Extraction de motifs dans les graphes de workflows scientifiques

Stéphanie Kamgnia Wonkap

Research Master Thesis

Research Master Thesis

Extraction de motifs dans les graphes de workflows scientifiques

Auteure :
Stephanie KAMGNIA Wonkap

Encadrant :
Sarah Cohen-Boulakia
AMIB

UNIVERSITÉ DE
RENNES 1

ENS
CACHAN
BRETAGNE

Résumé

La quantité de données données biologiques est en pleine croissance. Une fois ces données brutes collectées, il faut les analyser et les comparer aux données existantes pour obtenir de nouvelles connaissances. Cette analyse se fait grâce à l'utilisation de différents outils, scripts ou programmes que l'on peut aujourd'hui utiliser et enchaîner en utilisant un *système de gestion de workflows scientifique*.

L'utilisation des systèmes de workflows est en plein essor, donnant lieu à un important développement des bases de données pour stocker ces workflows. Cependant, très peu de techniques ont été mises en oeuvre pour pouvoir aider les utilisateurs dans la conception des workflows scientifiques, notamment pour les aider à tirer au mieux partie des workflows existants dans les bases qui pourraient être réutilisés. Un besoin fort est présent dans la communauté bioinformatique pour proposer des techniques pour guider les utilisateurs dans leur choix d'ensemble d'outils à (ré)utiliser. C'est dans ce cadre que s'articule les travaux de mon stage académique.

Plus précisément, notre objectif est d'extraire des workflows scientifiques des sous-structures fréquentes et intéressantes et constituer des librairies de motifs.

Dans nos travaux, nous avons étudié de façon approfondie les structures (grands graphes) des workflows scientifiques et nous avons fait un état de l'art des approches existantes d'extraction de motifs à base de graphes. Nous avons choisi d'étudier plus particulièrement les résultats fournis par deux algorithmes d'extraction qui ont fait leurs preuves sur des données réelles. Nous avons proposé d'utiliser et nous avons adaptés au contexte plusieurs métriques pour rendre compte de la qualité des motifs extraits.

Nos contributions sont les suivantes : (i) Nous avons mis en évidence des caractéristiques structurelles des workflows scientifiques ayant un impact sur le problème de l'extraction de motifs dans les workflows ; (ii) Nous proposons une étude comparative quantitative et qualitative des résultats obtenus par deux algorithmes d'extraction de motifs.

Table des matières

1 Introduction

La quantité de données scientifiques, plus particulièrement les données biologiques (par exemple les données produites par les techniques du séquencement haut débit), est en pleine croissance. Une fois ces données brutes collectées et conservées, il faut les analyser et les comparer aux données existantes pour obtenir de nouvelles connaissances. Cette analyse se fait grâce à l'utilisation de différents outils, scripts ou programmes. C'est ainsi que de plus en plus de travaux se développés pour permettre d'assembler, de concevoir et d'exécuter ces ensembles de programmes de traitement de données, à l'aide de *systèmes de gestion de workflows scientifiques*.

L'utilisation des systèmes de workflows pour analyser les données est en plein essor, donnant lieu à un important développement des bases de données pour stocker ces workflows. Cependant, très peu de techniques ont été mises en oeuvre pour pouvoir aider les utilisateurs dans la conception des workflows scientifiques, notamment pour les aider à tirer au mieux partie des workflows existants dans les bases qui pourraient être réutilisés. Un besoin fort est présent dans la communauté bio-informatique pour proposer des techniques pour guider les utilisateurs dans leur choix d'ensemble d'outils à (ré)utiliser. C'est dans ce cadre que s'articulera les travaux de mon stage académique.

Plus précisément, notre objectif est d'extraire des workflows scientifiques des sous-structures fréquentes et intéressantes pour pouvoir aider les utilisateurs lors de la conception de leurs workflows. L'objectif à court terme est de constituer des librairies de motifs, à partir des motifs extraits. À plus long terme, l'idée est d'utiliser ces motifs pour faire de la recommandation : en fonction d'un workflow particulièrement conçu, on pourrait suggérer à l'utilisateur de compléter son workflow par un (ensemble de) motif(s).

Dans nos travaux, nous avons étudié de façon approfondie les structures (grands graphes) des workflows scientifiques et nous avons fait un état de l'art des approches existantes d'extraction de motifs à base de graphes. Nous avons choisi d'étudier plus particulièrement les résultats fournis par deux algorithmes d'extraction, gSpan et GASTON, parce qu'ils font partie des algorithmes qui ont permis d'obtenir de bon résultats dans d'autres domaines d'application. Nous avons proposé d'utiliser et nous avons adaptés au contexte plusieurs métriques pour rendre compte de la qualité des motifs extraits.

Nos contributions sont les suivantes : (i) Nous avons mis en évidence des caractéristiques structurelles des workflows scientifiques ayant un impact sur le problème de l'extraction de motifs dans les workflows ; (ii) Nous proposons une étude comparative quantitative et qualitative des résultats obtenus par deux algorithmes d'extraction de motifs.

La suite de ce mémoire est oganisée comme suit. Nous introduisons en Section 2 de facon plus approfondie les différents concepts qui sont à la base de ce sujet de stage : les workflows et leur conception d'une part et les techniques existantes de fouille de motifs fréquents dans les graphes d'autre part. La Section 3 présente les différentes stratégies d'étiquetage que nous avons eu à considerer pour faire face au problème de l'identification des processeurs (noeuds) dans les workflows. En Section 4, nous présentons l'étude comparative que nous avons menée sur deux algorithmes d'extraction et nous discutons des résultats obtenus. Nous concluons et présenterons les nombreuses perspectives de recherche qui découlent de ce stage en Section 5.

2 Etat de l'art

2.1 Conception des workflows scientifiques

Les workflows Un workflow est une représentation d'une suite ordonnée de tâches éffectuées par une personne, une entreprise etc, servant à décrire et exécuter un processus ou permettant de décrire et exécuter de manière automatique les étapes de traitements complexes et pertinent des données. Il existe deux types de workflows : les *business workflows* et les workflows scientifiques [13]. Une meilleure définition, compréhension des workflows scientifiques découle d'une comparaison des deux types de workflows.

D'une part, les business workflows sont des workflows utilisés dans de nombreux domaines comme les domaines commerciaux, financiers, ou encore pharmaceutiques. Ils ont pour but de décrire de manière organisationnelle les étapes d'un processus : par exemple définir les étapes de validation d'un document ou d'une commande de produit, avec des étapes de validation par certains acteurs de l'entreprise, payements bancaires, génération de factures, transformation d'un document d'un format à un autre et à une heure précise. Notons que dans ce type de workflow, le résultat de l'exécution est connu à l'avance et le workflow ne sert donc qu'à traduire de manière explicite le processus qu'il implémente : on parle alors de chaîne de planification de travail. Dans un business workflow, les étapes (nécessitant parfois l'intervention humaine) sont reliées par des arcs qui représentent le flot de contrôle : on passe d'une étape à l'autre parce que des évènements déclenchent le passage. Ces évènements sont de type "une personne a cliqué sur un bouton ", "il est midi", "la banque a donné son accord pour le paiement par carte bancaire" etc.

D'autre part, les workflows scientifiques sont des workflows utilisés dans les protocoles expérimentaux dans des domaines comme la biologie ou la bioinformatique. Ils ont pour but d'analyser les données scientifiques. Ils sont donc représentés par un ensemble de tâches qui sont en fait des appels à outils (logiciels, scripts, programme etc) reliées entre elles par des arcs qui représentent le flot de données, c'est-à-dire l'ensemble des données qui transiteront dans le workflow lors de son exécution. Un workflow scientifique est donc un cadre présentant l'ensemble des tâches et l'ordre dans lequel elles pourront être exécutées. Chaque tâche, lors de l'exécution, consomme des données prises en entrée et génère des données consommées par la tâche suivante. Ainsi, l'exécution d'une tâche ne dépendra uniquement que des données prises en entrée par celle-ci : dès qu'elle aura en entrée toutes les données qu'elle attend, celle-ci pourra démarrer. Les workflows scientifiques sont donc complètement automatiques (il n'y a pas d'intervention nécessaire des utilisateurs durant toute l'exécution du workflow) et les résultats de leurs exécutions serviront à valider une hypothèse scientifique [13]. Ils offrent la possibilité aux utilisateurs de pouvoir traiter de grandes quantités de données hétérogènes, d'effectuer des calculs intensifs. La figure 1 donne un exemple de workflow scientifique. Les workflows scientifiques font l'objet des travaux de mon stage académique. Notons que dans la suite de ce rapport un noeud correspondra à une tâches qui est en fait un processeur dans *Taverna* ; un arc correspond au flot de données. **Aide à la conception de workflows**

La phase de conception du workflow consiste à le définir. Durant cette phase, les utilisateurs peuvent se baser sur des workflows préexistants qu'ils modifient pour construire leur modèle ou alors ils peuvent simplement réutiliser les workflows existants [13]. Une autre façon est de choisir dans des librairies des processeurs à enchaîner les uns aux autres. Cette phase se fait grâce à des systèmes de gestion de workflows tels que *Kepler*[4],*Triana* [19], ou *Taverna* [12], qui offrent une interface conviviale aux utilisateurs pour pouvoir construire le workflow en ajoutant ou enlevant des boîtes correspondants aux tâches du workflow.

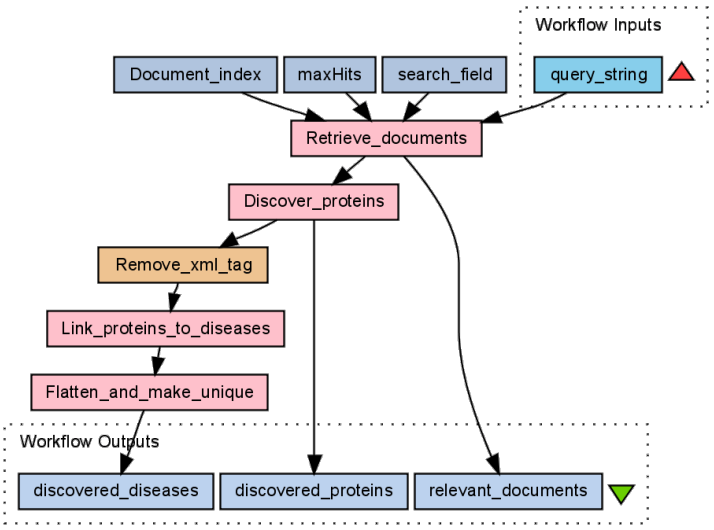


FIGURE 1: Exemple de workflow scientifique[16] issu de l'entrepôt myExperiment[12].

Permettre la réutilisation du workflow consiste à le rendre moins complexe à la lecture, et ainsi plus facile à exploiter par l'utilisateur. [6] propose une méthode pour distiller des sous-structures dites mauvaises (appelées "anti-patterns") dans les workflows scientifiques, présentant des répétitions. Ces sous-structures sont considérées comme directement associée à une mauvaise conception du workflow. La méthode présentée en [6] consiste donc à retirer ces sous-structures, à les modifier puis les réintégrer dans le workflow initial. Cette étape de transformation ne modifiera pas le comportement du workflow à l'exécution (le workflow original et le workflow réécrit produiront les mêmes données). *DistillFlow* est la première méthode se basant sur des méthode de réécriture pour réduire la redondance dans les Workflows scientiques. Il existe d'autres travaux portant sur l'aide à la conception des workflows. Parmis les travaux d'aide à la conception des workflows, on retrouve aussi [9], dans lequel l'auteur extrait des motifs des spécifications des workflows, qui résument les fonctionnalités des étapes dans ceux-ci. On peut aussi citer les travaux en [15] dans lesquels l'auteur propose des méthodes orientées graphes, pour aider à la conception des workflows, en créant de manière automatique le pipeline associé à partir d'un ensemble d'outils. D'autre part, une méthode basée sur du résumé de workflows pour le rendre moins complexe a été proposée en [3]

Il existe de nos jours de nombreux entrepôts de workflows. Ces entrepôts stockent des workflows pour permettre leur réutilisation. Parmi ces bases de données on peut citer *CrowdLabs* qui comporte des workflows utilisés pour la visualisation. On peut entre autres citer Kepler ou InforSense, ou enfin et surtout *MyExperiment* (http ://www.myexperiment.org/) qui est un grand projet sur les sciences de la vie, regroupant plus de 2000 workflows dont la majorité provient du système *Taverna*. Ce dernier constitue la source des données sur lesquelles nous avons travaillé pendant le stage.

Cette partie nous a permis de pouvoir faire un état de l'art sur les techniques existantes d'aide à la conception des workflows scientifiques. La section suivante présentera les objectifs de nos travaux de même que les différentes missions à remplir.

Ces approches n'ont pas utilisé les techniques classiques de fouilles de données dans les graphes. Nous introduisons ces méthodes et les quelques applications aux workflows dans la sous section suivante.

2.2 Extraction de motifs fréquents

L'extraction de motifs fréquents a été proposée pour la première fois par *Agrawal et al* [2] dans le cas de l'analyse du panier de la ménagère sous forme de règles d'association, pour analyser les habitudes d'achat des clients d'un supermarché. De manière formelle une règle d'association se définit ainsi :

Soit $I = i_1, i_2, \cdots, i_n$ un ensemble d'items et $T = t_1, t_2, \cdots, t_n$ un ensemble de transactions telle que $t_i \subseteq I$, une *règle d'association* est une implication de la forme $X \rightarrow Y$ où $X \in T$ et $Y \in T$ et $X \cup Y = \emptyset$.

