ÉCOLE POLYTECHNIQUE DE L'UNIVERSITÉ DE NANTES
DÉPARTEMENT D'INFORMATIQUE

RAPPORT DE RECHERCHE ET DÉVELOPPEMENT

Création de profils utilisateurs

Damien LE FLOHIC & Cyril CARON

février 2017

encadré par Antoine PIGEAU & Yannick PRIÉ

— Équipe DUKE —
LABORATOIRE D'INFORMATIQUE DE NANTES-ATLANTIQUE

coordinateur : Philippe LERAY

Création de profils utilisateurs

Damien LE FLOHIC & Cyril CARON

Résumé

Pour ce projet de recherche et développement, nous avons choisi le projet qui consiste à analyser des traces utilisateurs du site OpenClassrooms grâce à des techniques de process mining. La problématique principale est de construire des modèles à partir de ces traces, capables de prédire par la suite si un utilisateur va réussir ou échouer à un cours proposé par OpenClassrooms. Cette prédiction permettra alors au site de proposer un moyen à l'utilisateur de favoriser sa réussite, en proposant un système de tutorat par exemple. Pour créer ces modèles, nous allons étudier plusieurs algorithmes de process mining et choisir le plus adapté à notre problème. Une fois que ces modèles seront créés, nous réaliserons un outil permettant de comparer une trace utilisateur à ces modèles pour prédire la réussite ou non à un cours. Pour mener à bien ce projet, nous nous sommes aidés essentiellement d'un livre pour nos recherches, Process Mining Discovery, Conformance and Enhancement of Business Processes, de Wil M.P. van der Aalst, ainsi que de deux autres articles complémentaires pour nous aider à choisir l'algorithme le mieux adapté à notre problématique. Après recherches, nous avons décidé d'utiliser l'heuristic miner pour créer les modèles de réussite et d'échec à un cours (3). Les limites de ce travail sont dues au fait qu'il existe de nombreuses solutions possibles, mais qui nécessiteraient plus de connaissances techniques dans le process mining, et qui ne sont pas forcément toutes expliquées dans le livre, qui reste l'ouvrage de référence dans ce domaine. Néanmoins, ce travail a permis une bonne compréhension globale du process mining et de ses différentes techniques, et la présentation des différents algorithmes permettra potentiellement d'appliquer ces techniques à d'autres traces sans avoir à faire des recherches supplémentaires sur le sujet pour trouver un autre algorithme adapté, même s'ils ne sont pas tous présents dans le livre.

Remerciements

Nous tenons à remercier Antoine PIGEAU pour les informations qu'il nous a apportées, les éléments bibliographiques qu'il nous a fournis, mais aussi pour le temps qu'il nous a accordé pour les explications et les démonstrations des outils. Son aide a également été la bienvenue lorsque nous faisions face à de nombreux problèmes. Nous le remercions également pour les conseils qu'il nous a donnés dans la rédaction de notre rapport.

Table des matières

1 Introduction

La formation en ligne, ou e-learning, représente l'ensemble des solutions qui permettent un apprentissage par le biais de moyens électroniques, surtout internet, grâce à des plateformes d'apprentissage en ligne par exemple. Ce marché en pleine expansion permet de faciliter l'apprentissage en facilitant l'accès à des ressources, tout en n'ayant pas nécessairement besoin d'un professeur à proximité pour donner des explications. L'e-learning permet de donner de l'autonomie aux personnes durant leur apprentissage, mais aussi de faciliter l'apprentissage grâce à la rapidité d'accès, car une connexion internet suffit pour pouvoir suivre un cours, qu'il soit gratuit ou payant. Aujourd'hui il existe de nombreux domaines concernés par l'e-learning. Cette approche de la formation a un aspect plus ludique que certains cours traditionnels, ce qui explique le développement rapide de ce secteur depuis une quinzaine d'années.

Aujourd'hui, il est possible de faire évoluer encore plus l'e-learning en proposant des cours et des solutions qui sont de plus en plus adaptés à chaque personne, de manière individuelle. Il s'agit d'améliorer au mieux l'apprentissage en s'adaptant au rythme et aux envies de chacun dans le but que les cours suivis correspondent aux attentes des utilisateurs.

1.1 Présentation de la problématique

OpenClassrooms est une plateforme en ligne proposant de nombreux cours en ligne, des « Massive Open Online Course » (MOOC) dans plusieurs domaines. Ces cours sont gratuits mais il est également possible de payer pour avoir accès à un service de tutorat plus complet.

La lecture de ces cours génère de nombreuses traces, ainsi il est possible de connaître les sections qui ont été lues et dans quel ordre, le temps passé sur telle ou telle section, etc.

L'analyse de ces traces est donc intéressante, celles-ci pourraient permettre de prédire les utilisateurs étant susceptibles d'échouer à un cours.

1.2 Objectifs poursuivis

Pour cette étude, nous avons défini plusieurs objectifs afin de répondre à la problématique.

Tout d'abord, nous allons sélectionner l'algorithme le plus adapté à nos besoins pour cette problématique parmi une liste d'algorithmes de process mining.

Une fois que l'algorithme sera choisi, nous l'utiliserons pour créer un outil permettant de détecter l'échec d'un utilisateur à un cours auquel il sera inscrit à partir des traces utilisateurs qui nous ont été fournies.

Ce résultat permettra par la suite à Openclassrooms de proposer à un utilisateur qui semble en train de décrocher d'un cours des services personnalisés, comme un tutorat par exemple.

1.3 Travail réalisé

Nous avons terminé la partie étude bibliographique qui consiste à étudier des algorithmes de process mining afin de choisir le plus adapté à nos traces d'utilisateurs. Nous avons également étudié la partie qui consistera pour la suite de ce projet à comparer les traces avec le modèle que nous créerons grâce à l'algorithme. Dans la seconde partie du projet, suite à l'étude bibliographique, nous avons réalisé le développement nécessaire pour créer un outil qui permet de générer des modèles à partir de fichiers de logs. Celui-ci comprend une partie qui applique l'algorithme que nous aurons choisi, pour générer les différents modèles, puis une partie Conformance Checking, qui joue d'autres logs sur le modèle pour vérifier leur adéquation. En revanche, nous avons eu des problèmes avec le Conformance Checking pour vérifier le résultat de la conformance des traces.

1.4 Plan de l'étude

Le chapitre 2 fait état de nos recherches sur les algorithmes de process mining, avec pour chacun d'eux leurs avantages et inconvénients

Le chapitre 3 montre la proposition que nous avons faite quant au choix de l'algorithme le plus adapté pour créer notre modèle à partir de traces d'utilisateurs.

Le chapitre 4 montre le travail de développement qui a été réalisé suite à l'étude bibliographique et les propositions. Il contient également une partie qui présente les besoins de notre programme ainsi que les principales étapes de son déroulement. Enfin, une dernière sous partie montre les résultats de notre travail.

Enfin, une conclusion 5 est présente afin de faire un bilan de notre travail effectué, nos décisions prises, les problèmes rencontrés et les solutions potentielles pour résoudre le problème.

2

État de l'art

Ce chapitre présente la notion de process mining et les différents types existant. Les différents algorithmes qu'il est possible de trouver dans la littérature sont également étudiés et analysés afin d'en retirer les intérêts et les limites de chacun d'entre eux. Pour finir, un récapitulatif des différentes solutions sera présenté.

Il est important de noter que les informations présentes dans cet état de l'art proviennent principalement du livre [vdA11].

2.1 Le process mining

Le process mining est une technique de gestion de processus qui permet d'analyser des processus, comme ceux d'entreprises par exemple, à partir de journaux d'évènements. Il a pour but d'améliorer la compréhension et l'efficacité de ces processus.

L'extraction de connaissances à partir des données (*data mining*) est un domaine de l'informatique décisionnelle qui a pour objectif d'extraire de la connaissance à partir des données à l'aide d'algorithmes d'apprentissage (supervisé ou non). Ces algorithmes permettent de trouver des motifs et les structures cachées au sein des données, de réaliser des prédictions ou encore de discriminer.

Le process mining est semblable dans le sens où des algorithmes spéciaux de data mining peuvent être utilisés pour identifier des tendances, des modèles à partir de données, souvent sous la forme de journaux d'évènements.

2.1.1 Vocabulaire

Cette sous-section aborde les différents concepts qui seront abordés dans ce rapport ainsi que les différents acronymes utilisés.

— Un processus (*process* en anglais) est une suite d'activités (*activity*). Ces activités peuvent être réalisées séquentiellement ou en parallèle. La présence de boucles est également possible, ce qui permet de répéter une activité plusieurs fois de suite (boucle de longueur 1) ou de revenir à N activités

— Un cas (*case*) est une instance d'un processus, c'est-à-dire une réalisation de celui-ci.
— Un événement (*event*) est une activité qui a été réalisée par un « cas ». Plusieurs attributs autres que l'activité et le « cas » peuvent être associés à un événement comme par exemple un horodatage (*timestamp)* ou la ressource (*resource*) qui a exécuté l'activité.
— Un journal d'événements (*event logs*) est un fichier contenant des enregistrement d'événements ordonnés dans le temps.
— Une trace correspond à la séquence d'activités réalisées par un utilisateur, e.g. (a, b, c, f, h, j).
— Rappel : il s'agit d'une mesure. Elle permet de mesurer le pourcentage d'événements rejouable par un modèle.
— Le F-score est une mesure qui permet d'avoir un équilibre entre le rappel et la précision. Elle peut servir à évaluer un modèle généré par un algorithme de process mining.

2.2 Les différents types de process mining

Il existe trois objectifs inhérents au process mining : *Discovery*, *Conformance* et *Enhancement*. Il y a un élément commun à chacun des types de process mining : un journal d'événements.

2.2.1 Discovery

Cette technique de process mining utilise un journal d'évènements afin de produire un modèle sans connaissances a-priori, c'est à dire des données qui ne sont pas utilisées dans l'apprentissage du modèle. Le modèle peut être par exemple représenté sous la forme d'un réseau de Pétri, dépendant des informations contenues dans ces logs.

2.2.2 Conformance

Le second type de process mining, appelé conformance, peut être utilisé afin de comparer un évènement avec le modèle créé par un algorithme de type *discovery*. Cet évènement doit être du même type que ceux utilisés pour construire le modèle au préalable. Cette technique peut donc être utilisée pour détecter des « déviations » du log par rapport au modèle, c'est-à-dire une différence entre le modèle et l'événement. Grâce à un algorithme de conformance checking, il est aussi possible de mesurer ces déviations pour en donner l'intensité.

Dans le cadre de ce projet, après avoir utilisé un algorithme pour créer un modèle à partir d'un journal d'événements, nous allons utiliser un algorithme de *conformance checking* afin de confronter le modèle à un journal d'événements contenant les traces d'utilisateurs d'OpenClassrooms afin de prédire si ces utilisateurs vont échouer ou non à un cours.

2.2.3 Enhancement

Le troisième type de process mining, appelé « enhancement » (renforcement) a pour but d'améliorer un modèle déjà existant à partir de données nouvelles. Il existe plusieurs techniques de renforcement d'un modèle :

— Repair (Réparation) : Il s'agit de modifier le modèle pour qu'il soit plus représentatif de la réalité.
— Extension : Il s'agit de faire de la corrélation croisée entre le modèle et les logs afin d'ajouter une nouvelle perspective au modèle, par exemple une dimension sur les ressources d'une activité, une fréquence grâce au temps, etc.

2.3 Les critères de qualité d'un algorithme de process mining

Dans cette section, nous allons aborder les différents critères utilisés pour mesurer la qualité d'un algorithme de process mining.

2.3.1 Fitness

La « fitness » permet de mesurer la capacité d'un algorithme de process mining à pouvoir rejouer les événements d'un fichier de logs. Ainsi, la précision sera égale à 1 si toutes les traces présentes en entrée de l'algorithme sont rejouables.

2.3.2 Généralisation

La capacité d'un algorithme à généraliser est également un critère de qualité. Un algorithme qui produit un modèle qui ne peut rejouer que les traces présentes dans le journal n'est pas forcément un bon algorithme. En effet, certaines traces peuvent survenir bien qu'elles ne soient pas présentes dans les logs. Ainsi, un modèle qui généralise bien permet d'accepter certaines traces non présentes dans les logs. Il faut cependant éviter de trop généraliser car on serait alors dans une problématique d'« underfitting », c'est-à-dire que le modèle accepte n'importe quelle trace. L'algorithme Flower Miner produit de tels modèles (section 2.7).

2.3.3 Précision

La précision, contrairement à la généralisation, évalue la capacité d'un modèle à ne pas accepter trop de traces. Cependant, si le modèle devient trop précis, on se retrouve dans une situation d'« overfitting », c'est-à-dire qu'il n'accepte que les traces présentes dans les logs, et ne permet donc pas d'accepter certaines traces qui peuvent se produire mais qui n'étaient pas présentes dans le fichier de logs utilisé en entrée de l'algorithme.

2.3.4 Simplicité

La simplicité est également un critère de qualité d'un modèle. Il est en effet plus intéressant d'avoir un modèle avec peu de places et de transitions mais qui produit les mêmes résultats qu'un modèle plus complexe.

2.3.5 Limites de ces critères

Ces critères ne sont généralement pas maximisables en même temps. Par exemple, la généralisation et la précision sont opposées, le premier critère cherchant à étendre l'ensemble des traces pouvant être rejouées par le modèle tandis que le second cherche à limiter ce nombre à l'ensemble des traces présentes dans le fichier de logs.

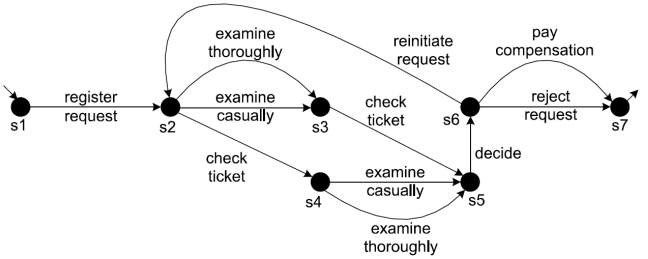

FIGURE 2.1 – Exemple de système de transition ([vdA11], p. 32)

2.4 Les modélisations de processus

Cette section présente différentes modélisations de processus (« process model ») qui peuvent être générées par des algorithmes de process mining. « L'objectif d'une modélisation de processus est de décider quelles activités ont besoin d'être exécutées et dans quel ordre » ([vdA11], p.31).

2.4.1 Systèmes de transition

Cette première modélisation se base sur un système d'états et de transitions. Les états sont représentés par des ronds noirs et les transitions par des flèches reliant un état à un autre. Pour changer d'état il est nécessaire d'activer une certaine transition. La figure 2.1 montre un exemple de système de transition. Ainsi pour passer de l'état initial s1 à l'état s2, il faut exécuter la transition « register request ».

2.4.2 Petri net

Les réseaux de Pétri sont constitués de places, représentées par des ronds, et de transitions qui sont représentées par des rectangles. Les transitions correspondent aux activités dans les journaux d'événements. Un système de jeton (« *token* » en anglais) permet d'activer ou non une transition. Initialement, il n'y a qu'une place qui est marquée par un jeton, la place initiale.

Pour activer une transition, il faut que toutes les places reliées à l'activité par des dépendances entrantes soient marquées. L'activation d'une activité consomme les jetons des places en entrée. Toutes les places sortantes de l'activité se voient attribuer un jeton lorsque cette activité est réalisée.

En revanche, il ne peut y avoir qu'une seule transition activée par une place à la fois (OU exclusif).

2.5 Les propriétés des algorithmes de process mining

Dans cette section, nous allons aborder les différentes propriétés auxquelles sont confrontés les modèles générés par les algorithmes de process mining.

2.5.1 Les boucles

Des boucles peuvent être présentes dans les traces. Il existe des boucles de longueurs différentes, les boucles de longueur 1 et 2 sont des boucles qui peuvent poser problèmes.

Une trace contient une boucle de longueur 1 lorsqu'une activité est réalisée plusieurs fois d'affilé, e.g. $\langle a, b, b, b, b, c\rangle$. Le modèle que l'on souhaite obtenir à partir de cette trace est celui représenté sur la figure 2.2.

Dans le cas où il y a une alternance entre deux activités dans une trace, il s'agit d'une boucle de longueur 2, e.g. $\langle a, b, c, b, c, b, d\rangle$. Le modèle que l'on souhaite obtenir à partir de cette trace est celui représenté sur la figure 2.3.

