façon simple serait d'exprimer des contraintes par niveau de structures, ce qui permet d'imposer la sélection d'un nombre fixe ou minimal d'extraits par « Set » ou par « Jeu ».

Application sur d'autres types de contenus. D'autres perspectives qui s'inscrivent dans la continuité de nos travaux de thèse concernent la validation de notre approche sur d'autres types de contenus, tel que les journaux TV et les jeux télévisés. Ceux-ci en particulier ont fait l'objet d'expériences lors du développement du modèle #1. En particulier, les journaux TV sont bien structurés. De façon générale, un journal télévisé

est composé de séquences (présentateur + reportage + interview) comme le montre la figure 7.3.

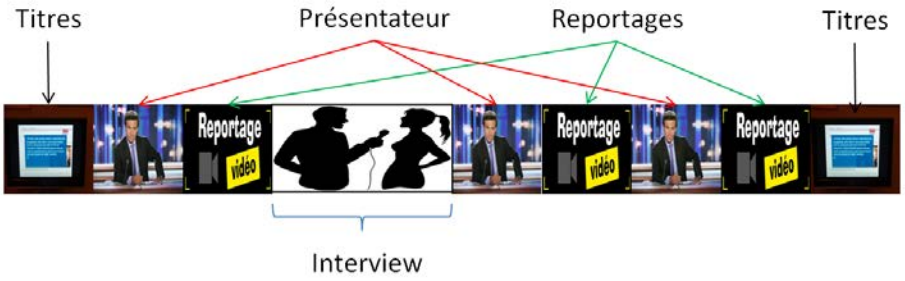

FIGURE 7.3 : Structure temporelle d'un journal TV.

La succession des scènes de plateau (présentateurs) et des reportages peut, non seulement, être utile dans la sélection des extraits significatifs et intéressants, mais aussi pour avoir une cohérence dans le résumé généré. Par exemple, si un extrait d'une scène de plateau est sélectionné dans le résumé, alors le reportage associé à cette scène doit aussi être représenté dans le résumé.

D'autre part, l'information textuelle, issue de la transcription de la parole, du texte incrusté dans la vidéo ou des sous-titres, peut être utilisée pour mieux répondre aux préférences des utilisateurs. En effet, l'information textuelle permet d'effectuer une segmentation thématique caractérisée par un ensemble de mots-clés. Cette segmentation permet de sélectionner les extraits qui correspondent au mieux aux préférences de l'utilisateur, préférences qui peuvent être déclarées ou déduites automatiquement en analysant les traces de consommation de l'utilisateur. Contrairement aux vidéos de sport, les centres d'intérêt des utilisateurs s'expriment fortement dans les vidéos de journaux TV. Un utilisateur peut s'intéresser uniquement aux actualités politiques alors qu'un autre peut s'intéresser à l'économie et aux actualités culturelles. De toute façon, dans un journal TV, il est indispensable de comprendre finement le contenu via l'analyse des informations textuelles car il n'existe pas dans journal TV d'indicateurs universels (comme les applaudissements) sur les séquences intéressantes.

D'autres expérimentations et d'autres attributs. Des expériences supplémentaires sur la flexibilité et l'adaptabilité de la méthode feront également partie de nos travaux futurs. En l'occurrence, la quantité d'information présente dans le résumé généré peut être mesurée en calculant la similarité entre le résumé et la vidéo d'entrée. D'autres attributs peuvent aussi être impliqués par des contraintes et intégrés dans le processus de génération de résumés tels que la détection des segments de ralenti (*replay*) pour les résumés

de vidéos de sport, la détection des sous-titres pour les résumés des journaux TV et la quantité de mouvement pour les résumés des films d'action. Une autre piste serait d'intégrer les résumés textuels générés par la méthode proposée dans le cadre du projet de master présenté dans la première partie de ce chapitre. Ces résumés textuels peuvent pointer vers les parties intéressantes et représenter une modalité de plus pour notre méthode de création de résumés vidéo, sur laquelle, de nouvelles contraintes peuvent être exprimées.