Cette idée a été adaptée et entendue à différents domaines d'application en prenant en compte non plus en compte les informations de type transactions classiques mais en considérant des données structurées sous forme de graphes ou d'arbres. Les transactions sont alors qualifiées de motifs ou patterns. Nous allons particulièrement nous intéresser à l'extraction de patterns fréquents dans les graphes.

L'idée dans la fouille de patterns dans les graphes est donc, à partir d'un ensemble de graphes D, d'extraire des sous-graphes g_i, tel que $freq(g_i) \geq \theta$ où $freq(g_i)$ est le support qui correspond la fraction de graphe de D contenant g_i.

Cependant, on peut noter que ce problème implique la recherche d'un sous-graphe dans un graphe. Or ce problème implique le problème d'isomorphisme de sous-graphe. L'isomorphisme de sous-graphes tel que présenté dans [20], s'énnonce ainsi : étant donné deux graphes $G_1 = (V_1, E_1, f_1)$ et $G_2 = (V_2, E_2, f_2)$ où f_i est une application de E_i dans $V_i \times V_i$, le problème revient à trouver deux sous-graphes $G_{s1} = (V_{s1}, E_{s1}, f_{s1})$ et $G_{s2} = (V_{s2}, E_{s2}, f_{s2})$ respectivement de G_1 et G_2 et une bijection I_{12} permettant de mettre en correspondance les sommets de G_{s1} et G_{s2} de sorte que ceux-ci soient identiques. C'est-à-dire que : $f_1(e_{1h}) = (u_1, v_1) \in E_{s1}$ si et seulement si $f_2(e_{2h}) = (u_2, v_2) \in E_{s2}$, où $u_2 = I_{12}(u_1)$ et $v_2 = I_{12}(v_1)$.

En étendant le problème à plusieurs graphes, on peut le définir ainsi : étant donné un ensemble de graphes $\{G_k = (V_k, E_k, f_k), k = 1, \cdots, n\}$, le problème d'isomorphisme entre les graphes G_k revient à trouver un sous graphe $G_s = (V_s, E_s, f_s)$, un ensemble de sous-graphes $G_{sk} = (V_{sk}, E_{sk}, f_{sk})$ et une bijection I pour faire correspondre les sommets des sous-graphes $G_{sk}, k = 1, \cdots, n$.

Le problème de déterminer l'existence d'un isomorphisme entre deux sous-graphes est **NP-complet** [20]. Par conséquent, déterminer si deux sous-graphes sont identiques est un problème difficile.

Dans la sous section suivante nous allons présenter quelques algorithmes de fouille de graphe

2.2.1 Méthodes et leurs applications

De nombreux algorithmes d'extraction de motifs dans les structures de graphes ont été proposés ces dernières années (voir [7] pour un état de l'art). Nous donnons plus de précisions sur les algorithmes gSpan, GASTON et SUBDUE ci-après (dont plusieurs variantes ont été proposées) et que nous avons choisi parce qu'ils avaient été utilisés sur des données réelles et ont produit de bon résultats.

2.2.1.1 L'algorithme gSpan [21, 7]

1. Les contraintes sur les entrées.

   L'algorithme prends en entrée plusieurs graphes respectant les contraintes suivantes :

- Étiquetés (plusieurs noeuds et arcs peuvent avoir les mêmes étiquettes). Notons que les étiquettes sont des chiffres.
- Non orientés
- Simplesc'est -à -dire ne contenant pas d'arêtes multiples, ni de boucles
- Connexes
- Cycliques ou pas

2. Les patterns retournés en sorties

   L'algorithme retourne en sortie des patterns fréquents ayant les caractéristiques suivantes :
   - Étiquetés : (plusieurs noeuds et arcs peuvent avoir les mêmes étiquettes)
   - Non orientés
   - Simples
   - Connexes
   - Cycliques ou pas

3. Description :

   L'algorithme gSpan permet l'extraction, dans un ensemble de graphes, des patterns fréquents, c'est à dire présents dans un certains nombre de graphes. La fréquence d'un pattern est définie par rapport à un seuil et par rapport au support du pattern, qui est la fraction de graphes dans laquelle le pattern apparaît. Ainsi un pattern sera jugé fréquent si son support est supérieure ou égal au seuil définit par l'utilisateur. L'algorithme prend en entrée l'ensemble de graphes et un seuil de fréquence

   La particularité de cet algorithme est qu'il utilise un parcours en profondeur pour explorer l'espace de recherche, et aussi celui-ci construit les patterns de manière progressive, il n y a donc pas de génération de candidats. Aussi, l'algorithme représente les patterns sous une forme particulière : le code DFS (issu du parcours en profondeur du graphe), qui lui permet de faciliter le test d'isomorphisme de sous graphe. De plus, la particularité de cet algorithme est qu'il étend les patterns de telle sorte qu'il ne génère pas de doublons : l'ensemble de patterns obtenus à la fin de l'algorithme ne contient donc pas de doublons.

2.2.1.2 L'algorithme GASTON [14]

1. Les contraintes sur les entrées.

   L'algorithme prends en entrée plusieurs graphes respectant les contraintes suivantes :
   - Étiquetés (plusieurs noeuds et arcs peuvent avoir les mêmes étiquettes). Notons que les étiquettes sont des chiffres.
   - Non orientés
   - Simples
   - Connexes
   - Cycliques ou pas

2. Les patterns retournés en sorties

   L'algorithme retourne en sortie des patterns fréquents qui présentent les caractéristiques suivantes :
   - Étiquetés : (plusieurs noeuds et arcs peuvent avoir les mêmes étiquettes)
   - Non orientés
   - Simples
   - Connexes

– Cycliques ou pas

3. Description :

   L'algorithme GASTON permet l'extraction, dans un ensemble de graphes, des patterns fréquents, c'est à dire présents dans un certains nombre de graphes. Comme pour l'algorithme gSpan la fréquence d'un pattern se juge par rapport à un seuil fixé et en utilisant son support. L'algorithme prend en entrée l'ensemble de graphes et un seuil de fréquence.

   La particularité de cet algorithme est la manière dont il construit les patterns. En effet, il recherche d'abord les chemins fréquents dans les graphes, ensuite les arbres fréquents et enfin il construit à partir de ces derniers les graphes cycliques fréquents. De plus, pour éviter un trop grand nombre de test d'isomorphisme, l'algorithme GASTON garde une liste de toute les variations de des patterns.

2.2.1.3 L'algorithme SUBDUE [11, 7]

1. Les contraintes sur les entrées.

   L'algorithme prends en entrée un ou plusieurs graphes respectant les contraintes suivantes :
   – Étiquetés (plusieurs noeuds et arcs peuvent avoir les mêmes étiquettes). Notons que les étiquettes sont des chiffres ou des chaînes de caractères.
   – Non orientés ou orientés
   – Simples [1]
   – Connexes
   – Cycliques ou pas

2. Les patterns retournés en sorties

   L'algorithme retourne en sortie des patterns fréquents ayant les caractéristiques suivantes :
   – Étiquetés : (plusieurs noeuds et arcs peuvent avoir les mêmes étiquettes)
   – Orientés ou pas suivant que le graphe ou les graphes en entrées le sont
   – Simples
   – Connexes
   – Cycliques ou pas

3. Description :

   L'algorithme *SUBDUE* permet l'extraction, à partir d'un ou de plusieurs graphes, de sous-structure qui minimise le nombre de bits nécessaire pour décrire le ou les graphe(s) pris en entrée après que ceux-ci aient été compressés par l'ensemble des patterns obtenus.

   La particularité de cet algorithme est qu'il recherche les motifs qui compressent très bien le ou les graphes de la base de données et ceci en utilisant le principe de la longueur de description minimale (Minimum Description Length) tel que défini dans l'équation 1, où $I(G|S)$ est la quantité de bits nécessaire pour coder le graphe une fois compressé en utilisant le motif et $I(S)$ est la quantité de bits nécessaire pour coder le motif. L'algorithme effectue une recherche contrainte appelée "beam search", c'est à dire qu'à chaque itération de l'algorithme, un nombre limité de motifs est considéré pour être étendus.

$$DL = I(S) + I(G|S) \tag{1}$$

1. c'est -à -dire ne contenant pas d'arêtes multiples, ni de boucles

2.2.2 Mesures de qualité

Lorsqu'on fait de l'extraction de motifs en se basant sur le support, l'ensemble des motifs obtenus ne sont pas toutes intéressants. Ceci est généralement le cas lorsqu'on fixe des seuils bas. On se retrouve avec un très grand nombre de motifs et il faudrait faire un compromis entre quantité et qualité. Ainsi, le support d'un motif ne rend pas compte de la qualité de celui-ci. Il existe différents critères pour juger de l'intérêt d'un motif. On peut citer notamment :

- La concision. Un motif sera considéré comme concis si on peut le décrire avec relativement peu d'attribut
- La généralité. Un motif sera général s'il est présent dans une grande fraction des transactions de la base de donnée.
- La sûreté. Un motif sera sûr si la relation qu'il décrit apparaît dans une grande proportion des transactions de la base de données. Si on prend l'exemple des règles d'association on dira qu'elle est sûre si elle a une grande confidence. Notons que la confidence d'une règle $A \rightarrow B$ d'association est le pourcentage de transaction contenant A qui contient aussi B.
- La diversité. Un motif sera considéré divers si ses éléments diffèrent les uns des autres.
- L'utilité. Il est défini dans un contexte bien précis et dans un but précis. Ainsi, en prenant en compte le contexte, un motif sera utile s'il permet d'atteindre l'objectif fixé.
- La nouveauté, la surprise

Afin de pouvoir rendre compte des différents critères cité plus hauts, il existe différentes mesures (technique) de qualité qu'on peut classer en trois principaux groupes :

- Les mesures objectives. Ces mesures se basent seulement sur les données afin de déterminer l'intérêt des motifs
- Les mesures subjectives. Ces mesures se basent non seulement sur les données, mais aussi sur les utilisateurs à travers l'intégration de la connaissance à priori.
- Les mesures basée sémantique. Celles-ci se basent sur une connaissance du domaine et la sens du motif pour calculer l'intérêt de celui-ci.

De nombreux travaux ont été développés autour des mesures d'intérêt des motifs et ceci principalement sur les règles d'associations. On peut notamment citer les travaux de Brin et al [5], dans lesquels ils ont proposés la mesure du lift pour les règles d'association. Elle mesure comment les prémisses et la conclusion apparaissent ensemble dans les transactions le plus souvent qu'on ne l'aurait espéré. De plus ont peut citer les travaux d'*Aggarwal et Yu* [1] dans lesquels ils étudient les limites d'utiliser simplement le support et la confiance pour extraire des règles d'associations. Ils proposent un modèle d'item pour la génération de règle d'association. *Tan et al* ont proposé dans [8] l'indice de *Jaccard* pour mesurer l'intérêt des règles d'association. Il mesure la similarité entre l'ensemble des transaction contenant les prémisses et celui contenant les transactions des conclusions. Enfin on peut citer les travaux de *Tan* et *Kumar* dans lesquels ils ont proposé le facteur de certitude [17] qui mesure le degré de confiance en la relation définie par une règle d'association. Dans nos travaux nous nous sommes intéressées aux mesures objectives, l'indice de *Shannon* et la mesure du cosinus que nous présenterons dans la section suivante

3 Identification des modules

L'objectif de ces travaux est d'extraire des motifs fréquents dans les graphes représentant les workflows scientifiques, afin d'aider à leur conception. Pour que des motifs fréquents puissent être extraits, les noeuds des workflows -qui jouent un rôle majeur dans leur structure- doivent être

étiquetés avec soin. En d'autres termes, on souhaite (autant que possible) qu'un même processeur soit étiqueté de façon identique dans l'ensemble des workflows qui le contiennent et qu'une même étiquette ne puisse pas être utilisée pour identifier deux processeurs différents.

L'objectif de cette section est de présenter les probllèmes rencontrés et les stratégies proposées relativement à l'étiquetage des processeurs des workflows.

Plus précisément, la section est organisée comme suit. Nous présenterons dans un premier temps les différentes caractéristiques des processeurs mis à disposition par *Taverna* (section 3.1). Ensuite, nous présenterons les difficultés rencontrées liées à la comparaison de ceux-ci dans la construction des graphes des workflows et les différentes stratégies proposées pour pallier les difficultés rencontrées (section 3.2).

3.1 Le problème de l'identification des modules

Les processeurs réalisent des tâches bioinformatiques sur les entrées qu'ils reçoivent pour générer des sorties. Un processeur peut contenir un script (Java, R,..) ou encore faire appel à un outil disponible (service web..). Pour chaque processeur, on dispose d'un ensemble d'informations décrit brièvement ci-après.

- Un nom. Le nom est attribué par le concepteur du workflow le plus souvent en fonction du contexte dans lequel il réalise son workflow.
- Un type. Un type décrit la nature du processeur : 45 types différents ont été recensés. On peut citer par exemple, le type constante, le type *Xpath* pour définir un chemin Xpath, le type *Local service* qui correspond aux services propres à *Taverna*, le type *SOAP* qui offre des services web, le type *Biomart* qui offre l'accès à des entrepôts de données, le type *Beanshell* qui interprète des scripts, etc.
- Un contenu. Il donne accès au code source du processeur. Certains types de processeurs ont un contenu disponible. Il s'agit des types script, services web et constante.
- Un identifiant unique. Il est attribué au processeur à l'intérieur d'un workflow. Cet identifiant permet notamment de différencier les différentes occurrences d'un même processeur dans un workflow.
- Des ports d'entrée et de sortie. Ils permettent de connecter les processeurs entre eux, pour ainsi former un workflow.