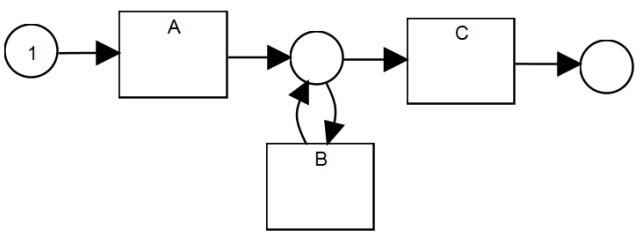

FIGURE 2.2 – Modèle désiré pour la trace $\langle a, b, b, b, b, c\rangle$

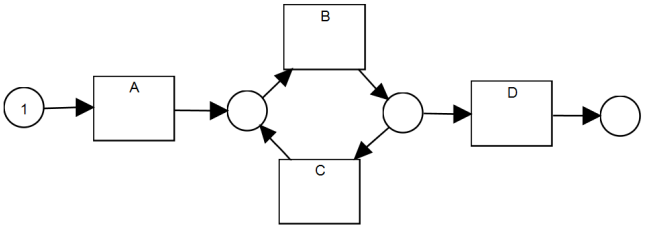

FIGURE 2.3 – Modèle désiré pour la trace $\langle a, b, c, b, c, b, d\rangle$

2.5.2 Les places implicites

Un modèle peut être confronté à des places qui ont été créées mais qui sont pourtant inutiles et devraient être supprimées pour rendre le modèle plus simple.

L'algorithme Alpha (2.6) construit le modèle présent sur la figure 2.4 à partir du journal d'événements suivant :

$$L = [\langle a, c, e, g\rangle^2, \langle a, e, c, g\rangle^3, \langle b, d, f, g\rangle^4, \langle b, f, d, g\rangle^4]$$

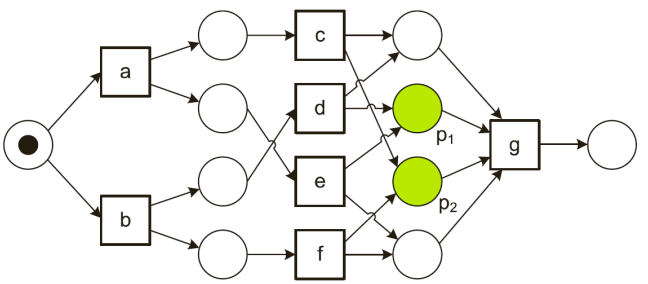

FIGURE 2.4 – Modèle généré grâce à l'algorithme Alpha ([vdA11], p. 137)

Les places *p1* et *p2* ne sont pas nécessaires, en les supprimant le modèle autoriserait les mêmes traces tout en étant plus simple.

2.5.3 Les dépendances non locales

Toutes les techniques de découverte de processus (« process discovery ») ne sont pas capables de construire des dépendances non locales. Ainsi le modèle généré par l'algorithme Alpha à partir du journal d'événements $L = [\langle a, c, d\rangle^{45}, \langle b, c, e\rangle^{42}]$, est celui représenté par la figure 2.5. Ce modèle accepte les traces $\langle a, c, e\rangle$ et $\langle b, c, d\rangle$, alors qu'elles n'étaient pourtant pas présentes dans le journal d'événements. Le modèle idéal est représenté par la figure 2.6. Les places *p1* et *p2* permettent de limiter les traces possibles à celles présentes dans le journal d'événements.

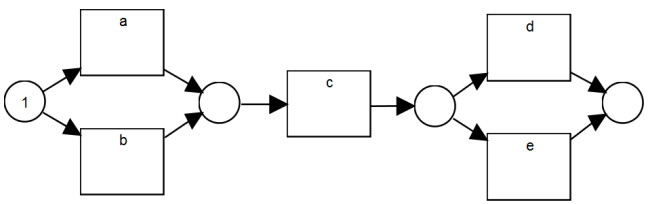

FIGURE 2.5 – Modèle généré grâce à l'algorithme Alpha ne détectant pas les dépendances locales

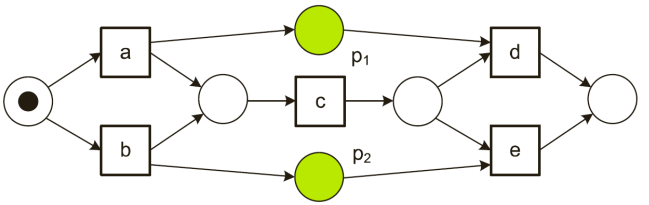

FIGURE 2.6 – Modèle idéal détectant les dépendances non locales ([vdA11], p. 139)

2.5.4 Biais de représentation

Selon la modélisation choisie et l'algorithme utilisé, il est possible de rencontrer des biais de représentation. Par exemple, un journal d'événements $L = [\langle a, a\rangle]$ devrait produire dans l'idéal le modèle présenté par la figure 2.7. En effet, ce modèle possède deux activités *a* qui se suivent afin de décrire la trace présente dans le journal.

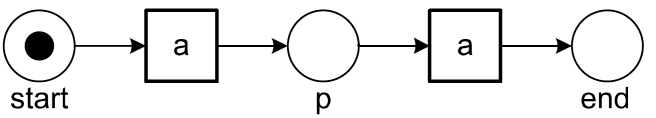

FIGURE 2.7 – Modèle idéal décrivant le journal d'événement ([vdA11], p. 145)

Certains modèles comme l'algorithme Alpha 2.6 ne sont pas capables de reproduire un tel comportement dans les modèles qu'ils produisent.

Par ailleurs, on peut noter qu'un modèle avec une boucle sur l'activité *a* permettrait de reproduire cette

trace (2.8), cependant elle autoriserait également d'autres traces, e.g. $\langle a, a, a, a, a \rangle$.

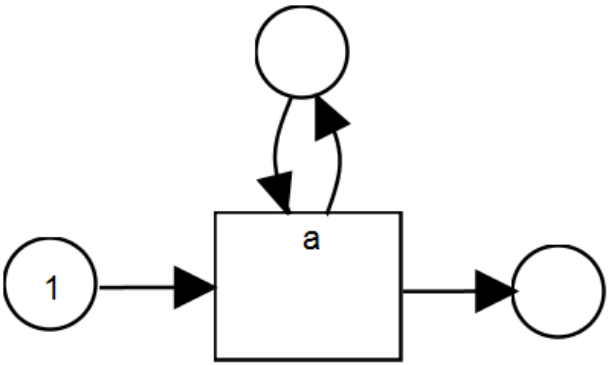

FIGURE 2.8 – Modèle avec une boucle sur l'activité *a*

2.5.5 La duplication d'activités

Certains algorithmes n'ont pas la capacité de pouvoir générer des modèles ayant deux activités avec le même label. Ainsi, une même activité présente plusieurs fois au cours d'un processus mais à différentes étapes sera représentée une seule fois dans le modèle généré par la plupart des algorithmes de *process discovery* (« In many notations, there cannot be two activities having the same label. » [1]).

1. Dans beaucoup de notations, il ne peut pas y avoir deux activités ayant le même label ([vdA11], p. 160).

2.5.6 Les OR-splits et les OR-joins

La plupart des représentations de modèles (Réseau de Pétri, etc.) admettent des jointures et des séparations de type « ET » (*AND-splits* et *AND-joins*) ou de type « OU exclusif » (*XOR-splits* et *XOR-joins*). Cette propriété a une conséquence sur les modèles générés : « If the representational bias of a discovery algorithm does not allow for OR-splits and OR-joins, then the discovered model may be more complex or the algorithm is unable to find a suitable model. » [2]

2.5.7 Les choix non libres

Les constructions de type « choix non libres » (*non-free choice*) désignent les constructions qui mélangent la concurrence (AND-splits) et le choix (XOR-splits) ([vdA11], p. 160). Sur la figure 2.9, nous pouvons constater que les places p1 et p2 créent de la concurrence (les activités b et c peuvent être réalisées simultanément) mais également un choix (il est possible de réaliser l'activité e ou alors, les activités b et c).

2. Si le biais de représentation d'un algorithme de *discovery* n'autorise pas les séparations ou des jointures de type « OU inclusif », alors le modèle découvert pourrait être complexe ou alors l'algorithme n'est pas capable de trouver un modèle adapté. ([vdA11], p. 160).

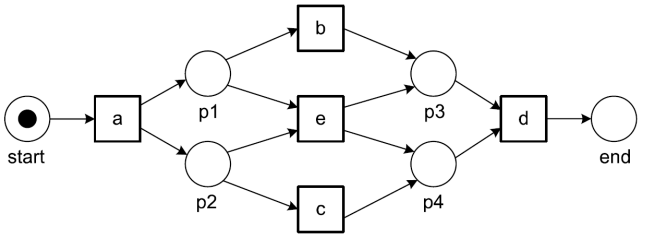

FIGURE 2.9 – Réseau de Pétri présentant une construction de type « choix non libres » ([vdA11], p. 126

2.6 L'algorithme Alpha

Le premier algorithme que nous allons étudier est l'algorithme Alpha. Il fait partie des premiers algorithmes spécialisés dans la découverte de processus et a été repris par la suite dans d'autres algorithmes plus élaborés, fournissant des résultats plus adaptés selon la configuration des logs. Il s'agit d'un algorithme de type *discovery* qui permet, à partir d'un journal d'événements, de modéliser les relations entre différentes activités et ainsi de construire un *« Workflow net »*.

2.6.1 Présentation

L'algorithme Alpha fonctionne comme présenté dans 1 ([vdA11], p.133)

2.6.2 Analyse

Cette sous-section a pour objectif d'analyser les intérêts ainsi que les limites de l'algorithme Alpha.

Intérêts de l'algorithme

L'intérêt majeur de cet algorithme est sa simplicité de compréhension. Il permet de se familiariser facilement avec le principe du process mining.

Limites de l'algorithme

Malgré sa simplicité, il possède quelques inconvénients. En effet, il est possible que le modèle généré puisse être trop complexe du fait que certaines places créées sont redondantes et donc inutiles pour le modèle. De plus, cet algorithme rencontre des problèmes lorsqu'il y a des boucles (de longueur une ou deux) dans les traces, e.g. (a, b, b, b, c) qui est une trace contenant une boucle de longueur une, ou alors (a, b, c, b, c, d) qui est une trace contenant une boucle de longueur deux. En revanche, il prend en compte les boucles de longueur trois ou plus dans les traces. Un autre problème est que l'algorithme ne prend pas en compte la fréquence des traces. Ainsi, une trace qui apparaît très rarement sera prise en compte avec le même poids que d'autres traces plus fréquentes. Cela rend l'algorithme très sensible au bruit, i.e. les traces peu fréquentes. Enfin, le dernier problème lié à l'algorithme Alpha est qu'il est très sensible à l'incomplétude, i.e il ne génère pas de modèle capable de prendre en compte certaines traces non présentes dans le journal d'événement mais qui pourraient pourtant être valides.

Algorithme 1 Algorithme Alpha

1: Construire un premier ensemble contenant toutes les activités présentes dans le journal d'événements.
2: Construire un deuxième ensemble contenant toutes les activités qui apparaissent au moins une fois en première position dans une trace.
3: Construire un troisième ensemble contenant toutes les activités qui apparaissent au moins une fois en dernière position dans une trace.
4: Calculer toutes les paires (A, B) où A correspond à l'ensemble des activités d'entrée d'une place et B correspond à l'ensemble des activités de sortie de cette même place.
5: Supprimer les paires non maximales, c'est à dire les paires dont les activités sont contenues dans un autre ensemble plus grand.
6: Générer les places à partir de ces paires ainsi que la place de début et celle de fin.
7: Générer les arcs et les transitions (activités).
8: Générer le *Workflow net*.

2.6.3 Extension

L'algorithme Alpha a reçu plusieurs extensions. Ainsi, l'algorithme Alpha+ est capable de construire des modèles prenant en compte les boucles, et le modèle Alpha++ est capable de générer des modèles ayant des constructions de type « choix non libres ».

2.7 L'algorithme Flower Miner

2.7.1 Présentation

L'algorithme Flower Miner permet de générer des modèles pouvant accepter n'importe quelle trace. L'algorithme est simple 2.

La figure 2.10 présente un modèle généré par cet algorithme à partir de plusieurs traces contenant les activités *a, b, c, d, e, f, g, h*.

Algorithme 2 Algorithme Flower Miner

1: créer une place initiale *p0* ;
2: créer une activité et créer une dépendance allant de la place initiale à cette activité ;
3: créer une nouvelle place *p1*, reliée à l'activité précédente par une dépendance (de l'activité vers la place nouvellement créée) ;
4: créer une activité pour chacune des activités présentes dans le journal d'évènements, et la relier à la place *p1* créée précédemment par deux dépendances, une allant de l'activité vers la place et une autre allant de la place vers l'activité ;
5: créer une activité ainsi qu'une dépendance allant de la place *p1* à cette activité.
6: créer la place finale ainsi qu'une dépendance allant de l'activité créée précédemment vers la place finale.

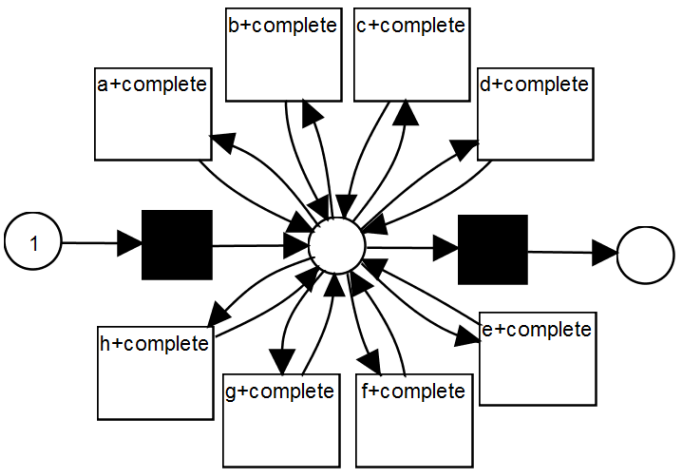

FIGURE 2.10 – Modèle généré par l'algorithme Flower Miner

2.7.2 Analyse

Cet algorithme n'apporte que peu d'intérêt. En effet, il permet de générer un modèle pouvant rejouer n'importe quelle trace. Il n'est donc pas très utile car il ne permet pas de créer un modèle représentatif d'un vrai processus.

2.8 L'algorithme Heuristic Miner

2.8.1 Présentation

Cet algorithme a pour principale caractéristique de prendre en compte la fréquence des évènements lors de la création du modèle. Cela induit que les traces non fréquentes ne doivent pas être ajoutées au modèle.

2.8.2 Analyse

Pour représenter un modèle créé avec l'heuristic miner, il convient d'utiliser une représentation sous forme

de réseau causal (« Causal net »). En effet, cette représentation a l'avantage de pouvoir afficher des dépendances qui ne sont pas gérées par les WF-nets. Il permet également de combler le problème principal de l'algorithme Alpha, à savoir la prise en compte des fréquences des traces. Afin de calculer cette fréquence, on cherche la dépendance entre chaque paire d'activités en calculant la matrice de dépendance. Tout d'abord, il faut réaliser la matrice qui indique, pour chaque activité, le nombre de fois où chaque autre activité la suit directement.

Par exemple, on considère L l'ensemble des traces suivantes ([vdA11], p. 164) :

$$L = [\langle a, e\rangle^5, \langle a, b, c, e\rangle^{10}, \langle a, c, b, e\rangle^{10}, \langle a, b, e\rangle^1, \\ \langle a, c, e\rangle^1, \langle a, d, e\rangle^{10}, \langle a, d, d, e\rangle^2, \langle a, d, d, d, e\rangle^1]$$

On commence par construire la matrice des activités qui se suivent directement, c'est à dire que pour chaque trace, on compte, pour chaque activité, le nombre de fois où chaque autre activité la suit. Ce nombre est noté $|a >_L b|$ Par exemple, en observant la liste des traces ci-dessus, on voit que l'activité a est suivie 11 fois par l'activité b, 11 fois par l'activité c, 13 fois par d, etc. On obtient le résultat dans le tableau 2.1.