Résumé d'un ensemble de vidéos. Les travaux que nous avons menés dans cette thèse ont porté sur des vidéos considérées individuellement. Il serait intéressant d'étudier l'extension de notre approche à la création automatique de résumé d'un ensemble de vidéos, appelé communément document multi-vidéo. Un document multi-vidéo est un condensé de vidéos pouvant être de différents types et/ou provenant de différentes sources. Un enregistrement continu du flux TV diffusé pendant une demi-journée sur une chaîne TV ou une collection de vidéos récupérées à partir d'une plateforme de partage de vidéos sur internet, sont des exemples de documents multi-vidéos. Plusieurs approches ont été proposées pour ce type de contenus [LM10, WM09]. La diversité thématique au sein d'un même document multi-vidéo est la principale difficulté. Une analyse « sémantique » du contenu du document est donc indispensable dans la création de résumés. Pour appliquer notre approche sur des documents multi-vidéo, plusieurs questions se posent : serait-il efficace, dans le cas d'un condensé de vidéos de différents types, de définir seulement des contraintes génériques qui seraient indépendantes du type de la vidéo ou bien structurer la vidéo par une étape de délinéarisation et exprimer, par la suite, pour chaque segment, ses contraintes appropriées ? Serait-il intéressant, dans le cas d'un condensé de vidéos du même type, créer un seul résumé du document multi-vidéo sans considérer la répartition temporelle des extraits sélectionnés ou bien créer un résumé par vidéo dans une première étape et assembler les résumés générés dans une deuxième étape ?

Un langage de haut niveau pour exprimer les contraintes. Il s'agit de proposer un langage de haut niveau qui permet aux utilisateurs de spécifier leurs contraintes sans avoir des connaissances en programmation. Actuellement, même si les règles de production et le solveur sont séparés, l'utilisateur doit formuler ses contraintes en respectant la syntaxe du langage du solveur de contraintes choisi (CHOCO dans notre cas). Il serait intéressant de proposer un nouveau langage de haut niveau qui permet de rendre le processus de génération de résumés encore plus transparent pour l'utilisateur. La

proposition d'un langage de plus haut niveau est indispensable pour permettre aux utilisateurs *non experts* en programmation par contraintes d'exprimer aisément leurs contraintes. Cela consiste à proposer une **grammaire** contenant les règles définissant le nouveau langage. L'utilisation d'un outil de génération automatique d'analyseur permettant de reconnaître les mots et les phrases d'un programme écrit en langage naturel, est donc indispensable. ANTLR (*ANother Tool for Language Recognition*) est un exemple d'outil puissant pour la gestion des langages qui permet de générer un analyseur à partir de descriptions grammaticales. La difficulté se présente dans l'étape de définition de la grammaire et le nombre de mots qu'elle contiendra afin de mettre en place un langage simple et intuitif. En d'autres termes, la grammaire définira un mot pour chaque contrainte globale (une relation d'Allen) ou elle rentrera dans les détails de la contrainte en l'exprimant à l'aide d'un ensemble de mots pour permettre au nouveau langage de traiter la spécialisation des contraintes (one-to-one et one-to-all par exemple).

Déploiement de la solution. Dans la continuité des travaux réalisés dans cette thèse, il serait intéressant également de lancer des tests à large échelle sur une sélection de programmes TV dans le cadre d'une application ou d'un service en version beta, destinés à une sélection d'utilisateurs, et d'étudier l'industrialisation de l'approche avec des unités d'affaires du Groupe Orange.

Dans le contexte de cette thèse, les articles suivants ont été publiés :

Conférences internationales

- Sid-Ahmed Berrani, Haykel Boukadida et Patrick Gros. Constraint satisfaction programming for video summarization. Dans *Proceedings of the IEEE International Symposium on Multimedia (ISM)*, Anaheim, CA, USA, December, 2013, pp.195-202.

- Haykel Boukadida, Sid-Ahmed Berrani et Patrick Gros. A novel modeling for video summarization using constraint satisfaction programming. Dans *Proceedings of the* 10^{th} *International Symposium on Advances in Visual computing (ISVC)*, Las Vegas, NV, USA, December, 2014, pp.208-219.

Journal international

- Haykel Boukadida, Sid-Ahmed Berrani et Patrick Gros. Automatically creating adaptive video summaries using constraint satisfaction programming : Application to Sport Content. Dans *IEEE Transactions on Circuits and Systems for Video Technology (TCSVT)*, December, 2015.