En conséquence, il n'y a pas de moyen d'identifier un processeur donné à travers les workflows puisqu'il peut être renommé (nom) et identifié de plusieurs façons (selon le workflow).

Attribuer une étiquette aux processeurs de façon à identifier la présence d'un même processeur à travers les workflows (ou à l'intérieur d'un même workflow) est le problème que nous considérons ici. Notons que les étiquettes sont des chiffres car c'est une contrainte imposée par les algorithmes de fouille de graphe que nous avons présenté à la section 2. Nous avons rencontré 4 cas de difficultés, que nous présenterons dans les sous-sections suivantes.

3.2 Propositions de stratégies d'étiquetage

3.2.1 Similarité de noms des processeurs

L'on serait tenté d'attribuer les étiquettes aux sommets des graphes des workflows en se basant sur une égalité stricte des noms des processeurs auxquels ils sont rattachés : c'est-à-dire deux processeurs de noms différents seraient systématiquement considéré comme différents. Cette solution n'est

TABLE 1: Statistiques présentant les observations des différentes difficultés rencontrées.

Observation	Nombre observé
# Nombre de processeurs total	16 819
Pourcentage de processeurs ayant le même type et des contenu identiques	73%
# Classes observées pour un seuil de similarité sur les noms de processeurs à :	
0.8	5 802
0.7	5 409
Pourcentage de processeurs n'ayant pas de contenu	31%
# Processeurs ayant des noms différents mais contenu identiques	32%

pas envisageable car comme le montre la table 1, 32% des processeurs ont des contenus identiques alors qu'ils ont des noms différents. Ce choix ferait donc perdre une information précieuse.

En effet, les utilisateurs rajoutent des majuscules, des caractères spéciaux ou des chiffres afin de pouvoir distinguer les occurrences d'un même processeur, comme illustré dans la figure 2.

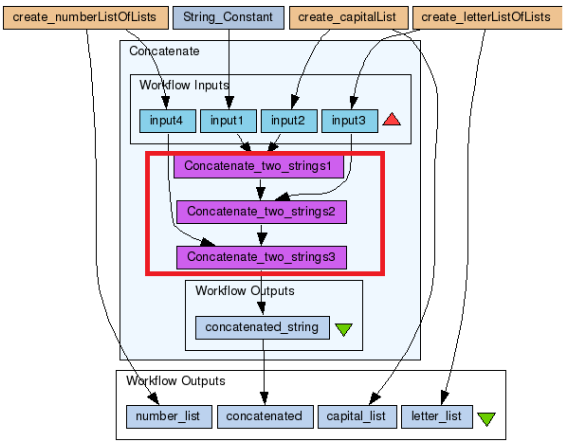


FIGURE 2: *Exemple de workflow (myExperiment 100)*, illustrant (partie encadrée en rouge) un cas où des chiffres et des caractères spéciaux ont été ajoutés aux noms du processeur pour différencier les différentes occurrences d'un même processeur.

Afin de résoudre ce problème nous avons considéré une similarité entre les noms des processeurs. Ainsi deux processeurs P1 et P2, seront considérés identiques s'ils ont des noms distants de L suivant la distance d'édition (distance de levenshtein), auquel cas P1 et P2 auront la même étiquette. En d'autres termes, nous avons adopté la stratégie suivante :

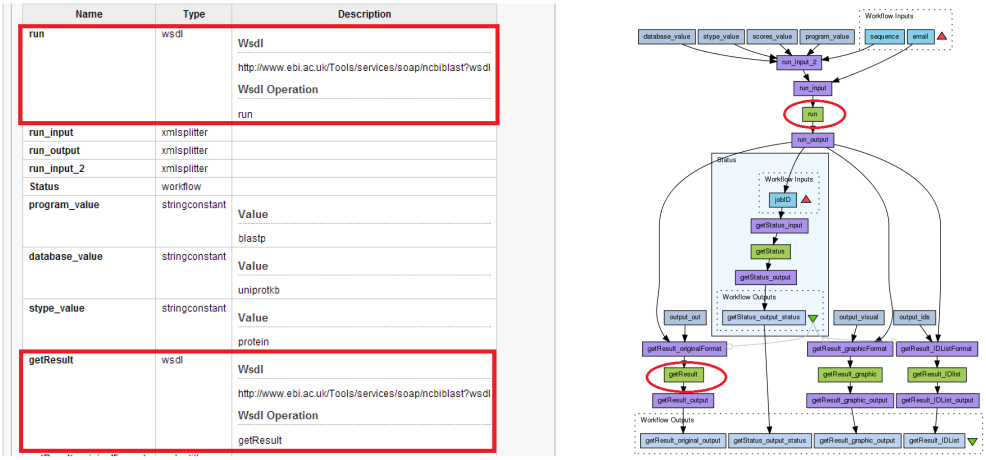
Stratégie: Similarité des noms. *Si Dist(P1.nom,P2.nom)≤ L alors Etiquette1 = Etiquette2*

Considérer la similarité entre les noms des processeurs nous a permis de construire des classes d'équivalence. À l'intérieure d'une classe d'équivalence, tous les sommets (processeurs) ont la même

étiquette. En prenant par exemple un seuil de 0.7 comme de similarité pour la distance de levenshtein [2], on obtient 5 409 classes d'équivalence (cf table 1). Ce qui revient à dire que 32% de processeurs sont similaires.

3.2.2 Exploitation du contenu des processeurs

En observant les workflows, nous avons remarqué que les utilisateurs ont tendance à attribuer un nom à un processeur en fonction du contexte dans lequel ils réalisent leurs workflows. Ainsi on pourrait se retrouver dans un cas où deux processeurs ont des noms cette fois très différents mais de contenus identiques comme l'illustre la figure 3.

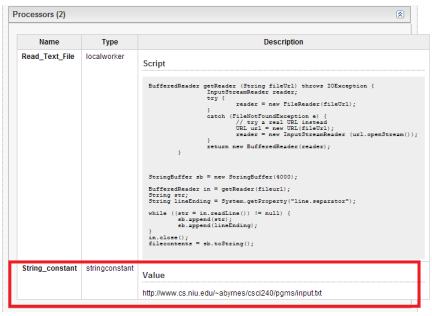


(a) Description des Processeurs

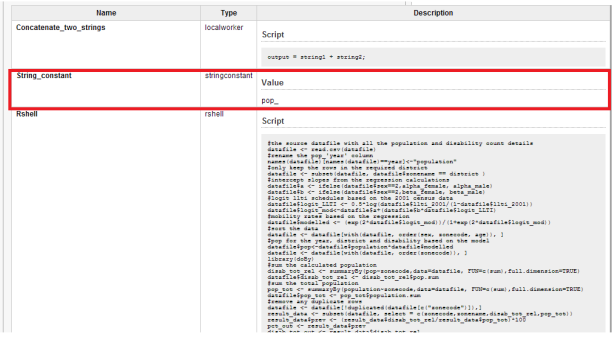
(b) Workflow 1765 v1

FIGURE 3: *Exemple de workflow (myExperiment 1765 version 1)*. Exemple de workflow de *Taverna*(à droite) et la description de ses processeurs (à gauche), illustrant (parties encadrée en rouge) un cas où deux processeurs ont des noms totalement différents mais des contenus identiques.

2. Distance donnant une mesure de la similarité entre deux chaînes



(a) Description des processeurs du workflow 933 de myExperiment



(b) Description des processeurs du workflow 2770 de myExperiment

FIGURE 4: *Exemple de description de processeurs de workflows.* Cette figure illustre (parties encadrées en rouge) un cas où deux processeurs ont des noms similaires (ici égaux) mais des contenus différents.

Dans ce cas les noms ne sont même plus similaires. Une autre stratégie doit être proposée.

Pour résoudre le problème posé par ce cas de figure, nous proposons de prendre en compte les contenus des processeurs pour déterminer leur étiquette. Afin d'avoir une comparaison plus exacte, nous n'avons comparé que les contenus de processeurs ayant le même type. Ainsi, deux processeurs P1 et P2 seront similaires s'ils ont le même type et le même contenu, auquel cas P1 et P2 ont la même étiquette. Plus explicitement on a :

Stratégie: Égalité des contenus. *Si (P1.type = P2.type et P1.content = P2.content) alors Etiquette1=Etiquette2*

3.2.3 Prise en compte des différences dans le contenu des processeurs

Une observation plus fine des processeurs montre qu'ils se différencient parfois par très peu d'éléments : la manière de nommer les variables dans un code, la façon dont celui-ci est indenté ou le fait de contenir des commentaires, l'utilisation des majuscules (comme illustré dans la figure 5) ou d'autres caractères spéciaux. Bien que ces processeurs aient des contenus syntaxiquement différents

ils peuvent réaliser la même opération. On souhaiterait donc que ces processeurs soient considérés comme similaires.

(a) Description des processeurs du workflow 2394 de myExperiment

(b) Description des processeurs du workflow 2390 de myExperiment

FIGURE 5: *Exemple de description de processeurs de workflows.* Cette figure illustre (parties encadrée en rouge) un cas où deux processeurs ont des contenus qui diffèrent par l'utilisation de majuscules.

Nous proposons donc ici de considérer, non plus une égalité stricte entre les contenus, mais une similarité de contenu . Ainsi, deux processeurs P1 et P2 seront similaires s'ils ont le même type et des contenus distants d'une distance L. De manière plus précise on a :

Stratégie: Similarité des contenus. *Si (P1.type = P2.type et Dist(P1.content,P2.content)<=L) alors Etiquette1=Etiquette2*

Cette notion de similarité a été réalisée en utilisant à nouveau la distance d'édition (distance de levenshtein). Afin d'appliquer la distance de levenshtein avec de faibles valeurs, nous avons nettoyé

les contenus des processeurs, en retirant tous les caractères spéciaux des contenus, les espaces et enfin le contenu a été mis en minuscule. Étant donné que les chaînes des scripts sont pour la plupart bien plus grandes que les contenus des autres processeurs, nous avons avons utilisé des seuils différents selon qu'on compare des scripts ou d'autres types de contenus.

3.2.4 Exploitation conjointe des noms et des contenus

Comme souligné plus haut, il existe des types pour lesquels nous ne disposons pas de contenu. D'après la table 1, 30% de processeurs n'ont en effet pas de contenu. Ainsi comparer les processeurs en prenant pas en compte le fait que certains ont des contenus vides, pourrait biaiser les résultats obtenus.

Nous proposons donc ici une stratégie dans laquelle deux processeurs, P1 et P2 seront égaux selon les deux cas suivant : soient ils n'ont pas de contenu auquel cas leurs noms doivent être séparés d'une distance L1 (confère stratégie 1), soient leurs contenus sont séparés d'une distance L2 (confère stratégie 2). Autrement dit, nous avons adopté la stratégie suivante :

Stratégie: Similarité des contenus ou Similarité des noms. *Si ((P1.content="" et P2.content ="" et Dist(P1.name, P2.name)<=L1) ou*
((Dist(P1.content,P2.content)<=L2)) alors P1=P2

Les prétraitement décrits pour les chaînes relatives au contenu, présentés dans la stratégie précédente sont identiques ici. Ils ont été aussi effectués sur les noms des processeurs. Dans ce qui précède, nous avons introduit 4 stratégies d'étiquetage pour les processeurs des workflows. La section suivante évalue deux algorithmes d'extraction de motifs fréquents dans une base de workflows étiquetée suivant chacune de ces 4 stratégies.

4 Évaluation d'algorithmes d'extraction de motifs fréquents dans les workflows scientifiques

Dans cette partie, nous évaluerons les deux algorithmes utilisés pour l'extraction des motifs dans les workflows scientifiques. Cette évaluation se fera suivant différents critères structurels ou mesurant la qualité des motifs retournés. Cette évaluation a pour but de pouvoir comparer ces algorithmes, afin de déterminer lequel des algorithmes serait le mieux adapté à l'extraction des motifs dans les graphes des workflows de façon à utiliser ces motifs pour constituer une base de motifs ré-utilisables à conseiller aux utilisateurs.

4.1 Contexte

L'objectif de cette section est de présenter les critères que nous allons utiliser pour comparer les résultats obtenus par les deux algorithmes d'extraction de motifs, *gSpan* et *GASTON*. Nous distinguerons les critères relatifs à la structure des motifs, des critères capables de rendre compte de l'*intérêt* d'un pattern.

Nous allons nous intéresser aux critères basés sur l'*intérêt* des patterns. En effet, les critères structurels ne suffisent pas à eux seuls à déterminer la qualité d'un pattern. On peut avoir un pattern qui a une petite taille mais celui-ci a un grand intérêt, parce que celui-ci possède peu (ou pas) de répétitions de même processeurs ; ou encore celui-ci est intéressant parce que les associations

de sommets, telles que définies par ses arcs, sont très fortes. C'est-à-dire que si on retrouve un noeud du couple dans un graphe, on a de grandes chances de le retrouver rattaché à son compère dans le couple tel que défini par l'arc dans le pattern.

Nous souhaiterons juger de l'intérêt d'un pattern en se basant sur le fait que celui-ci représente très bien la base de données, c'est-à-dire qu'il comporte un grand nombre des étiquettes utilisées dans l'ensemble des graphes des workflows. Ensuite, on aimerait pouvoir juger l'intérêt d'un pattern par rapport aux associations définies par les arcs dans le motif. Ainsi un motif devra être jugé intéressant si les corrélations entre les noeuds telles que définies par ses arcs sont fortes. En effet ceci nous aidera dans la mesure où on souhaiterait faire conception assistée de workflows à travers de la recommandation de motifs à utiliser.

