



**Université libanaise**  
**Faculté des sciences**



**Université Paul Sabatier**  
**Toulouse III-I.R.I.T**

**DEA d'informatique**  
**Coopération entre les Sciences de Traitement de l'Information**

Année universitaire 2005/2006

**Evaluation d'une chaîne d'outils d'indexation  
multimédia**

Préparé par : Hassan WEHBÉ

Responsable : Dr. Bassem HAIDAR

Jury : Ali AWADA  
Kabalan BARBAR  
Bilal CHEBARO  
Siba HAIDAR



## *Remerciements*

Mes premiers remerciements vont pour mon encadrant M. Bassem HAIDAR. Mes sentiments de reconnaissance et de gratitude, ainsi que mes expressions d'amitiés, vont pour lui. Je le remercie pour son aide, sa direction, et son suivi tout le temps, ainsi que pour la confiance qu'il m'a accordée au cours du sujet.

Je remercie mon prof M. Bilal CHEBARO pour ses idées innovantes qui m'ont aidé à évoluer ma recherche et à progresser dans le vrai chemin. Je remercie encore tous les profs de ma classe de DEA qui ont simplifié le cours pour bien comprendre et qui sont fatigués durant leurs voyages.

Mes sentiments les plus profonds et les plus aimables vont vers mon père Ali et ma mère Souad qui sont fatigués dans le travail pour m'offrir les bonnes conditions de recherche, ainsi que pour leurs encouragements pour continuer ma recherche. Merci aussi à mes sœurs et frères : Housein, Nour el-ein, Mohamed et Fatmeh que j'aime beaucoup. Je remercie chaleureusement tous les membres de ma famille et spécialement à ma bonne grand-mère.

Mes remerciements amicaux s'adressent à chaque membre de notre classe sans aucune exception. Je remercie fraternellement mon meilleur ami Hassan CHOUAIB qui m'a accompagné dans les bons et les mauvais jours.

Finalement, je remercie chaque personne qui m'a aidé à terminer mon projet avec succès.

## *Résumé*

*Avec l'augmentation de nombre des documents multimédia utilisés, le besoin d'identifier et de trouver facilement ces documents à partir des index a devenu très grand, d'où le domaine d'indexation multimédia. Les index sont générés en partant d'un document multimédia primitif, par un programme appelé outil d'indexation. Mais certains index ne peuvent pas être générés directement en utilisant un seul outil, d'où le besoin de chaîner un ensemble d'outils pour atteindre notre but. Généralement, ce chaînage s'effectue manuellement mais récemment un algorithme permettant d'automatiser ce processus a été proposé, mais sans classifier les chaînes résultants. Notre objectif dans ce sujet est de définir et mettre en place des critères visant à évaluer et classifier les chaînes d'outils d'indexation trouvées par cet algorithme.*

## *Abstract*

*With the number increase of the multimedia used documents, the need to identify and to find easily these documents from the indices have become very big, from which the indexation multimedia domain. The indices are generated while leaving a primitive multimedia document, by a program called indexation tool. However, certain indices cannot be generated directly using an alone tool, from which the need to chain a collection of tools to attain our goal. Generally, this chaining carries is manual, but recently an algorithm aiming to automate this process was proposed, but without classifying the resultant chains. Our objective in this subject is to define criteria aiming to evaluate and classify the chains of indexation tools found by this algorithm.*

***Mots-clés : Indexation multimédia, outil d'indexation, chaînage, évaluation.***

# TABLE DES MATIERES

<b>CHAPITRE 1 : INTRODUCTION.....</b>	<b>1</b>
<b>CHAPITRE 2 : ETAT DE L'ART.....</b>	<b>5</b>
<b>CHAPITRE 3 : LES PARAMETRES D'EVALUATION.....</b>	<b>9</b>
<b>1. INTRODUCTION .....</b>	<b>9</b>
<b>2. DEFINITIONS DES PARAMETRES .....</b>	<b>10</b>
2.1 LE TEMPS .....	10
2.2 LE COUT.....	10
2.3 LA FIABILITE.....	11
2.4 LA FIDELITE .....	11
<b>3. LA MISE A JOUR DES PARAMETRES.....</b>	<b>12</b>
<b>4. METHODE DE CALCUL DES PARAMETRES D'UNE CHAINE.....</b>	<b>13</b>
4.1 MODELE DE REDUCTION.....	13
4.2 LE CALCUL DES PARAMETRES .....	14
4.2.1 <i>Le temps</i> .....	14
4.2.2 <i>Le coût</i> .....	14
4.2.3 <i>La fiabilité</i> .....	15
4.2.4 <i>La fidélité</i> .....	15
4.3 LE CALCUL DU SCORE.....	16
<b>CHAPITRE 4 : METHODOLOGIE DE CHAINAGE.....</b>	<b>18</b>
<b>1. INTRODUCTION .....</b>	<b>18</b>
<b>2. REPRESENTATION DES REGLES DE COMPATIBILITE.....</b>	<b>18</b>
2.1 PREMIERE APPROCHE : UN SEUL GRAPHE POUR TOUS LES INDEX (FIGURE 1).....	19
2.2 DEUXIEME APPROCHE : UN GRAPHE PAR INDEX (FIGURE 2).....	20
<b>3. EXTRACTION DES CHAINES « SOLUTIONS ».....</b>	<b>21</b>
3.1 EXTRACTION DE LA MEILLEURE CHAINE .....	21
3.2 EXTRACTION DE TOUTES LES CHAINES.....	23
3.3 PROBLEME DE SURETE .....	23
<b>CHAPITRE 5 : IMPLEMENTATION .....</b>	<b>26</b>
<b>1. INTRODUCTION .....</b>	<b>26</b>
<b>2. REPRESENTATION DES OUTILS ET DES DOCUMENTS .....</b>	<b>27</b>
<b>3. REPRESENTATION D'UN GRAPHE .....</b>	<b>29</b>
3.1 EXEMPLE D'UN GRAPHE .....	30
<b>4. REPRESENTATION D'UNE CHAINE.....</b>	<b>31</b>
<b>5. ALGORITHME DE CONSTRUCTION DU GRAPHE.....</b>	<b>32</b>
5.1 PROBLEME DES CYCLES INFINIS .....	34
5.2 REDUCTION DE LA REDONDANCE DU CALCUL .....	36

<b>6. ALGORITHME D'EXTRACTION DES CHAINES SOLUTIONS .....</b>	<b>37</b>
<b>CHAPITRE 6 : RESULTATS.....</b>	<b>39</b>
<b>CONCLUSION.....</b>	<b>43</b>
<b>BIBLIOGRAPHIE .....</b>	<b>45</b>

# *Chapitre 1 : Introduction*

Le domaine d'indexation multimédia a été trouvé afin d'aider les utilisateurs à identifier, trouver, et filtrer les informations multimédia. Ce domaine a constitué récemment un domaine de recherche aux enjeux importants. Des applications permettant la sélection d'émissions télévisées, la recherche dans des catalogues de données culturelles (images, musiques, etc.) sont requises pour accompagner l'essor de la communication numérique.

Du fait de cette forte évolution, l'indexation manuelle des contenus n'est plus envisageable. Par conséquent, des algorithmes de traitement automatique permettant par exemple la segmentation et la structuration des contenus, la reconnaissance de la parole, l'identification automatique des langues, etc. deviennent nécessaires pour l'indexation et la recherche de données multimédia.

Les ressources d'indexation<sup>1</sup> ne sont pas souvent regroupées sur une seule machine, mais ils sont distribués sur plusieurs sites. Dans le but d'intégrer les technologies d'indexation distribuée d'une part, et d'améliorer la qualité des résultats produits, ou de générer de nouveaux index, dont la production aurait été impossible d'autre part, une plateforme partageant les ressources a été définie.

Dans le domaine de l'indexation multimédia, plusieurs types d'index ne peuvent pas être générés par un seul outil d'indexation. Une séquence d'outils chaînés dans un ordre précis doit être construite. Prenons l'exemple de la transcription de la parole à partir d'un contenu audio quelconque : pour arriver à extraire le texte en français, supposé être prononcé dans un enregistrement numérique, plusieurs étapes de traitement successives sont nécessaires : tout d'abord, le contenu audio doit être analysé pour identifier les segments de Parole/Musique/Bruit; ensuite les segments de parole font l'objet d'un traitement spécifique pour isoler ceux qui correspondent à la parole en langue

---

<sup>1</sup> Ressource d'indexation : outil d'indexation ou un document ou une métadonnée décrivant des contenus multimédia.

française ; Enfin l’algorithme de transcription peut être appliqué sur ces segments. On peut même imaginer qu’une segmentation et une identification du locuteur intervienne dans ce chaînage dans le but d’adapter les paramètres de la transcription (voir la figure 1). D’où le besoin du chaînage.

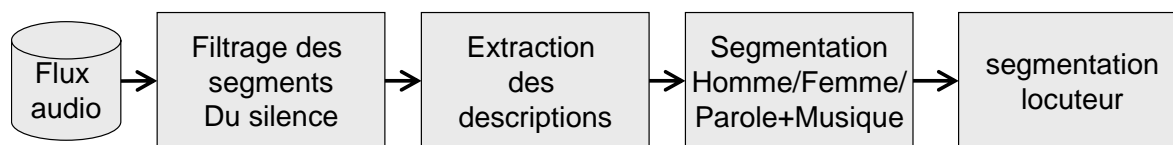


Figure 1 : un exemple de chaînage

Les travaux menés dans ce domaine consistent généralement à chaîner manuellement les outils d’indexation à travers des scénarios prédéfinis, dans des contextes bien définis. Mais récemment, la thèse de [Haidar05] a proposé un algorithme de chaînage automatique visant à trouver toutes les chaînes capables de générer un index donné.

La thèse de [Haidar05] a proposée l’idée consistant à intégrer les outils d’indexation dans des services distribués accessibles à distance, l’ensemble de ces services est regroupé par une plateforme centralisée.

L’algorithme capable de chaîner ces services prend en entrée l’index à générer. Ensuite, en partant de cet index, il cherche à établir toutes les relations de compatibilité entre les outils d’indexation<sup>2</sup> disponibles, ce qui nous permet de trouver toutes les chaînes solutions capables de générer cet index.

Une chaîne d’outils d’indexation est considérée comme une solution si, et seulement si, l’index qu’elle génère est obtenu en partant d’un seul document multimédia (audio, vidéo, image).

Cependant l’exécution distribuée d’une chaîne peut être arrêtée par l’absence d’un service (panne, maintenance, blocage, etc.). Or, l’algorithme proposé ne permet pas d’identifier la chaîne remplaçante. En effet, aucun critère d’évaluation d’outil d’indexation n’a pas été proposé, et par conséquent, il est impossible de classer les chaînes.

D’autre part, les critères de génération d’un index donné peuvent varier selon les préférences et les besoins de l’utilisateur. Par exemple un utilisateur

---

<sup>2</sup> Compatibilité des outils : Deux outils compatibles sont des outils dont le premier a parmi ces types d’entrée un type de donnée généré par le deuxième.

préfère un résultat en un temps minimal même si cette génération va coûter plus chère en terme de coût d'exécution, et un autre utilisateur peut exiger une grande fiabilité d'exécution sans s'intéresser au coût et au temps d'exécution.

Notre étude se focalise sur deux principaux objectifs :

- œ *Définir un ensemble de paramètres capable d'évaluer les outils d'indexation et par suite les chaînes d'outils* : un outil d'indexation est un programme exécutable, intégré dans un service accessible à distance. Notre objectif est de définir les propriétés caractérisant un service distribué intégrant un outil d'indexation multimédia, comme par exemple le temps, coût,...
- œ *Évaluer les chaînes selon les préférences de l'utilisateur* : l'algorithme proposé par [Haidar05] vise à découvrir toutes les chaînes capables de générer un index donné sans aucune différence entre une chaîne et une autre. Par conséquent, un problème apparaît qui est comment faut-il distinguer ces chaînes selon les préférences de l'utilisateur ? donc notre deuxième objectif est de classer ces chaînes selon les critères proposés par l'utilisateur. Un degré d'importance (pondération) peut être attribué pour chaque critère par l'utilisateur.

Cette classification de chaînes permettra, dans le cas de non disponibilité d'une chaîne, de choisir la meilleure parmi les chaînes fonctionnelles.