Bibliographie

[AKT03] Y. Ariki, M. Kumano et K. Tsukada : Highlight scene extraction in real time from baseball live video. *In Proceedings of the 5th international workshop on Multimedia information retrieval*, pages 209–214, New York, NY, USA, 2003. ACM. 28

[All83] J.F Allen : Maintaining knowledge about temporal intervals. *Communications of the ACM*, 26(11):832–843, 1983. 79

[BCD+05] N. Beldiceanu, M. Carlsson, R. Debruyne *et al.* : Reformulation of global constraints based on constraints checkers. *Constraints*, 10(4):339–362, 2005. 110

[BCR05] N. Beldiceanu, M. Carlsson et J.X. Rampon : Global constraint catalog. *Technical Report T2005 :08, SICS*, 2005. 82

[BPS99] S.C. Brailsford, C.N Potts et B.M Smith : Constraint satisfaction problems : Algorithms and applications. *European Journal of Operational Research*, 119(3):557–581, 1999. 32

[BPS+09] S. Bengio, F. Pereira, Y. Singer *et al.* : Group sparse coding. *In Advances in Neural Information Processing Systems*, pages 82–89, 2009. 18

[BR96] J.S. Boreczky et L.A. Rowe : Comparison of video shot boundary detection techniques. *Journal of Electronic Imaging*, 5(2):122–128, 1996. 49

[BSR10] R. Bartak, M.A. Salido et F. Rossi : Constraint satisfaction techniques in planning and scheduling. *Journal of Intelligent Manufacturing*, 21(1):5–15, 2010. 32

[BT08] W. Bailer et G. Thallinger : Comparison of content selection methods for skimming rushes video. *In Proceedings of the 2nd ACM TRECVid Video Summarization Workshop*, pages 85–89, New York, USA, 2008. 22, 23

[Cai14] C. Caio : *Résumé automatique de texte appliqué au résumé multi-modal de vidéos par programmation par contraintes*. Thèse de doctorat, Université Rennes 1, 2014. 108

[CDV11] F. Chen et C. De Vleeschouwer : Automatic summarization of broadcasted soccer videos with adaptive fast-forwarding. *In IEEE International Conference on Multimedia and Expo*, pages 1–6, 2011. 19

[CF02] M. Cooper et J. Foote : Summarizing video using non-negative similarity matrix factorization. *In Proceedings of the IEEE Workshop on Multimedia Signal Processing, St. Thomas, Virgin Islands, USA*, pages 25–28, 2002. 16

[CHG02] P. Chang, M. Han et Y. Gong : Extract highlights from baseball game video with hidden markov models. *In Proc. of the International Conference on Image Processing*, volume 1, pages 609–612, Rochester, NY, USA, 2002. 19, 28

[CHL+08] M.G. Christel, A.G. Hauptmann, W. Lin *et al.* : Exploring the utility of fast-forward surrogates for bbc rushes. *In Proceedings of the 2nd ACM TRECVid Video Summarization Workshop*, pages 35–39, New York, USA, 2008. 25

[dAdLdA08] S. de Avila, A. da Luz et A .and others de Araujo : Vsumm : An approach for automatic video summarization and quantitative evaluation. *In XXI Brazilian Symposium on Computer Graphics and Image Processing*, pages 103–110, Campo Grande, Brazil, 2008. IEEE. 95

[Del06] E. Delakis : *Multimodal Structuring of Tennis Videos using Segment Models*. Thèse de doctorat, Université Rennes 1, 2006. 111

[DFP+15] A. Derrien, J.G. Fages, T. Petit *et al.* : A global constraint for a tractable class of temporal optimization problems. *In Principles and Practice of Constraint Programming*, pages 105–120, Ireland, 2015. Springer. 109

[DR14] C.T. Dang et H. Radha : Heterogeneity image patch index and its application to consumer video summarization. *IEEE Transactions on Image Processing*, 23(6):2704–2718, 2014. 17

[DRGN08] L. De Raedt, T. Guns et S. Nijssen : Constraint programming for itemset mining. *In Proceedings of the 14th international conference on Knowledge discovery and data mining*, pages 204–212. ACM, 2008. 32