Pour ce faire, nous nous sommes intéressés à deux principales mesures : l'indice de *Shannon* et la mesure du *cosinus*, nous motiverons ce choix dans les sous sections correspondantes.

4.2 Analyse en tenant compte des critères structurels : Taille des motifs et taille de l'ensemble résultat

Dans cette section, nous allons comparer les algorithmes *gSpan* et *GASTON*, en tenant compte de la structure des motifs qu'ils renvoient, de la taille des ensembles résultats et de la taille des motifs qu'ils renvoient.

Cette comparaison se fera suivant les 4 cas d'étiquetage des graphes de workflows présenté à la section 3. Nous avons considéré différentes valeurs de seuils que nous indiquons systématiquement ci-après. Ces valeurs ont été choisies en prenant en compte le contexte de notre étude : Les workflows étant relatifs à des domaines scientifiques (à l'intérieur même de la bioinformatique) très variés, un même motif ne pouvait pas se retrouver avec une fréquence élevée dans la base considérée.

En observant les tableaux ci-dessous, on note une variation de la taille de l'ensemble des motifs retournés par les deux algorithmes. Sans surprise on remarque que le cas où l'on considère l'étiquetage basé sur une égalité stricte des contenus des processeurs ne renvoie que peu de résultats. L'ensemble des motifs retournés est assez petit et les motifs de taille maximale sont assez petits, ils ne contiennent que 6 arcs.Si on considère maintenant le cas d'étiquetage où on s'intéresse à une similarité sur le contenu, on observe que c'est dans ce cas de figure qu'on obtient le plus grand nombre de motifs quelque soit l'algorithme et quelque soit le seuil choisi. Ceci traduirait le fait que cette stratégie d'étiquetage est souple. Aussi, on observe que c'est dans ce cas de figure qu'on obtient les motifs de plus grands, avec une taille maximale atteinte de 16 arcs. Avec le dernier cas qui fait un compromis entre le nom et le contenu, on obtient des résultats assez équilibrés pour les deux algorithmes. La différence de taille entre les ensembles solutions des deux algorithmes n'est pas notable.

De manière générale, on peut remarquer que l'algorithme *GASTON* retourne un plus grand nombre de motifs que l'algorithme *gSpan*. De plus, on peut remarquer que les deux algorithmes ont à peu près le même comportement, étant donné que pour les deux algorithmes, les plus grands motifs ont à peu près la même taille (à un arc près). En outre, on remarque que le support des motifs de plus grande taille, est quasiment le même pour dans les 4 cas d'étiquetage choisi.

Algorithmes	Seuil	Nombre de motifs	Taille du plus grand motif	Nombre de graphes contenant le plus grand motif
gSpan	0.02 (26 graphes)	156	8 arcs (voir figure 18)	31 graphes
	0.01 (13 graphes)	510	15 arcs (voir figure 19)	13 graphes
GASTON	0.02 (26 graphes)	177	8 arcs (voir figure 18)	31 graphes
	0.01 (13 graphes)	912	16 arcs	13 graphes

TABLE 3: Nombre de motifs obtenus avec la stratégie similarité de contenu pour un seuil de distance d'édition à 0.6 pour les scripts et un seuil à 0.7 pour les autres types

Algorithmes	Seuil	Nombre de motifs	Taille du plus grand motif	Nombre de graphes contenant le plus grand motif
gSpan	0.02 (26 graphes)	125	8 arcs (voir figure 20)	29 graphes
	0.01 (13 graphes)	663	13 arcs (voir figure 21)	14 graphes
GASTON	0.02 (26 graphes)	153	8 arcs (voir figure 20)	29 graphes
	0.01 (13 graphes)	1047	14 arcs	17 graphes

TABLE 4: Nombre de motifs obtenus avec la stratégie similarité de contenu pour un seuil de distance d'édition à 0.7 pour les scripts et un seuil à 0.8 pour les autres types

Algorithmes	Seuil	Nombre de motifs	Taille du plus grand motif	Nombre de graphes contenant le plus grand motif
gSpan	0.02 (26 graphes)	39	4 arcs (voir figure 16)	28 graphes
	0.01 (13 graphes)	247	6 arcs (voir figure 17)	13 graphes
GASTON	0.02 (26 graphes)	116	4 arcs (voir figure 16)	28 graphes
	0.01 (13 graphes)	347	6 arcs (voir figure 17)	15 graphes

TABLE 2: Nombre de motifs obtenu avec la Stratégie égalité des contenus.

Dans la section suivante nous allons présenter les critères d'évaluation des algorithmes basés sur l'intérêt des motifs extraits par les algorithmes *gSpan* et *GASTON*.

Algorithmes	Seuil	Nombre de motifs	Taille du plus grand motif	Nombre de graphes contenant le plus grand motif
gSpan	0.02	66	2 arcs	30 graphes
	0.01	569	10 arcs (voir figure 12)	16 graphes
GASTON	0.02	164	3 arcs	40 graphes
	0.013	311	10 arcs	13 graphes

TABLE 5: Nombre de motifs obtenus avec la stratégie similarité de noms pour un seuil de distance d'édition à 0.8.

Algorithmes	Seuil	Nombre de motifs	Taille du plus grand motif	Nombre de graphes contenant le plus grand motif
gSpan	0.02 (26 graphes)	68	2 arcs	30 graphes
	0.01 (13 graphes)	679	12 arcs (voir figure 15)	16 graphes
GASTON	0.02 (26 graphes)	126	3 arcs	27 graphes
	0.01 (13 graphes)	748	11 arcs	17 graphes

TABLE 6: Nombre de motifs obtenus avec la stratégie similarité de noms pour un seuil de distance d'édition à 0.7.

Algorithmes	Seuil	Nombre de motifs	Taille du plus grand motif	Nombre de graphes contenant le plus grand motif
gSpan	0.02 (26 graphes)	125	7arcs (voir figure 22)	graphes
	0.01 (13 graphes)	523	10 arcs (voir figure 23)	14 graphes
GASTON	0.02 (26 graphes)	145	7arcs (voir figure 22)	27 graphes
	0.01 (13 graphes)	470	9 arcs	19 graphes

TABLE 7: Nombre de motifs obtenus avec la stratégie similarité de contenu ou similarité de noms pour un seuil à 0.7 pour les noms, à 0.7 pour les script et à 0.7 pour les autres.

Algorithmes	Seuil	Nombre de motifs	Taille du plus grand motif	Nombre de graphes contenant le plus grand motif
gSpan	0.02 (26 graphes)	123	7arcs (voir figure 22)	27 graphes
	0.01 (13 graphes)	549	10 arcs (voir figure 13)	14 graphes
GASTON	0.02 (26 graphes)	153	7 arcs (voir figure 22)	27 graphes
	0.013 (16 graphes)	501	9 arcs	19 graphes

TABLE 8: Nombre de motifs obtenus avec la stratégie similarité de contenu ou similarité de noms pour un seuil à 0.7 pour les noms, à 0.6 pour les script et à 0.7 pour les autres.

4.3 Comparaison en tenant compte de l'intérêt des motifs

Dans cette partie nous allons présenter les résultats obtenus après calcul de l'intérêt des motifs obtenus avec les algorithmes *gSpan* et *GASTON* en utilisant l'indice de Shannon et la mesure de cosinus. Afin de comparer les deux algorithmes, nous avons, selon les 4 stratégies d'étiquetage de graphes présentées à la section 3.2, établi des intervalles de valeurs et comparé pour les deux algorithmes, les pourcentage de workflows présents dans chacun des intervalles. Cette tâche a été réalisé en prenant séparément l'indice de Shannon et la mesure de cosinus.

4.3.1 Comparaison en utilisant l'indice de Shannon

Motivation.

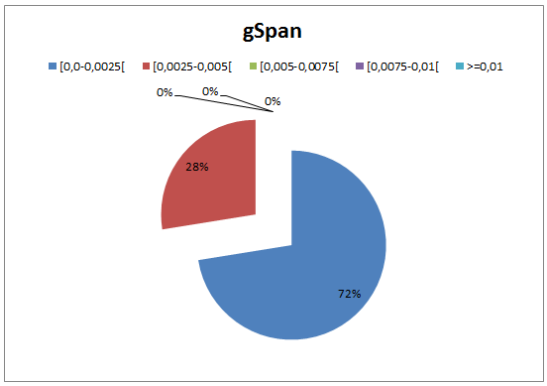
L'indice de *Shannon* [10, 18] est une mesure utilisée pour calculer la diversité d'un échantillon d'étude. Elle est utilisée par exemple dans le domaine de la synthèse pour juger de l'intérêt du résumé. Dans le cadre des patterns des workflows, l'indice de Shannon permet de caractériser à quel point un motif est diverse : un motif est diverse s'il contient un grand nombre d'étiquettes différentes. Ceci nous permettra de montrer à quel point un motif couvre l'ensemble des processeurs présents dans la base de données, c'est-à-dire à quel point le motif est représentatif de la base de données. Plus un motif contiendra les différentes classes de processeurs (c'est-à-dire plus il y aura différentes étiquettes pour les processeurs), plus celui-ci sera jugé intéressant. L'indice de *Shannon* d'un motif s'obtient grâce à l'équation 2. Dans cette équation S est le nombre total d'étiquettes différentes dans l'ensemble des workflows, P est le motif considéré, p_i est la proportion de sommets dans la classe i par rapport au nombre total de classes que l'on obtient comme suit : $p_i = \frac{n_i}{N}$, où n_i est le nombre de sommets dans la classe i, et N est le nombre total sommets. L'indice de *Shannon* prend ses valeurs dans l'intervalle $[0, 1]$. Ainsi si on a $IS(P) = 0$ alors le motif ne contient qu'une seul classe d'étiquette, et si au contraire $IS(P) = 1$ alors le motif couvre toutes les étiquettes présentes dans les workflows.

$$IS(P) = -\sum_{i=1}^{S} p_i log_2(p_i) \tag{2}$$

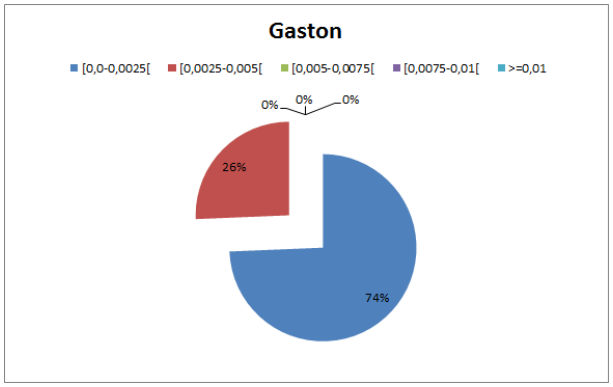
Résultats Après calcul de l'indice de Shannon sur les motifs obtenus avec *gSpan* et *GASTON*, nous avons pu remarquer que la valeur maximale de l'indice de *Shannon* ne dépasse pas 0.02. Ceci est tout à fait normal étant donné que les motifs contiennent en moyenne 10 noeuds et que l'on a total de 16 819 noeuds (processeurs). Nous avons donc établi les intervalles en tenant compte du fait que les valeurs de l'indice de Shannon se concentrent dans l'intervalle [0.0 ;0.02[. Nous avons donc défini un pas de 0.0025 et nous obtenons les intervalles suivants [0.0 ;0.00025[, [0.0025 ;0.005[, [0.005 ; 0.0075[, [0.0075 ;0.01[. Dans les sous parties suivantes, nous allons présenter les résultats obtenu suivant les 4 cas de figure.

4.3.1.1 Stratégie égalité des contenus La figure 6 montre deux graphiques, qui présentent comment sont répartis les motifs obtenus pour un seuil de 0.01 avec l'algorithme *gSpan* (figure6a) et *GASTON* (figure6b). Dans cette figure, on peut constater que les motifs obtenus avec les deux algorithmes ne sont pas très divers. En effet le plafond de cette valeur est de 0.005. Cependant, bien qu'ils ne le soient pas, pour les deux algorithmes, très diverses, les motifs de *gSpan* sont un

peu plus divers que ceux de *GASTON*. Ceci se montre par le fait que la zone réservée à l'intervalle [0.0025 ;0.005[est plus grande dans le cas de l'algorithme *gSpan* comparée a celle dans le cas de l'algorithme *GASTON*.



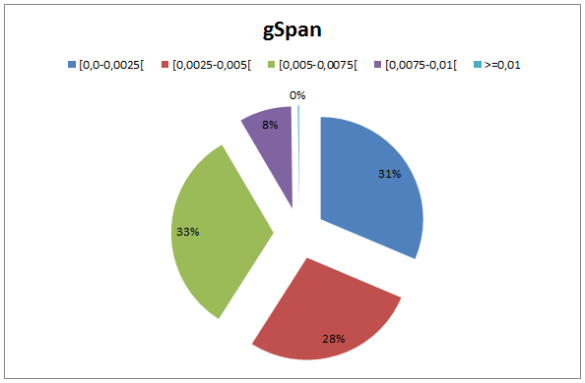
(a) Répartition des motifs de *gSpan* en pourcentage des workflows suivant l'indice de Shannon



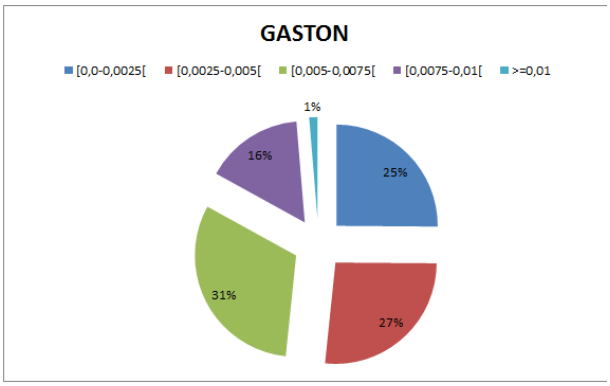
(b) Répartition des motifs de *GASTON* en pourcentage des workflows suivant l'indice de Shannon

FIGURE 6: Répartition du nombre de motifs, suivant la valeur de l'indice de shannon, dans les intervalles choisis, 6a présente les résultats pour *gSpan* tandis que 6b ceux de l'algorithme *GASTON*. Les motifs sont obtenus en considérant la stratégie égalité des contenus. Le seuil de fréquence choisi pour les deux algorithmes est de 0.01.