Dans le *chapitre 3* nous définissons les paramètres d'évaluation caractérisant chaque outil, nous présentons ensuite les mécanismes visant à calculer ces paramètres pour une chaîne entière. Finalement nous proposons une méthode capable de tenir compte de tous ses paramètres dans une seule valeur qui sera le score final caractérisant cette chaîne.

Le *chapitre 4* présente les approches proposées pour trouver et extraire les chaînes d'indexation en tenant compte du problème d'évaluation de ces chaînes. C'est-à-dire mettre en place la meilleure approche facilitant notre travail, pour présenter après les avantages et les inconvénients de chaque approche. Ensuite nous déterminons notre choix parmi les approches présentées.

Dans le *chapitre 5* nous détaillons l'implémentation de l'algorithme suivi pour trouver et évaluer les chaînes d'outils d'indexation. Nous définissons

durant ce chapitre, les structures de données nécessaires à la représentation des outils et des chaînes.

Un exemple de résultats générés clarifiant la méthodologie suivie par notre travail aura lieu dans le *chapitre 6*. En montrant l'intérêt de l'évaluation et de la classification des chaînes selon le choix de l'utilisateur.

Finalement, nous présentons la conclusion de nos travaux, et discutons les perspectives qu'ils ouvrent.

## *Chapitre 2 : Etat de l'art*

La forte utilisation des objets et des services distribués sur Internet a entraîné un besoin d'une composition et d'un chaînage de ces outils ou de ces services, dans le but de produire des résultats qui ne peuvent pas être produits par un simple outil, et qui ont besoin d'une chaîne d'outils pour être générés. Les travaux menés dans ce domaine ont consisté au début à chaîner manuellement les outils à travers des scénarios prédéfinis, dans des contextes bien définis. Le chaînage automatique des services est considéré récemment comme un sujet de recherche, pour cela peu de travaux concernant l'évaluation des chaînes résultantes ont été effectués.

Dans le domaine de l'indexation multimédia, beaucoup de travaux sont basés sur l'utilisation de plusieurs outils d'indexation pour générer un index donné. Dans [Haidar05] une étude concernant l'état de l'art du chaînage d'outils dans le domaine de l'indexation multimédia a été effectuée. Les travaux menés dans ce domaine consistent généralement à chaîner manuellement les outils d'indexation à travers des scénarios prédéfinis, dans des contextes bien définis.

Le chaînage des services est utilisé dans le domaine de services web, domaine qui est de plus en plus répandu sur Internet. Les services web sont des applications auto descriptives, modulaires, pouvant être publiées, localisées, et invoquées à travers le web. Les services web sont proposés pour répondre au besoin d'une architecture distribuée, indépendante des plateformes et des langages de programmation.

Les services web sont créés par des compagnies et des organisations hétérogènes, pour répondre à des tâches allant d'une simple requête à des processus complexes. Cependant, certaines requêtes utilisateurs ne peuvent pas être satisfaites par un simple service. Ces requêtes doivent être traitées par deux ou plusieurs services pour répondre à l'attente de l'utilisateur. De là découle la nécessité d'un mécanisme qui permet de chaîner plusieurs services pour répondre à une requête particulière. Cet ensemble de services constitue

une chaîne (ou un workflow) qui représente l'ordre d'exécution de ces services.

Généralement, ces chaînes doivent être évaluées pour aider l'utilisateur à choisir la meilleure chaîne à exécuter selon ses préférences. Selon [Cardoso02a] l'évaluation d'un workflow peut être divisée en deux types d'analyses :

- *Analyse qualitative* : c'est l'analyse de tous les facteurs concernant la qualité d'un service web comme l'absence des boucles infinies dans un service web, la validation d'un service web, le test que le service web se comporte comme attendu, la vérification de l'existence de certaines propriétés d'un service web
- *Analyse quantitative* : c'est l'analyse des paramètres caractérisant la plateforme, comme le délai, la gigue ou la variation du délai de bout en bout, le débit maximum atteint, la disponibilité ou le taux moyen d'erreurs d'une liaison

Concernant l'évaluation qualitative d'un workflow, ce sont les chercheurs de la communauté des réseaux de Petri [Diaz01] qui sont les plus actifs. Les travaux de [Narayanan02] ont aboutis à des résultats intéressants concernant ce problème, il propose des solutions pour décrire, simuler, composer automatiquement, tester et vérifier la composition de Web services. Cependant, les principaux travaux évoquent le problème de l'évaluation des performances de la QoS d'un Web service, sans s'occuper réellement du problème de l'évaluation quantitative, parce que les réseaux de Petri considérés n'intègrent pas le temps.

Très peu de travaux de recherche traitent le problème de l'évaluation des performances quantitatives de la QoS d'un Workflow. Un de ces travaux est celui basé sur un modèle de qualité proposée par la norme ISO 8402 (International Organisation for Standardization). Les auteurs proposent un modèle de QoS d'un Workflow construit en définissant des métriques de QoS des tâches pouvant composer un tel Workflow, ces métriques sont le temps, ainsi que le coût, la fiabilité, et la fidélité d'une tâche. De plus il faut connaître des caractéristiques supplémentaires comme la manière dont ces activités s'enchaînent, cependant l'obtention de ces caractéristiques n'est jamais facile, surtout si le fournisseur de service ne les précise pas.

La solution proposée par [Musa93] consiste à effectuer des mesures sur le système pour tenter, à l'aide de techniques statistiques, d'estimer ces

caractéristiques, puis de les injecter dans un modèle d'évaluation. En bref, ce modèle fournit une approche multidimensionnelle, dont les dimensions sont le temps, le coût, la fiabilité et la fidélité, ces métriques sont calculées automatiquement. Le modèle QoS proposé permet de calculer la QoS d'un Workflow à partir des métriques de ces composants selon deux techniques. Or, le choix d'une de ces techniques dépend essentiellement du compromis entre le temps de calcul des QoS d'un Workflow et la qualité des résultats obtenus :

- Première technique : en utilisant une modélisation mathématique [Cardoso02a] : ce modèle utilise des règles mathématiques pour calculer la QoS d'un workflow. Par conséquent, nous réduisons le workflow selon des règles prédéfinies, cependant la règle à utiliser sera choisie selon la situation du service à réduire par rapport à l'autres qui sont en contact avec lui (en séries, en parallèles). Ce modèle consomme un temps important car il parcourt tout le workflow plusieurs fois pour le réduire, avant d'atteindre l'objectif. D'autre part, les résultats obtenus sont sûrs, exacts, et de bonne qualité.
- Deuxième technique : en utilisant la simulation aléatoire à événements discrets [Narayanan02] : ce modèle ne calcule pas la QoS d'un workflow, mais il l'estime. Les données sur lesquelles ce modèle se base, sont obtenues à partir du comportement du service durant les simulations faites ou durant ses exécutions passées. Au contraire du premier modèle, celui là réduit le temps de production des résultats car il ne parcourt pas tout le workflow, mais la qualité des résultats est mauvaise, comme tous les cas où on estime les résultats, et par suite ils ne sont ni sûres ni exacts ce qui nous empêche de se baser sur ces résultats pour juger les chaînes générées.

Dans [Klingemann98], les auteurs proposent d'évaluer les performances quantitatives d'un Web service avec des chaînes de Markov à temps continu et espace d'état discret. Il est donc nécessaire de collecter des informations concernant ce service pendant son exécution (ce service étant réalisé par un fournisseur de services). Une hypothèse forte est alors que le comportement observé d'un service représente son comportement futur (l'objectif étant de faire des prédictions). Dans ces travaux, un service est constitué d'un ensemble de tâches qui pourront être activées lors d'une exécution de ce service. Les auteurs définissent l'état d'un service (observé) comme l'ensemble des tâches actives d'un service à un instant donné. L'exécution d'un service est alors modélisée par une chaîne de Markov dont les états sont

obtenus à partir du journal de ce service et dont les caractéristiques sont calculées par un processus d'agrégation des données [Klingemann98].

# *Chapitre 3 : Les paramètres d'évaluation*

## **1. Introduction**

Le but d'un utilisateur d'une plateforme de ressources d'indexations multimédia est de trouver la meilleure chaîne d'outils capable de générer son index. La recherche et la génération des chaînes d'outils capables de produire un index donné seront effectuées grâce à un algorithme de chaînage automatique qui sera expliqué dans les chapitres suivants, cet algorithme ne permet pas de différencier une chaîne (solution) d'une autre. Cependant l'utilisateur doit être capable de classifier les chaînes, afin de pouvoir en choisir la meilleure, ou la mieux adaptée à ses préférences. Les caractéristiques d'une chaîne dépendent fortement des caractéristiques des outils formant la chaîne, on peut constater que la spécification des chaînes exige la spécification des ces outils.

Nous proposons de spécifier un outil par l'ensemble des critères suivants : le temps d'exécution, le coût d'exécution, la fiabilité et la fidélité de l'outil. Les paramètres d'évaluation d'une chaîne sont calculés en suivant des règles précises.

Les critères de choix d'une chaîne donnée peuvent varier selon les préférences et les besoins de l'utilisateur. Par exemple un utilisateur préfère un résultat en un temps minimal même si cette génération va coûter plus chère en terme de coût d'exécution, mais un autre utilisateur peut exiger une grande fiabilité d'exécution sans s'intéresser au coût et au temps d'exécution. Dans le but de bien représenter les préférences de l'utilisateur, un degré d'importance (pondération) peut être attribué pour chaque critère.

Nous définissons au début les paramètres d'évaluation d'un outil. Ensuite, ces paramètres peuvent être modifiés par l'administration de la plateforme d'une façon qui prend en compte les comportements de l'outil durant ses

exécution passées. Et puis, nous décrivons les règles capables de calculer ses paramètres pour une chaîne d'outils d'indexation.

## 2. Définitions des paramètres

Les paramètres d'évaluations seront considérés comme des critères qui caractérisent chaque outil. C'est au fournisseur de l'outil de préciser ses valeurs après un ensemble de tests.

### 2.1 Le temps

C'est la mesure universelle qui vise à mesurer la performance d'un système de Workflow (travail en séquence), il peut être défini comme le temps total séparant la demande d'un service et le début de l'apparition de résultats. Pour rendre les résultats plus précis on peut le décomposer en deux facteurs le *delay time* et le *Process time*.

Le premier est dit le temps retardé (DT), c'est le délai nécessaire pour qu'une tâche soit traitée, autrement dit c'est le temps supplémentaire nécessaire pour qu'une demande soit traitée par un outil.

Le temps d'exécution (PT) est le temps brut essentiel pour traiter une tâche demandée.

Donc, le temps de réponse d'une tâche  $t$  peut être calculé comme suit :

$$T(t) = DT(t) + PT(t)$$

Ce facteur est initialisé avant l'ajout du service sur la plateforme, c'est-à-dire il est fixé par le fournisseur lors de la création. Durant le travail sur la plateforme, cette valeur peut être modifier par l'administration en le remplaçant par une valeur estimée et calculée en se basant sur la comportement de ce service durant ses anciennes exécutions.

### 2.2 Le coût

C'est le coût (C) demandé par le fournisseur d'un service moyennant l'exécution de ce service par un client. Ce facteur est essentiel pour les clients car c'est très important d'estimer le coût d'une tâche, avant qu'elle s'exécute, afin de garantir les financements. Le coût d'un service comporte plusieurs composants, parmi eux : le *enactment cost* et le *realization cost*. Le premier est le coût associé à la direction et aux contrôleurs de la plateforme. Le deuxième est le coût de réalisation (RC) qui est le coût associé à l'exécution du

service comme le coût d'utilisation d'équipement, le coût d'engagement humain, et autres. La plateforme proposée dans [Haidar05] peut être évoluée pour avoir des services payants, en y intégrant par exemple des services de surveillance payants (reconnaissance des personnes, surveillance des bébés, surveillance basée sur la reconnaissance des gestes).

La formule exportant le coût d'une tâche  $t$  sera défini comme suit :

$$C(t) = CE(t) + RC(t)$$

### 2.3 La fiabilité

Chaque tâche a un état initial, un état d'exécution, et deux états terminant distincts. Un des états indiquent qu'une tâche a échouée ou a été avortée, pendant que l'autre indique qu'une tâche est faite ou est commise. Le modèle utilisé pour représenter chaque tâche indique qu'il existe un seul point de départ, mais il y en a deux états différents qui peuvent être atteints à la fin d'exécution. D'où la dimension de fiabilité apparaît. Cette dimension fournit l'information à propos de la relation entre le nombre de fois où l'état fait/dévoqué est atteint et le nombre de fois où l'état raté/avorté est atteint après l'exécution d'une tâche.