[ERP+08] G. Evangelopoulos, K. Rapantzikos, A. Potamianos *et al.* : Movie summarization based on audiovisual saliency detection. *In 15th IEEE Internatio-*

nal Conference on Image Processing, pages 2528–2531, San Diego, California, USA, 2008. 13

[ETM03] A. Ekin, A.M. Tekalp et R. Mehrotra : Automatic soccer video analysis and summarization. *IEEE Transactions on Image Processing*, 12(7):796–807, 2003. 19

[EZS+09] G Evangelopoulos, A Zlatintsi, G Skoumas *et al.* : Video event detection and summarization using audio, visual and text saliency. *In Proceedings of the International Conference on Acoustics, Speech and Signal Processing*, pages 3553–3556, Taipei, Taiwan, 2009. 13

[GD04] C. Garcia et M. Delakis : Convolutional face finder : A neural architecture for fast and robust face detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(11):1408–1423, 2004. 51

[GG12] A. Gersho et R.M. Gray : *Vector quantization and signal compression*, volume 159. Springer Science & Business Media, 2012. 50

[GGR+14] M. Gygli, H. Grabner, H. Riemenschneider *et al.* : Creating summaries from user videos. *In Computer Vision-ECCV*, pages 505–520. Springer, 2014. 14

[GL01] Y. Gong et X. Liu : Video summarization with minimal visual content redundancies. *In Proceedings of the International Conference on Image Processing*, volume 3, pages 362–365, Thessaloniki, 2001. IEEE. 16

[Gon03] Y. Gong : Summarizing audiovisual contents of a video program. *EURASIP Journal on Advances in Signal Processing*, 2003(2):160–169, 2003. 16

[Han02] A. Hanjalic : Shot-boundary detection : unraveled and resolved ? *IEEE Transactions on Circuits and Systems for Video Technology*, 12(2):90–105, 2002. 49

[HCL+07] A.G. Hauptmann, M.G. Christel, W. Lin *et al.* : Clever clustering vs. simple speed-up for summarizing rushes. *In Proceedings of the international workshop on TRECVID video summarization*, pages 20–24, 2007. 25

[HE80] R.M Haralick et G.L Elliott : Increasing tree search efficiency for constraint satisfaction problems. *Artificial intelligence*, 14(3):263–313, 1980. 39

[HMD04] M. Huang, A.B. Mahajan et D.F. DeMenthon : Automatic performance evaluation for video summarization. Rapport technique, DTIC Document, 2004. 28

[HZ05] X.S. Hua et H.J. Zhang : An attention-based decision fusion scheme for multimedia information retrieval. *In Advances in Multimedia Information Processing-PCM 2004*, pages 1001–1010. Springer, 2005. 14

[IS08] K. Ishihara et Y. Sakai : Bbc rush summarization and high-level feature extraction in trecvid2008. *In TRECVID*, 2008. 24

[JHvD08] M. Jansen, W. Heeren et B. van Dijk : Videotrees : Improving video surrogate presentation using hierarchy. *In International Workshop on Content-Based Multimedia Indexing*, pages 560–567, 2008. 27

[JRL08] N. Jussien, G. Rochart et X. Lorca : Choco : an open source java constraint programming library. *In Workshop on Open-Source Software for Integer and Contraint Programming*, pages 1–10, 2008. 39

[Kij03] E. Kijak : *Structuration multimodale des vidéos de sport par modèles stochastiques*. Thèse de doctorat, Université Rennes 1, 2003. 111

[KSV+08] M. Koskela, M. Sjoberg, V. Viitaniemi *et al.* : Picsom experiments in trecvid 2008. *In TRECVID*, 2008. 22, 24

[LG13] Z. Lu et K. Grauman : Story-driven summarization for egocentric video. *In IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2714–2721, 2013. 19

[LG15] Y.J. Lee et K. Grauman : Predicting important objects for egocentric video summarization. *International Journal of Computer Vision*, pages 1–18, 2015. 19

[LGF+10] K. Li, L. Guo, C. Faraco *et al.* : Human-centered attention models for video summarization. *In International Conference on Multimodal Interfaces and the Workshop on Machine Learning for Multimodal Interaction*, page 27. ACM, 2010. 14

[LHC10] D. Liu, G. Hua et T. Chen : A hierarchical visual model for video object summarization. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(12):2178–2190, 2010. 18