Une fois cette matrice complétée, on calcule la dépendance entre chaque activité, notée $|a \Rightarrow_L b|$ à l'aide de la formule

$$|a \Rightarrow_L b| = \begin{cases} \frac{|a>_L b| - |b>_L a|}{|a>_L b| + |b>_L a| + 1} & \text{si} \quad a \neq b \\ \frac{|a>_L a|}{|a>_L b| + 1} & \text{si} \quad a = b \end{cases}$$

Cela permet de calculer la matrice de dépendance entre chaque activité. Pour construire le modèle, on utilise ensuite deux seuils. un seuil va permettre de déterminer le nombre minimal de répétitions d'une trace pour qu'elle ne soit pas considérée comme du bruit. Le second seuil détermine la valeur minimale que doit avoir la dépendance entre deux activités pour intégrer la relation dans le modèle. La relation est créée dans le modèle si et seulement si les deux conditions des seuils sont respectées. En faisant varier ces seuils, il est donc possible de faire varier le modèle qui en résulte.

|>L|	a	b	c	d	e
a	0	11	11	13	5
b	0	0	10	0	11
c	0	10	0	0	11
d	0	0	0	4	13
e	0	0	0	0	0

TABLE 2.1 – Fréquence des activités qui se suivent directement ([vdA11], p. 165)

Un graphe de dépendances, noté C est formalisé de la forme $C = (A, a_i, a_o, D, I, O)$ [vdA11]. A correspond à l'ensemble des activités présentes dans les traces. D correspond à l'ensemble des relations de dépendance entre les activités. a_i et a_o correspondent respectivement aux activités d'entrée et de sortie du graphe. Toutes les activités par la suite doivent suivre un chemin entre a_i et a_o. Si ce n'est pas le cas alors elles ne doivent pas être ajoutées au modèle car elles sont considérées comme inutiles dans la réalisation du processus. Nous avons vu lors de l'étape précédente que le graphe de dépendances était créé, mais pour être complet il manque la création des relations entre les activités. Le graphe est donc de la forme

$C = (A, a_i, a_o, D)$. I et O sont les fonctions des ensembles des activités d'entrée (Input) et de sortie (Output) de chaque activité. Une des méthodes de l'heuristique consiste à définir pour chaque activité (sauf les entrées pour a_i et les sorties pour a_o) l'ensemble des liens possibles avec les activités qui les entourent, puis de compter le nombre d'occurrences qui surviennent. En comparant ces sous-ensembles créés avec les logs, on peut éliminer ceux qui n'apparaissent pas dans les logs pour ne garder que les meilleurs. On peut également utiliser un seuil pour ne pas prendre en compte certains liens qui existent entre des activités, mais qui sont quand même présents dans les traces.

Intérêts de l'heuristic miner

Comme nous l'avons vu précédemment, l'intérêt majeur de l'utilisation de l'heuristic miner pour générer un modèle réside dans le fait que celui-ci prend en compte la fréquence des traces pour créer le modèle. Le bruit est donc bien géré. Il gère également les boucles dans les traces, car la dépendance peut être mesurée si deux activités successives sont les mêmes. De plus, les activités cachées sont gérées par l'heuristic miner, car il est possible d'exécuter des activités en parallèle. ces tâches sont difficiles à trouver car elles ne sont pas forcément présentes dans les logs. Les « causal nets » n'affichent pas les activités cachées contrairement aux réseaux de Pétri. C'est pour cette raison que les causal nets sont utilisés par l'heuristic miner.

Enfin, d'après l'article [WBVB12] qui a réalisé une comparaison de plusieurs algorithmes sur des vrais journaux d'événements, l'heuristic miner possède la meilleure précision et un F-score élevé. De plus, il fait parti des algorithmes ayant une meilleure note en terme de compréhensibilité.

Limites de l'heuristic miner

L'une des limites de l'heuristic miner est que bien qu'il puisse gérer le bruit présent dans les logs, le seuil se détermine de façon arbitraire, et il est possible qu'il soit mal réglé et que l'on perde de l'information lors de la création du modèle, et que des traces qui sont considérées comme du bruit ne le sont pas vraiment en réalité.

2.9 Genetic Process Mining

Contrairement à l'algorithme Alpha et les méthodes heuristiques, qui construisent des modèles de manière déterministe, le process mining génétique (Genetic process mining) suit une méthode itérative pour suivre une évolution naturelle de la construction du modèle, et donc de manière non déterministe. Il y a une part d'aléatoire dans la découverte de nouvelles possibilités de construction de modèles.

2.9.1 Présentation

Cette technique utilise les méthodes des algorithmes génétiques dans le cadre de la découverte de processus.

Celle-ci se décompose en quatre étapes principales (figure 2.11) :

— L'initialisation
— La sélection
— La reproduction
— Conditions d'arrêt et choix du modèle à retourner

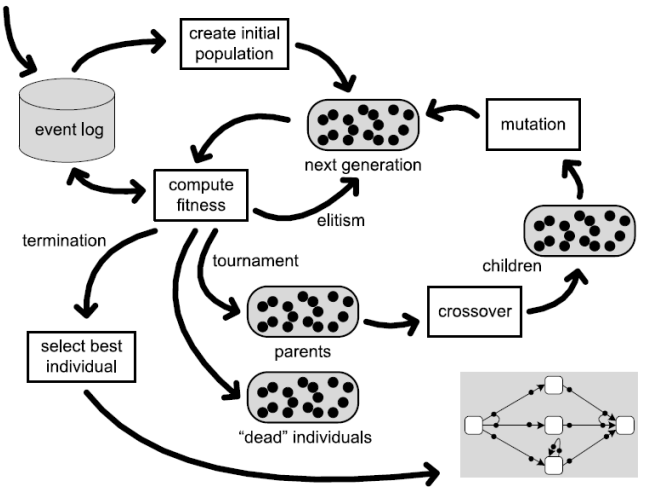

FIGURE 2.11 – Schéma représentant les différentes étapes du Genetic Process Mining ([vdA11], p.170)

L'initialisation

Tout d'abord, l'étape de l'initialisation consiste à créer la population initiale de process models (individus). Différents modèles sont générés aléatoirement à partir des activités présentes dans le journal d'événements. Ainsi, dans certains modèles il existera une dépendance entre deux activités tandis que dans d'autres modèles, ce lien ne sera pas présent. Ces process models (individus) ne sont donc pas forcément conformes à la représentation qui en est faite dans les logs, cependant quelques uns peuvent correspondre aux logs si un grand nombre d'individus sont générés par l'algorithme, toujours de manière aléatoire.

La sélection

L'étape suivante est la sélection. Celle-ci consiste à calculer l'adéquation (« fitness ») de chaque individu par rapport aux logs grâce à une fonction qui détermine la qualité de cet individu. Il existe différentes manières de mesurer l'adéquation d'un modèle aux évènements des logs. Une fonction d'adéquation adaptée doit prendre en compte l'adéquation partielle du modèle ainsi que les quatre critères de qualité (section 2.3). Par la suite, les meilleurs individus (les process models qui ont la meilleure valeur d'adéquation) sont utilisés pour la génération d'après. Cette étape s'appelle l'élitisme. Étape après étape, le reste des meilleurs individus (les parents) sont utilisés pour créer de nouveaux individus. Le reste n'est plus utilisé et les individus sont considérés comme morts (« dead individuals »). Ainsi, seuls les meilleurs individus sont conservés à chaque itération.

La reproduction

Vient ensuite la phase de reproduction. Ici, les parents sont utilisés pour créer une nouvelle génération d'individus. On utilise deux opérateurs utilisés dans les algorithmes génétiques : le croisement (« crossover ») (figure 2.12) et la mutation (figure 2.13). Pour le croisement, deux individus sont sélectionnés, en résulte la création de deux nouveaux modèles possédant chacun une partie du « patrimoine génétique » du parent. Ces deux nouveaux modèles sont ensuite modifiés aléatoirement grâce à la mutation, où des relations entre les activités sont ajoutées ou supprimées.

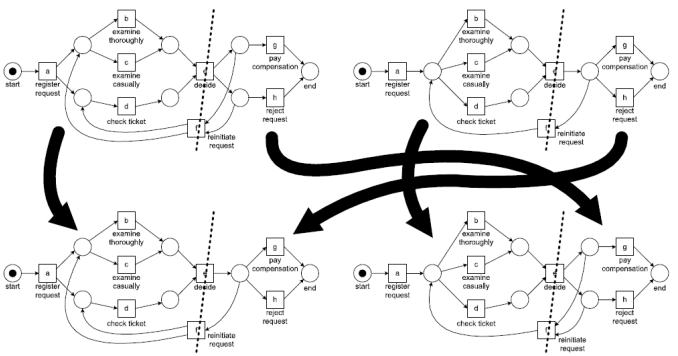

FIGURE 2.12 – Schéma illustrant le principe de crossover ([vdA11], p.173)

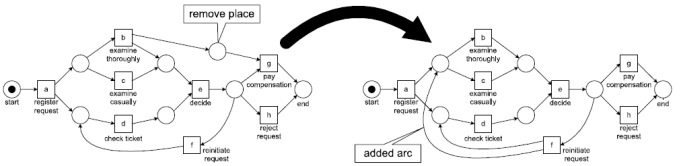

FIGURE 2.13 – Schéma illustrant le principe de mutation ([vdA11], p.173)

Conditions d'arrêt et choix du modèle à retourner

Les deux étapes précédentes, la sélection et la reproduction, sont répétées plusieurs fois, générant à chaque itération (génération) une nouvelle population. L'algorithme s'arrête lorsqu'une génération possède un modèle ayant une note supérieure à un seuil passé en paramètre de l'algorithme. Le meilleur modèle de cette population est retourné par l'algorithme.

Cependant, arrêter l'algorithme lorsque ce seuil a été atteint n'est pas forcément envisageable. En effet, il est parfois nécessaire d'ajouter d'autres paramètres car il n'est pas toujours possible d'atteindre le seuil à cause du bruit ou d'un temps de calcul trop élevé. Pour cela, il est possible de limiter le nombre d'itérations ou d'arrêter l'algorithme lorsqu'un certain nombre d'itérations n'a pas produit de meilleurs modèles ([vdA11], p.171).

Paramètres de l'algorithme

Nous allons désormais nous intéresser aux différents paramètres de l'algorithme ([vdA11], p.171).

- Le langage de modélisation à utiliser (réseaux de Pétri, Causal net, etc.)
- L'initialisation. Par exemple, il est possible de créer la population initiale aléatoirement ou alors en utilisant le résultat d'un autre algorithme de process mining en entrée de l'algorithme genetic process mining.
- La fonction d'adéquation utilisée, c'est-à-dire la fonction utilisée pour calculer la note d'un modèle.
- Le pourcentage de modèles à garder pour l'itération suivante (élitisme).
- La probabilité d'une mutation.

2.9.2 Analyse

Après la présentation du fonctionnement de l'algorithme genetic process mining effectuée dans la sous-section précédente, les intérêts et les limites de cet algorithme sont étudiés ici.

Intérêts du genetic process mining

Il s'agit d'un algorithme robuste qui tient compte du bruit dans les données ainsi que de l'incomplétude. Il est également flexible car en modifiant la fonction d'adéquation il est possible d'orienter le modèle construit ([vdA11], p.172).

Limites du genetic process mining

Cet algorithme possède quelques inconvénients. En effet, de part sa nature il est possible que cet algorithme converge vers une bonne solution, mais en beaucoup trop de temps. C'est surtout le cas lorsque la taille du modèle à générer est très grande, c'est-à-dire lorsqu'il y a beaucoup d'activités. En effet, la solution est construite petit à petit à partir de modèles générés aléatoirement à partir d'un journal d'événements.

Cependant, pour réduire le temps nécessaire pour converger vers une solution, il est possible de démarrer avec une population initiale composée de modèles construits à l'aide d'un autre algorithme, comme l'heuristic miner 2.8. Ainsi, la population initiale possède déjà des modèles relativement bons, ce qui augmente les chances de converger rapidement vers une bonne solution.

2.9.3 Duplicate Task Genetic Miner

Il existe également un algorithme génétique, le *Duplicate Task Genetic Miner*, qui contrairement à l'algorithme *Genetic Miner* permet de générer des modèles ayant des duplications d'activités.

2.10 AGNEs Miner

2.10.1 Présentation

Il s'agit d'un algorithme qui est capable de générer un réseau de Pétri. Il utilise en plus du journal d'évé-

nements, des évènements négatifs créés artificiellement, afin d'apprendre les conditions pour discrimer les évènements ([WBVB12], p.4).

2.10.2 Analyse

Cette sous-section a pour objectif de présenter les intérêts et les limites de l'algorithme AGNEs Miner.

Intérêts de l'algorithme AGNEs Miner

D'après l'article [WBVB12], cet algorithme possède une bonne précision et possède une bonne note en terme de compréhensibilité.

Limites de l'algorithme AGNEs Miner

Cet algorithme ne possède pas beaucoup d'inconvénients, il est cependant important de noter qu'il est un peu moins bon que l'Heuristic Miner 2.8 ([WBVB12]).

2.11 ILP Miner

2.11.1 Présentation

L'algorithme ILP Miner est une application d'optimisation linéaire en nombres entiers au domaine du process mining ([WBVB12], p. 6). Cette approche est indépendante du nombre d'événements à traiter, « this approach allows for parallelization and is shown to be independent of the number of events registered in the event log, the technique can be expected to be useful in practice. » [3]

2.11.2 Analyse

Cette sous-section a pour objectif de présenter les intérêts et les limites de l'algorithme ILP Miner.

Intérêts de l'algorithme ILP Miner

Sur des journaux d'événements réels, c'est-à-dire issus d'une organisation (entreprises, universités, etc.) et non pas créés artificiellement, cet algorithme créé des modèles avec un fort rappel (pourcentage d'événements pouvant être rejoués par le modèle).

Limites de l'algorithme ILP Miner

Le principal inconvénient de cet algorithme est qu'il produit des modèles qui sont difficilement compréhensibles.

2.12 Region-Based Mining

2.12.1 Présentation

Cette méthode permet de construire un réseau de pétri à partir d'un système de transitions déterminé à partir des

3. Cette approche autorise la parallélisation et est démontrée être indépendante du nombre d'événements présents dans le journal d'événements, cette technique peut s'avérer utile en pratique ([WBVB12], p. 6)

logs. Ce système de transitions, qualifié de bas niveau, va être synthétisé en découvrant des régions, c'est-à-dire découvrir un ensemble d'états qui s'accordent entre eux.

2.12.2 Analyse

Afin de découvrir un réseau de pétri avec l'approche du Region-Based mining, il est tout d'abord nécessaire de découvrir un système de transitions.

Ce système est formulé de la manière suivante : $TS = (S, A, T)$, où *S* représente l'ensemble des états, *A* l'ensemble des activités et *T* l'ensemble des transitions. Pour représenter ce système de transitions à partir de logs, on utilise une fonction de représentation des états. Ici, un état correspond à un ensemble d'activités qui respectent la règle définie par la fonction de représentation d'un état. Il existe de nombreuses fonctions de représentation d'un état. Une première fonction de représentation prend en paramètre une trace et une position *k*, qui va retourner les k premières activités de la trace en partant du début de celle-ci. Elle prend donc en compte le « passé » des traces, c'est à dire les activités qui se sont déroulées avant la k^{ème} activité. Une seconde fonction est assez similaire à la première, sauf qu'elle représente le « futur » de la trace à la position k, en inscrivant les activités qui suivent celle située à la position *k*. Une autre fonction de représentation ne prend pas en compte l'ordre des évènements, seulement la fréquence des activités. Elle compte, à une position *k* dans la trace, les activités qui sont apparues et le nombre de fois où elles sont apparues. Une autre ne prend en compte ni l'ordre, ni la fréquence, et est juste un ensemble d'activités apparues au moins une fois à un instant *k* dans la trace.

Grâce à ces fonctions, il est ensuite possible de construire les transitions où chaque transition entre deux noeuds correspond à une activité et chaque noeud l'ensemble des activités représentant l'état à un instant *k*.

Il existe d'autres fonctions de représentation d'un état pour s'adapter au besoin du modèle que l'on va créer.

Une fois que ce système de transitions a été créé à partir des logs, on va pouvoir le représenter sous forme de réseau de pétri, qui sera plus petit et plus lisible. Le principe est de découvrir des régions, c'est à dire un ensemble d'états qui s'accordent, et chaque région va correspondre à une place dans le réseau de pétri.

Une région, nommée *R* est donc un sous-ensemble d'états. Elle est classée par trois règles :

- Les transitions entrantes.
- Les transitions sortantes.
- Les transitions qui sont contenues dans la région, c'est-à-dire ni entrantes, ni sortantes.

Elle peut être considérée comme telle si elle respecte trois conditions :

- Toutes les transitions entrantes du même nom ne doivent pas être sortantes (Par exemple, toutes les transitions avec l'activité a doivent entrer dans la région mais pas en sortir.
- Idem pour les transitions sortantes, elles ne doivent pas entrer dans la région mais seulement en sortir.
- Les transitions qui ne traversent pas la région ne doivent ni entrer, ni en sortir.

Si toutes ces conditions sont respectées, alors une place

est créée. En entrée, on retrouvera les activités qui satisfont la règle des activités entrantes, en sortie, celles des activités sortantes, et les activités qui sont contenues dans la région sont ignorées dans la place.

Intérêts du Region-based mining

Lors de la construction du système de transition, il est possible de prendre en compte les nombreuses informations disponibles dans des logs, comme des ressources, qui peuvent être utilisées pour nommer des transitions plutôt que les activités.

Limites du Region-based mining

Les systèmes de transitions sont considérés comme des modèles bas niveau, car le nombre d'états et de transition peut vite exploser en fonction du nombre d'activités. C'est pour cela que le réseau de pétri est ensuite construit à partir de ce système de transitions, pour pouvoir le synthétiser. De plus, des fonctions de représentation d'états, comme la première présentée ci-dessus, peut créer un modèle qui ne peut rejouer que les traces des logs, donc en situation d'« overfitting » ou d'autres fonctions qui peuvent rejouer n'importe quelle trace, en situation d'« underfitting ». Il faut donc trouver un juste équilibre dans le choix de ces fonctions.