La Fiabilité (F) correspond à la probabilité que le service soit exécuté quand l'utilisateur en demande, et donc elle est calculée en fonction de taux d'échec. La valeur de la fiabilité est calculée par la formule suivante :

$$R(t) = 1 - \text{le taux d'échec.}$$

### 2.4 La fidélité

C'est une mesure de qualité, elle se réfère à la caractéristique d'un bon produit ou d'un bon service. La fidélité est souvent difficile à définir et à mesurer car c'est subjectif aux jugements. Elle est souvent prédite, quand possible. Ils n'existent pas souvent un seul paramètre pouvant décrire la fidélité d'un outil, Les outils ont un vecteur de fidélité composé par une série d'attributs de fidélité ( $F(t).a_i$ ), chaque attribut se réfère à une propriété ou à une caractéristique du service étant créé, transformé, ou analysé. Dans le cas de l'indexation multimédia, les paramètres Rappel/précision sont utilisés pour évaluer certains outils d'indexation, mais on ne peut pas considérer qu'ils sont les seuls paramètres utilisés pour évaluer un outil.

Mais parfois c'est nécessaire d'utiliser la fidélité comme une valeur unique, et pas comme un vecteur de valeurs, ce qui exige la définition d'une fonction qui normalise la fidélité. Cette fonction sera défini selon le besoin de

l'administration de la plateforme et selon le type des services localisés sur cette plateforme.

### 3. La mise à jour des paramètres

Lors de la création d'un outil, le fournisseur précise la valeur de chaque paramètre d'évaluation selon les données utilisées par cet outil, mais ces valeurs ne sont pas toujours exacts, par exemple, le coût d'exécution est précis et sûr, or la fiabilité et la fidélité sont variables car elles sont obtenues d'une action d'estimation.

Donc, il faut que ces paramètres soient mis à jour à chaque intervalle de temps. Trois facteurs influent sur la nouvelle valeur selon leurs poids, qui sont :

- *Le fournisseur (FR)* : le premier facteur c'est la valeur  $FR$  du paramètre  $p$  calculée en se basant seulement sur les données introduites par le fournisseur de l'outil  $o$ .
- *Comportements anciens (CA)* : c'est la valeur du paramètre  $p$  calculée en se référant aux anciens comportements de l'outil  $o$  sans prendre en compte le graphe dans lequel a été exécuté. Pour atteindre cet objectif, nous sauvegardons la valeur ancienne de chaque paramètre d'un outil  $o$ .
- *Comportements anciens dans un graphe précis (CAG)* : ce facteur prend en considération que l'outil peut comporter différemment dans deux graphes différents. Nous sauvegardons la valeur ancienne de chaque paramètre d'un outil  $o$  dans un graphe précis.

Comme on a dit, chaque facteur a un poids qui signifie son degré d'importance. La formule générale capable de re-calculer la valeur d'un paramètre  $p$  c'est :

$$Valeur_p = w_f \times V_f + w_{CA} \times V_{CA} + w_{CAG} \times V_{CAG}$$

Les  $w$  sont les poids de chaque facteur utilisé à condition que leur sommation égale à 1.

#### 4. Méthode de calcul des paramètres d'une chaîne

Dans la partie 2 nous avons défini les paramètres permettant l'évaluation de chaque outil à part. Cependant, nous n'avons pas besoin de ces paramètres indépendamment des chaînes mais on va chercher à les utiliser pour calculer les paramètres d'évaluation de chaque chaîne d'outils d'indexation et par suite les classer.

Maintenant, nous proposons les règles capables de calculer les paramètres d'évaluation d'une chaîne toute entière. Mais au début on va expliquer c'est quoi un modèle de réduction d'une chaîne ? À quoi il sert ? Et comment il nous aide pour calculer les critères d'évaluation d'une chaîne donnée.

##### 4.1 Modèle de réduction

Le modèle de réduction d'une chaîne est un modèle proposé par [Cardoso02], ce modèle vise à réduire une chaîne donnée en un seul nœud ayant les critères de la chaîne toute entière. Ces critères peuvent être le coût de cette chaîne, le temps nécessaire pour exécuter la chaîne, et autres. Pour réduire une chaîne, le modèle définit un ensemble de règles permettant de réduire la chaîne pas à pas (voir la figure 1). Le choix de la règle à appliquer dépend de la situation de l'outil à réduire par rapport à ces voisins de la forme suivante : si les nœuds sont en série on appliquera une formule prédéfinie et s'ils sont en parallèle on appliquera une autre formule.

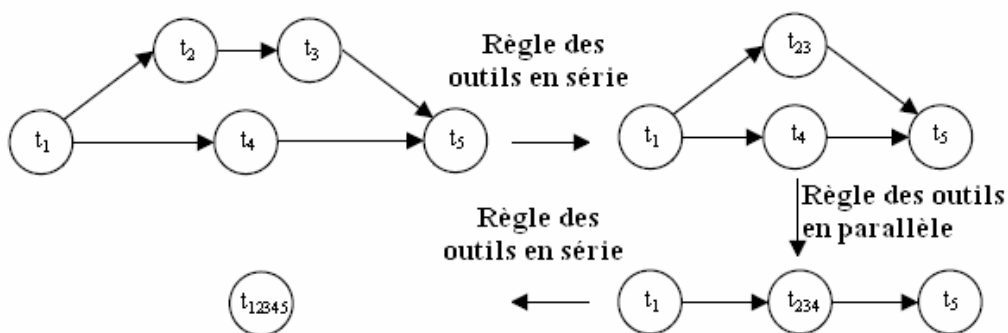


Figure 1 : réduction d'une chaîne

Ce modèle nous déclare un principe de travail pour calculer les paramètres d'évaluation correspondant à une chaîne donnée, c'est-à-dire qu'à la fin de la réduction de la chaîne nous aurons un nœud ayant les valeurs de paramètres correspondant à toute la chaîne, c'est comme si on a trouvé un seul outil, qui

génère l'index demandé directement à partir d'une entrée de type document multimédia primitifs.

Dans ce qui suit, nous dressons, pour chaque paramètre d'évaluation, les règles visant à calculer son valeur dans les deux cas : outils en série et outils en parallèle.

## 4.2 Le calcul des paramètres

### 4.2.1 Le temps

- Outils en série

Deux outils en série ne s'exécutent jamais simultanément, c'est-à-dire que le deuxième ne commence qu'après la fin du premier, donc le temps correspondant à l'outil résumant les deux, est égal à la sommation de leurs temps d'exécution.

$$T_{O1, O2} = T_{O1} + T_{O2}$$

- Outils en parallèle

Contrairement au premier cas, ici deux outils en parallèle s'exécutent simultanément, donc le temps nécessaire pour terminer les deux est égal au temps consommé par l'outil le plus lent, c'est-à-dire égal au maximum du temps pris par chacun.

$$T_{O1, O2} = \text{Max} \{T_{O1}, T_{O2}\}$$

### 4.2.2 Le coût

L'exécution d'une chaîne consiste à exécuter tous les outils constituant cette chaîne. Chaque fois qu'un outil est exécuté, on augmente le coût d'exécution de la chaîne d'une valeur égale au coût d'exécution de cet outil, or si cet outil est exécuté deux fois il faudra donc ajouter son coût deux fois sur le coût total. Donc, Une fois on connaît les outils formant une chaîne et le nombre de fois où cet outil est utilisé on pourra simplement calculer son coût d'exécution en utilisant la formule suivante :

$$C_{chaîne} = \sum_i (\text{Nombre d'utilisation d'outil } i) \times C(O_i)$$

Cette formule réduit beaucoup le travail représenté par le parcours difficile de la chaîne en distinguant le cas où on a des outils en série et l'autre où on a des outils en parallèle.

#### 4.2.3 La fiabilité

La fiabilité, comme nous l'avons déjà mentionné, correspond à la probabilité que l'outil soit exécuté quand l'utilisateur en demande. Une fois la demande d'un outil, parmi les outils formant une chaîne, est avortée, toute la chaîne sera avortée. En effet, les entrées d'un outil d'indexation sont indispensables pour que l'outil fonctionne, or dans le cas où une de ces entrées est manquée car l'outil qui génère ce type d'entrée est avorté, l'exécution de cet outil n'aura pas lieu et par suite tous les outils qui utilisent ses sorties ne seront plus exécutés, ce qui exige l'élimination de cette chaîne de la collection des chaînes solutions car elle est incomplète et inutilisable.

En se basant sur ce qui précède, on constate qu'il faille multiplier les fiabilités des outils pour obtenir la fiabilité de la chaîne qu'ils forment. De plus nous devons utiliser l'idée annoncée durant le calcul du coût, disant qu'il faille considérer la répétition d'un outil comme étant un autre outil indépendant. D'où la formule visant à calculer la fiabilité d'une chaîne :

$$F_{chaîne} = \prod_i (\text{Nombre d'utilisation d'outil } i) \times F(O_i)$$

#### 4.2.4 La fidélité

La fidélité est la probabilité d'avoir des bonnes réponses parmi toutes les réponses fournies par un service. La condition de fonctionnement de l'algorithme de production de chaîne exige d'une part une condition de convergence à la tête de la chaîne (un seul outil produit l'index demandé), et d'autre part il n'existe pas des entrées facultatives pour un outil intégré dans une chaîne. Ainsi, Pour avoir une réponse correcte dans une chaîne en série ou en parallèle, il faut que tous les outils composant la chaîne fournis des résultats corrects. D'où la fidélité totale est multiplicatif dans les deux cas (série et parallèle). Figure 2.

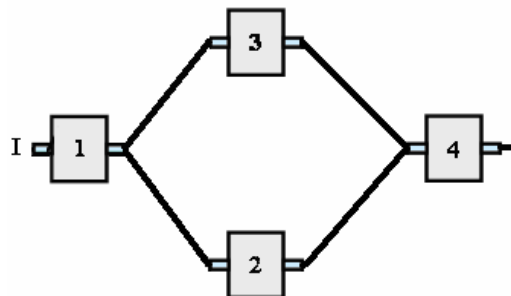


Figure 2 : deux outils en parallèle

Donc la formule est :

$$Fd_{chaîne} = \Pi_i (\text{Nombre d'utilisation d'outil } i) \times Fd(O_i)$$

### 4.3 Le Calcul du score

Les quatre facteurs cités ci-dessus constituent un vecteur capable d'évaluer un ensemble des chaînes, or il est difficile d'évaluer des chaînes avec quatre paramètres, d'où vient l'idée d'unir ces facteurs en une seule valeur prenant en compte les préférences de l'utilisateur.