[LK03] R.T.M. LYU et I.K.C. KING : Video summarization using greedy method in a constraint satisfaction framework. *In In Proceedings of the 9th International Conference on Distributed Multimedia Systems*, pages 456–461, 2003. 14

[LKL05] S. Lu, I. King et M.R. Lyu : A novel video summarization framework for document preparation and archival applications. *In IEEE Aerospace Conference*, pages 1–10, Big Sky, Montana, 2005. 16

[LL09] Y. Li et Z. Lu : Video abstraction via attention model and on-line clustering. *In Proc. of the 4th Int. Conf. on Innovative Computing, Information and Control*, pages 627–630, Kaohsiung, Taiwan, 2009. 13

[LLK04] S. Lu, M.R. Lyu et I. King : Video summarization by spatial-temporal graph optimization. *In Proceedings of the International Symposium on Circuits and Systems*, volume 2, pages 197–200, 2004. 16

[LLR+08] Y. Liu, Y. Liu, T. Ren *et al.* : Rushes video summarization using audio-visual information and sequence alignment. *In Proceedings of the 2nd ACM TRECVid Video Summarization Workshop*, pages 114–118, 2008. 24

[LM10] Y. Li et B. Merialdo : Multi-video summarization based on av-mmr. *In International Workshop on Content-Based Multimedia Indexing (CBMI)*, pages 1–6. IEEE, 2010. 113

[LMR+11] Y. Li, B. Merialdo, M. Rouvier *et al.* : Static and dynamic video summaries. *In Proceedings of the 19th ACM international conference on Multimedia*, pages 1573–1576, 2011. 16

[Low99] D.G. Lowe : Object recognition from local scale-invariant features. *In The proceedings of the seventh IEEE international conference on Computer vision*, volume 2, pages 1150–1157, 1999. 23

[LRvD10] C. Lecoutre, O. Roussel et M. R. van Dongen : Promoting robust black-box solvers through competitions. *Constraints*, 15(3):317–326, 2010. 40

[LSK05] Z. Li, G.M. Schuster et A.K. Katsaggelos : Minmax optimal video summarization. *IEEE Transactions on Circuits and Systems for Video Technology*, 15(10):1245–1256, 2005. 15

[LYG+08] Z. Longfei, C. Yuanda, D. Gangyi *et al.* : A computable visual attention model for video skimming. *In Proc. of the 10th International Symposium on Multimedia*, pages 667–672, Berkeley, CA, USA, 2008. 13

[LZF+04] T. Liu, X. Zhang, J. Feng *et al.* : Shot reconstruction degree : a novel criterion for key frame selection. *Pattern recognition letters*, 25(12):1451–1457, 2004. 28

[MA08] A.G. Money et H. Agius : Video summarisation : A conceptual framework and survey of the state of the art. *Journal of Visual Communication and Image Representation*, 19(2):121–143, 2008. 11

[MBB+11] A. Marcus, M.S. Bernstein, O. Badar *et al.* : Twitinfo : aggregating and visualizing microblogs for event exploration. *In Proceedings of the annual*

conference on Human factors in computing systems, pages 227–236. ACM, 2011. 20

[MG+07] F. Mamalet, C. Garcia *et al.* : Real-time video convolutional face finder on embedded platforms. *EURASIP Journal on Embedded Systems*, 2007. 51

[MG08] E. McCrum-Gardner : Which is the correct statistical test to use ? *British Journal of Oral and Maxillofacial Surgery*, 46(1):38–41, 2008. 97

[MHL+05] Y. Ma, X. Hua, L. Lu *et al.* : A generic framework of user attention model and its application in video summarization. *IEEE Transactions on Multimedia*, 7(5):907–919, 2005. 14

[NDM+08] S.U. Naci, U. Damnjanovic, B. Mansencal *et al.* : The cost292 experimental framework for rushes summarization task in trecvid 2008. *In Proceedings of the 2nd ACM TRECVid Video Summarization Workshop*, pages 40–44, Vancouver, Canada, 2008. 22, 24

[NMZ03] C.W. Ngo, Y.F. Ma et H. Zhang : Automatic video summarization by graph modeling. *In Proceedings in 9th IEEE International Conference on Computer Vision*, pages 104–109, 2003. 27