2.13 Présentation du Conformance Checking

Le conformance checking est une technique qui permet de comparer un modèle - qu'il ait été construit à la main ou via un algorithme de process mining - à une trace utilisateur présente dans un journal d'évènements. Le but est de trouver les points communs et les divergences entre les deux. Cela permet par exemple de détecter de la fraude car il mesure les déviations par rapport au comportement attendu. Les techniques de conformance checking peuvent également permettre d'améliorer des modèles qui ne seraient pas conformes à la réalité, ou encore mesurer la performance d'un algorithme de process mining.

Afin de pouvoir mesurer cette différence, plusieurs techniques sont utilisées.

2.13.1 Utilisation des jetons

Pour calculer la différence entre des logs et un modèle, une solution simple serait de calculer le rapport entre le nombre de traces qui peuvent être rejouées complètement et le nombre de traces total. Cependant, cette solution ne prend pas en compte le fait que certaines traces qui ne passent pas complètement puissent potentiellement être pratiquement jouées en totalité, c'est à dire qu'il n'y a qu'une activité ou deux qui bloque la réalisation de la trace. On utilise donc une méthode qui va calculer l'adéquation de la trace avec le modèle en parcourant chaque trace, et en forçant l'exécution de celle-ci même lorsque

ce n'est pas permis par le modèle. Quatre compteurs vont être utilisés pour compter les jetons. On compte tout d'abord les jetons produits (p). Chaque passage dans une activité incrémente ce nombre. On compte ensuite les jetons consommés (c), où chaque passage d'une place à une autre incrémente ce compteur, les jetons manquants (m), où dès qu'un jeton est créé pour satisfaire une relation, ce compteur est incrémenté, et les jetons restants (r), c'est à dire ceux qui n'ont pas besoin d'être utilisés pour satisfaire une condition de passage à une activité suivante. La formule du calcul de l'adéquation (« fitness ») d'une trace σ à un modèle N s'écrit

$fitness(\sigma, N) = \frac{1}{2}(1 - \frac{m}{c}) + \frac{1}{2}(1 - \frac{r}{p})$

Ce calcul est donc compris entre 0 et 1 et peut être généralisé à l'ensemble du journal d'évènements en adaptant la formule ci-dessus, en replaçant *m* par la somme de toutes les *m*fréquence de la trace* et *c* par la somme de toutes les *c*fréquence de la trace*. Le même changement vaut pour r et p. On obtient donc une valeur entre 0 et 1 qui représente globalement le pourcentage de traces qui sont conformes au modèle. Le résultat peut être interprété de deux manières, si par exemple on obtient une valeur de fitness de 0.95 : soit les logs ont une fitness de 95%, c'est-à-dire que 5% des traces dévient par rapport au modèle, soit le modèle est incapable d'expliquer 5% des traces et que donc il est potentiellement non correct. Cette vision des choses dépend de la considération du modèle.On peut le considérer soit normatif (le modèle est correct, et ce sont les logs qui ne sont pas conformes), soit descriptif (le modèle n'est pas conforme à la réalité).

Cette méthode de conformance checking s'appuie sur la valeur du fitness pour évaluer des logs par rapport au modèle, car il s'agit du critère le plus important pour cette partie. Les trois autres critères de qualité sont moins importants.

Il est possible de l'améliorer en ajoutant des règles, comme par exemple deux activités qui doivent se suivre obligatoirement, ou qui ne doivent pas être effectuées par la même personne, etc.

2.13.2 Empreintes causales

Une autre manière de calculer la conformance est de comparer l'empreinte (« footprint ») des logs et celle du modèle. Pour ce faire, on réalise la matrice similaire à la matrice 2.1, où on va formaliser la relation entre chaque activité, de la manière suivante :

- $a \rightarrow b$ si a est toujours juste avant b et b jamais avant a.
- $a \leftarrow b$ si a est toujours juste après b et jamais avant.
- $a \parallel b$ si a et b peuvent être parallèles, c'est-à-dire si a est avant b et b avant a selon les traces.
- $a \# b$ si a et b n'ont pas de relation particulière.

Par la suite on créé la même matrice, mais à partir du modèle. On compare ensuite les différences entre les deux matrices, en comptant le nombre de cellules où le résultat est différent. La formule de la conformance avec les empreintes est 1-$\frac{\text{Nombre de cellules différentes}}{\text{Nombre de cellules totales}}$

Les limites de cette techniques sont qu'elle ne prend pas en compte la fréquence des traces, juste les relations entre les activités. Aussi, la limitation est due au fait qu'elle prend en compte la relation « suit directement » et

non pas « suit éventuellement ». Il s'agit d'une des améliorations possibles, et une mesure de conformance doit être adaptée selon ce que l'on cherche.

2.14 Récapitulatif

Certains algorithmes de process mining sont moins performants que d'autres, notamment l'algorithme alpha et ses extensions. En effet, l'algorithme alpha souffre de beaucoup de problèmes, comme le fait qu'il ne détecte pas les boucles où qu'il ne prend pas en compte les dépendances non locales.

D'après l'article [WBVB12] qui compare sept algorithmes dans un contexte réel, c'est-à-dire que les journaux d'événements utilisés pour créer les modèles proviennent de véritables organisations, et ne sont donc pas créés artificiellement, l'algorithme ILP Miner possède le meilleur rappel. C'est donc l'algorithme qui permet de rejouer le plus de traces. Cependant, il souffre d'une mauvaise note en compréhensibilité car les modèles qu'il génère possèdent beaucoup d'arcs. Du côté de la précision, c'est l'Heuristic Miner qui a le meilleur score, et il a également un bon F-score, tout comme l'algorithme AGNEs Miner qui obtient de bons résultats.

L'Heuristic Miner fait également parti des meilleurs algorithmes en terme de compréhensibilité des modèles générés, avec les algorithmes génétiques. Cependant, ceux-ci souffrent de divers problèmes, comme le fait qu'ils mettent beaucoup de temps à produire un résultat, ce qui les rend peu adaptés dans un contexte réel.

Le Conformance Checking regroupe l'ensemble des algorithmes qui permettent de rejouer des traces sur un modèle afin de vérifier leur conformité par rapport au modèle. Plusieurs méthodes sont possibles, plus ou moins efficaces comme nous l'avons vu précédemment. Par exemple, l'approche consistant à utiliser des jetons pour rejouer une trace semble plus adaptée à notre problème que celle basée sur les empreintes causales étant donné qu'elle prend en compte les fréquences des traces, c'est à dire qu'une trace fréquente aura une meilleure conformance qu'une qui ne l'est pas.

2.15 Conclusion

Cet état de l'art fait état de nos recherches sur les différents algorithmes de process mining ainsi que des techniques de conformance checking, qui est une étape importante dans notre projet puisque c'est cette partie qui permettra de comparer une trace d'utilisateur à notre modèle. Ici, les algorithmes présentés ont tous leurs avantages et inconvénients par rapport à notre problème, et c'est suite à cet état de l'art que nous pourrons choisir un algorithme pour créer notre modèle.

3

Propositions

Ce chapitre a pour objectif de présenter notre proposition suite à l'état de l'art.

3.1 Présentation des traces

Le but de la recherche bibliographique étant de choisir l'algorithme le plus adapté à nos besoins pour créer les modèles qui vont permettre de prédire l'échec ou la réussite à un cours par un utilisateur, il faut que cet algorithme soit adapté aux fichiers de logs qui ont été mis à notre disposition pour réaliser cette étude. Pour chaque trace, nous avons les données suivantes à notre disposition :

- UserID : l'id de l'utilisateur
- CourseID : l'id du cours
- PartID : l'id de la partie du cours suivi
- StartTime et EndTime : les dates et heures de début et de fin de l'évènement
- CourseName : Le nom du cours
- PartName : Le nom de la partie du cours suivi

A partir de ces différents champs, nous pouvons choisir certains d'entre eux pour être adaptés à notre problème lié au process mining. Dans notre cas, un évènement correspond à une ligne dans ces données. Le cas correspond ici à l'utilisateur. Une activité est représentée ici par le partID. Le StartTime et EndTime sont les Timestamp associés à l'évènement.

3.2 Utilisation de l'algorithme Heuristic Miner

Comme nous l'avons vu dans l'état de l'art 2, l'Heuristic Miner semble être la meilleure solution pour notre problématique, à savoir créer un modèle à partir des traces d'utilisateurs. En effet, suite à l'étude bibliographique, nous avons pu constater que l'Heuristic Miner était capable de gérer les boucles dans une trace. Or, dans les données que nous avons, des boucles sont présentes, ce qui implique que l'algorithme que nous choisissons doit absolument prendre en compte la gestion des boucles, ce

qui élimine l'algorithme alpha dans sa version initiale.

Il prend également en compte les fréquences, ce qui lui permet de ne pas tenir compte du bruit présent dans les journaux d'évènements. Il permet également de réaliser des modèles contenant des structures de « OU inclusif » ainsi que des constructions de choix non libres.

De plus, d'après l'article [WBVB12] qui compare différents algorithmes de process mining dont l'Heuristic Miner, désigne celui-ci comme étant le plus performant dans un contexte réel. En effet, il possède la meilleure précision, et il a également un bon F-score, c'est-à-dire qu'il réalise un bon compromis entre précision et rappel.

3.3 Conformance Checking

Afin de valider les traces sur le modèle que nous aurons établi, nous avons décidé d'utiliser l'algorithme de conformance checking basé sur l'utilisation de jetons. Nous en avons conclu que cette méthode était la plus adaptée pour notre besoin étant donné que cet algorithme permet de prendre en compte le fait qu'une trace peut ne pas être totalement rejouée sur un modèle. Une fois que les modèles de réussite et d'échec à un cours seront définis, on pourra appliquer le conformance checking sur de nouvelles traces, et comparer avec quel modèle la trace se rapproche le plus.

3.4 Création des modèle et classification d'un utilisateur

Afin de classifier des utilisateurs en leur prédisant un échec ou une réussite à un cours, nous allons générer deux modèles par type d'utilisateurs. Le premier sera construit à partir des utilisateurs ayant réussi le cours, et le second sera généré à partir de ceux ayant échoué à ce cours.

Pour cela, nous allons créer un module python qui se chargera de construire les ensembles d'apprentissage et les ensembles de test afin d'appliquer les différents algorithmes retenus, à savoir l'Heuristic Miner pour la construction du modèle et l'algorithme de Conformance Checking basé sur l'utilisation de jetons.

Ensuite, pour prédire la réussite ou non d'un nouvel utilisateur, il faut le comparer aux deux modèles générés pour ce cours et pour le groupe auquel l'utilisateur appartient. Ainsi un utilisateur non premium suivant un cours sur Java sera comparé à d'autres utilisateurs ayant suivi ce cours et étant non premium. On applique ensuite l'algorithme de conformance checking sur chacun des modèles avec les données de cet utilisateur. Si un meilleur score est obtenu sur le modèle construit avec des utilisateurs ayant réussi le cours, alors nous prédisons que l'utilisateur va réussir le cours.

3.5 Conclusion

Nous avons donc prévu d'utiliser l'algorithme Heuristic Miner. Il nous semble bien adapté à notre jeu de données, et il semble être le plus adapté pour un contexte réel. De plus, il possède l'avantage d'être disponible publiquement et implémenté dans le logiciel ProM. Nous utiliserons également l'algorithme de Conformance Checking basé sur l'utilisation de jetons, car il semble également bien adapté pour ce que nous souhaitons faire.

4

Expérimentations et résultats

Cette partie est destinée à présenter le développement effectué suite au travail bibliographique présenté jusqu'à maintenant. Elle montrera notre approche pour réaliser l'outil permettant de générer les modèles suite à nos propositions, et les résultats obtenus. Dans un premier temps nous présenterons les attentes de l'outil réalisé ainsi que le contexte dans lequel il a été réalisé. Nous développerons par la suite la modélisation et la structuration de notre code. Nous présenterons les différentes étapes du développement et enfin les résultats obtenus.

4.1 Cahier des charges

Nous axerons le développement de cette partie sur les besoins qui ont été exprimés quant aux fonctionnalités attendues de notre outil.

4.1.1 Besoins

Afin de pouvoir faire nos expérimentations, nous avons dû créer un outil qui permet de générer les modèles nécessaires qui permettront de dire si un utilisateur va réussir ou non à un cours. Pour cela, le programme devra permettre de spécifier un cours et un groupe d'utilisateurs pour récupérer les fichiers de logs correspondant puis d'appliquer notre algorithme choisi. La liste des groupes d'utilisateurs disponibles est la suivante :

— Utilisateurs premiums
— Utilisateurs non premiums
— Tous les utilisateurs

Pour chacun de ces groupes, on définit également les deux catégories, soit un total de 6 groupes d'utilisateurs :

— Utilisateurs qui réussissent (Succeed)
— Utilisateurs qui échouent (Failed)

Le programme devra générer un modèle pour chaque groupe d'utilisateurs, et pour chaque cours. On devra pouvoir ainsi par la suite comparer un nouvel utilisateur qui s'inscrit à un cours au modèle correspondant selon sa catégorie (premium ou non) avec les deux modèles « Succeed » et « Failed ». Ainsi, si cet utilisateur obtient un meilleur score avec le modèle « Succeed », on peut alors prédire qu'il va réussir le cours. Dans le cas contraire, la prédiction que l'on fera sera qu'il va échouer.

4.1.2 Contraintes liées au projet

Le développement de notre projet est caractérisé par plusieurs contraintes.

La première est de pouvoir appliquer les algorithmes que nous avons choisi d'utiliser. Pour appliquer l'algorithme Heuristic Miner et le Conformance Checking, nous utiliserons deux plugins issus du logiciel ProM. Il s'agit d'un logiciel open source qui permet d'appliquer des algorithmes de process mining et de conformance checking, d'analyse de modèle, etc. C'est le logiciel de référence dans le domaine du process mining. Il a été conçu pour être modifiable par n'importe qui, et les différents algorithmes sont créés sous forme de plugin, que chacun peut développer.

La deuxième contrainte est que le code produit devra être intégré sous la forme d'un module à un projet existant. En effet, ce projet contient notamment des modules python permettant de générer tous les fichiers au format CSV contenant les logs utiles à notre projet à partir d'une base de données.

Notre projet consistera donc en la réalisation d'un package « processMiningManager » qui sera intégré au projet existant et contenant un module python permettant d'appliquer les deux algorithmes.

4.1.3 Architecture globale

La figure 4.1 montre le déroulement et la logique de notre travail. Notre module python permet à partir d'un fichier CSV de générer deux nouveaux fichiers CSV : un ensemble d'apprentissage et un ensemble de test.

Ce code python appelle ensuite un premier programme Java en lui fournissant l'ensemble d'apprentissage. Ce programme transforme le fichier en un objet exploitable par l'algorithme Heuristic Miner. En appliquant cet algorithme, on obtient un Heuristic net qui est ensuite converti en Petri net puis exporté dans un fichier.

Le module python appelle ensuite un deuxième programme Java qui importe le fichier créé à l'étape précédente, le convertit en réseau de Pétri et applique un algorithme de Conformance Checking en appliquant l'en-

semble de test sur le réseau de Pétri.

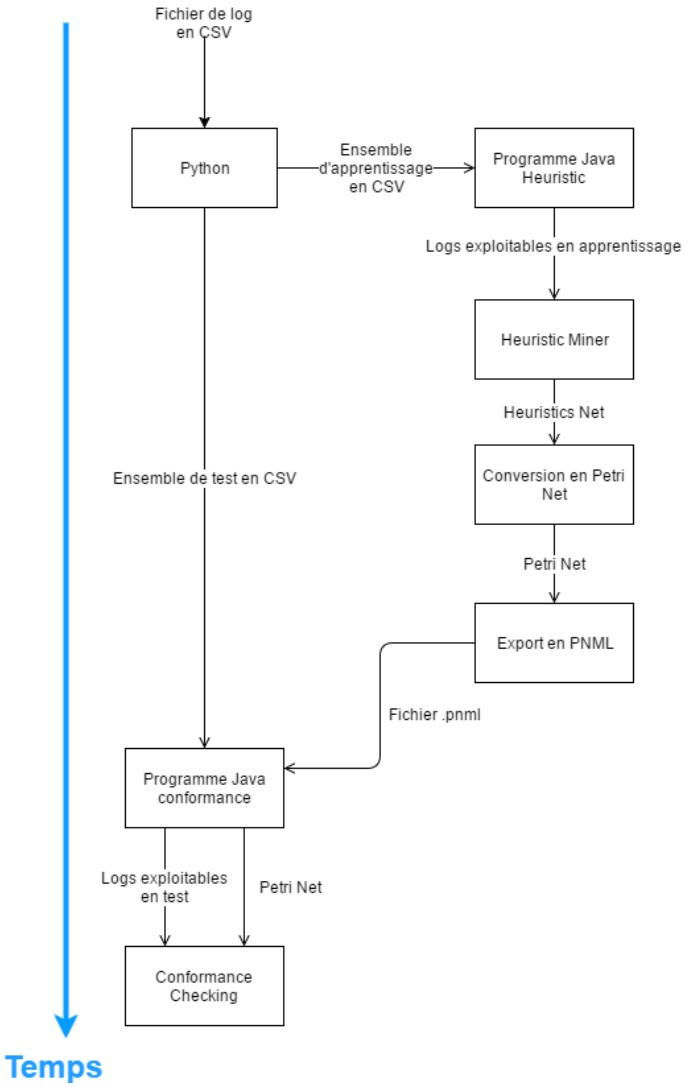

FIGURE 4.1 – Schéma illustrant la logique du code

4.2 Développement

Cette section a pour objectif de présenter le développement qui a été réalisé lors de ce projet de recherche et développement. Nous allons dans un premier temps aborder la récupération des plugins contenant les algorithmes que nous allons utiliser par la suite. Nous détaillerons ensuite chacune des grandes étapes du développement, puis nous terminerons en présentant les résultats que nous avons obtenus.