L'idée fréquemment utilisée c'est de demander à l'utilisateur de préciser pour chaque paramètre un nombre entre 0 et 1 (pondération) de façon qu'il reflète le degré d'importance de ce paramètre, à condition que la sommation de ces nombres soit égale à 1. Donc, chaque utilisateur doit avoir une liste de préférences, par exemple, un utilisateur qui préfère évaluer les chaînes premièrement selon leurs coûts, deuxièmement selon leurs fidélités, et finalement selon leurs temps et leurs fiabilités, peut proposer les coefficients suivants :

$$\text{Coefficient du temps} = C\ae f_{Temps} = 0.1$$

$$\text{Coefficient du coût} = C\ae f_{Coût} = 0.6$$

$$\text{Coefficient de la fiabilité} = C\ae f_{Fiabilité} = 0.1$$

$$\text{Coefficient de la fidélité} = C\ae f_{Fidélité} = 0.2$$

Le score d'une chaîne sera donc déterminé suivant ces coefficients et suivant les valeurs de paramètres de chaque chaîne de la façon suivante :

$$\text{Score chaîne} = C\ae f_{Temps} \times T_{chaîne} + C\ae f_{Coût} \times C_{chaîne} + C\ae f_{Fiabilité} \times F_{chaîne} + C\ae f_{Fidélité} \times Fd_{chaîne}$$

Mais, un problème apparu lors de l'utilisation de ce score. Que signifie un grand score ? Une bonne chaîne ou bien une chaîne inacceptable. Les règles déjà déclarées pour calculer le temps et le coût d'une chaîne correspondent des grandes valeurs à une mauvaise chaîne, or les règles calculant la fiabilité et la fidélité correspondent des grandes valeurs à une bonne chaîne, d'où le problème. Dans le but de donner une seule signification pour tous les paramètres, nous proposons d'utiliser à la place des valeurs réelles de la fiabilité et de la fidélité leurs compléments à 1, ce changement donne pour

tous les paramètres la même signification (grandes valeurs impliquent une mauvaise chaîne). Donc la formule du score sera :

$$\text{Score}_{chaîne} = \text{Cœf}_{Temps} \times T_{chaîne} + \text{Cœf}_{Coût} \times C_{chaîne} + \text{Cœf}_{Fiabilité} \times (1 - F_{chaîne}) + \text{Cœf}_{Fidélité} \times (1 - Fd_{chaîne})$$

Ce score est une valeur illimitée car le temps et le coût d'exécution ont cette propriété. Cela crée un problème de normalisation car nous ne pouvons pas utiliser des grandes valeurs (le temps et le coût) avec des très petites valeurs (la fiabilité et la fidélité) dans une même formule, parce que les petites valeurs seront considérées négligeables devant les grandes, ce qui génère des résultats non significatifs. Pour résoudre ce problème, il faut normaliser les valeurs des paramètres.

D'autre part, l'évaluation des chaînes qui génèrent des index différents n'a pas de sens. Donc l'évaluation aura lieu en se basant sur les valeurs maximales de paramètres caractérisant l'ensemble des chaînes produisant un index précis.

Donc, pour chaque paramètre on prend la valeur maximale atteinte par les chaînes. Ensuite, ces valeurs seront utilisées comme des limites, ce qui nous permet de normaliser toutes les valeurs (valeur du paramètre d'une chaîne divisée par la valeur maximale, du même paramètre, atteinte par toutes les chaînes). Par conséquent, notre formule sera cohérente et toutes les valeurs auront le même impact sur le score final. Les détails de calcul du score seront expliqués dans un exemple dans le chapitre 6.

# *Chapitre 4 : Méthodologie de chaînage*

## **1. Introduction**

Après avoir défini dans le chapitre précédent les critères d'évaluation d'un chaîne d'outils, nous discutons dans ce chapitre les méthodologies utilisées pour trouver le ou les chaînes capables de générer un index donné. Nous rappelons le but principal de la plateforme d'indexation, qui consiste à fournir à l'utilisateur la possibilité de demander pour un index donné, le ou les chaînes capables de le générer.

Dans ce chapitre nous présentons les deux principaux problèmes qu'on cherche à résoudre et nous discutons les approches proposées en dressant les avantages et les inconvénients de chacune. Le premier problème est la représentation des règles de compatibilité entre les différents outils d'indexation présentés sur la plateforme. Le deuxième problème est la méthodologie d'extraction des chaînes « solutions » en tenant compte le problème d'évaluation discuté dans le chapitre précédent

## **2. Représentation des règles de compatibilité**

Deux outils d'indexation sont considérés comme compatibles si un des types de sortie de l'un fait partie des types d'entrées de l'autre. La construction automatique d'une chaîne capable de générer un index donné, passe au début par l'établissement des règles de compatibilité entre les outils. L'ensemble d'outils ainsi que les relations de compatibilité entre ces outils, constituent une forme de graphe d'outils.

Deux approches principales peuvent être utilisées pour représenter un graphe : la première consiste à créer, lors de la construction de la plateforme, un graphe qui représente toutes les relations de compatibilité possibles entre les outils disponibles sur la plateforme. La deuxième consiste à construire le graphe pour chaque index demandé par l'utilisateur. Ce graphe représente les

relations de compatibilité nécessaires à produire l'index demandé, seuls les outils qui contribuent à ces relations sont présentes dans le graphe.

### 2.1 Première approche : Un seul graphe pour tous les index (figure 1)

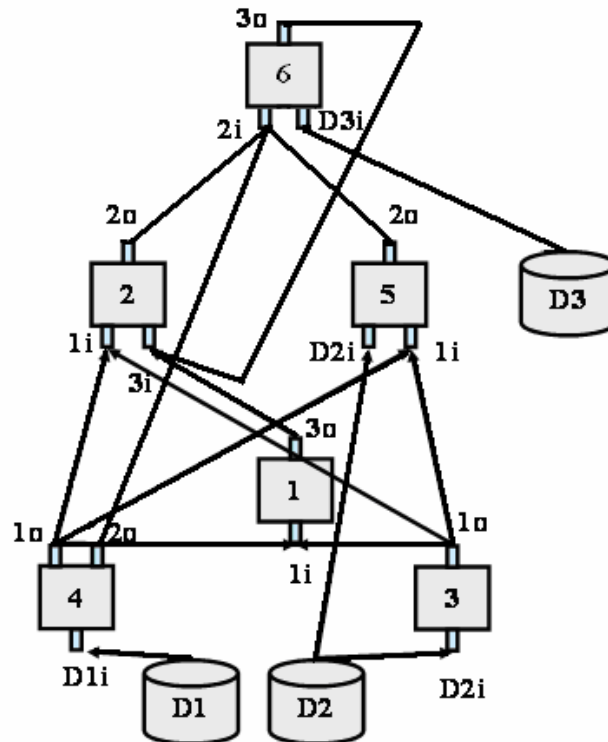


Figure 1 : un grand graphe initial

Lors de l'initialisation de la plateforme les relations de compatibilité seront établies entre tous les outils de la plateforme pour constituer un seul graphe. Cette méthode de construction possède un avantage important représenté par l'économise du temps durant l'extraction des chaînes solutions. En effet, le temps nécessaire à l'extraction des chaînes sera réduit à un temps de parcours du graphe, puisque la phase de la construction du graphe a été faite à la construction de la plateforme. Le problème d'extraction est réduit à un problème de parcours simple du graphe, et par conséquent, un simple algorithme de parcours de graphe peut être utilisé.

Malgré l'avantage cité ci-dessus, cette approche possède un ensemble d'inconvénients. Le premier inconvénient est la complexité de la mise à jour du graphe suite à l'ajout, à la modification et à la suppression d'un outil de la base d'outils présents sur la plateforme, surtout avec l'évolution de nombre des outils disponibles sur la plateforme. Par exemple, quand on décide d'ajouter un nouvel outil d'indexation à la plateforme, il faut modifier tout le

graphe pour qu'il prenne en considération ce changement, en mettant à jour toutes les nouvelles relations de compatibilité avec l'outil concerné. De plus il faut s'assurer que l'ajout de cet outil ne crée pas un cycle infini.

Un autre inconvénient essentiel est que la construction initiale d'un graphe devient très coûteuse en terme de temps de traitement, et de ressources mémoire, notamment si la plateforme dispose d'un grand nombre d'outil.

## 2.2 Deuxième approche : un graphe par index (figure 2)

Pour générer un index donné un ensemble précis d'outils d'indexation peut être contribué à cette génération. Notre idée est de construire un graphe qui contient seulement les outils nécessaires à la génération de cet index. Ce graphe sera constitué d'un ou de plusieurs chaînes dont chacune peut produire l'index demandé. (Figure 2). De même cette approche a des avantages et des inconvénients.

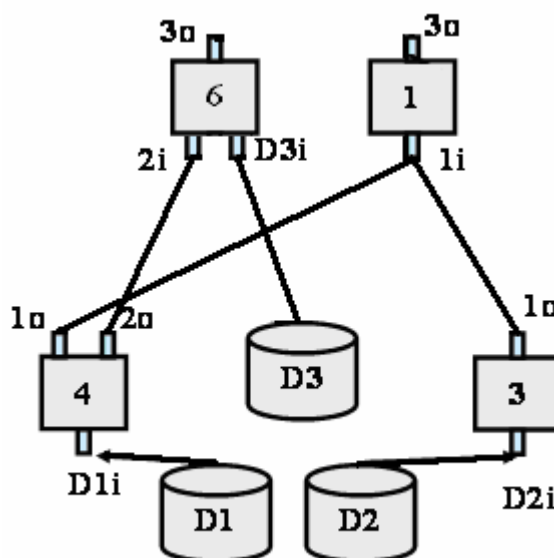


Figure 2 : un graphe contenant les chaînes capables de générer l'index 3

Cette approche est dynamique et simple, elle construit le graphe en se basant sur l'état actuel de la base d'outils. Cela permet de prendre en considération toutes les mises à jours antérieures. D'autre part comme ce graphe est construit seulement par les outils pouvant contribuer à la génération d'un index donné, sa construction est alors plus facile et plus vite en comparaison avec la première approche.

Si on sauvegarde les graphes construits par cette approche, on aura un ensemble de graphes capable de générer plusieurs index dont chacun peut former un sous graphe dans un graphe visant à générer un nouvel index, ce qui résume le temps nécessaire pour construire le nouvel graphe.

Cette approche présente aussi quelques inconvénients, notamment la nécessité de la construction du graphe à chaque demande d'index. Donc l'algorithme d'extraction des chaînes doit être précédé par une phase de construction du graphe. Ce qui ajoute un temps d'attente supplémentaire pour l'utilisateur.

La plateforme d'indexation est une plateforme ouverte, et par conséquent, la mise à jour des outils d'indexation sur la plateforme sera fréquente. En tenant compte de cette exigence ainsi que des avantages et des inconvénients de chaque approche, nous avons choisi la deuxième approche qui nous semble la plus efficace pour notre application.

### **3. Extraction des chaînes « solutions »**

La partie la plus difficile dans notre implémentation était la partie d'extraction des solutions et des chaînes. Nous présentons dans cette partie deux points de vue concernant l'extraction des chaînes. Le premier consiste à retirer seulement la meilleure chaîne solution et le deuxième consiste à extraire et classifier toutes les chaînes possibles selon les préférences de l'utilisateur.

#### **3.1 Extraction de la meilleure chaîne**

Cette approche cherche à retirer seulement la meilleure chaîne solution en se référant sur les préférences de l'utilisateur. L'algorithme capable d'appliquer cette approche est basé sur l'idée suivante : en partant de l'outil ayant le plus grand score, parmi ceux qui génèrent dans leurs sorties l'index cible, nous choisissons pour chaque entrée de cet outil l'outil compatible ayant le score le plus élevé, et puis on applique la même règle d'une façon récursive sur tous les outils choisis. Par exemple dans la figure 3, l'algorithme trouve que la chaîne 1-> [3-> [4-> [D2]]] est la meilleure, car en partant de l'outil 1 capable de générer l'index I, nous choisissons l'outil 3 ayant le score le plus élevé, ensuite nous choisissons l'outil 4, ayant un score supérieur à celui de l'outil 5.

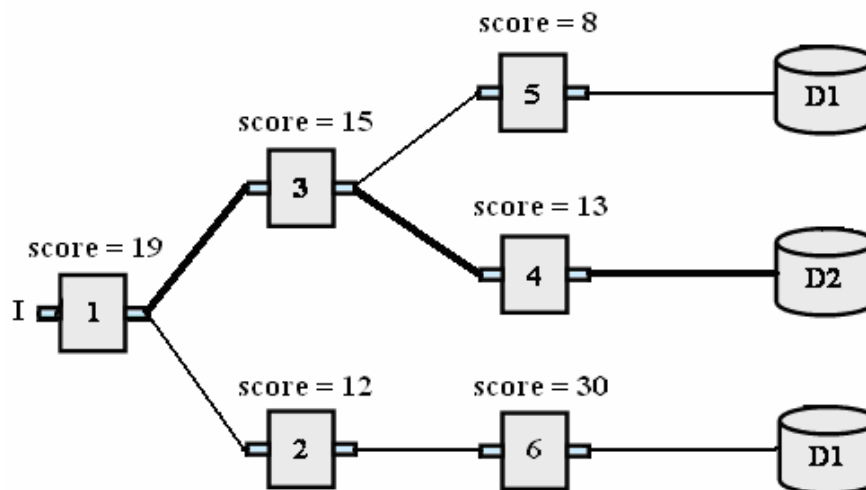


Figure 3 : extraction de la meilleure chaîne

Un des avantages est la réduction du temps de calcul et de l'espace mémoire. Comme cet algorithme ne parcourt pas le graphe tout entier, le temps nécessaire pour trouver la solution est très court en le comparant avec le temps en besoin dans l'algorithme utilisant la deuxième approche présentée dans le paragraphe suivant. L'espace mémoire demandé est réduit à l'espace essentiel pour sauvegarder une, et une seule chaîne.