[NMZ05] C.W. Ngo, Y.F. Ma et H.J. Zhang : Video summarization and scene detection by graph modeling. *IEEE Transactions on Circuits and Systems for Video Technology*, 15(2):296–305, 2005. 14

[NT99] J. Nam et A.H. Tewfik : Video abstract of video. *In IEEE 3rd Workshop on Multimedia Signal Processing*, pages 117–122, Copenhagen, 1999. 16

[OSA08] P. Over, A.F Smeaton et G. Awad : The trecvid 2008 bbc rushes summarization evaluation. *In Proceedings of the 2nd ACM TRECVid Video Summarization Workshop*, pages 1–20, Vancouver, BC, Canada, 2008. 22

[OSK07] P. Over, A.F. Smeaton et P. Kelly : The trecvid 2007 bbc rushes summarization evaluation pilot. *In Proceedings of the international workshop on TRECVID video summarization*, pages 1–15, New York, USA, 2007. 22

[PAS+99] D. Ponceleon, A. Amir, S. Srinivasan *et al.* : Cuevideo : automated multimedia indexing and retrieval. *In Proceedings of the seventh ACM international conference on Multimedia (Part 2)*, page 199, New York, NY, USA, 1999. 15, 25

[PLF+96] S. Pfeiffer, R. Lienhart, S. Fischer *et al.* : Abstracting digital movies automatically. *Journal of Visual Communication and Image Representation*, 7(4): 345–353, 1996. 18

[PN05] Y. Peng et C. Ngo : Hot event detection and summarization by graph modeling and matching. *In Image and Video Retrieval*, pages 257–266. Springer, 2005. 18

[QBPM+08] G. Quenot, J. Benois-Pineau, B. Mansencal *et al.* : Rushes summarization by irim consortium : redundancy removal and multi-feature fusion. *In Proceedings of the 2nd ACM TRECVid Video Summarization Workshop*, pages 80–84, 2008. 24

[RB08] A. Ramsden et A. Bate : Using word clouds in teaching and learning. *Opus : University of Bath Online Publication Store*, 2008. 104

[Rey09] D. Reynolds : Gaussian mixture models. *In Encyclopedia of Biometrics*, pages 659–663. Springer, 2009. 50

[RGA00] Y. Rui, A. Gupta et A. Acero : Automatically extracting highlights for tv baseball programs. *In Proceedings of the eighth ACM international conference on Multimedia*, pages 105–115, New York, USA, 2000. 19

[RJ09] J. Ren et J. Jiang : Hierarchical modeling and adaptive clustering for real-time summarization of rush videos. *IEEE Transactions on Multimedia,*, 11(5): 906–917, 2009. 22, 24

[RK14] S.P. Rajendra et N. Keshaveni : A survey of automatic video summarization techniques. *International Journal of Electronics, Electrical and Computational System*, 2, 2014. 11

[RVBW06] F. Rossi, P. Van Beek et T. Walsh : *Handbook of constraint programming*. Elsevier, Oct 2006. 31

[San07] S. Santini : Who needs video summarization anyway ? *In International Conference on Semantic Computing*, pages 177–184, Irvine, CA, 2007. IEEE. 27, 95

[SC01] H. Sundaram et S. Chang : Condensing computable scenes using visual complexity and film syntax analysis. *In Proceedings of the IEEE International Conference on Multimedia and Expo*, pages 273–276, Tokyo, Japan, 2001. 95

[SC03] H. Sundaram et S. Chang : Video analysis and summarization at structural and semantic levels. *Multimedia Information Retrieval and Management : Technological Fundamentals and Applications*, pages 75–94, 2003. 19

[SJ72] K. Sparck Jones : A statistical interpretation of term specificity and its application in retrieval. *Journal of documentation*, 28(1):11–21, 1972. 20, 104

[SK95] M.A. Smith et T. Kanade : *Video skimming for quick browsing based on audio and image characterization*. Citeseer, Pittsburgh, PA, USA, 1995. 17

[SKC10] D.A. Shamma, L. Kennedy et E.F. Churchill : Summarizing media through short-messaging services. *In Proceedings of the ACM conference on computer supported cooperative work*, pages 551–552, USA, 2010. 20

[SM00] J. Shi et J. Malik : Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):888–905, 2000. 14