4.2.1 Récupération des plugins

Afin de pouvoir appliquer l'algorithme Heuristic Miner et le Conformance Checking, nous avons dû importer du code existant pour pouvoir l'utiliser. Ces algorithmes sont disponibles dans des plugins de ProM, sous forme d'archives JAR sur internet. Pour les récupérer, nous avons dû parcourir la liste de tous les packages de proM (il en existe plus d'une centaine) et récupérer le code, via le répertoire SVN de ProM.

Cependant, les plugins sont pensés pour fonctionner avec ProM, et pas forcément dans un outil externe tel que celui que l'on souhaite créer. Ainsi, de nombreux plugins ne sont pas compatibles avec une utilisation externe à ProM, car les paramètres requis pour les faire fonctionner nécessitent une interface graphique.

Néanmoins, certains d'entre-eux fonctionnent grâce à l'interface de commandes en ligne (« command-line interface », ou « CLI », de ProM. Dans ce cas, l'interface graphique n'est plus nécessaire et le code peut donc être utilisé à l'extérieur.

Nous avons trouvé un moyen de lister les plugins utilisables en interface de lignes de commandes. Cependant, les noms des plugins affichés correspondent au nom utilisé dans l'interface graphique de ProM, et non au nom de l'archive JAR. De plus, plusieurs plugins sont parfois contenus dans une même archive, ce qui complexifie la recherche d'un plugin particulier.

La récupération du code nous a posé d'autres problèmes. En effet, la plupart des plugins font eux-mêmes référence à d'autres plugins. Dans ce cas il était alors nécessaire de récupérer tous les plugins concernés pour pouvoir faire fonctionner le code. Pour chaque plugin, un fichier liste les dépendances aux autres, ce qui nous a permis de les récupérer, en les cherchant un par un.

Les deux plugins que nous avons utilisé sont donc l'« HeuristicsMiner », et le plugin « PNConformaceAnalysis » pour le Conformance Checking. Ils nécessitent aussi 67 autres packages de ProM pour pouvoir être utilisable, que nous avons récupérés également.

4.2.2 Programme java

Heuristic Miner

Ce programme comporte toutes les étapes qui permettent, à partir d'un fichier de logs, de générer un modèle au format .pnml qui sera ensuite utilisé par un deuxième programme Java permettant d'appliquer un al-

gorithme de Conformance Checking. Pour générer ce modèle, notre code utilise donc plusieurs plugins de ProM.

Tout d'abord, nous générons un objet XLog à partir du fichier CSV contenant un ensemble d'apprentissage. Cet objet est utilisé par l'algorithme Heuristic Miner pour générer le modèle sous forme d'un Heuristic Net. Le plugin de ProM propose plusieurs méthodes pour configurer les paramètres de l'algorithme.

Une fois que l'algorithme a été lancé, celui-ci retourne donc un objet « HeuristicsNet », qui est converti par la suite en réseau de pétri (PetriNet) puis exporté sous forme de fichier .pnml. Ces étapes sont visibles dans la figure 4.1.

Les principales difficultés rencontrées pour la création de ce programme ont été de réussir à utiliser le plugin « HeuristicMiner ». En effet, celui-ci nécessitait un objet de type « PluginContext » pour fonctionner (relatif à ProM). Nous avons donc récupéré et adapté le code de la classe de façon à retirer les éléments qui nous posaient problème, sans pour autant empêcher le bon fonctionnement du plugin, étant donné que l'objet PluginContext a pour fonction principale de stocker des informations pour l'intégration du plugin à l'interface de ProM.

Nous avons finalement réussi à utiliser un objet « CLIPluginContext » héritant de « PluginContext » qui est proposé par l'un des plugins de ProM. Cet objet permet de ne pas avoir à utiliser l'interface graphique de ProM. Ainsi, nous avons pu utiliser directement le plugin « HeuristicsMiner » en utilisant cet objet.

Conformance Checking

Une fois que le modèle est créé, nous pouvons ensuite utiliser un algorithme de Conformance Checking pour comparer de nouvelles traces à notre modèle. Pour ce faire, nous utilisons le plugin « PNConformaceAnalysis ». Celui-ci prend en paramètres un objet XLog et un PetriNet pour pouvoir calculer la proportion de traces rejouées sur le modèle (« fitness »). Tout comme pour l'Heuristic Miner, le fichier CSV contenant l'ensemble de test passé en paramètre est converti en « XLog » pour être utilisé par l'algorithme, et le fichier .pnml est importé et reconverti en objet « PetriNet ».

Tout comme l'Heuristic Miner, le Conformance Checking nous a posé problème. Tout d'abord, il existe assez peu d'algorithmes de Conformance Checking disponibles dans ProM, et un seul est compatible avec l'interface en lignes de commandes. Cela nous a énormément limité dans le choix des algorithmes, même si celui choisi correspond à celui de notre proposition.

Malgré qu'il soit adapté à notre besoin, ce plugin n'a pas fonctionné sur nos fichiers de logs. A de nombreuses reprises, nous avons tenté de les modifier pour voir où était le problème, mais sans succès. L'exécution du plugin se bloque, et il n'analyse pas toutes les traces. Ce comportement est le même en utilisant le plugin directement dans ProM. Malheureusement, il s'agissait du seul

plugin disponible en lignes de commandes que nous pouvions utiliser. L'exploitation des résultats a donc été rendue impossible car cette étape est bloquante pour le reste de notre projet.

4.2.3 Python

Comme présenté sur la figure 4.1, le module python est le réel point de lancement des algorithmes de process mining. Nous allons nous intéresser à son fonctionnement, puis aux diverses difficultés rencontrées lors du développement du module.

Fonctionnement

Le module s'accompagne d'une fonction permettant à l'utilisateur de renseigner l'identifiant d'un cours et un groupe d'utilisateurs (4.1.1) afin de récupérer le fichier CSV correspondant. Le programme appelle ensuite une méthode qui génère deux fichiers : un fichier qui constituera l'ensemble d'apprentissage, et un autre celui de test. La proportion apprentissage/test est paramétrable. Pour notre projet, nous l'avons réglé à 80% pour l'apprentissage, c'est à dire que l'on récupère 80% des traces dans le fichier de logs. Une trace correspond à l'ensemble des données pour un « cas » (CF vocabulaire 2.1.1), qui est un utilisateur du site dans notre cas. Les derniers 20% sont destinés au fichier contenant l'ensemble de test.

Le programme python appelle ensuite notre premier programme Java afin de générer le modèle, en lui passant les paramètres nécessaires, et notamment le fichier utilisé, qui est donc l'ensemble d'apprentissage. Une fois le fichier contenant le modèle généré, le deuxième programme Java est appelé afin d'appliquer le conformance checking à partir de ce fichier et du fichier contenant l'ensemble de test.

4.3 Résultats

4.3.1 Résultats de l'Heuristic Miner

Notre premier programme Java permet donc de générer un fichier au format .pnml représentant un réseau de Pétri. La figure 4.2 montre un extrait du réseau de Pétri obtenu sur un jeu de données. Les paramètres de l'algorithme sont configurables à partir du code Java.

Nous sommes confiants dans le modèle généré par notre programme Java car il est identique à un modèle généré à l'aide de l'interface graphique de ProM. Notre programme a donc l'avantage de pouvoir automatiser tout le processus, de l'importation du fichier csv contenant les logs à la génération du modèle et à l'exportation de celui-ci.

4.3.2 Résultats du Conformance Checking

Comme nous l'avons expliqué dans la partie Conformance Checking (4.2.2), nous avons rencontré des problèmes pour la partie conformance checking de notre programme. En effet, nous ne sommes pas parvenus à appliquer l'algorithme que nous avions choisi sur nos traces.

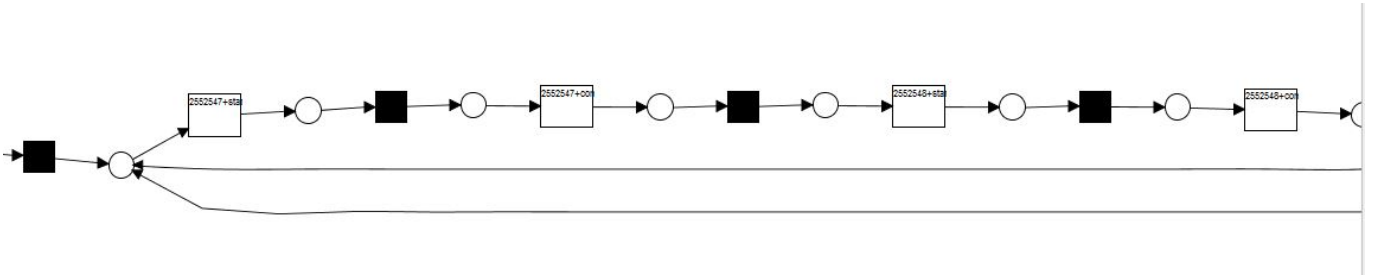

FIGURE 4.2 – Extrait d'un réseau de Pétri obtenu

Cependant, le problème ne semble pas venir directement de notre programme car nous observons le même résultat en faisant la même expérience avec le logiciel ProM. Étonnamment nous sommes arrivés à appliquer notre algorithme sur un de nos fichiers CSV en lui retirant certaines lignes (évènements). Nous n'arrivons cependant pas à comprendre la raison pour laquelle le programme ne s'exécute pas sur l'ensemble des traces.

Afin de pouvoir tester tout de même une solution sur nos traces, nous avons pu appliquer un algorithme de Conformance Checking en utilisant un autre plugin de ProM, s'utilisant avec l'interface graphique. Il s'agit du plugin « PNetReplayer ». Ce plugin applique un algorithme de type A* qui calcule la fitness et est basé sur des coûts (le poids associé à une transition dans le processus). Dans notre cas, les coûts de passage d'une place à une autre ont été laissées par défaut.

Pour appliquer ce plugin sur nos traces, nous avons tout d'abord généré les deux ensembles de test et d'apprentissage du cours « Audace Entreprendre », ainsi que les modèles correspondant générés avec les données d'apprentissage, sous forme de PetriNet à l'aide de notre programme.

Les données de test utilisées sont celles représentant un seul utilisateur premium ayant échoué à ce cours, car il s'agit d'une prédiction que l'on souhaite faire pour cet utilisateur (le cas) particulier.

On peut donc s'attendre à ce que la valeur du fitness soit plus élevée en comparant ces données avec le modèle « failed » qu'avec le modèle « premium ». Une fois ces données importées dans ProM sous forme de Logs ainsi que les deux modèles (succeed et failed), nous avons pu appliquer cet algorithme à la main pour chaque modèle.

Nous avons paramétré l'algorithme de manière à ne pas pénaliser les traces qui ne sont pas rejouées complètement, étant donné qu'il s'agit d'une possibilité vu que nous avons utilisé l'Heuristic Miner pour générer notre modèle et qu'il peut ne pas prendre en compte toutes les traces exactement pour créer le modèle.

Nous obtenons donc le résultat des figures 4.3 et 4.4

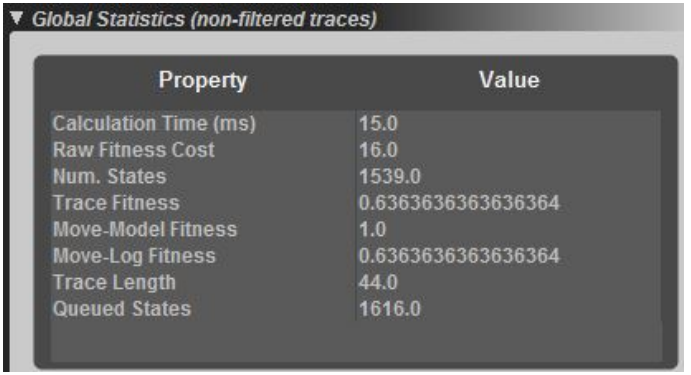

FIGURE 4.3 – Résultat de l'application d'une trace de test sur le modèle premium failed

En comparant la valeur de la ligne « Trace Fitness » pour chaque résultat, on peut voir que la valeur qui correspond au Conformance Checking du modèle « failed » est plus élevée que l'autre modèle, on peut donc conclure que la prédiction a été correcte pour cette trace.

Pour évaluer notre modèle, nous avons également rejoué cet algorithme avec l'ensemble des traces qui ont servi pour générer le modèle. de la même manière qu'avec les données de test, nous avons appliqué celles d'apprentissage sur les deux modèles.

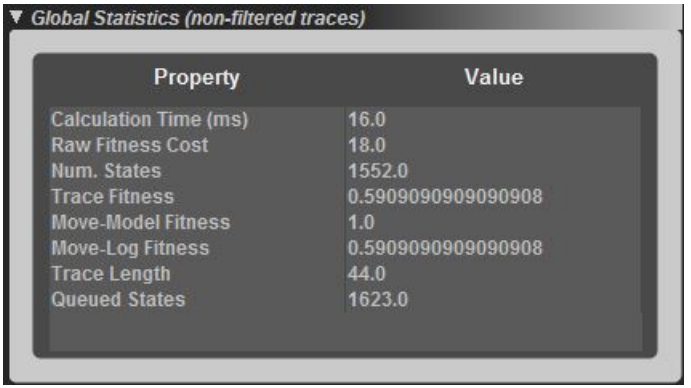

FIGURE 4.4 – Résultat de l'application d'une trace de test sur le modèle premium succeed

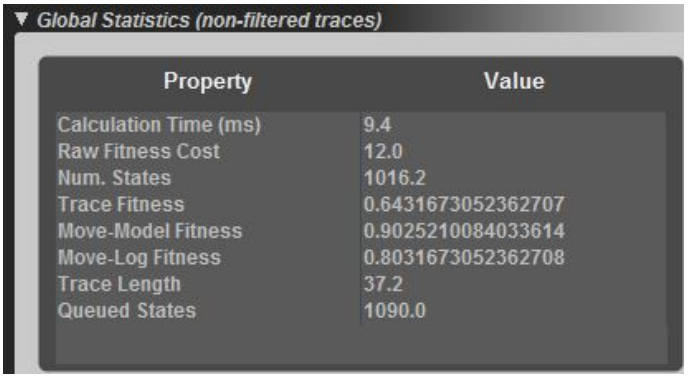

FIGURE 4.5 – Résultat de l'application des traces d'apprentissage sur le modèle premium succeed

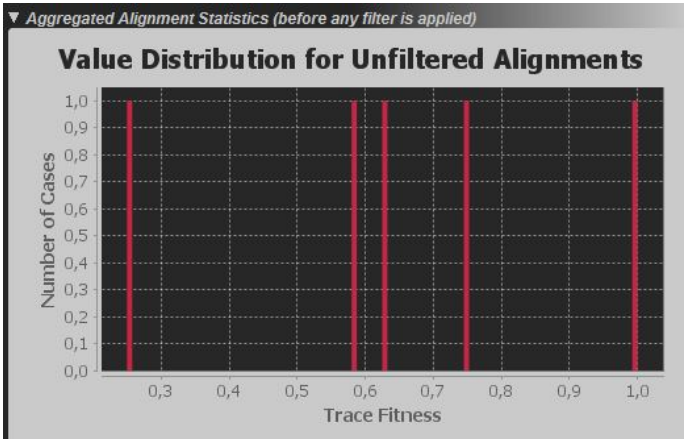

FIGURE 4.6 – valeur du fitness pour chaque trace d'apprentissage sur le modèle premium succeed

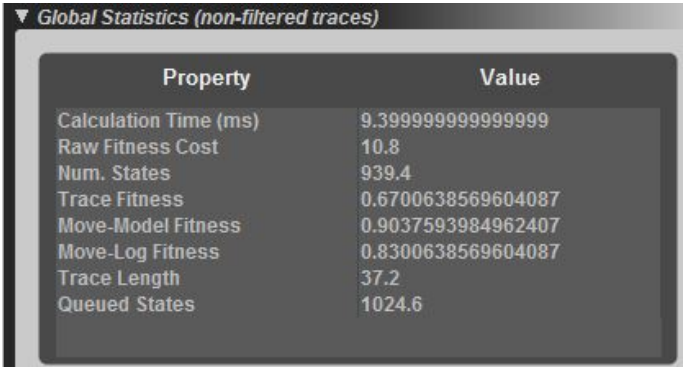

FIGURE 4.7 – Résultat de l'application des traces d'apprentissage sur le modèle premium failed

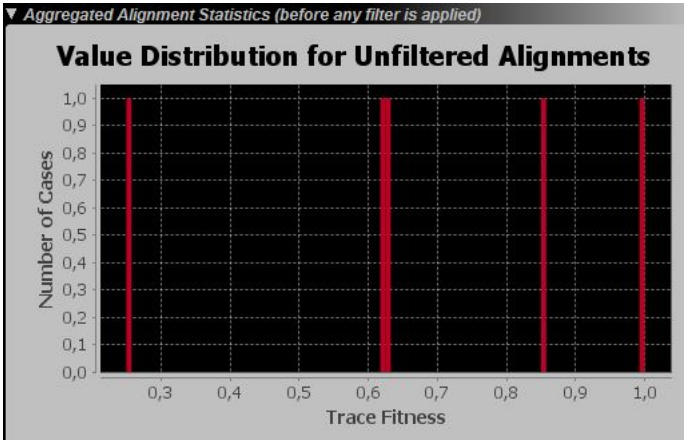

FIGURE 4.8 – valeur du fitness pour chaque trace d'apprentissage sur le modèle premium failed

Comme on peut le voir sur les figures 4.5 et 4.7, la valeur du fitness pour le modèle premium failed est plus élevée que le modèle succeed. Il s'agit d'une valeur de fitness moyenne, calculée à partir du fitness de chaque trace, que l'on peut voir sur les figures 4.6 et 4.8 Dans tous les cas, toutes les traces ne sont pas rejouées entièrement, sauf une. Il s'agit d'un comportement normal, étant donné que la fitness ne représente qu'un des critères de qualité d'un modèle en Process Mining, elle ne peut pas être de 100%, car cela pénaliserait les autres critères.