Cet algorithme se base sur le principe de prédiction et ne permet pas de produire la meilleure chaîne d'une manière sûre. En effet, cet algorithme ne teste pas tout le graphe, mais seulement les successeurs directs d'un nœud. Par exemple, dans la figure 3, quand l'algorithme arrive sur le nœud 1 il ne voit et teste que les nœuds 2 et 3 et selon leurs scores il décide qui va être pris. Cette méthode permet de réduire au maximum le temps de traitement de l'algorithme.

Cet algorithme calcule le score de chaque outil, au départ du système, en fonction des coefficients de préférences de l'utilisateur et en fonction des paramètres d'évaluation. Une fois les scores sont prêts, l'algorithme peut chercher la meilleure chaîne, or si on change un des paramètres d'évaluation d'un outil appartenant au graphe ou un des préférences de l'utilisateur, il faudra alors recalculer le score de chaque outil avant de re-extraire la nouvelle meilleure chaîne, ce qui peut être considéré comme un point faible dans cet algorithme. De plus il ne faut pas oublier la complexité du travail notamment dans les cas où on a des outils tombés en pannes ou inaccessibles.

L'inconvénient le plus essentiel est la faible sûreté de l'algorithme, c'est-à-dire que cet algorithme estime la chaîne solution sans donner la meilleure chaîne exacte.

### 3.2 Extraction de toutes les chaînes

Le premier algorithme possède un grand inconvénient représenté par la faible sûreté, pour résoudre ce problème nous avons cherché à trouver une méthode d'extraction de résultats qui produit des résultats plus sûre. Pour avoir une solution exacte en tenant compte des paramètres d'évaluation, il faut parcourir tout le graphe afin de tester tous les cas possibles avant de choisir la meilleure chaîne. D'où vient l'idée de chercher toutes les chaînes capables de générer notre index cible en calculant, pour chacune, les quatre valeurs correspondantes à chaque paramètre d'évaluation (coût, fiabilité, fidélité, et le temps d'exécution), ce qui nous permettra de calculer simplement le score de la chaîne toute entière en fonction des préférences de l'utilisateur. Nous aurons à la fin, un ensemble de chaînes triées selon la valeur du score. Cet algorithme possède un avantage important qui est la simplicité de la mise à jour des scores si un paramètre d'évaluation d'un outil change ou si les préférences de l'utilisateur changent, dans ce cas il suffit de recalculer ce paramètre seulement pour les chaînes contenant cet outil, ce qui réduit beaucoup le temps du travail. De plus cet algorithme donne le choix à l'utilisateur à choisir la meilleure chaîne par rapport à lui sans l'obliger par celle choisie par l'algorithme comme le cas de l'approche précédente. Il peut aussi changer ces préférences et recalculer les scores facilement.

Cet algorithme permet de gérer le cas où la meilleure chaîne sera non utilisable à cause de la panne d'un de ces outils. L'utilisateur choisira simplement la seconde chaîne dans le tableau des chaînes triées comme étant la meilleure.

L'avantage le plus important est la sûreté des solutions. Cet algorithme génère toujours, dans les cas ordinaires, des résultats corrects en calculant toutes les solutions possibles.

Dans l'autre côté, l'inconvénient de cette approche est la consommation de beaucoup de temps et d'un grand espace mémoire pour produire les résultats.

### 3.3 Problème de sûreté

La sûreté d'un algorithme est un facteur très intéressant chez les utilisateurs. On verra dans l'exemple suivant l'effet de ce facteur dans le cas des algorithmes présentés ci-dessus.

Dans cet exemple nous supposons qu'un grand score correspond à un bon outil. La figure 4 montre la démarche suivie par le premier algorithme afin d'estimer la meilleure chaîne capable de générer l'index 3.

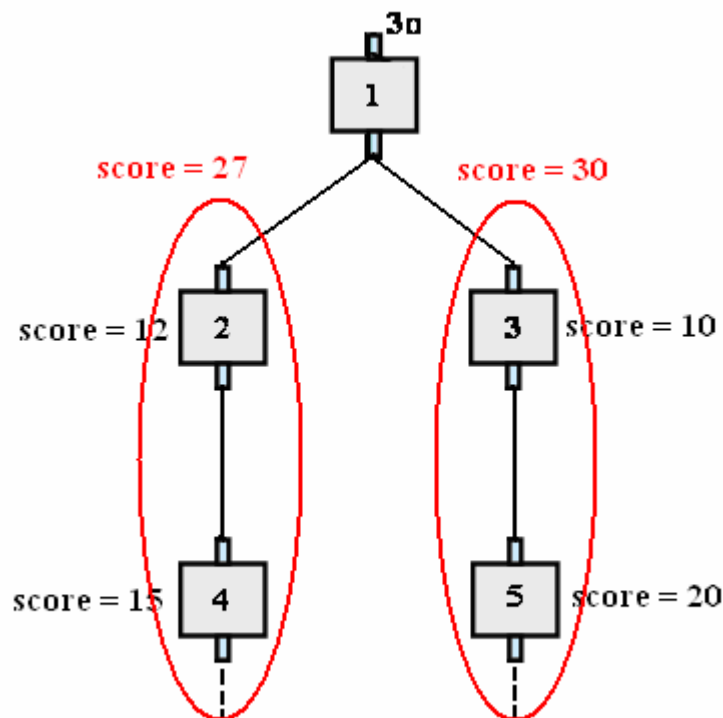


Figure 4 : estimer la meilleure solution en suivant l'approche 1

Mais attention, un outil ayant un grand score n'implique pas toujours que la chaîne dont il est le sommet est la meilleure chaîne, par exemple dans la figure 4, en partant de l'outil 1 nous cherchons le successeur ayant le plus grand score, c'est l'outil 2. De même en partant maintenant de l'outil 2 nous aurons l'outil 4. Finalement la chaîne 1->2->4 sera choisie comme étant la meilleure. Or, ce n'est pas le cas car la chaîne 1->3->5 a un score total plus grand que celui de la chaîne choisie par cet algorithme ( $30 > 27$ ), d'où l'erreur.

Le problème est que si on ne construit pas un graphe complet, on n'aura pas à priori des informations sur le chemin qu'on va suivre. Ce problème peut être résolu si on effectue plusieurs parcours en avant et en arrière pour tester tous les chemins, mais ceci nous ramène à un algorithme proche de celui proposé dans la deuxième approche.

La sûreté de l'évaluation des chaînes est un paramètre important pour choisir la meilleure chaîne, de plus un des principaux buts de notre étude est

de permettre à l'utilisateur de choisir une chaîne parmi plusieurs chaînes possibles. Ceci nous conduit à choisir d'implémenter le deuxième algorithme qui répond à ces critères.

# Chapitre 5 : Implémentation

## 1. Introduction

Dans ce chapitre nous présentons et discutons l'implémentation des approches discutées et choisies dans le chapitre précédent. Nous avons proposé de définir un graphe pour chaque index demandé, ce graphe contient toutes les solutions « chaînes » capables de générer cet index. D'autre part nous avons choisi d'implémenter un algorithme qui permet d'extraire et de classier toutes les chaînes possibles à partir du graphe.

Dans ce chapitre, nous définissons des classes représentant les données de base dont chacun contient une structure de donnée caractérisant l'objet et des méthodes capables de générer les résultats nécessaires pour atteindre l'objectif du travail (évaluer les chaînes).

Au départ, nous choisissons pour réaliser notre objectif le langage de programmation JAVA, qui est un langage orienté objet, ce qui permet de modéliser sous forme d'objets, l'ensemble des éléments constituant notre problème (outil, graphe, chaîne, type de donné, etc.). Et par conséquent, faciliter le développement de l'application.

Le programme qu'on cherche à implémenter doit avoir un ensemble de caractéristiques qui nous aident à produire les résultats souhaitables. Premièrement, il faut proposer une bonne représentation de la plateforme d'indexation, ce qui facilite et simplifie l'extraction des chaînes pouvant produire un index donné, pour cela nous avons représenté une plateforme comme étant un graphe dont les nœuds représentent les outils de cette plateforme et les arcs représentent les relations de compatibilité entre ces outils.

Deuxièmement, l'objet outil doit hériter toutes les caractéristiques d'un outil d'indexation réel, d'où vient l'idée de correspondre pour chaque caractéristique d'un outil un attribut dans la classe *Outil* qui prend la valeur de ce caractéristique.

Troisièmement, seulement une bonne représentation d'une chaîne « solution » permet une évaluation exacte et parfaite. Cette représentation doit nous permet de parcourir facilement la chaîne et d'appliquer les formules capables de calculer les paramètres d'évaluation facilement. Pour cela, nous avons choisi de représenter une chaîne comme étant un arbre n-aire.

Nous citons maintenant les classes définies dans notre programme en expliquant, en bref, le rôle de chacune.

- La classe *Type* est une classe qui représente un des types de métadonnée.
- La classe *Outil* vise à représenter un outil d'indexation avec toutes ses caractéristiques.
- La classe *Type Outil* est utilisée pour traduire une relation de compatibilité entre deux outils sur un type précis.
- La classe *Grappe* est un type de données visant à représenter le graphe qui contient toutes les chaînes capables de générer un index donné. Cette classe comporte les méthodes capables d'extraire les solutions du graphe.
- Une chaîne capable de produire un index donné est représentée par la classe *Arbre N-aire*.

Dans les parties suivantes nous détaillerons les attributs et les méthodes formant chaque classe, puis nous expliquerons le rôle et le but de chaque composant.

Au départ nous commençant par la classe représentant un outil ou un document, puis nous arrivons à la classe *Grappe* qui vise à crée le graphe contenant toutes les chaînes capable de produire un index donné et finalement nous expliquerons les composantes de la classe *Arbre N-aire* qui nous permet de sauvegarder les chaînes qu'on cherche à extraire du graphe.

## 2. Représentation des outils et des documents

La représentation d'un outil d'indexation et d'un document multimédia doit être semblable à celle des outils et des documents réels. Donc il faut encapsuler toutes les caractéristiques réelles d'un outil et d'un document dans

une classe afin d'obtenir la meilleure représentation. Pour cela nous déclarons la classe *Outil* représentant un outil ou un document. En ce qui concerne les documents, notre proposition consiste à considérer qu'un document est un outil mais sans aucune donnée d'entrée et avec des paramètres d'évaluation neutres, ce qui permet une meilleure cohérence dans le parcours du graphe en considérant un document comme un outil.

En général, un outil est spécifié par un numéro et un label qui représente le nom complet ou la description de cet outil. D'autre part, les outils sont des programmes exécutables ce qui exige la présence des paramètres d'entrées et des paramètres de sorties qui seront représentés comme des vecteurs dont chaque élément est un type de métadonnées ou un type de données. Quand l'outil n'a pas des entrées (cas des documents) le vecteur correspondant sera vide.

Par ailleurs, la représentation d'un graphe, comme on va voir, consiste à spécifier pour chaque outil les outils compatibles avec lui. Pour cela, nous définissons dans la classe *Outil* un vecteur appelé *Fils* dont chaque élément est un objet représentant un couple formé d'un outil *o* et d'un type *t*. le type *t* représente l'élément de compatibilité entre l'outil *o* et l'outil initial. C'est-à-dire que l'outil initial possède le type *t* comme type de donnée d'entrée, et l'outil *o* possède le même type en sortie. On a de plus quatre attributs dont chacun représente un paramètre d'évaluation.

À côté de ces attributs, cette classe comporte plusieurs méthodes d'importances variées qu'on vient à expliquer les plus importantes parmi eux. Premièrement, on a la fonction « *GetCopie* » qui retourne une copie d'un outil (copier les données et non seulement la référence), deuxièmement, on a la méthode *Afficher* qui vise à afficher les caractéristiques de l'outil appelant. Finalement, nous avons une fonction importante qui est la fonction *Combiner*, cette fonction forme une étape essentielle dans l'ensemble d'étapes permettant d'extraire les solutions (voir le paragraphe Extraction des chaînes). Son rôle est clarifié dans la Figure 1.