4.3.1.2 Stratégie similarité des contenus Dans la figure 7, on peut constater qu'en prenant en compte l' étiquetage des graphes basé sur une similarité sur le contenu, les algorithmes renvoient tous les deux des motifs dont la valeur de l'indice de Shannon est supérieure à 0.01. Ceci est intéressant par rapport au cas précédent. En effet les motifs retournés dans ce cas de figure sont assez divers. On peut le remarquer dans la figure, pour le cas *GASTON*, la zone dans l'intervalle [0.0 ;0.0025[est la plus petite des zones. De plus, en observant la figure on pourrait conclure que, si on utilisait l'indice de shannon pour filtrer les résultats avec une valeur minimale de 0.0075, pour ce cas, l'algorithme *GASTON* aurait tendance à renvoyer des motifs plus divers, comparé à l'algorithme *gSpan*



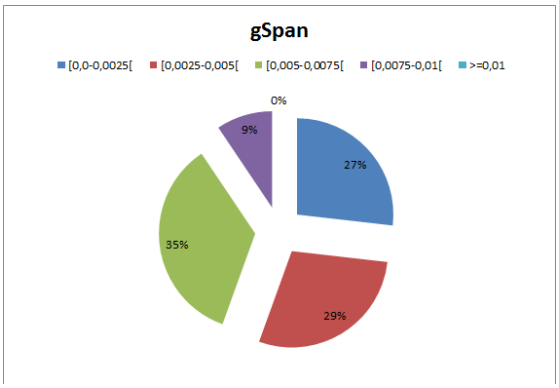
(a) Répartition des motifs de *gSpan* en pourcentage des workflows suivant l'indice de Shannon



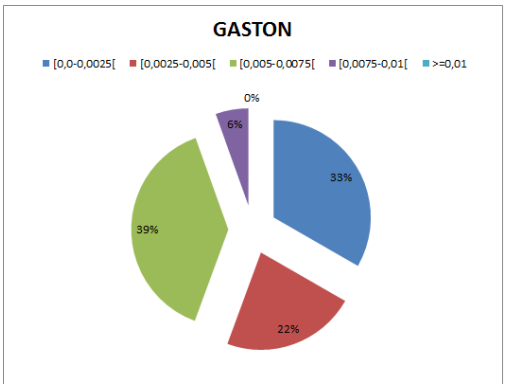
(b) Répartition des motifs de *GASTON* en pourcentage des workflows suivant l'indice de Shannon

FIGURE 7: Nombre de motifs, suivant la valeur de l'indice de shannon, dans les intervalles choisis en 7a pour l'algorithme *gSpan* tandis qu'en 7b pour *GASTON*. Les motifs sont obtenus en considérant la stratégie similarité des contenus pour un seuil choisi est de 0.6 pour les script et 0.7 pour les autres types de processeurs. Le seuil de fréquence des algorithmes et celui des deux algorithmes est de 0.01.

4.3.1.3 Stratégie similarité des noms La figure 8 montre que les motifs retournés par *gSpan* ont plus tendance à couvrir la base de donnée que ceux de *GASTON*. En effet, la portion de motifs tombant dans l'intervalle $[0.0075\,;0.01[$ est plus grande pour l'algorithme *gSpan* que celle de l'algorithme *GASTON*. De plus, si on utilisait l'indice de shannon comme filtre et qu'on fixait la valeur minimale à avoir à 0.005, on aurait put constater que les motifs de *gSpan* seraient plus diverses que ceux de *GASTON* car en effet, une grande partie des motifs de *gSpan* se concentre dans l'intervalle $[0.005; 0.0075[\cup[0.0075; 0.01[$ ceci n'est pas le cas pour l'algorithme *GASTON*.



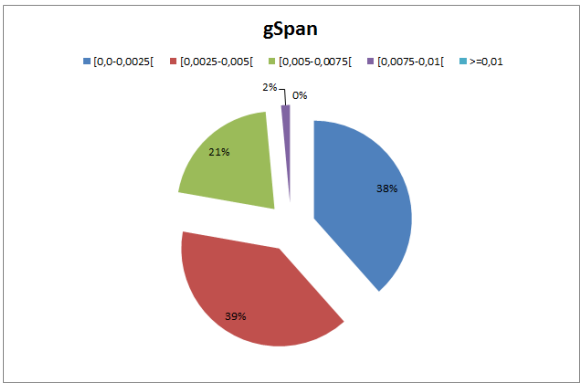
(a) Répartition des motifs de *gSpan* en pourcentage des workflows suivant l'indice de Shannon



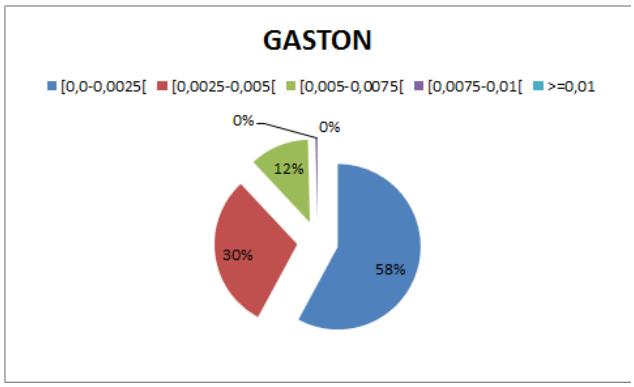
(b) Répartition des motifs de *GASTON* en pourcentage des workflows suivant l'indice de Shannon

FIGURE 8: Répartition du nombre de motifs, suivant la valeur de l'indice de shannon, dans les intervalles choisis en 8a pour l'algorithme *gSpan* et en 8b pour l'algorithme *GASTON*. Les motifs sont obtenus en considérant la stratégie similarité des noms , pour un seuil de similarité pour les noms des processeurs de 0.7 et celui des deux algorithmes est de 0.01.

4.3.1.4 Stratégie similarité des contenus ou similarité des noms D'après la figure 9 on peut remarquer qu'en considérant un mixte entre la similarité sur les noms et celle sur les contenus des processeurs pour l'étiquetage des graphes des workflows, on obtient des motifs avec des indices de similarité supérieur à 0.01. Aussi on peut constater que l'algorithme *gSpan* retourne des motifs plus divers que ceux de l'algorithme *GASTON*. En effet, comme le montre le graphique, dans le cas de la répartition de l'algorithme *GASTON*, la majorité des motifs ont leur indice de *Shannon* dans l'intervalle [0.0 ;0.0025[, contrairement à l'algorithme *gSpan* où la majorité des motifs ont leur indice de *Shannon* ≥ 0.0025.



(a) Répartition des motifs de *gSpan* en pourcentage des workflows suivant l'indice de Shannon



(b) Répartition des motifs de *GASTON* en pourcentage des workflows suivant l'indice de Shannon

FIGURE 9: La figure présente la répartition du nombre de motifs, suivant la valeur de l'indice de Shannon, dans les intervalles choisis, en 9a pour l'algorithme *gSpan* tandis qu'en 9b pour l'algorithme *GASTON*. Les motifs sont obtenus en considérant la stratégie similarité de contenus ou de noms pour un seuil de similarité de 0.6 pour les script, 0.7 pour les autres types et 0.7 pour les noms des processeurs. Le seuil des algorithmes et celui des deux algorithmes est de 0.01.

En conclusion, on remarque une différence notable entre le cas où l'étiquetage des graphes de workflows se fait en considérant une égalité stricte du contenu des workflows et les autres cas d'étiquetage. Ceci est du au fait qu'en considérant une égalité stricte on est plus rigide en taille. On obtient donc des motifs assez restreints. On remarque que la taille maximale des motifs est de 6 arcs ce qui donne un nombre maximum de noeuds de 7 (étant donné que les motifs sont connexes), et on a aussi pu noter en faisant la comparaison basée sur la possibilité de contenir des répétitions que dans le cas d'une égalité stricte sur le contenu, 30% des motifs contiennent des répétitions. Ceci pourrait donc expliquer pourquoi les valeurs de l'indice de *Shannon*, sont concentrées dans l'intervalle [0.0 ;0.0025[. Lorsqu'on passe aux autres cas d'étiquetage, on devient assez souple. En effet on remarque une augmentation notable de la taille des motifs allant jusqu'à 16 arcs dans le cas où on considère l'étiquetage basé sur la similarité sur le contenu des processeurs, offrant ainsi une possibilité de 17 noeuds. En effet les motifs grandissent, et offre la possibilité de contenir une plus grande diversité de noeuds. Ceci expliquerait pourquoi les autres intervalles ont reçu des candidats. Cependant, la comparaison basée sur les répétitions d'étiquettes dans les motifs indique qu'en moyenne 50% des motifs contiennent des répétitions. Mais en observant les graphiques ci-dessus, il en ressort que ces répétions ne sont pas grandes au sens du nombre de fois qu'une étiquette se répète.

Après observation des résultats obtenus dans les différents cas figure précédents, il en ressort,

d'après l'indice de Shannon, qu'il est plus intéressant d'utiliser l'algorithme *gSpan*, pour l'extraction des motifs dans les workflows. En effet, l'algorithme a tendance à retourner des motifs plus diverses, c'est-à-dire comportant des étiquettes diverses.

Dans la section qui suit, nous allons présenter les résultats de la comparaison des algorithmes *gSpan* et *GASTON* obtenus en utilisant comme mesure d'intérêt la mesure du cosinus.

4.3.2 Comparaison en utilisant la mesure du cosinus

Motivation. La seconde mesure considérée, est la mesure du *cosinus* [10, 18]. C'est une mesure objective utilisée dans l'extraction des règles d'association, mais aussi dans la fouille de document. Elle permet de mesurer la distance entre les prémisses et la conclusion d'une règle d'association, pour ainsi qualifier le degré de liaison des prémisses et la conclusion, ceci à travers les chevauchements possibles des transactions contenant les prémisses et celles contenant la conclusion. Ainsi si on a une règle R de la forme $A \rightarrow B$, sa mesure de cosinus s'obtient en suivant l'équation 3. Dans cette équation $P(AB)$ représente le support de A et B, $P(A)$ celui de A et enfin $P(B)$ le support de B. La mesure de *cosinus* prend ses valeurs dans l'intervalle $[0; 1]$. Quand cette mesure atteint son minimum, l'on conclut qu'il n'y a pas de chevauchement entre les transactions contenant les prémisses et celles contenant la conclusion, auquel cas les prémisses et la conclusion sont indépendantes. Cependant, quand cette mesure atteint son maximum, alors le chevauchement entre les transactions des prémisses et celles des conclusions est total ; c'est-à-dire que dans toutes les transactions qui contiennent les prémisses, on retrouve aussi la conclusion.

$$cos(R) = \frac{P(AB)}{\sqrt{P(A) * P(B)}} \tag{3}$$

Afin d'adapter cette mesure aux motifs dans les workflows scientifiques, nous avons procédé de la manière suivante : un arc peut être vu comme une règle d'association dans laquelle il n'y a qu'une seule prémisse. Si $S_1 \rightarrow S_2$ est un arc, alors son cosinus sera obtenu tel que décrit dans l'équation 3. Une valeur à zéro de cette mesure signifie que dans les graphes qui contiennent le noeud (processeur) S_1, on a jamais de liaison directe avec S_2. Une valeur maximale de cette mesure signifie que si les processeurs S_1 et S_2 apparaissent dans un graphe, alors il y a nécessairement un arc entre S_1 et S_2. La mesure de cosinus d'un motif sera la moyenne des mesures de cosinus de ses arcs.

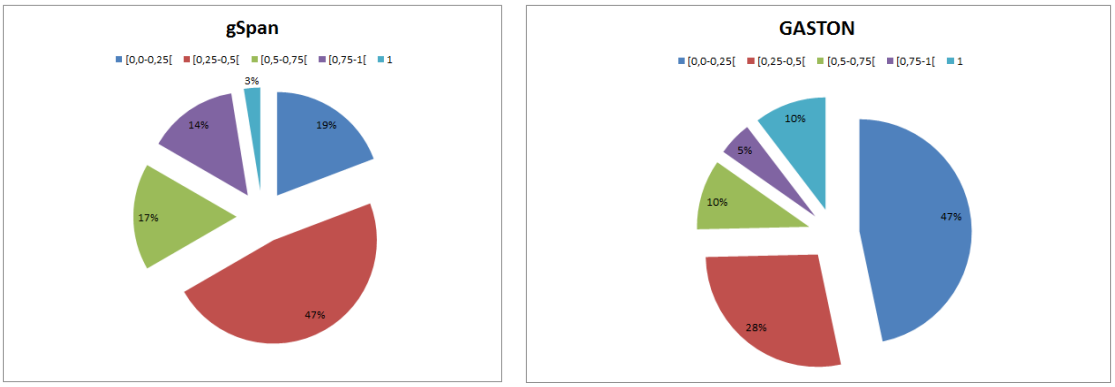
Résultats.