La conformance des résultats peut cependant être imprécise. En effet, il peut être intéressant de comprendre les paramètres de coût proposés par l'algorithme. Cela permettrait de prendre en compte le fait que sauter une section particulière d'un cours pénaliserait l'utilisateur, qui pourrait manquer une notion clé du cours et favoriser son échec, par exemple. De plus, selon la valeur choisie pour le paramètre pénalisant les traces rejouées entièrement ou non, le résultat final est différent. Pour notre trace de test, la valeur du fitness est plus élevée pour le modèle succeed que failed si on pénalise les traces qui ne se rejouent pas complètement, ce qui n'est pas correct étant donné que l'on sait que cette trace concerne un utilisateur qui a échoué.

4.4 Conclusion

Nous avons réussi à recréer la même chaîne de transformation d'un fichier csv que sous l'interface graphique de ProM, avec l'avantage de pouvoir automatiser les différentes étapes. De plus, notre module permet la séparation de notre fichier initial en deux fichiers. Le premier est pour l'apprentissage et donc la construction du modèle à l'aide de l'algorithme Heuristic Miner, et le deuxième pour les tests et le calcul du score de notre modèle en appliquant le Conformance Checking.

Nous sommes certains que notre programme fonctionne correctement, car il retourne des résultats identiques à ceux obtenus directement sous ProM.

En revanche, nous ne sommes pas capables d'évaluer nos modèles, étant donné que le problème vient du plugin

de ProM et qu'il n'est pas possible d'utiliser l'algorithme de conformance checking sous l'interface graphique de celui-ci. Nous avons donc utilisé un plugin alternatif avec l'interface graphique de ProM pour pouvoir réaliser des tests, qui se sont avérés concluants bien qu'ils ne soient pas forcément fiables à 100%.

5 Conclusion

5.1 Résumé du travail effectué

Tout au long de ce projet, nous avons donc étudié de nombreux algorithmes de process mining dans le but de choisir le plus adapté à notre besoin. Suite à cette étude, il s'est avéré que celui qui était le plus en accord avec notre problématique est l'heuristic miner. Pour le Conformance Checking, nous avons également conclu que l'algorithme basé sur l'utilisation de jetons était bien adapté. Nous avons donc pu trouver ces algorithmes en ligne, dont le code était sous la forme de plugins dans le logiciel spécialisé en process mining, ProM.

Nous avons pu réaliser un outil permettant de générer des modèles à partir de traces d'utilisateurs, en appliquant l'algorithme Heuristic Miner. Une fois ces modèles générés, nous avons également pu implémenter l'algorithme de Conformance Checking. Cependant, nous avons été bloqués dans son utilisation car celui-ci n'arrivait pas à rejouer nos traces, pour une raison inconnue. Cette étape ne nous a donc pas permis d'analyser les résultats comme nous l'aurions voulu. Cependant, nous avons tout de même pu effectuer quelques tests en utilisant un plugin différent de celui prévu au départ, qui applique un autre algorithme de Conformance Checking, via l'interface graphique de ProM.

5.2 Enseignements

Ce projet a été l'occasion de tirer des enseignements sur le déroulement d'un projet de recherche et développement. Tout d'abord, nous avons pu découvrir l'importance de la recherche bibliographique afin de bien cerner le domaine de recherche. C'est une étape très importante mais qui prend du temps, d'autant que tout n'est pas forcément utile dans la compréhension des éléments de réponse qui nous aident pour notre sujet. Dans la partie développement, nous avons également été confrontés à la complexité qu'est l'implémentation d'un code extérieur au nôtre, qui nous a retardé dans notre projet.

5.3 Perspectives de recherche

Nous avons précédemment que nous n'avions pas réussi à analyser les résultats de l'application du Conformance Checking avec notre application. En revanche, il est possible d'appliquer ce Conformance Checking avec l'interface graphique de ProM et l'utilisation d'autres plugins. Pour chaque cours, il est ainsi possible de comparer manuellement chaque modèle à une trace, et de trouver le modèle dont la trace se rapproche le plus. Une amélioration possible serait donc de trouver une alternative au plugin trouvé qui permettrait l'utilisation d'un autre algorithme de Conformance Checking.

Bibliographie

[vdA11] Wil M. P. van der Aalst. *Process Mining : Discovery, Conformance and Enhancement of Business Processes*. Springer Publishing Company, Incorporated, 1st edition, 2011. 10, 13, 14, 15, 16, 17, 20, 22, 23, 24, 47, 49, 51, 54

[WBVB12] Jochen De Weerdt, Manu De Backer, Jan Vanthienen, and Bart Baesens. A multi-dimensional quality assessment of state-of-the-art process discovery algorithms using real-life event logs. *Information Systems*, 37(7) :654 – 676, 2012. 21, 25, 29, 31, 52

Table des figures

Liste des tableaux

Liste des algorithmes

Fiches de lecture

A.1 Process Mining : Discovery, Conformance and Enhancement of Business Processes

Ce livre ([vdA11]) est une initiation aux techniques de process mining ainsi que tout ce qui est relatif à ce domaine pour fournir une compréhension générale des enjeux et fonctionnement du process mining.

A.1.1 Résumé

Ce livre est découpé en plusieurs parties majeures. Dans l'introduction, qui reste assez générales, l'auteur montre les grands principes du process mining et ses enjeux actuels, quel est son rôle dans une organisation et comment il s'intègre. Il familiarise le lecteur avec les éléments les plus importants dans la compréhension du livre, notamment le vocabulaire technique. Dans la première partie, l'auteur présente les différentes façons de modéliser les process models et fait un état sur les techniques de data mining.

Dans une seconde partie, L'auteur commence par exposer un algorithme de process mining simple pour expliquer le principe. Il présente les critères d'importance pour un algorithme de process mining, ainsi que d'autres algorithmes plus avancés, avec leur limitations et leurs avantages. Dans une troisième partie, il expose d'autres techniques liées au process mining, notamment le conformance checking, qui est une méthode d'évaluation de modèle. Il présente ainsi plusieurs moyens de calculer cette conformité.

A.1.2 Analyse

Ce livre est une très bonne référence dans l'apprentissage des techniques de process mining, son auteur étant l'une des références dans ce domaine, avec de nombreuses études à ce sujet. Malgré sa technicité, ce livre reste compréhensible et présente une approche simple sur

le process mining, ses techniques, ses enjeux. Il a été d'une grande aide dans notre projet car c'est lui qui explique tous les fondamentaux dans ce domaine, et tout ce qu'il y a à savoir dessus. Dans son livre, l'auteur présente de nombreux algorithmes, que nous avons donc dû étudier dans le cadre de notre projet. Les explications étaient donc d'une grande utilité. Il nous a permis d'avoir une vision d'ensemble plus claire sur le sujet, car le but de cette recherche bibliographique est de choisir l'algorithme le plus adapté à nos besoins, et pour cela les explications que l'auteur a données sont d'une grande utilité.

A.2 A multi-dimensional quality assessment of state-of-the-art process discovery algorithms using real-life event logs

Cet article ([WBVB12]) propose une description succinte de plusieurs algorithmes de process mining, puis effectue une comparaison de certains d'entre eux.

A.2.1 Résumé

Les auteurs effectuent une comparaison de plusieurs algorithmes de process mining. Ils expliquent que les journaux d'événements artificiels sont très différents des réels journaux d'événements produits par les organisations, car ils sont souvent plus petits et moins complexes que ces derniers. L'objectif de l'article est donc de réaliser une comparaison sur des journaux d'événements provenant de plusieurs organisations comme une université ou des entreprises.

Pour se faire, ils utilisent une évaluation multi-dimensionnelle en se focalisant sur des mesures d'exactitudes et de compréhensibilité.

La comparaison a été réalisé sur sept algorithmes différents :

— Alpha+ 2.6.3
— Alpha++ 2.6.3
— AGNEsMiner 2.10
— Genetic Miner 2.9
— Duplicate Task Genetic Miner 2.9.3
— Heuristic Miner 2.8
— ILP Miner 2.11

L'algorithme Flower Miner 2.7 a également été utilisé afin de servir de modèle de référence. En effet, il possède un rappel maximal (généralisation élevée) car il autorise toute les traces. En revanche, il n'est pas beaucoup précis.

D'autres algorithmes présentés dans l'article n'ont pas été comparés car soit ils ne sont pas disponibles publiquement, soit ils ne génèrent pas un réseau de Pétri ou une autre modélisation pouvant être traduit en réseau de Pétri (et donc étant difficilement comparable avec les autres algorithmes), soit les algorithmes n'ont pas une grande scalabilité, et donc ne sont pas applicables à des journaux d'événements réels.

Le résultat de l'étude est que les algorithmes produisent des résultats assez proches dans le cas où les journaux d'événements utilisés ont été créés artificiellement, à l'exception faite de l'algorithme Heuristic miner qui est un peu moins bon.

Cependant, l'algorithme Heuristic miner est le plus performant sur des journaux d'événements réels. L'algorithme ILP possède quant à lui le meilleur taux de rappel. Au niveau des F-Score, c'est l'heuristic miner qui est le meilleur, même si l'AGNEsMiner et les algorithmes génétiques produisent de bons scores également.

Cependant, lorsqu'on compare au niveau de la compréhensibilité, les modèles générés par l'ILP Miner sont très mauvais. En effet, ils contiennent énormément d'arcs, ce qui les rend difficilement lisibles et compréhensibles. L'heuristic miner et les deux algorithmes génétiques possèdent des activités qui ont été rajoutées du fait de la traduction des modèles générés en réseaux de Pétri, cependant ils possèdent beaucoup moins d'arcs.

Les algorithmes génétiques produisent des modèles qui sont très compliqués à comprendre et prennent beaucoup de temps à s'exécuter, ce qui ne les rends pas adaptés aux journaux d'événements réels. En effet, ceci sont plus grands et plus complexes que les journaux d'événements créés artificiellement.

Enfin, les algorithmes alpha (alpha+ et alpha++) sont les moins bons.

A.2.2 Analyse

Cet article est intéressant pour notre projet de recherche et développement car il effectue une comparaison sur des journaux d'événements générés par de véritables organisations, et non pas créés artificiellement. En effet, les événements sur lesquels nous allons devoir créer des modèles à l'aide d'algorithmes de process mining sont également générés par une entreprise, et non pas créés artificiellement.

De plus, l'article nous a permis de dégager un algorithme qui est particulièrement efficace sur de vrais journaux d'événements, à savoir l'algorithme *Heuristic Miner* 2.8.

B

Planification

La figure B.1 présente le planning prévisionnel. Ce planning a notamment été réalisé à l'aide des conseils et indications d'avancements présents sur le cours « Projet de recherche et développement » disponible sur la plateforme pédagogique Madoc.

En toute logique, nous avons estimé que le projet débuterait par une courte période d'acquisition du sujet. Ensuite, une phase d'étude bibliographique sera réalisée suivie par des propositions de solutions. Nous avons prévu deux semaines pour la rédaction du rapport intermédiaire ainsi que pour la préparation de la soutenance. Après cette partie plus théorique, une phase de développement devrait être initiée, suivie par des tests d'intégrations et de validation. Enfin une analyse des résultats sera effectuée, tout de suite suivie par la rédaction du rapport final, puis par une dernière semaine de corrections.

La figure B.2 présente le planning réalisé, c'est-à-dire le planning qui a été tenu jusqu'à présent et la mise à jour des tâches restantes dans le cas où du retard aurait été pris.

Une rapide comparaison des plannings nous montre que la phase d'étude bibliographique a pris plus de temps que prévu. Il y a eu plusieurs raisons à cela. En effet, nous avons eu à lire un livre centré sur le thème du *process mining* ([vdA11]), qui a une taille assez conséquente et étant sur un domaine que nous ne connaissions pas. La taille de ce livre est un des éléments qui nous a fait prendre du retard. La recherche de stage et les entretiens réalisés sont les autres raisons de ce retard.

Nous pouvons également remarquer que lors de la semaine du 28 novembre au 4 décembre, nous avons cumulé plusieurs tâches (étude bibliographique, proposi-

tions de solution, rapport intermédiaire) afin de rattraper notre retard.

La figure B.3 présente le planning réalisé final. La partie développement a pris beaucoup plus de temps que prévu, et cela pour plusieurs raisons : nous avons eu du mal à comprendre comment récupérer les archives JAR des plugins contenant les algorithmes que nous souhaitions utiliser. De plus, nous avons également mis du temps à comprendre comment utiliser ces derniers.

Le nombre de projets / TP à rendre en parallèle de ce projet est l'autre principale raison de notre retard pris sur cette phase du projet. En effet, nous avons cumulé beaucoup de projets différents sur la même période.

Nous pouvons notamment remarquer que la phase d'analyse des résultats n'est pas présente. En effet, comme nous l'avons vu dans le rapport, nous avons rencontré un problème lié à un plugin du logiciel ProM, qui nous empêche d'utiliser un algorithme de Conformance Checking pour évaluer notre modèle.