Selon la figure, deux ensemble d'outils peuvent formés l'entrées de l'outil 2 qui sont : (outil4 et outil1) ou (outil3 et outil1), la fonction *Combiner* utilise le vecteur des fils de l'outil 2 pour trouver ces combinaisons ( $4 \times 1$  et  $3 \times 1$ ). La compréhension du rôle de cette fonction aura lieu durant l'explication de l'algorithme capable d'extraire des chaînes solutions.

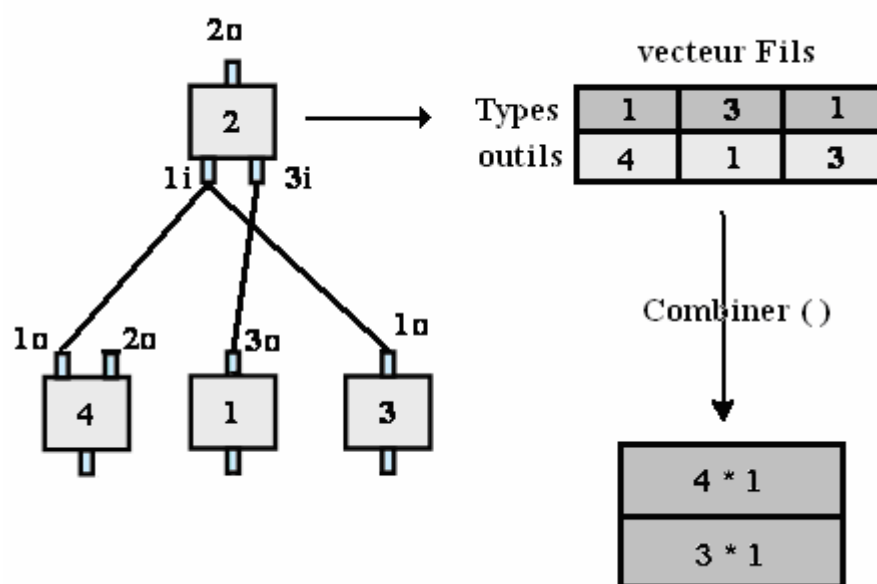


Figure 1 : l'objectif de la fonction Combiner

### 3. Représentation d'un graphe

Un graphe est un ensemble des nœuds connectés entre eux (relations de compatibilité), de manière qu'à partir d'un nœud on peut se déplacer vers le successeur qu'on veut atteindre. Dans notre cas, un nœud appartenant au graphe correspond à un outil localisé sur la plateforme, ce nœud possède toutes les critères caractérisant l'outil d'indexation (le temps d'exécution, la fiabilité, les types de métadonnées d'entrée, ..).

Un graphe peut être vu comme étant un ensemble de chemins liant les documents multimédia d'une part et les outils d'indexation finals générant dans leurs sorties l'index cible (l'index que l'utilisateur souhaite trouver la chaîne qui le génère) d'autre part. Dans la partie précédente, nous avons mentionné que chaque outil peut accéder à ses successeurs, donc pour atteindre tous les nœuds du graphe, il suffit de sauvegarder les outils finals (outils qui génèrent l'index désiré) pour qu'on aura accès à tous les outils du graphe. Pour cela un graphe sera représenté par un vecteur contenant les références de tous les outils finals (ce vecteur est appelé *Les Outils Sommets*). De plus on a trois vecteurs contenant des outils :

- *Les outils* : est un vecteur qui contient les outils utilisés dans ce graphe.

- *Les outils prêts* : est le vecteur comportant tout outil dont on a déjà cherché les outils compatibles avec lui (il est utilisé pour éviter le parcours inutile durant la construction du graphe).
- *Les outils main* : est la collection d'outils inclus dans la base d'outils.

### 3.1 Exemple d'un graphe

Cet exemple montre graphiquement comment nous représentons un graphe dans la mémoire. La figure 2 est un graphe construit pour trouver toutes les chaînes capables de générer l'index 2.

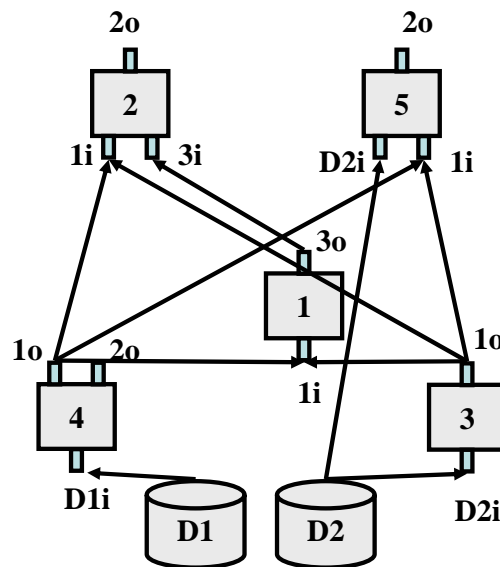


Figure 2 : un graphe incluant toutes les chaînes générant l'index 2

La représentation utilisée pour ce graphe est comme l'indique la figure 3.

LesOutilsSommet

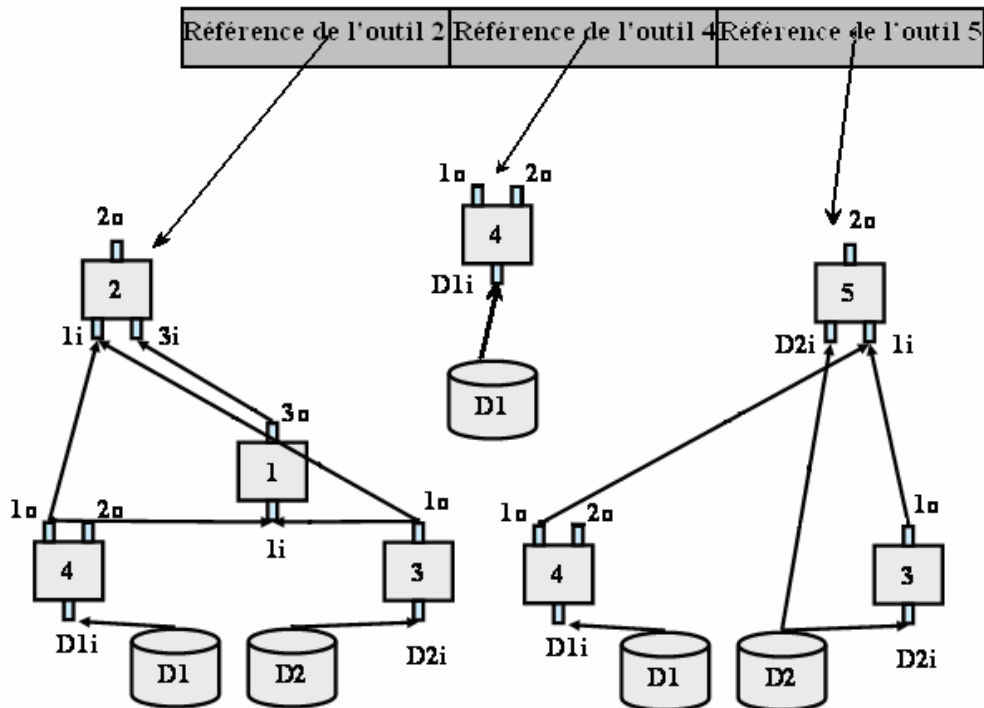


Figure 3 : représentation d'un graphe

#### 4. Représentation d'une chaîne

La dernière représentation est celle d'une chaîne. On peut définir une chaîne comme un graphe ayant un seul point de départ. Cette condition ne suffit pas de point de vue de l'indexation multimédia, car une chaîne qui génère un index donné ne doit jamais partir de plus qu'un document multimédia primitif, sinon l'indexation n'aura pas un sens. Donc, la définition sera : une chaîne est un graphe qui part d'un outil générant l'index désiré et arrive à un seul document multimédia. La figure 4 suivante est une chaîne d'outil capable de générer l'index INDEX.

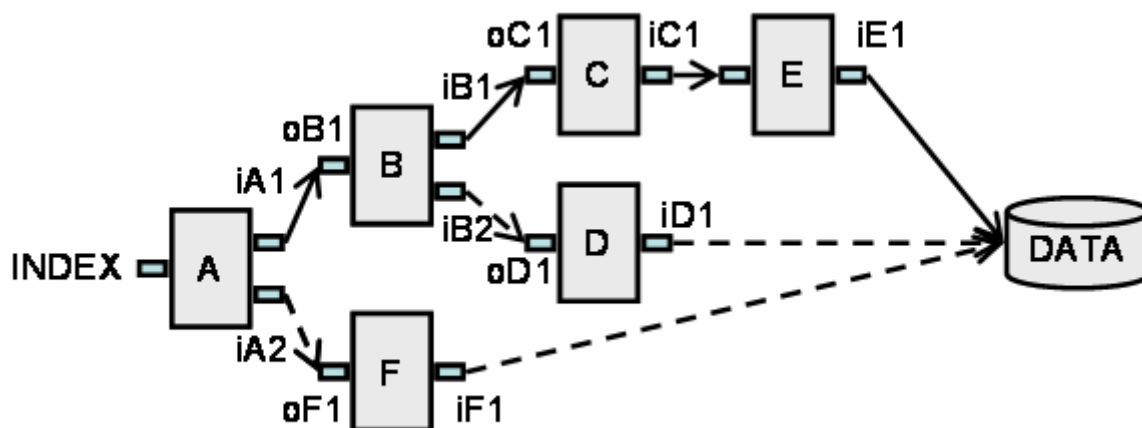


Figure 4 : chaîne d'outils d'indexation produisant l'index INDEX

Cette représentation est très proche de la structure de données d'un arbre n-aire. Un arbre n-aire est défini comme étant un ensemble de nœuds connectés entre eux de façon que chacun n'a qu'un seul père, ce qui favorise une chaîne avec un seul point de départ et un ensemble de points d'arrivé, mais cette dernière propriété ne convient pas avec notre besoin. Pour cela nous avons ajouté une condition de convergence qui exige que les derniers nœuds (feuilles) représentent un seul type de document multimédia primitif.

D'autre part, Chaque nœud dans l'arbre contient : un outil, un vecteur regroupant les nœuds fils, les documents accessibles par ce nœud, le premier nœud dans cet arbre (le sommet). Or, le nœud sommet est un nœud spécial qui a des attributs supplémentaires qui sont :

- les paramètres visant à évaluer la solution : chaque arbre sauvegarde les valeurs de chaque paramètre d'évaluation correspondantes à toute la chaîne.
- l'ensemble d'outils utilisés par cet arbre : c'est la collection d'outils utilisés par cette chaîne, elle nous aidera pour calculer les paramètres d'évaluation.

## 5. Algorithme de construction du graphe

L'algorithme proposé pour construire le graphe est un algorithme récursif qui permet de regrouper les chaînes d'outils capables de générer l'index désiré. Cet algorithme basé sur un parcours en arrière (backward) s'exécute de la façon suivante :

En partant de l'index cible *INDEX*, on cherche à trouver le ou les outils capables de générer cet index dans ses sorties, et pour chaque outil trouvé (*OUTIL*), on applique la démarche suivante :

- Si cet outil a un seul paramètre d'entrée, qui est un document multimédia primitif (non généré par un autre outil) *DATA* alors on s'arrête. Par conséquent, l'index *INDEX* pourra être généré en traitant tous les documents de type *DATA* à partir de l'outil *OUTIL*.
- Si cet outil a un ou plusieurs paramètres d'entrée qui ne sont pas des types de documents multimédia primitifs (*In1, In2,...*), on applique la même démarche en partant chaque fois d'un des types d'entrée (*In1, In2,...*), et en ajoutant chaque fois un nouvel outil à la chaîne construite, jusqu'à trouver un outil qui satisfait la condition précédente.

À la fin on ajoute l'outil *OUTIL* sur le vecteur *Les Outils Sommets*, et puis on essaie de refaire la même chose pour un autre outil capable de générer l'index cible de graphe. Cet algorithme est traduit comme le suivant.