Étant donné que la mesure du cosinus prend ses valeurs dans l'intervalle [0,1], pour définir nos intervalles, nous avons choisi un pas de 0.25. Ainsi les intervalles que nous obtenons sont les suivants : [0.0,0.25[, [0.25,0.5[, [0.5,0.75[,[0.75,1[. Notons que pour ne pas biaiser les résultats de la répartition des motifs obtenus avec l'algorithme *gSpan*, nous n'avons considéré que les motifs ayant au moins un arc. Nous avons adopté cette stratégie parce que l'algorithme *GASTON* ne renvoie que des motifs ayant au moins un arc. Aussi, nous ne sommes pas intéressés par des motifs ne comportant qu'un seul sommet, car ceux-ci ne peuvent être exploité pour l'aide à la conception des workflows. En plus nous savons à l'avance que leur mesure de cosinus sera toujours nulle. Dans ces conditions de test, il est à noter que la mesure de cosinus sera toujours strictement positive.

Ainsi, nous allons dans ce qui suit, présenter les différents résultats obtenus en considérant la mesure du cosinus pour la comparaison de *gSpan* et *GASTON*, suivant les 4 cas d'étiquetage des graphes des workflows.

4.3.2.1 Cas de l'égalité stricte des contenus des processeurs des workflows : La figure 10 présente la répartition du nombre de motifs pour les algorithmes *gSpan* (figure10a) et *GASTON*(figure10b), selon que leur valeur de leur mesure de cosinus tombe dans les différents intervalles.

En observant cette figure, on peut noter que près de la moitié (43%) des motifs obtenus avec *GASTON* ont leur valeur cosinus strictement inférieure à 0.25, contrairement à l'algorithme *gSpan* où seul 19% des motifs ont leur valeur inférieure à 0.25. On peut aussi remarquer que pour les deux algorithmes nous obtenons des motifs ayant des valeurs de mesure de cosinus égale à 1. Ceci traduisant une corrélation parfaite entre les sommets des arc de ces motifs. Aussi on peut remarquer que les motifs retournés par l'algorithme *gSpan* sont plus corrélés que ceux de l'algorithme *GASTON*. En effet 34% des motifs ont leur valeur de la mesure du cosinus supérieure à 0.5, contre 25% pour l'algorithme *GASTON*.

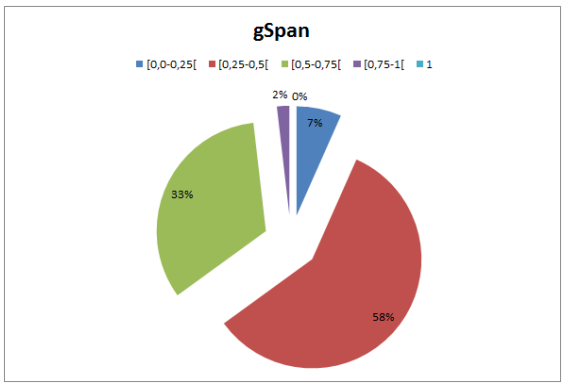


(a) Répartition des motifs de *gSpan* en pourcentage des workflows suivant la mesure du cosinus

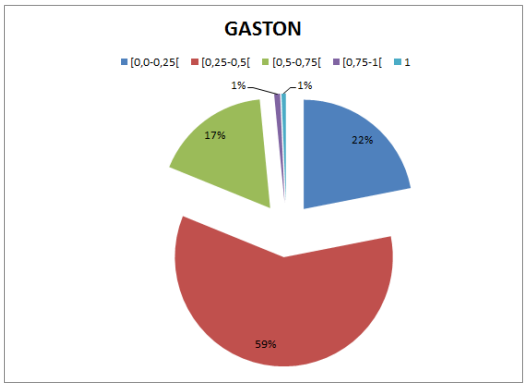
(b) Répartition des motifs de *GASTON* en pourcentage des workflows suivant la mesure du cosinus

FIGURE 10: Répartition du nombre de motifs, suivant la valeur de leur mesure du cosinus, dans les intervalles choisis, en 10a pour l'algorithme *gSpan* tandis qu'en 10b présente celle pour l'algorithme *GASTON*. Les motifs sont obtenus la stratégie égalité des contenu pour un seuil choisi des deux algorithmes de 0.01.

4.3.2.2 Stratégie similarité des contenus Une observation de la figure 11, permet de constater que pour les deux algorithmes, la moitié de leurs motifs ont leur de cosinus dans l'intervalle [0.25,0.5[. Ceci montre que les motifs obtenus dans ce cas de figure sont assez corrélés et que les motifs ont a peu près le même comportement. Néanmoins, aucun des motifs de l'algorithme *gSpan* n'a une valeur de la mesure de cosinus égal à 1 contre 1% dans le cas des motifs obtenus avec *GASTON*. Cependant si on fixe un seuil pour la valeur de la mesure de cosinus à 0.5, on peut nettement voir que les motifs de *gSpan* sont plus corrélés, que ceux de *GASTON*.



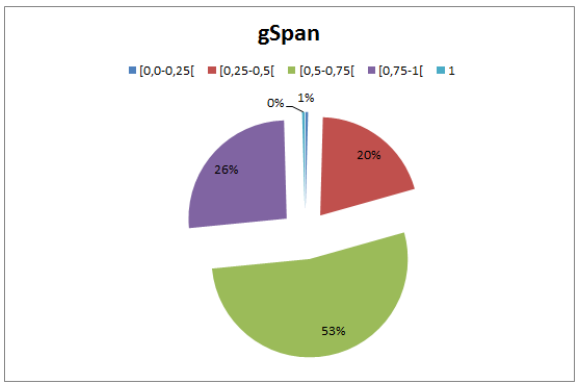
(a) Répartition des motifs de *gSpan* en pourcentage des workflows suivant l'indice de Shannon



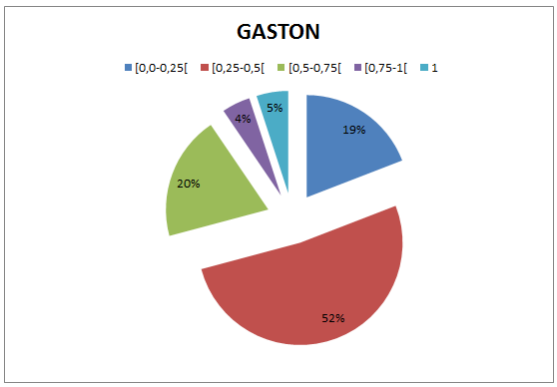
(b) Répartition des motifs de *GASTON* en pourcentage des workflows suivant l'indice de Shannon

FIGURE 11: La figure présente la répartition du nombre de motifs, suivant la valeur de leur mesure du cosinus, dans les intervalles choisis. La figure11a présente la répartition pour l'algorithme *gSpan* tandis que celle en 11b présente celle pour l'algorithme *GASTON*. Les motifs sont obtenus en considérant une similarité sur les contenus des processeurs pour l'étiquetage des graphes des workflows. Le seuil de similarité choisi est de 0.6 pour les script et 0.7 pour les autres types. Le seuil des algorithmes et celui des deux algorithmes est de 0.01.

4.3.2.3 Stratégie similarité de noms En observant la figure 12, on peut remarquer que la majorité (53%) des motifs de *gSpan* ont une valeur de cosinus dans l'intervalle [0.5 ;0.75[ce qui traduit une forte corrélation entre les arcs dans les motifs. C'est le cas contraire avec l'algorithme *GASTON* où la majorité (52%) des motifs ont leur de cosinus dans l'intervalle [0.0-0.25[, ce qui traduit une moins forte corrélation entre les noeuds des arcs des motifs de *GASTON*, bien que pour *GASTON*, on a 13% des motifs qui ont une valeur de cosinus égale à 1 contre 0% pour l'algorithme *gSpan*. Ainsi si on fixait le seuil pour la valeur de la meure de cosinus à 0.5, alors les motifs de gSpan seront plus corrélés que les motifs de *GASTON*.



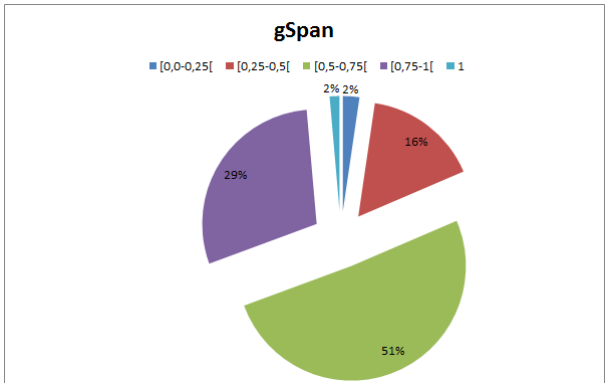
(a) Répartition des motifs de *gSpan* en pourcentage des workflows suivant la mesure du cosinus



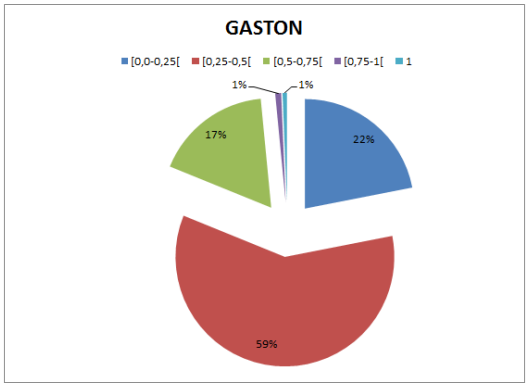
(b) Répartition des motifs de *GASTON* en pourcentage des workflows suivant la mesure du cosinus

FIGURE 12: La figure présente la répartition du nombre de motifs, suivant la valeur de leur mesure du cosinus, dans les intervalles choisis, en12a pour l'algorithme *gSpan* tandis qu'en 12b pour l'algorithme *GASTON*. Les motifs sont obtenus en considérant la stratégie similarité des noms pour un seuil de similarité choisi de 0.7 et celui des deux algorithmes est de 0.01.

4.3.2.4 Cas de l'étiquetage des graphes des workflows en tenant compte de la similarité des noms et contenu des processeurs : En observant la figure 11, on peut remarquer que la moitié (51%) des motifs obtenus avec l'algorithme *gSpan* ont leur valeur de cosinus dans l'intervalle [0.5-0.75[contre seulement 29% pour l'algorithme *GASTON*. Ainsi si on fixait un seuil pour la valeur de cette mesure de cosinus à 0.5, alors les motifs de *gSpan* seraient bien plus corrélés que ceux de l'algorithme *GASTON*.



(a) Répartition des motifs de *gSpan* en pourcentage des workflows suivant l'indice de Shannon



(b) Répartition des motifs de *GASTON* en pourcentage des workflows suivant l'indice de Shannon

FIGURE 13: La figure présente la répartition du nombre de motifs, suivant la valeur de la mesure du cosinus, dans les intervalles choisis, en 13a pour l'algorithme *gSpan* tandis qu'en 13b celle de l'algorithme *GASTON*. Les motifs sont obtenus en considérant la stratégie similarité des noms ou similarité des contenus pour un seuil de similarité choisi de 0.6 pour les script, 0.7 pour les autres types et 0.7 pour les noms des processeurs. Le seuil des algorithmes et celui des deux algorithmes est de 0.01.

Après observation des résultats ci-dessus, obtenus en considérant la mesure du cosinus comme mesure d'intérêt des motifs, on peut conclure que du point de vu corrélation, l'algorithme *gSpan* rend des motifs plus intéressants que l'algorithme *GASTON*.

De manière générale, bien qu'à la section précédente nous avons pu constater que l'algorithme *GASTON* a tendance à renvoyer un plus grand nombre de motifs, l'utilisation des mesures de l'indice de *Shannon* et la mesure du *Cosinus* pour qualifier de l'intérêt des motifs, nous a permis de statuer que les motifs obtenus avec *gSpan* sont plus intéressants que ceux de l'algorithme *GASTON*.

4.4 Analyse de l'intérêt des motifs pour l'utilisateur

Dans cette section nous analysons les patterns obtenus avec l'algorithme *gSpan* en prenant en compte cette fois des critères plus objectifs, relatifs au point de vue de l'utilisateur.

Les différents patterns sont donnés en annexe 1. Deux points nous semblent particulièrement intéressants. D'abord nous avons étudié la présence de motifs extraits à l'intérieur ou autour de sous-workflow. Les sous-workflows sont des workflows ré-utilisés dans le contexte de différents workflows. Ils sont en quelque sorte des motifs à réutiliser et sont déjà proposés aux utilisateurs.

De façon intéressante, certains motifs extraits sont directement des sous-workflows. Mais de façon encore plus intéressante, la plupart des motifs extraits sont plus grands que les sous-workflows : ils montrent donc que plutôt que de ne réutiliser que les sous-workflows exu=istant, d'autres sous-workflows, strictement plus grands pourraient être réutilisés. La figure 14 illustre ce cas. Une hiérarchie de sou-workflows pourrait donc être proposée aux utilisateurs.

Un second point d'intérêt est que beaucoup de motif extraits auraient été complètement ignorés par un utilisateur ou par une stratégie de recherche de workflows basée sur les noms. Certains motifs sont en effet basés sur les noms totalement différents. La figure 18 illustre ce cas. Pouvoir les mettre en évidence, c'est ici montrer qu'une série de processeurs peut être réutilisé dans des contextes totalement différents (exprimés par les renommages complets des processeurs).