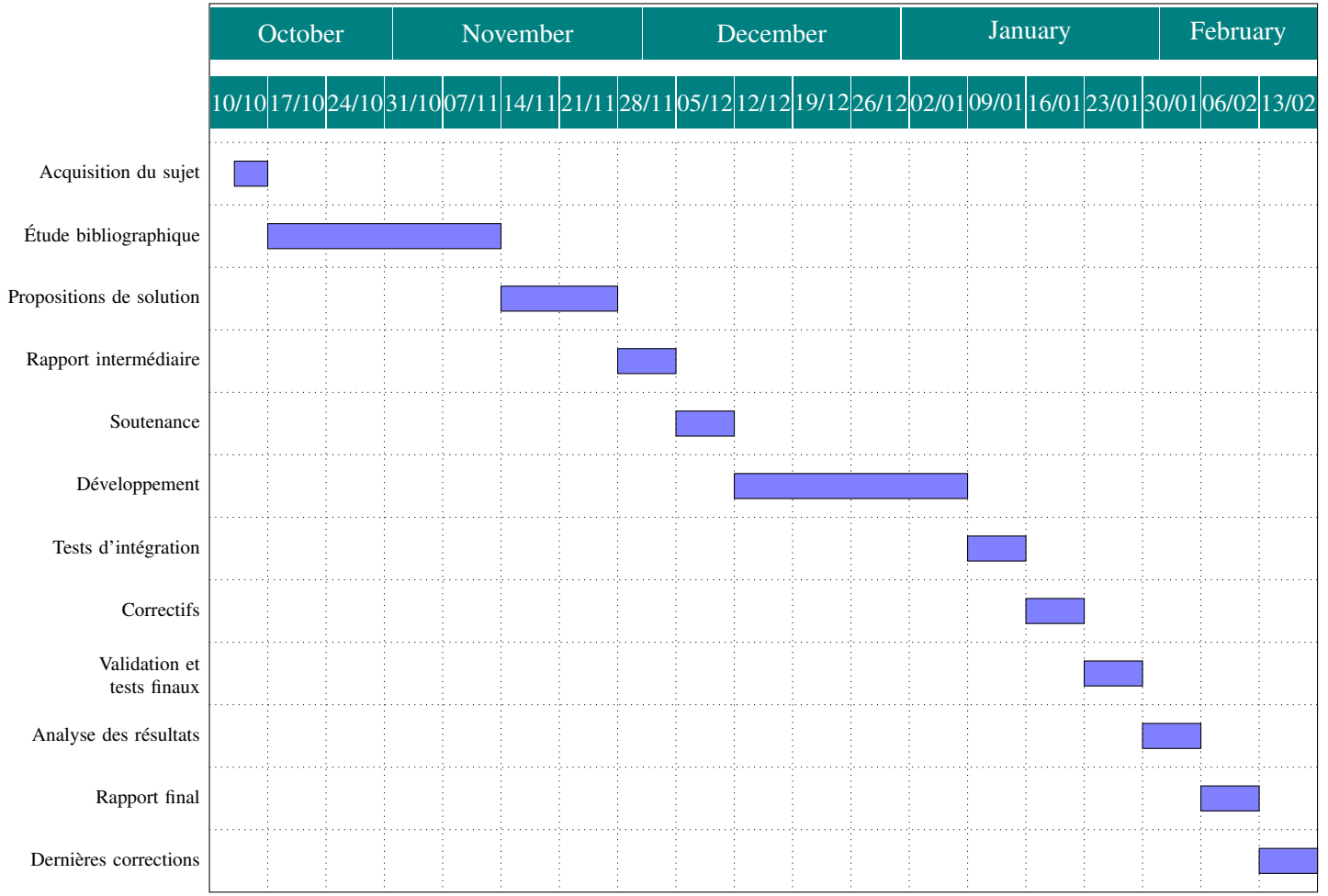

FIGURE B.1 – Planification prévisionnelle

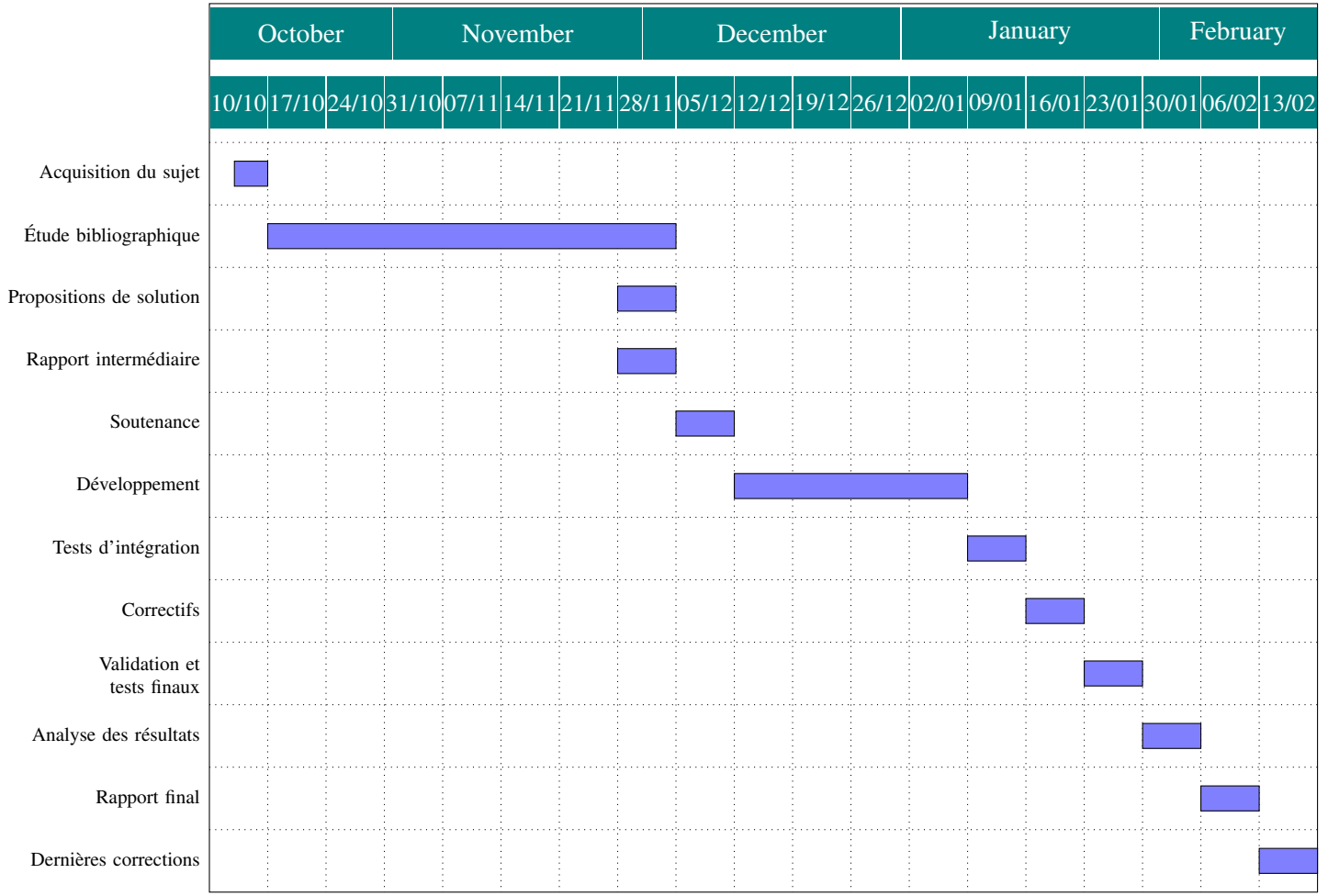

FIGURE B.2 – Planning effectif

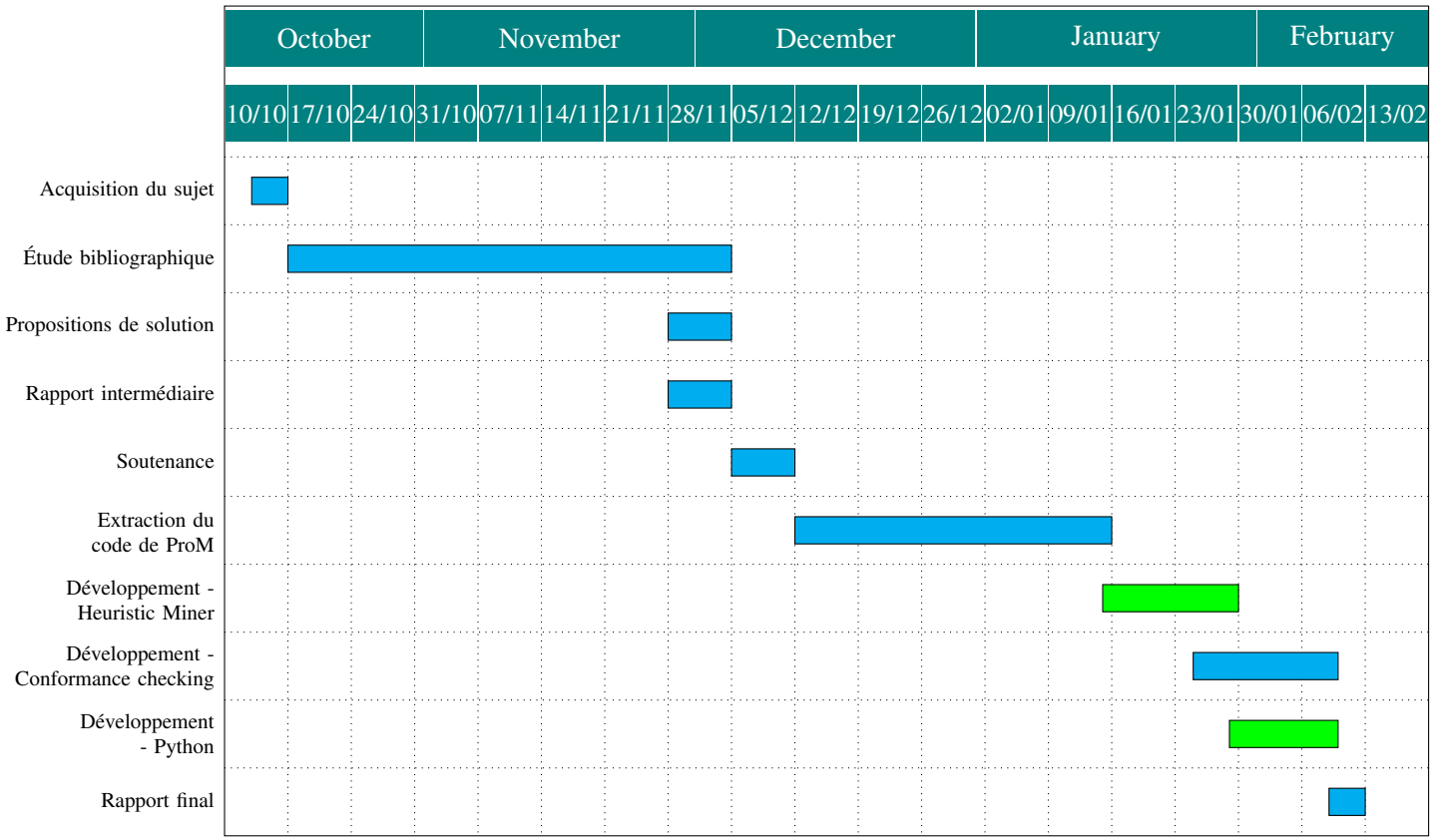

FIGURE B.3 – Planning effectif final (Damien et Cyril ; Damien)

C

Fiches de suivi

Fiche de suivi de la semaine 1
du 13 octobre 2016 au 16 octobre 2016

Temps de travail de Damien LE FLOHIC: 2 h 00 m
Temps de travail de Cyril CARON: 2 h 00 m

Travail effectué.

- Prise de contact avec le coordinateur (Antoine Pigeau) ;
- Compte-rendu de la première réunion ;
- Rédaction de l'introduction (première ébauche).

Échanges avec le commanditaire.

- Réunion avec le coordinateur (Antoine Pigeau) le 13/10/2016 ;
- Partage des éléments de travail (google docs, bibliographie, code).

Planification pour la semaine prochaine.

- Planifier une réunion pour la découverte des outils à utiliser ;
- Commencer à lire le livre et les articles de la bibliographie ;
- Début de la rédaction de l'étude de la bibliographique ;
- Installation et prise en main de Disco ;
- Regarder le code fourni.

Fiche de suivi de la semaine 2
du 16 octobre 2016 au 23 octobre 2016

Temps de travail de Damien LE FLOHIC: 6 h 00 m
Temps de travail de Cyril CARON: 6 h 00 m

Travail effectué.

- Prise en main des outils de travail (Disco, ProM) ;
- Étude des données fournies ;
- Lecture du livre « Process mining - Discovery, Conformance and enhancement of Business Processes »

— Amélioration de l'introduction et de la problématique du projet (deuxième version).

Échanges avec le commanditaire.

— Réunion avec Antoine Pigeau le 20/10/2016 pour la présentation du code déjà réalisé ainsi que des outils de process mining (Disco et ProM).

Planification pour la semaine prochaine.

— Avancer l'étude bibliographique (lecture du livre) ;
— Début de la rédaction de la bibliographie ;

Fiche de suivi de la semaine 3
du 30 octobre 2016 au 6 novembre 2016

Temps de travail de Damien LE FLOHIC: 3 h 00 m
Temps de travail de Cyril CARON: 1 h 00 m

Travail effectué.

— Poursuite de la lecture du livre de Wil van der Aalst.

Travail non effectué.

— Début de la rédaction de la bibliographie. Nous avons pris du retard sur cette tâche car nous avons été pris par les simulations d'entretiens et la recherche de stage. Ce retard sera rattrapé la semaine du 7 novembre au 13 novembre.

Planification pour la semaine prochaine.

— Avancer l'étude bibliographique (livres + articles sur le Process Mining) ;
— Début de la rédaction de la bibliographie ;

Fiche de suivi de la semaine 4
du 14 novembre 2016 au 20 novembre 2016

Temps de travail de Damien LE FLOHIC: 9 h 00 m

Temps de travail de Cyril CARON: 7 h 00 m

Travail effectué.

— Poursuite de la lecture du livre de Wil van der Aalst.
— Rédaction de l'état de l'art (notamment les algorithmes Alpha et Genetic Process Mining)

Planification pour la semaine prochaine.

— Avancer l'étude bibliographique (livres + articles sur le Process Mining) ;
— Finir l'état de l'art
— Rechercher l'algorithme à retenir par la suite

Fiche de suivi de la semaine 5
du 20 novembre 2016 au 27 novembre 2016

Temps de travail de Damien LE FLOHIC: 9 h 00 m
Temps de travail de Cyril CARON: 9 h 00 m

Travail effectué.

- Poursuite de la lecture du livre de Wil van der Aalst.
- Avancement de l'état de l'art (Étude et rédaction de la bibliographie)

Échanges avec le commanditaire.

- Vendredi 25 novembre : réunion avec Antoine Pigeau sur l'avancement du projet.

Planification pour la semaine prochaine.

- Finir l'état de l'art : Rédaction de la bibliographie, comparaison des algorithmes et choix du meilleur algorithme.

Fiche de suivi de la semaine 6
du 27 novembre 2016 au 04 décembre 2016

Temps de travail de Damien LE FLOHIC: 24 h 30 m
Temps de travail de Cyril CARON: 25 h 30 m

Travail effectué.

- Rédaction de l'algorithme Region based mining et Conformance checking
- Rédaction des fiches de lecture
- Correction du rapport suite aux remarques
- Ajout des plannings dans le rapport
- Lecture d'un article comparant différents algorithmes de process mining (multi-dimensional quality assessment of state-of-the-art process discovery algorithms using real-life event logs)
- Rédaction de nouveaux algorithmes (Agnes, ILP, Alpha+, Alpha++, DT genetic miner)
- Rédaction des remerciements
- Rédaction du résumé
- Rédaction de la partie « Propositions »

Planification pour la semaine prochaine.

- Préparer la soutenance
- Corriger le rapport suite aux remarques reçues pendant la soutenance.

Fiche de suivi de la semaine 7
du 04 décembre 2016 au 11 décembre 2016

Temps de travail de Damien LE FLOHIC: 11 h 30 m
Temps de travail de Cyril CARON: 10 h 00 m

Travail effectué.

- Préparer la soutenance

— Corriger le rapport suite aux remarques reçues pendant la soutenance.
— Chercher et récupérer les classes Java de l'algorithme Heuristic Miner.
— Se documenter sur l'API du logiciel ProM
— Suivi du MOOC « Process Mining : Data science in Action » sur Coursera

Planification pour la semaine prochaine.

— Se familiariser avec le code existant
— Établir un rendez-vous pour organiser la suite du projet

Fiche de suivi de la semaine 8
du 09 janvier 2017 au 15 janvier 2017

Temps de travail de Damien LE FLOHIC: 12 h 00 m
Temps de travail de Cyril CARON: 12 h 00 m

Travail effectué.

— Récupération du code du plugin de l'heuristic miner de ProM et des dépendances associées.
— Recherche d'informations sur les plugins de ProM liés au Conformance Checking (papiers des auteurs pour décrire leurs méthodes).

Travail non effectué.

— Nous n'avons pas encore construit des modèles en utilisant le plugin Heuristic Miner de ProM.

Échanges avec le commanditaire.

— Nous avons eu une réunion pour faire état de l'avancement du projet et faire le point sur les difficultés que nous avons rencontrées.

Planification pour la semaine prochaine.

— Organiser une réunion pour faire le point sur nos recherches de cette semaine.
— Poursuivre nos recherches sur le code de ProM et essayer de comprendre comment utiliser le plugin de l'Heuristic Miner.

Fiche de suivi de la semaine 9
du 15 janvier 2017 au 22 janvier 2017

Temps de travail de Damien LE FLOHIC: 10 h 00 m
Temps de travail de Cyril CARON: 10 h 00 m

Travail effectué.

— Tentative d'utilisation du CLI de ProM (Command-Line Interface)
— Conversion des fichiers CSV en fichier XES (format de logs exploitable par l'algorithme Heuristic Miner)
— Application de l'algorithme Heuristic Miner sur les fichiers XES générés.
— Prise en main de RapidMiner, outil pour appliquer un algorithme de Process Mining.

Travail non effectué.

- Nous n'avons pas encore construit des modèles en utilisant le plugin Heuristic Miner de ProM (en cours).
- Approfondir l'utilisation du code, notamment le réglage des paramètres de l'algorithme Heuristic Miner.

Échanges avec le commanditaire.