La procédure *Construire* cherche les outils capables de générer l'index cible dans leurs sorties, et puis chaque *Outil* trouvé appelle la fonction *ChercherIn (Outil)*, cette fonction appelle à son tour la fonction *ChercherCompatibles* récursivement. Le rôle de chaque fonction est comme le suivant :

- *ChercherIn (OUTIL)* : cette fonction, et pour chaque entrée *I*, appelle la fonction *ChercherCompatibles (OUTIL, I)*. Elle ne fait aucune action si *OUTIL* est un document.
- *ChercherCompatibles (OUTIL\_PERE, INDEX)* : elle cherche tous les outils compatibles avec *OUTIL\_PERE* sur l'index *INDEX*, et puis elle appelle la première fonction récursivement pour chaque outil trouvé. Finalement, elle ajoute l'outil trouvé (avec son vecteur *Fils* chargé) au vecteur *Fils* de l'outil *OUTIL\_PERE*.

Au bout de cet algorithme, nous aurons dans le vecteur *Les Outils Sommets* les sommets de tous les sous graphes trouvés. Ces sous graphes seront utilisés dans ce qui suit pour extraire les chaînes capables de générer l'index cible.

### 5.1 Problème des cycles infinis

Le graphe construit dans la partie précédente doit être un graphe acyclique direct. Donc, il faut éviter les cycles infinis durant la construction du graphe. Cependant, en étudiant les cas où on peut avoir des cycles infinis, nous constatons qu'ils se produisent quand un outil est le père (prédécesseur) de lui-même comme le cas de l'outil 2 dans la figure 5

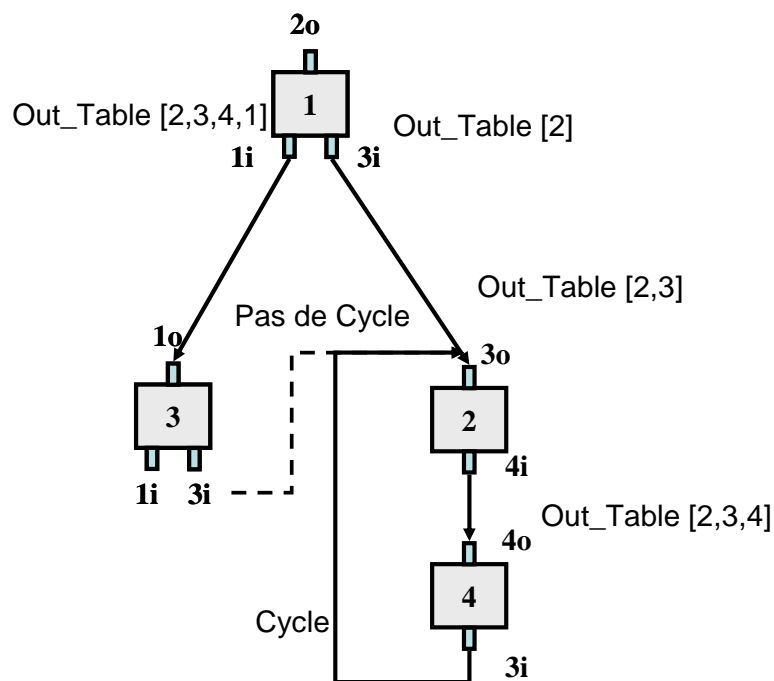


Figure 5 : Cycle infini dans le graphe recherché

Pour éviter ce cas, [Haidar05] a défini une méthode qui consiste à créer un tableau appelé `OUT_TABLE`, et pour chaque outil rencontré on ajoute ses types de sorties sur le tableau et puis on teste cet outil selon ces trois règles :

- Un outil prédécesseur de lui-même mais aucun type d'entrée n'appartient à `OUT_TABLE`, donc rien à faire car dans ce cas il est impossible de trouver un cycle partant de cet outil, comme le cas de l'outil 2 dans la figure 5.
- Un outil qui est un prédécesseur de lui-même mais un des entrées appartient à `OUT_TABLE`, comme le cas de l'outil 4, doit être éliminé de notre parcours car elle produit un

cycle infini. c'est-à-dire ignorer cet outil durant la recherche sur un outil compatible avec l'outil 2 sur l'index 4 et par conséquent, ne pas ajouter cet outil sur le vecteur de *Fils* de l'outil 2.

- Finalement, si un des entrées appartient au tableau, mais cet outil n'est pas un prédécesseur de lui-même comme la cas de l'outil 3, dans ce cas cet outil ne produit pas un cycle infini, et par suite l'outil 3 sera ajouté sur notre chemin.

Cette méthode permet d'éviter les cycles, cependant elle élimine parfois des solutions possibles. La figure 6 est un des exemples où cet algorithme ne prend pas en compte une solution possible (1->2->4->5). Par exemple, après l'ajout d'un outil 5 (voir la figure 6) sur le graphe de la figure 5, nous aurons deux chemins partant de l'outil 4, un vers l'outil 2 (qui forme un cycle infini) et un autre vers l'outil 5 (ne forme pas un cycle infini). Dans ce cas, l'algorithme proposé élimine le chemin 2 - 4 complètement du graphe, dans un temps où il faut supprimer seulement le chemin 4 - 2. D'où, la perte d'une solution possible (la chaîne commençant par l'outil 4 et passant par l'outil 5 est perdue).

Dans le but de résoudre ce problème, nous avons proposé une idée consistant à ne pas éliminer l'outil directement quand on trouve un de ses entrées dans le tableau `OUT_TABLE`, mais on attend jusqu'on trouve un outil ayant un de ses entrées et un de ses sorties dans le tableau, dans ce cas on élimine seulement cet outil (voir la figure 6). Comme ça, nous n'éloignons que l'outil dont on a sûre qu'il produise un cycle infini.

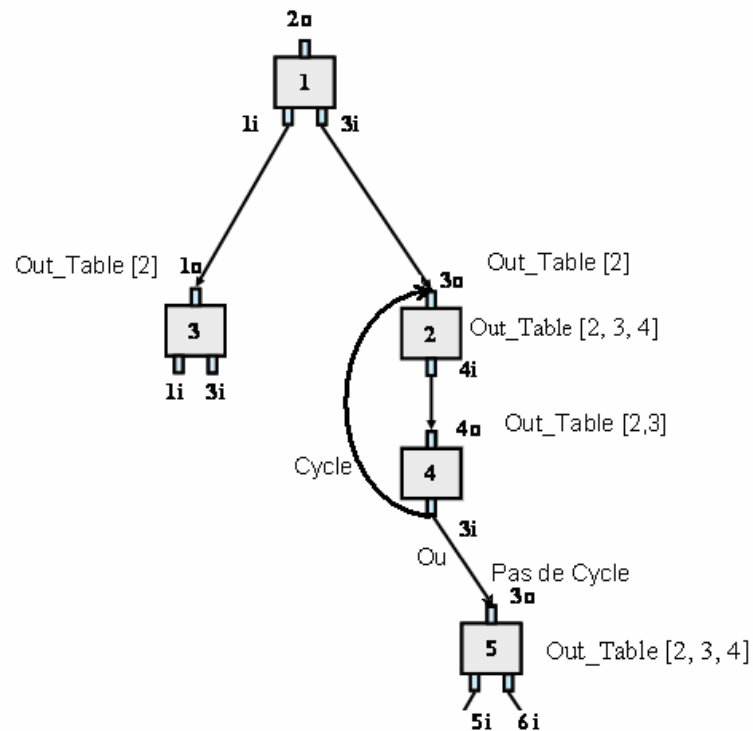


Figure 6 : cas non pris en compte

## 5.2 Réduction de la redondance du calcul

Pour optimiser l'algorithme de parcours et gagner plus de temps, nous avons ajouté un vecteur dans la classe *Graphe* appelé *Les Outils Prêts*, ce vecteur contient les outils qu'on a déjà cherché ses successeurs (Fils). En effet, dans le graphe produit par l'algorithme de construction de graphe, nous pouvons avoir un outil compatible avec deux autres (prédécesseurs) sur un de ses types de sorties (il est le fils de deux outils), c'est-à-dire lors du parcours nous atteindrons cet outil deux fois, une fois en cherchant les outils compatibles avec le premier prédécesseur et une autre en les cherchant pour le deuxième prédécesseur, ce qui implique qu'on va chercher ses fils deux fois. D'où vient l'idée consistant à sauvegarder les outils qu'on a déjà parcouru pour ne pas tomber dans la redondance.

Par exemple, dans la figure 7 l'outil 4 est un outil compatible avec l'outil 2 et avec l'outil 1. D'autre part, une fois qu'on trouve que le document D2 est compatible avec l'outil 4, on n'a plus besoin de rechercher les relations de compatibilité de l'outil 4.

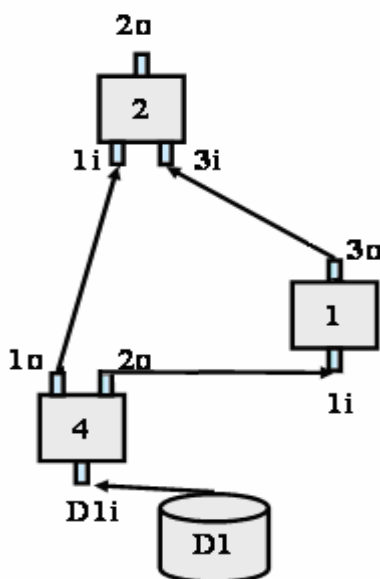


Figure 7 : éviter la redondance

## 6. Algorithme d'extraction des chaînes solutions

Une fois le graphe est construit, le but maintenant est d'extraire les chaînes capables de générer un index donné (chaînes solutions). Par exemple, la figure 7 est une solution pour générer l'index 2, car en partant de cet index nous arrivons à un document multimédia unique qui est D1.

L'idée principale visant à extraire les solutions d'un graphe c'est de chercher et puis séparer toutes les combinaisons d'outils qui peuvent former l'entrée de chaque outil, par exemple dans le graphe de la figure 2 l'outil 2 peut avoir deux entrées possibles :  $O4 \times O1$  ou  $O3 \times O1$ .

Comme le cas de la représentation d'un graphe, nous représentons la solution comme un ensemble (vecteur) contenant les sommets des chaînes capables de générer l'index demandé.

L'idée principale consiste à chercher pour chaque outil toutes les combinaisons possibles d'entrée pour tenir compte de tous les successeurs possibles. Par exemple, si un outil  $o$  possède deux possibilités d'entrée (comme l'outil 2 dans la figure 2), nous ferons une copie de  $o$  pour que chacune prendre une possibilité. La figure 8 clarifie la démarche de l'algorithme.

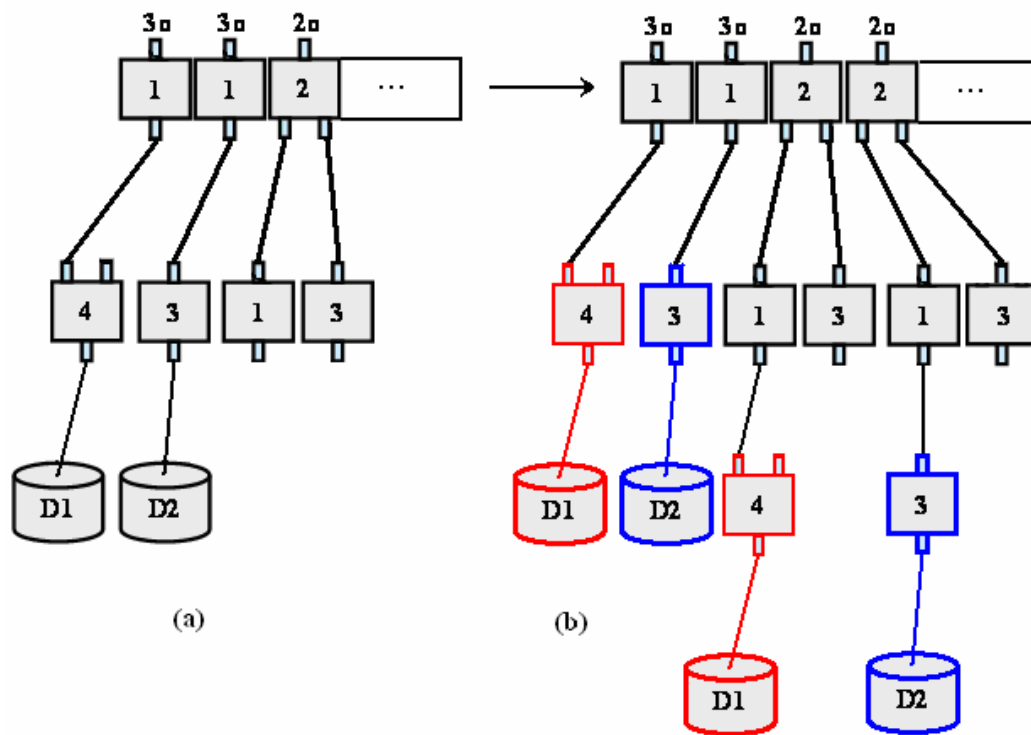


Figure 8 : La démarche de l'algorithme d'extraction des solutions

Dans la figure 8 partie (a), l'outil 1 a deux entrées possibles qui sont : la chaîne 4->D1 et la chaîne 3->D2. Donc pour chaque outil 1 dans n'importe quel arbre A, nous copions A en ajoutant la possibilité 4->D1 comme un fils de l'outil 1 dans le premier copie et la deuxième possibilité (3->D2) comme un fils de l'outil 1 dans la deuxième copie. Nous répétons cette démarche pour chaque outil utilisé dans notre graphe. .