4.5 Particularité des workflows scientifiques

Dans cette section, nous nous proposons d'insister sur une caractéristique structurelle des workflows scientifiques qui rend le problème d'extraction de motifs fréquents particulièrement difficile : la présence de la même étiquette à plusieurs reprise dans les workflows.

Cette situation traduit simplement le fait qu'un même outils peut être utilisé plusieurs fois dans un même workflow. Notons que cette situation n'apparaît pas dans les graphes des réseaux sociaux où chaque individu n'apparaît qu'une seule fois ceci est aussi le cas pour les réseaux biologiques où chaque protéines (ou gène) n'a généralement qu'une seule occurrence.

L'objectif de cette section est de monter la forte présence de ces répétions en fournissant des des chiffres concrets. L'étude s'est faite sur les patterns obtenus à partir des algorithmes *gSpan* et *GASTON*. Les résultats obtenus sont présentés respectivement dans les tables 9, 10, 11, 12

Algorithmes	Pourcentage de répétions observé dans les patterns
gSpan 0.02 (26 graphes)	30%
0.01 (13 graphes)	39%
GASTON 0.02 (26 graphes)	25%
0.01 (13 graphes)	35%

TABLE 9: Motifs obtenus avec la stratégie égalité de contenu avec au moins une étiquette répétée

Algorithmes	Pourcentage observé pour les seuils de similarité 0.6 et 0.7	Pourcentage de répétition observé pour les seuils de similarité 0.7 et 0.8
gSpan 0.02 (26 graphes)	60%	52%
0.01 (13 graphes)	70%	67%
GASTON 0.02 (26 graphes)	60%	50%
0.01 (13 graphes)	74%	77%

TABLE 10: Motifs obtenus avec la stratégie similarité des contenus de contenu avec au moins une étiquette répétée. Les seuils de similarité correspondent au seuil choisis pour la distance de levenshtein sur les contenus des processeurs

Algorithmes	Pourcentage de répétitions observé pour le seuil de similarité 0.7	Pourcentage de répétitions observé pour le seuil de similarité 0.8
gSpan 0.02 (26 graphes)	1%	2%
0.01 (13 graphes)	54%	8%
GASTON 0.02 (26 graphes)	17%	22%
0.01 (13 graphes)	54%	30%

TABLE 11: Motifs obtenus avec la stratégie similarité des noms avec au moins une étiquette répétée. Les seuils de similarité correspondent au seuil choisis pour la distance de levenshtein sur les noms des processeurs

Algorithmes	Pourcentage de répétition observé pour les seuils de similarité 0.7, 0.7 et 0.6	Pourcentage répétition observé pour les seuils de similarité 0.7, 0.7 et 0.7
gSpan		
0.02 (26 graphes)	37%	35%
0.01 (13 graphes)	49%	51%
GASTON		
0.02 (26 graphes)	35%	35%
0.01 (13 graphes)	43%	46%

TABLE 12: Motifs obtenus avec la stratégie similarité de contenu ou similarité de noms avec au moins une étiquette répétée. Les seuils de similarité correspondent au seuil choisis pour la distance de levenshtein sur les contenus et les noms des processeurs

Après observation des tableaux précédents, on remarque que pour la majorité des cas, au moins 30% des motifs retournés contiennent des répétitions de processeurs. Ceci est du, comme précisé plus haut, au fait que les workflows eux même contiennent de nombreuses répétitions. Mais aussi ceci est du au fait que, comme les algorithmes font du *"motif growth"*, alors les motifs obtenus ne sont pas maximaux. Mais ceci n'est pas un problème pour nous, car comme notre objectif est de faire de la proposition plus tard, on souhaiterait avoir des motifs de différentes taille. C'est pourquoi on conservent tous les motifs. De manière générale, l'algorithme *gSpan* a plus tendance à retourner des motifs contenant plusieurs fois un même noeud comparé à l'algorithme *GASTON*

5 Conclusion et perspectives

5.1 Conclusion

Ce stage a eu pour objectif d'étudier le problème de l'extraction de motifs fréquents dans les workflows scientifiques dans le but de constituer une librairie de motifs. C'est la première fois que ce travail est effectué sur ce type de données. Nous avons été amenées à considérer une base de près de 2 000 workflows, à mettre en place diverses stratégies d'étiquetages et nous avons choisi deux algorithmes robustes de fouille de graphes, ayant montré leur capacité à gérer des données réelles dans d'autres contextes.

L'étude que nous avons menée nous a permis de mettre en évidence deux points.

D'abord, les workflows scientifiques ont des particularités que l'on ne retrouve pas dans d'autres graphes comme les réseaux sociaux ou les réseaux biologiques. En particulier, on retrouve dans les workflows la présence de multiples occurrences d'un même processeur parce qu'un même outil bio-informatique peut être utilisé plusieurs fois dans un même workflow. Ce cas de figure ne se présente pas dans les réseaux sociaux qui représentent des personnes (chacune présente une unique fois) ou dans des réseaux biologiques. Cette situation est particulièrement forte dans les workflows scientifiques où certains motifs extraits ne contenait que quelques étiquette alors que plus d'une dizaine de noeuds était présents. Dans de telle situation de le test d'isomorphisme est particulièrement

coûteux puisque leur de la mise en correspondance des noeuds de deux graphes un grand nombre d'appariements est possible.

Ensuite, nous avons pu mettre en évidence de façon très intéressante trois points.
Premièrement, les motifs extraits sont peu divers (au sens de l'indice de Shannon). En d'autres termes, certains ensembles d'outils bioinformatiques sont très (ré)utilisés alors que d'autres ensembles ne le sont pas. Cette information va dans le sens d'une autre étude faite sur la réutilisation des processeurs dans les workflows (qui n'a pas considéré les motifs mais uniquement les processeurs, de facon atomique) qui montre que la distribution de réutilisation est telle que certains processeurs sont très réutilisés alors que d'autres le sont beaucoup moins.

Deuxièmement, les motifs obtenus, en particulier par gSPAN, sont assez fortement corrélés (mesure de cosinus). Cette information va dans le sens de notre objectif : utiliser ces motifs pour faire de la recommandation. En d'autres termes si l'utilisateur en cours de conception de workflow utilise une série d'outils qui apparaît dans un motif déjà extrait, nous pourrions lui recommander de compléter son workflow avec "la suite" du motif.

Troisièmement, nous avons mis en évidence que les motifs extraits correspondaient non seulement souvent à des sous-workflows (c'est-à-dire à des workflows déjà disponibles pour les utilisateurs et déjà réutilisés dans le cadre de workflows plus importants) mais surtout à des sur-ensembles de sous-workflows. En d'autres termes, alors que de relativement petits workflows sont proposés aux utilisateurs aujourd'hui par les systèmes de workflows, de plus grands workflows pourraient être proposés. on pourrait même envisager de proposer une hiérarchie de workflows à réutiliser.

5.2 Perspectives

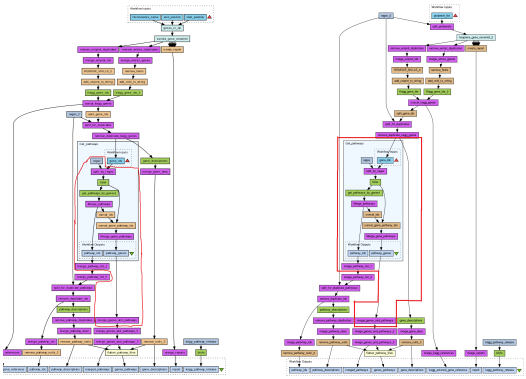
De ce stage découlent de nombreuses perspectives et pistes de recherche. A court terme, il faudrait étendre notre étude comparative pour considérer un nombre plus important d'algorithmes de fouille de graphes, de nouvelles mesures de qualités voire des workflows issus d'autres systèmes de workflows scientifiques. La démarche que nous avons suivie permet de pouvoir rapidement effectuer cette extension. Des résultats additionnels seront d'ailleurs disponibles dès la soutenance (évaluation complète de SUBDUE) mais il n'a pas été possible d'obtenir plus de résultats en 4 mois.

À moyen terme, une étude plus poussée des mesures de qualité nous semble un point très intéressant à développer. Il s'agirait d'évaluer plus généralement la capacité des mesures de qualité disponibles à rendre compte de la qualité des motifs extraits en terme de "potentiel" à être réutilisés, en terme d'intérêt du point de vue de l'utilisateur. D'autres données pourraient alors être exploitées comme la note associée par les utilisateurs à certains processeurs ou à certains workflows ou encore les statistiques associés aux précédentes exécutions faites d'un workflow ou d'un processeur.

Un autre point intéressant serait d'étudier de façon plus précise les motifs extraits et d'analyser en particulier s'il serait possible d'optimiser ces motifs dans le but de les rendre plus performant et/ou plus facile à maintenir. Ce travail pourrait se faire en collaboration avec des bioinformaticiens.

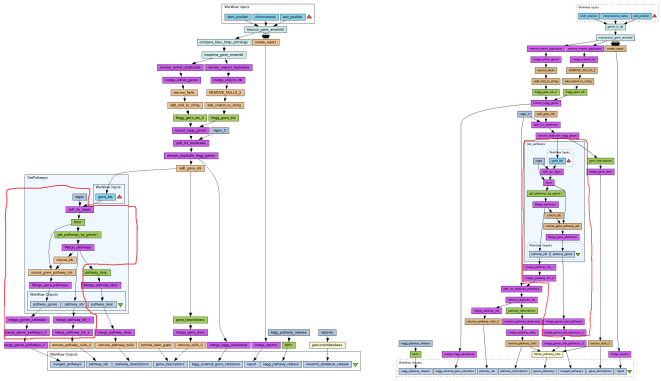
À plus long terme, la proposition d'un mécanisme complet de recommandation à la volée, lors de la conception de workflows répondrait à un besoin fort. Enfin, et de façon orthogonale, la conception d'un algorithme rapide et capable de fournir des motifs intéressants fait partie des défis à relever dans ce contexte.

6 Annexe : exemples de motifs extraits



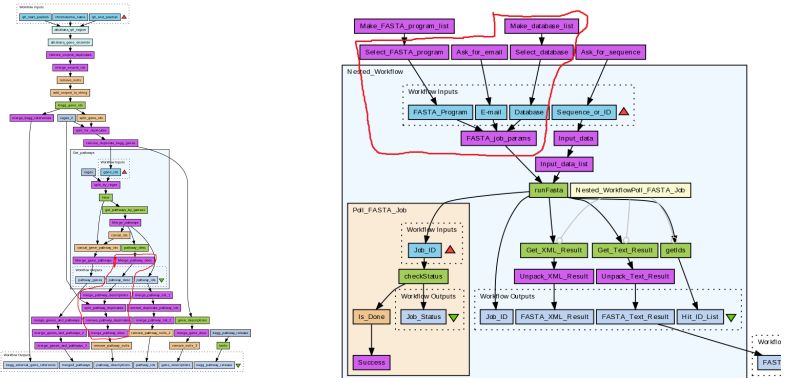
(a) Workflow de myExperiment 931 v2 (b) Workflow de myExperiment 1977 v1

FIGURE 14: *Exemple de motifs obtenus.* La partie surlignée en rouge présente des motifs obtenus en étiquetant les graphes en se basant sur la similarité des noms des processeurs. Ici le motif correspond à l'utilisation d'un seuil de 0.8 pour la distance de levenshtein et un seuil à 0.01 pour gSpan.



(a) Workflow de myExperiment 14 v2 (b) Workflow de myExperiment 16 v7

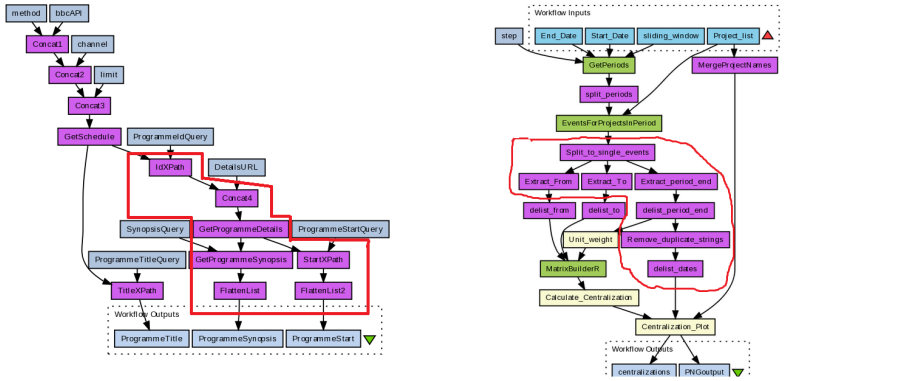
FIGURE 15: *Exemple de motifs obtenus.*Les parties surlignées en rouge présentent des occurences d'un motif obtenu en étiquetant les graphes en se basant sur la similarité des noms des processeurs. Ici le motif correspond à l'utilisation d'un seuil de 0.7 pour la distance de levenshtein et un seuil à 0.01 pour gSpan.