[SRT08] J. Sasongko, C. Rohr et D. Tjondronegoro : Efficient generation of pleasant video summaries. *In Proceedings of the 2nd ACM TRECVid Video Summarization Workshop*, pages 119–123, New York, USA, 2008. 22

[SS97] E. Scheirer et M. Slaney : Construction and evaluation of a robust multifeature speech/music discriminator. *In Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, volume 2, pages 1331–1334, Munich, 1997. 50

[STC10] N. Shroff, P. Turaga et R. Chellappa : Video précis : Highlighting diverse aspects of videos. *IEEE Transactions on Multimedia*, 12(8):853–868, 2010. 17

[TB12] A. Tang et S. Boring : Epicplay : crowd-sourcing sports video highlights. *In Proceedings of the annual conference on Human Factors in Computing Systems*, pages 1569–1572, New York, USA, 2012. ACM. 19

[TKL+13] C.M. Tsai, L.W. Kang, C.W. Lin *et al.* : Scene-based movie summarization via role-community networks. *IEEE Transactions on Circuits and Systems for Video Technology*, 23(11):1927–1940, 2013. 18

[TPA+06] C. Taskiran, Z. Pizlo, A. Amir *et al.* : Automated video program summarization using speech transcripts. *IEEE Transactions on Multimedia*, 8(4):775–791, 2006. 27

[TV06] B.T. Truong et S. Venkatesh : Curtin at trecvid 2006-rushes summarization. *In TRECVID Workshop*, Gaithersburg, Md, 2006. 23

[TV07] B.T. Truong et S. Venkatesh : Video abstraction : A systematic review and classification. *ACM Transactions on Multimedia Computing, Communications, and Applications*, 3(1):3, 2007. 27

[TYO11] H. Takamura, H. Yokono et M. Okumura : Summarizing a document stream. *In Advances in Information Retrieval*, pages 177–188. Springer, 2011. 20

[TZL+06] S. Tang, Y. Zhang, J. Li *et al.* : Rushes exploitation 2006 by cas mcg. *In TRECVID Workshop*, 2006. 24

[WFW+08] T. Wang, S. Feng, P.P. Wang *et al.* : Thu-intel at rushes summarization of trecvid 2008. *In Proceedings of the 2nd ACM TRECVid Video Summarization Workshop*, pages 124–128, 2008. 23

[WM09] F. Wang et B. Merialdo : Multi-document video summarization. *In International Conference on Multimedia and Expo*, pages 1326–1329. IEEE, 2009. 113

[XRD03] Z. Xiong, R. Radhakrishnan et A. Divakaran : Generation of sports highlights using motion activity in combination with a common audio feature extraction framework. *In Proceedings of the International Conference on Image Processing*, volume 1, pages 1–5. IEEE, 2003. 19

[YCK+08] N.C. Yang, W.H. Chang, C.M. Kuo *et al.* : A fast mpeg-7 dominant color extraction with new similarity measure for image retrieval. *Journal of Visual Communication and Image Representation*, 19(2):92–105, 2008. 50

[YCZ+03] H. Yang, L. Chaisorn, Y. Zhao *et al.* : Videoqa : question answering on news video. *In Proceedings of the eleventh ACM international conference on Multimedia*, pages 632–641, 2003. 19

[YMH03] I. Yahiaoui, B. Merialdo et B. Huet : Comparison of multiepisode video summarization algorithms. *EURASIP Journal on applied signal processing*, pages 48–55, 2003. 28

[YYW+95] M.M. Yeung, B.L. Yeo, W.H. Wolf *et al.* : Video browsing using clustering and scene transitions on compressed sequences. *In Symposium on Electronic Imaging : Science and Technology*, pages 399–413. International Society for Optics and Photonics, 1995. 14

[ZE01] D. Zhang et D. Ellis : Detecting sound events in basketball video archive. *Dept. Electronic Eng., Columbia Univ., New York*, 2001. 19

[ZWV11] S. Zhao, J. Wickramasuriya et V. Vasudevan : Analyzing twitter for social tv : Sentiment extraction for sports. *In Proceedings of the 2nd International Workshop on Future of Television*, 2011. 20

[ZX14] B. Zhao et E.P. Xing : Quasi real-time summarization for consumer videos. *In IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2513–2520, 2014. 18