## Chapitre 6 : Résultats

Après avoir proposé et implémenter les algorithmes capables d'extraire et d'évaluer les chaînes d'outils d'indexation dans les chapitres précédents, nous discutons dans ce chapitre les résultats produits par ces algorithmes en définissons à titre d'exemple un ensemble de ressources d'indexation (outils d'indexation, documents multimédia primitifs), et nous essayons d'extraire les résultats à partir de cette base de ressources.

Nous proposons une plateforme constituée de sept outils et de deux documents multimédia, ces ressources utilisent quatre types de métadonnée (numéroté de un à quatre) et deux types de donnée primitive non générée (type de donnée multimédia brute).

Les types d'entrées et les types de sorties de chaque outil sont déclarés dans la figure 1.

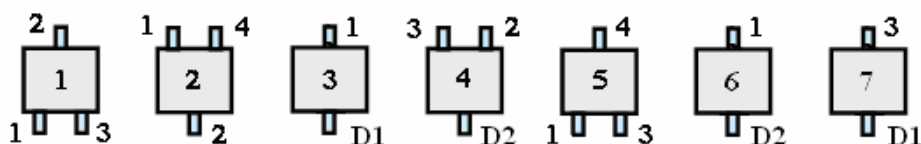


Figure 1 : les outils disponibles

Les paramètres d'évaluation de chaque outil sont affichés dans la figure 2

```
nom : O2
Temps = 1.5, Cout = 27.0, Fiabilité = 0.93, Fidélité = 0.88

nom : O1
Temps = 1.6, Cout = 25.0, Fiabilité = 0.95, Fidélité = 0.9

nom : O3
Temps = 1.0, Cout = 30.0, Fiabilité = 0.89, Fidélité = 0.91

nom : D1
Temps = 0.0, Cout = 0.0, Fiabilité = 1.0, Fidélité = 1.0

nom : O6
Temps = 5.5, Cout = 18.0, Fiabilité = 0.87, Fidélité = 0.92

nom : D2
Temps = 0.0, Cout = 0.0, Fiabilité = 1.0, Fidélité = 1.0

nom : O4
Temps = 6.0, Cout = 28.0, Fiabilité = 0.8, Fidélité = 0.95

nom : O7
Temps = 5.5, Cout = 21.0, Fiabilité = 0.89, Fidélité = 0.94

nom : O5
Temps = 5.6, Cout = 38.0, Fiabilité = 0.94, Fidélité = 0.9
```

Figure 2 : Les paramètres d'évaluation des outils

Dans le but de chercher les chaînes solutions capables de générer l'index 4, notre algorithme consiste à trouver les relations de compatibilité entre les outils (construire le graphe) en partant des outils générant l'index 4 (les outils 2 et 5). Le graphe obtenu est dans la figure 3.

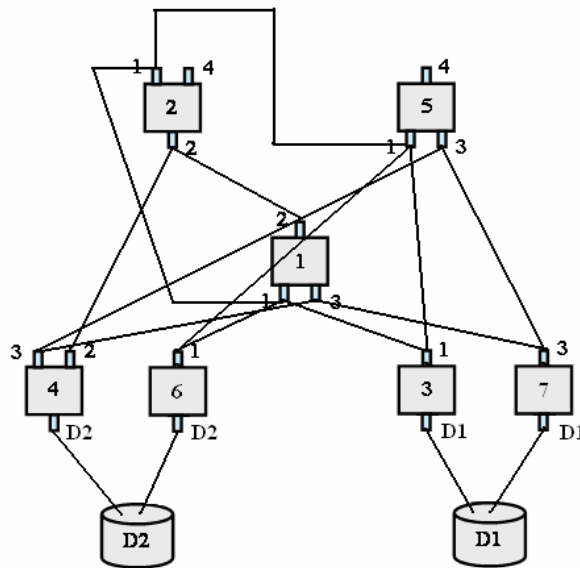


Figure 3 : le graphe obtenu

Une fois le graphe est construit, l'algorithme cherche à extraire les chaînes solutions de ce graphe. Les chaînes trouvées par l'algorithme sont décrites dans la figure 4.

```

Les solutions :
1. O2->[O4->[D2]]
   Score => (64.51924340467895) / 100

2. O5->[O3->[D1],O7->[D1]]
   Score => (48.229701702339476) / 100

3. O2->[O1->[O3->[D1],O7->[D1]]]
   Score => (48.098203372324534) / 100

4. O5->[O6->[D2],O4->[D2]]
   Score => (46.78952573419612) / 100

5. O2->[O1->[O6->[D2],O4->[D2]]]
   Score => (46.74653240418118) / 100

6. O5->[O2->[O4->[D2]],O4->[D2]]
   Score => (32.09067100049775) / 100

7. O5->[O2->[O1->[O3->[D1],O7->[D1]],O7->[D1]]
   Score => (18.949060865514085) / 100

8. O5->[O2->[O1->[O6->[D2],O4->[D2]],O4->[D2]]
   Score => (15.166715039999985) / 100
    
```

Figure 4 : l'ensemble des chaînes trouvées et évaluées

Cependant, les scores sont calculés en se basant sur les préférences de l'utilisateur qui a précisé les coefficients des paramètres d'évaluation comme suivant :

$$\text{Coefficient de temps} = 0.3,$$

$$\text{Coefficient de coût} = 0.4,$$

$$\text{Coefficient de fiabilité} = 0.2,$$

$$\text{Coefficient de fidélité} = 0.1.$$

L'algorithme proposé évite la production d'un cycle infini (le chemin 2→1→2→...) et réduit le temps en évitant la redondance (le calcul des outils successeurs compatibles avec l'outil 2).

Par une simple vérification basée sur les préférences de l'utilisateur, on pourra obtenir les scores affichés dans la figure 4. Par exemple, pour la meilleure chaîne obtenue O2-> [O4-> [D2]]. Le temps d'exécution de cette chaîne égal à 7.5 secondes ( $T_{O2} + T_{O4} + T_{D2}$ ), son coût d'exécution égal à 55 \$ ( $C_{O2} + C_{O4} + C_{D2}$ ), ensuite elle a une fiabilité égale à 0.744 ( $Fia_{O4} \times Fia_{O4} \times Fia_{D2}$ ) et enfin sa fidélité est égale à 0.836 ( $Fid_{O4} \times Fid_{O4} \times Fid_{D2}$ ). En effet le score sera calculé, après la pondération (diviser la valeur de chaque paramètre par la valeur maximale de ce paramètre) des valeurs, comme suivant :

$$\text{Score} = 0.3 \times 0.510 + 0.4 \times 0.335 + 0.2 \times (1-0.744) + 0.1 \times (1-0.836) = 0.355$$

En prenant le complémentaire à 1 (pour correspondre un grand score pour une bonne chaîne) le score sera égal à **0.645/1** et puis on le multiplie par 100 pour ne pas perdre des chiffres après la virgule, donc la note de cette chaîne sera => **64.5/100**.

Dans le cas où on a un outil tombé en panne, il suffit d'éliminer les chaînes qu'ils contiennent, pour choisir la meilleure chaîne remplaçante. Par exemple, si l'outil 4 tombe en panne la meilleure chaîne devient indisponible, ce qui pousse l'utilisateur à choisir la chaîne numéro deux qui ne contient pas l'outil en panne, et la considérée comme étant la meilleure chaîne.

## *Conclusion*

Dans notre étude nous avons proposé un algorithme permettant d'évaluer un ensemble de chaînes d'outils d'indexations. Une chaîne d'outils est un ensemble d'outils regroupés dans un ordre précis dans le but de générer un index donné. Cet algorithme est considéré comme étant le complémentaire de celui proposé par la thèse de [Haidar05] qui nous a permis de trouver les chaînes capables de générer un index donné, sans se préoccuper de les évaluer.

Généralement, une évaluation implique la définition de tous les facteurs et les critères caractérisant l'objet à évaluer, par exemple l'évaluation d'un groupe de voitures nécessite la définition des paramètres caractérisant chaque voiture (vitesse maximal, consommation d'essence, ..). Pour cela, le deuxième chapitre de notre étude a été consacré à la définition des paramètres caractérisant un outil d'indexation, et par la suite de définir des paramètres caractérisant une chaîne d'outils. Pour aboutir finalement à un score qui caractérise cette chaîne, et qui prend en compte, autre que les paramètres d'évaluation de la chaîne, les préférences de l'utilisateur.

Nous avons proposé et discuté ensuite plusieurs approches qui peuvent être utilisées dans la représentation des solutions possibles, ainsi que les méthodes d'extraction des ces solutions « chaînes ». Les avantages et les inconvénients de chaque approche ont été aussi discutés, pour aboutir à un choix des approches à implémenter. Nous avons présenté ensuite l'implémentation des algorithmes de chaînage et d'extraction des chaînes.

Finalement, nous avons présenté un exemple de résultats produits par l'algorithme d'évaluation, en montrant le processus de chaînage et le classement des chaînes « solutions » selon les préférences de l'utilisateur. L'exemple a montré le but principal de notre étude, qui est l'évaluation des chaînes d'outils. Un utilisateur qui demande une chaîne produisant un index donné pourra, après la mise en place de cet algorithme, choisir une chaîne à exécuter parmi une liste des chaînes classifiées selon leurs scores. De plus,

cette classification permet de choisir facilement une meilleure chaîne si la chaîne en exécution tombe en panne.

Notre algorithme se base sur les valeurs de paramètres d'évaluation prédéfinis pour évaluer les chaînes, mais pour plus de précision, ces valeurs doit être changées et modifiées avec le temps (les mettre à jour). En sauvegardant les valeurs réelles des paramètres résultants de chaque exécution d'un outil, nous pouvons associer à cet outil des valeurs plus proche de la réalité. Cette idée pourra être un sujet qu'on peut développer dans le futur.

Des travaux futurs peuvent aussi porter sur l'exécution de la chaîne produite d'une façon distribuée, au sein d'une plateforme distribuée.

# Bibliographie

- [Cardoso02a] J. Cardoso (2002a). Stochastic Workflow Reduction Algorithm. LSDIS Lab, Department of Computer Science, University of Georgia.
- [Cardoso02] J. Cardoso, J. Miller, A. Sheth & J. Arnolf (2002). Modeling Quality of Service for workflows and Web Service Processes. *Technical Report# 02-2002, LSDIS Lab. Computer Science*, University of Georgia, 44 pages.
- [Diaz01] M. Diaz (2001). *Les réseaux de Petri*. Hermès.
- [Haidar05] B. Haidar (2005). Services d'Indexation Multimédia Distribués. Département d'Informatique, Université Libanaise.
- [Klingemann98] J.Klingemann, J. Wasch & K. Aberer (1998). Adaptive outsourcing in cross-organizational workflows. GMD Report 30, GMD – German National Research Center for Information Technology.
- [Musa93] J.D. Musa (1993). Operational Profiles in Software-Reliability Engineering. *IEEE Software*, 10(2): 14-32.
- [Narayanan02] S. Narayanan & S. McIlraith (2002). Simulation, Verification and Automated Composition of Web Services, *Eleventh International World Wide Web Conference (WWW2002), Honolulu*.