(a) Workflow de myExperiment 62 v1

(b) Workflow de myExperiment 200 v3

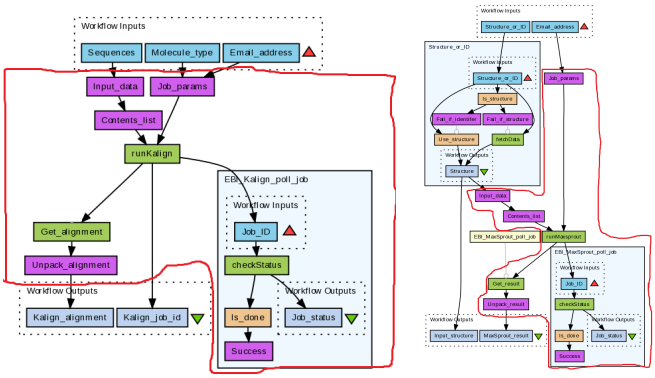
FIGURE 16: *Exemple de motifs obtenus.* Les parties surlignées en rouge présentent des occurrences d'un motif obtenu en étiquetant les graphes en se basant sur l'égalité des contenus des processeurs et de leur type des noms des processeurs. Ici le motif correspond à l'utilisation d'un seuil de 0.02 pour gSpan.



(a) Workflow de myExperiment 384 v1

(b) Workflow de myExperiment 170 v2

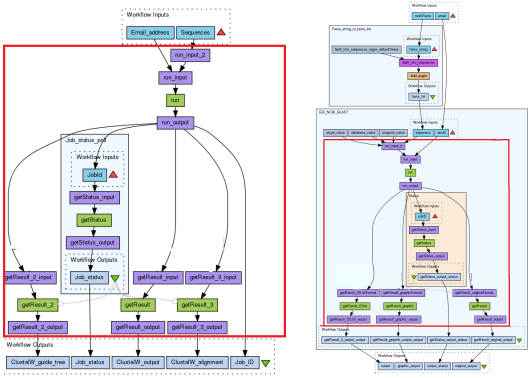
FIGURE 17: *Exemple de motifs obtenus.* Les parties surlignées en rouge présentent des occurrences d'un motif obtenu en étiquetant les graphes en se basant sur l'égalité des contenus des processeurs et de leur types. Ici le motif correspond à l'utilisation d'un seuil de 0.01 pour gSpan.



(a) Workflow de myExperiment 217 v1

(b) Workflow de myExperiment 224 v2

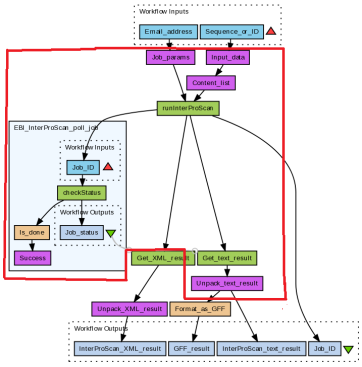
FIGURE 18: *Exemple de motifs obtenus.*Les partie surlignées en rouge présentent des occurrences d'un motif obtenu en étiquetant les graphes en se basant sur la similarité entre les contenus des processeurs et l'égalité de leur types. Ici le motif correspond à l'utilisation d'un seuil de 0.02 pour gSpan, un seuil à 0.6 pour les scripts et 0.7 pour les autres types sur la distance d'édition.



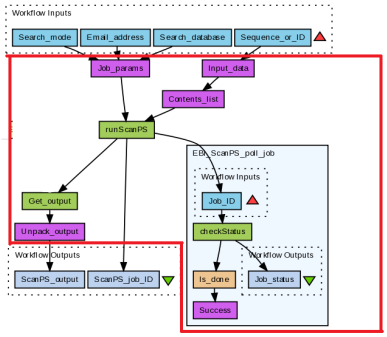
(a) Workflow de myExperiment 1705 v1

(b) Workflow de myExperiment 2258 v2

FIGURE 19: *Exemple de motifs obtenus.*La partie surlignée en rouge présente des occurrences d'un motif obtenu en étiquetant les graphes en se basant sur la similarité entre les contenus des processeurs et l'égalité de leur type des noms des processeurs. Ici le motif correspond à l'utilisation d'un seuil de 0.01 pour gSpan, un seuil à 0.6 pour les scripts et 0.7 pour les autres types sur la distance d'édition.

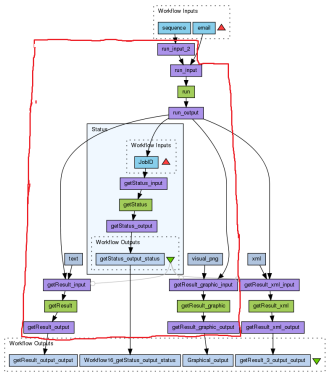


(a) Workflow de myExperiment 204 v1

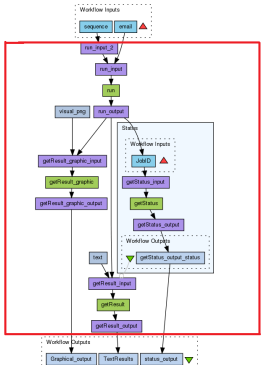


(b) Workflow de myExperiment 220 v1

FIGURE 20: *Exemple de motifs obtenus.*Les parties surlignées en rouge présentent deux occurrences d'un motif obtenu en étiquetant les graphes en se basant sur la similarité entre les contenus des processeurs et l'égalité de leur type des noms des processeurs. Ici le motif correspond à l'utilisation d'un seuil de 0.02 pour gSpan, un seuil à 0.7 pour les scripts et 0.8 pour les autres types sur la distance d'édition.

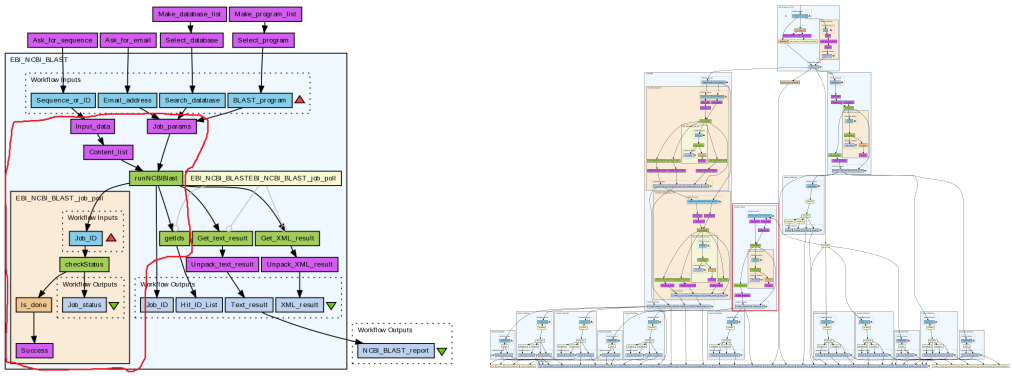


(a) Workflow de myExperiment 1767 v2



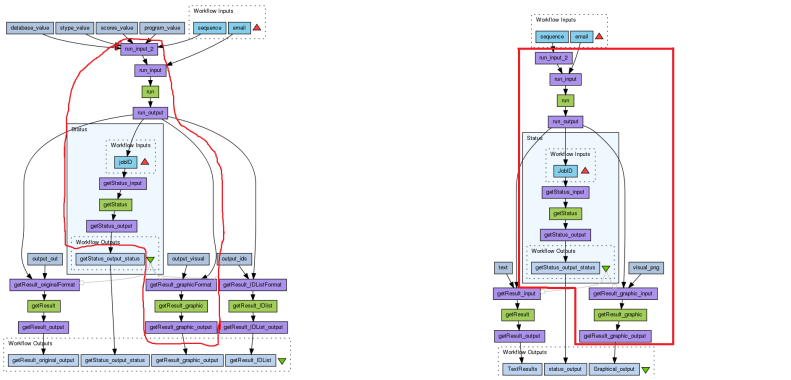
(b) Workflow de myExperiment 1953 v1

FIGURE 21: *Exemple de motifs obtenus.* Les parties surlignées en rouge présentent deux occurrences d'un motif obtenu en étiquetant les graphes en se basant sur la similarité entre les contenus des processeurs et l'égalité de leur type des noms des processeurs. Ici le motif correspond à l'utilisation d'un seuil de 0.01 pour gSpan, un seuil à 0.7 pour les scripts et 0.8 pour les autres types sur la distance d'édition.



(a) Workflow de myExperiment 202 v1 (b) Workflow de myExperiment 939 v2

FIGURE 22: *Exemple de motifs obtenus.* Les parties surlignées en rouge présentent deux occurrences d'un motif obtenu étiquetant les graphes en se basant sur le nom des processus lorsque leur contenu est indisponible, sur similarité entre les contenus (si disponible) et l'égalité de leur type. Ici le motif correspond à l'utilisation d'un seuil de 0.02 pour gSpan, un seuil à 0.6 pour les scripts 0.7 pour les autres types et 0.7 pour les noms sur la distance d'édition. On trouve le même résultat en mettant le seuil à 0.7 sur la similarité pour les scripts.



(a) Workflow de myExperiment 1765 v1 (b) Workflow de myExperiment 3374 v3

FIGURE 23: *Exemples de motifs.* Les parties surlignées en rouge présentent les occurrences d'un motif obtenu en étiquetant les graphes en se basant sur le nom des processus lorsque leur contenu est indisponible, sur similarité entre les contenus (si disponible) et l'égalité de leur type. Ici le motif correspond à l'utilisation d'un seuil de 0.01 pour gSpan, un seuil à 0.6 pour les scripts 0.7 pour les autres types et 0.7 pour les noms sur la distance d'édition.On trouve le même résultats en mettant le seuil à 0.7 sur la similarité pour les scripts.

Références

[1] Charu C Aggarwal and Philip S Yu. A new framework for itemset generation. In *Proceedings of the seventeenth ACM SIGACT-SIGMOD-SIGART symposium on Principles of database systems*, pages 18–24. ACM, 1998.

[2] Rakesh Agrawal, Tomasz Imieliński, and Arun Swami. Mining association rules between sets of items in large databases. In *ACM SIGMOD Record*, volume 22, pages 207–216. ACM, 1993.

[3] Pinar Alper, Khalid Belhajjame, Carole Goble, and Pinar Karagoz. Small is beautiful : Summarizing scientific workflows using semantic annotations. In *Big Data (BigData Congress), 2013 IEEE International Congress on*, pages 318–325. IEEE, 2013.

[4] Ilkay Altintas, Chad Berkley, Efrat Jaeger, Matthew Jones, Bertram Ludascher, and Steve Mock. Kepler : an extensible system for design and execution of scientific workflows. In *Scientific and Statistical Database Management, 2004. Proceedings. 16th International Conference on*, pages 423–424. IEEE, 2004.

[5] Sergey Brin, Rajeev Motwani, and Craig Silverstein. Beyond market baskets : Generalizing association rules to correlations. In *ACM SIGMOD Record*, volume 26, pages 265–276. ACM, 1997.

[6] Sarah Cohen-Boulakia, Jiuqiang Chen, Paolo Missier, Carole Goble, Alan R Williams, and Christine Froidevaux. Distilling structure in taverna scientific workflows : a refactoring approach. *BMC Bioinformatics*, 15(Suppl 1) :S12, 2014.

[7] Diane J Cook and Lawrence B Holder. *Mining graph data*. John Wiley & Sons, 2006.

[8] William B Frakes and Ricardo Baeza-Yates. Information retrieval : data structures and algorithms. 1992.

[9] Daniel Garijo, Pinar Alper, Khalid Belhajjame, Oscar Corcho, Yolanda Gil, and Carole Goble. Common motifs in scientific workflows : An empirical analysis. *Future Generation Computer Systems*, 2013.

[10] Liqiang Geng and Howard J Hamilton. Interestingness measures for data mining : A survey. *ACM Computing Surveys (CSUR)*, 38(3) :9, 2006.

[11] Lawrence B Holder, Diane J Cook, Surnjani Djoko, et al. Substucture discovery in the subdue system. In *KDD workshop*, pages 169–180, 1994.

[12] Duncan Hull, Katy Wolstencroft, Robert Stevens, Carole Goble, Mathew R Pocock, Peter Li, and Tom Oinn. Taverna : a tool for building and running workflows of services. *Nucleic acids research*, 34(suppl 2) :W729–W732, 2006.

[13] Bertram Ludäscher, Mathias Weske, Timothy McPhillips, and Shawn Bowers. Scientific workflows : Business as usual ? In *Business Process Management*, pages 31–47. Springer, 2009.

[14] Siegfried Nijssen and Joost N Kok. A quickstart in frequent structure mining can make a difference. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 647–652. ACM, 2004.

[15] Maíra R Rodrigues, Wagner CS Magalhães, Moara Machado, and Eduardo Tarazona-Santos. A graph-based approach for designing extensible pipelines. *BMC bioinformatics*, 13(1) :163, 2012.

[16] Marco Roos. Bioaid : Disease discovery workflow.

[17] Edward H Shortliffe and Bruce G Buchanan. A model of inexact reasoning in medicine. *Mathematical biosciences*, 23(3) :351–379, 1975.

[18] Pang-Ning Tan, Vipin Kumar, and Jaideep Srivastava. Selecting the right interestingness measure for association patterns. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 32–41. ACM, 2002.

[19] Ian Taylor, Matthew Shields, Ian Wang, and Andrew Harrison. The triana workflow environment : Architecture and applications. In *Workflows for e-Science*, pages 320–339. Springer, 2007.

[20] Takashi Washio and Hiroshi Motoda. State of the art of graph-based data mining. *Acm Sigkdd Explorations Newsletter*, 5(1) :59–68, 2003.

[21] Xifeng Yan and Jiawei Han. gspan : Graph-based substructure pattern mining. In *Data Mining, 2002. ICDM 2003. Proceedings. 2002 IEEE International Conference on*, pages 721–724. IEEE, 2002.