- Nous avons réalisé une réunion pour faire le point sur les difficultés que nous avons rencontrées.

Planification pour la semaine prochaine.

- Finir de réaliser le code permettant d'appliquer l'heuristic miner sur nos données (exportation des résultats obtenus).
- Appliquer le conformance checking sur les modèles obtenus.

Fiche de suivi de la semaine 10
du 22 janvier 2017 au 29 janvier 2017

Temps de travail de Damien LE FLOHIC: 24 h 30 m
Temps de travail de Cyril CARON: 12 h 00 m

Travail effectué.

- Comprendre plus en détail les étapes pour créer un Heuristics Net
- Paramétrer l'algorithme Heuristics Miner
- Vérifier les résultats à l'aide de ProM (résultats identiques)
- Création du répertoire Gitlab
- Création du fichier Readme.md expliquant comment obtenir les différents fichiers .jar (plugins de prom).
- Début de la création du code pour le conformance checking

Travail non effectué.

- Appliquer le conformance checking sur les modèles obtenus (en cours)

Échanges avec le commanditaire.

- Échange de mails liés à l'avancement du projet et au gitlab du projet.

Planification pour la semaine prochaine.

- Appliquer le conformance checking sur les modèles obtenus.
- Appeler le code à partir d'un programme python
- Réaliser les tests
- Rédiger le rapport et commentaires sur les résultats obtenus

---

Fiche de suivi de la semaine 11
du 29 janvier 2017 au 05 février 2017

Temps de travail de Damien LE FLOHIC: 17 h 00 m

Temps de travail de Cyril CARON: 10 h 00 m

Travail effectué.

- Programme python créé
- Programme java permettant de générer un Petri Net à partir d'un fichier de logs au format .csv puis de l'exporter en fichier .pnml
- Programme java permettant d'importer le fichier .pnml généré, puis applique un algorithme de conformance checking (plugin "PN Conformace Analysis" dans ProM). Cependant, le plugin met beaucoup trop de temps à s'exécuter (c'est également le cas dans ProM)
- Programme python qui génère un train set et un test set et qui appelle ensuite les deux fichiers java
- Amélioration du code (refactoring + commentaires)

Travail non effectué.

- Analyse des résultats

Planification pour la semaine prochaine.

- Rédaction du rapport final
- Création d'un manuel technique

Fiche de suivi de la semaine 12
du 05 février 2017 au 12 février 2017

Temps de travail de Damien LE FLOHIC: 17 h 00 m

Temps de travail de Cyril CARON: 16 h 00 m

Travail effectué.

- Résolution des problèmes liées à l'intégration de notre package au projet déjà existant.
- Modification du code python (interface)
- Rédaction du rapport

Travail non effectué.

- Application du Conformance Checking
- Analyse des résultats

Planification pour la semaine prochaine.

- Préparation de la soutenance

Le tableau C.1 récapitule le taux d'avancement du projet. Rappelons que le temps de travail théorique *minimal* correspond au temps indiqué sur la maquette pédagogique auquel on ajoute un strict minimum de 20 % correspondant au travail personnel hors emploi du temps. La partie « haute » de la fourchette correspond à 50 % de temps supplémentaire au titre du travail personnel.

Nous pouvons remarquer que la première semaine nous n'avions travaillé que deux heures sur les dix de prévues. Cela est normal car c'était la première semaine, où il y avait une partie de prise en main du sujet, de rendez-vous avec le tuteur pour expliquer un peu plus en détail le projet.

	Temps prévu		Damien LE FLOHIC			Cyril CARON		
	bas	haut	hebdo.	Σ	%	hebdo.	Σ	%
Semaine	h : m	h : m	h : m	h : m		h : m	h : m	
1	10 : 00	12 : 30	2 : 00	2 : 00	20 (16)	2 : 00	2 : 00	20 (16)
2	20 : 00	25 : 00	6 : 00	8 : 00	40 (32)	6 : 00	8 : 00	40 (32)
3	30 : 00	37 : 30	3 : 00	11 : 00	36 (29)	1 : 00	9 : 00	30 (24)
4	40 : 00	50 : 00	9 : 00	20 : 00	50 (40)	7 : 00	16 : 00	40 (32)
5	50 : 00	62 : 30	9 : 00	29 : 00	58 (46)	9 : 00	25 : 00	50 (40)
6	60 : 00	75 : 00	24 : 30	53 : 30	89 (71)	25 : 30	50 : 30	84 (67)
7	70 : 00	87 : 30	11 : 30	65 : 00	92 (74)	10 : 00	60 : 30	86 (69)
8	80 : 00	100 : 00	12 : 00	77 : 00	96 (77)	12 : 00	72 : 30	90 (72)
9	90 : 00	112 : 30	10 : 00	87 : 00	96 (77)	10 : 00	82 : 30	91 (73)
10	100 : 00	125 : 00	24 : 30	111 : 30	111 (89)	12 : 00	94 : 30	94 (75)
11	110 : 00	137 : 30	17 : 00	128 : 30	116 (93)	10 : 00	104 : 30	95 (76)
12	120 : 00	150 : 00	17 : 00	145 : 30	121 (97)	16 : 00	120 : 30	100 (80)

TABLE C.1 – Avancement du projet par rapport au temps de travail théorique minimal (respectivement haut)

Auto-contrôle et auto-évaluation

Les figures D.1 et D.2 permet d'énumérer un certain nombre de points importants dans les trois composantes du travail :

1. rapport ;
2. présentation orale ;
3. travail de fond ;

ainsi que d'évaluer notre niveau de satisfaction à l'issue de la phase I, composée de trois étapes :

1. étude préalable ;
2. étude bibliographique ;
3. conception générale.

Les points de satisfaction ou d'insatisfaction peuvent être approfondis.

PRO	Prénom	Nom
Binome 1	Damien	LE FLOHIC
Binome 2	Cyril	CARON

Notation	
A	Maîtrise dans l'application du savoir-faire requis
B	Application du savoir-faire requis
C	Insuffisances, lacunes à corriger dans l'application du savoir-faire requis *a minima*
D	Insuffisances flagrantes, inacceptables, voire travail absent

PROPOSITION DE NOTE (I)	**#DIV/0 !**
PROPOSITION DE NOTE (II)	**#DIV/0 !**
NOTE PROJET	**#NOM ?**

Phase I : Etude préalable, étude bibliographique et conception générale

				A	B	C	D	Remarque / Note / Commentaire
Rapport	Organisation	Plan	Equilibre	x				
			Cohérence	x				
		Fluidité	Introductions (partielles)	x				
			Transitions		x			
			Conclusions (partielles)	x				
		Tableaux, figures	Numérotés	x				
			Légendés	x				
			Référencés (non "en ligne")	x				
	Rédaction	Orthographe	Coquilles	x				
			Fautes évitables	x				
			Franglais, jargon		x			
		Rédaction	Aisée	x				
			Absence de plagiat !	x				
	Bibliographie	Références	Suffisantes (nombre, intérêt)	x				
			Pérennes	x				
			Complètes (auteurs, pages...)	x				
			Conséquentes (volume)	x				
		Références dans le texte		x				

Proposition de note haute	18,44
Proposition de note basse	13,44
Proposition de note du jury	

Projection	Organisation	Plan						Soutenance pas encore réalisée
		Liaisons						Soutenance pas encore réalisée
		Numérotation						Soutenance pas encore réalisée
	Contenu	Informatif						Soutenance pas encore réalisée
		Concis						Soutenance pas encore réalisée
		Clair						Soutenance pas encore réalisée
		Orthographe						Soutenance pas encore réalisée
		Illustrations						Soutenance pas encore réalisée
Oral	Présentation	Aisance						Soutenance pas encore réalisée
		Tenue						Soutenance pas encore réalisée
		Articulation, compréhension						Soutenance pas encore réalisée
	Durée	Respect						Soutenance pas encore réalisée
		Temps de parole équilibré						Soutenance pas encore réalisée
	Réponses	Pertinence						Soutenance pas encore réalisée
		Argumentation						Soutenance pas encore réalisée

Proposition de note haute	0,00
Proposition de note basse	0,00
Proposition de note du jury	

Travail	Etude	Bibliographie	Adéquate		x			
			Suffisante		x			
		Cahier des charges	Clair	x				
			Formalisé		x			
		Hypothèses envisagées	Nombre			x		
			Pertinence		x			
			Analysée *a priori*			x		
		Validation	Tableau comparatif			x		
			Choix argumenté(s)		x			
			Faisabilité	x				
	Complexité	Temps consacré			x			
		Résultats obtenus		x				
		Difficulté	Intrinsèque		x			
			Vis-à-vis du binôme	x				
	Annexes	Fiches d'avancement	Régularité			x		
			Détaillées		x			
		Gantt	Prévisionnel et justifications	x				
			Effectifs, erreurs, corrections		x			

Proposition note haute	15,30
Proposition note basse	10,30
Proposition de note du jury	

Phase II : Conception détaillée, réalisation et recette

Rapport	Organisation	Plan	Equilibre	?				
			Cohérence	?				
		Fluidité	Introductions (partielles)	?				
			Transitions	?				
			Conclusions (partielles)	?				
		Tableaux, figures	Numérotés	?				
			Légendés	?				
			Référencés (non "en ligne")	?				
	Rédaction	Orthographe	Coquilles	?				
			Fautes évitables	?				
			Franglais, jargon	?				
		Rédaction	Aisée	?				
			Absence de plagiat !	?				

Proposition de note haute	0,00
Proposition de note basse	0,00
Proposition de note du jury	

Projection	Organisation	Plan		?				
		Liaisons		?				
		Numérotation		?				
	Contenu	Informatif		?				
		Concis		?				
		Clair		?				
		Orthographe		?				
		Illustrations		?				
Oral	Présentation	Aisance		?				
		Tenue		?				
		Articulation, compréhension		?				
	Durée	Respect		?				
		Temps de parole équilibré		?				
	Réponses	Pertinence		?				
		Argumentation		?				

Proposition de note haute	0,00
Proposition de note basse	0,00
Proposition de note du jury	

Travail	Conception	Générale	Clarté	?				
			Niveau de détail adéquat	?				
			Argumentation	?				
			Validation	?				
		Détaillée	Formalisation adéquate (algorithme...)	?				
			Validation	?				
	Réalisation	Développement	"Fonctionnalités"	?				
			Volume, environnement...	?				
		Tests (unitaires, d'intégration, fonctionnel, de performance...)		?				
	Recette	Livraison du paquetage		?				
		Redéploiement du paquetage		?				
	Complexité	Temps consacré		?				
		Résultats obtenus		?				
		Difficulté	Intrinsèque	?				
			Vis-à-vis du binôme	?				
	Annexes	Fiches d'avancement	Régularité	?				
			Détaillées	?				
		Gantt	Prévisionnel et justifications	?				
			Effectif, erreurs, corrections	?				

Proposition de note haute	0,00
Proposition de note basse	0,00
Proposition de note du jury	

FIGURE D.1 – Points à contrôler à l'issue de la phase I

PRO	Prénom	Nom
Binome 1	Damien	LE FLOHIC
Binome 2	Cyril	CARON

Notation	
A	Maîtrise dans l'application du savoir-faire requis
B	Application du savoir-faire requis
C	Insuffisances, lacunes à corriger dans l'application du savoir-faire requis *a minima*
D	Insuffisances flagrantes, inacceptables, voire travail absent

PROPOSITION DE NOTE (I)	**#DIV/0 !**
PROPOSITION DE NOTE (II)	**#DIV/0 !**
NOTE PROJET	**#NOM ?**

Phase I : Etude préalable, étude bibliographique et conception générale

				A	B	C	D	Remarque / Note / Commentaire
Rapport	Organisation	Plan	Equilibre	x				
			Cohérence	x				
		Fluidité	Introductions (partielles)	x				
			Transitions			x		
			Conclusions (partielles)	x				
		Tableaux, figures	Numérotés	x				
			Légendés	x				
			Référencés (non "en ligne")	x				
	Rédaction	Orthographe	Coquilles	x				
			Fautes évitables	x				
			Franglais, jargon			x		
		Rédaction	Aisée	x				
			Absence de plagiat !	x				
	Bibliographie	Références	Suffisantes (nombre, intérêt)	x				
			Pérennes	x				
			Complètes (auteurs, pages...)	x				
			Conséquentes (volume)	x				
		Références dans le texte		x				

Proposition de note haute	18,44
Proposition de note basse	13,44
Proposition de note du jury	

Projection	Organisation	Plan			X			
		Liaisons		X				
		Numérotation		X				
	Contenu	Informatif		X				
		Concis			X			
		Clair			X			
		Orthographe		X				
		Illustrations		X				
Oral	Présentation	Aisance			X			
		Tenue			X			
		Articulation, compréhension			X			
	Durée	Respect			X			
		Temps de parole équilibré		X				
	Réponses	Pertinence				X		
		Argumentation				X		

Proposition de note haute	15,56
Proposition de note basse	10,74
Proposition de note du jury	

Travail	Etude	Bibliographie	Adéquate		x			
			Suffisante		x			
		Cahier des charges	Clair	x				
			Formalisé		x			
		Hypothèses envisagées	Nombre			x		
			Pertinence		x			
			Analyse *a priori*			x		
		Validation	Tableau comparatif			x		
			Choix argumenté(s)		x			
			Faisabilité	x				
	Complexité	Temps consacré			x			
		Résultats obtenus		x				
		Difficulté	Intrinsèque		x			
			Vis-à-vis du binôme	x				
	Annexes	Fiches d'avancement	Régularité			x		
			Détaillées		x			
		Gantt	Prévisionnel et justifications	x				
			Effectifs, erreurs, corrections		x			

Proposition note haute	15,30
Proposition note basse	10,30
Proposition de note du jury	

Phase II : Conception détaillée, réalisation et recette

Rapport	Organisation	Plan	Equilibre	X				
			Cohérence	X				
		Fluidité	Introductions (partielles)	X				
			Transitions			X		
			Conclusions (partielles)	X				
		Tableaux, figures	Numérotés	X				
			Légendés	X				
			Référencés (non "en ligne")	X				
	Rédaction	Orthographe	Coquilles	X				
			Fautes évitables	x				
			Franglais, jargon		X			
		Rédaction	Aisée	X				
			Absence de plagiat !	X				

Proposition de note haute	19,09
Proposition de note basse	14,09
Proposition de note du jury	

Projection	Organisation	Plan		?				
		Liaisons		?				
		Numérotation		?				
	Contenu	Informatif		?				
		Concis		?				
		Clair		?				
		Orthographe		?				
		Illustrations		?				
Oral	Présentation	Aisance		?				
		Tenue		?				
		Articulation, compréhension		?				
	Durée	Respect		?				
		Temps de parole équilibré		?				
	Réponses	Pertinence		?				
		Argumentation		?				

Proposition de note haute	0,00
Proposition de note basse	0,00
Proposition de note du jury	

Travail	Conception	Générale	Clarté			x		
			Niveau de détail adéquat		x			
			Argumentation		x			
			Validation	x				
		Détaillée	Formalisation adéquate (algorithme...)	x				
			Validation	x				
	Réalisation	Développement	"Fonctionnalités"			x		
			Volume, environnement...			x		
		Tests (unitaires, d'intégration, fonctionnel, de performance...)				x		
	Recette	Livraison du paquetage		x				
		Redéploiement du paquetage		x				
	Complexité	Temps consacré				x		
		Résultats obtenus			x			
		Difficulté	Intrinsèque			x		
			Vis-à-vis du binôme		x			
	Annexes	Fiches d'avancement	Régularité		x			
			Détaillées			x		
		Gantt	Prévisionnel et justifications		X			
			Effectif, erreurs, corrections			X		

Proposition de note haute	15,00
Proposition de note basse	10,00
Proposition de note du jury	

FIGURE D.2 – Points à contrôler à l'issue de la phase II

PRO	Prénom	Nom
Binome 1	Damien	LE FLOHIC
Binome 2	Cyril	CARON

Notation	
A	Maîtrise dans l'application du savoir-faire requis
B	Application du savoir-faire requis
C	Insuffisances, lacunes à corriger dans l'application du savoir-faire requis *a minima*
D	Insuffisances flagrantes, inacceptables, voire travail absent

PROPOSITION DE NOTE (I)	**#DIV/0 !**
PROPOSITION DE NOTE (II)	**#DIV/0 !**
NOTE PROJET	**#NOM ?**

Phase I : Etude préalable, étude bibliographique et conception générale

				A	B	C	D	Remarque / Note / Commentaire
Rapport	Organisation	Plan	Equilibre	x				
			Cohérence	x				
		Fluidité	Introductions (partielles)	x				
			Transitions		x			
			Conclusions (partielles)		x			
		Tableaux, figures	Numérotés	x				
			Légendés	x				
			Référencés (non "en ligne")	x				
	Rédaction	Orthographe	Coquilles	x				
			Fautes évitables	x				
			Franglais, jargon		x			
		Rédaction	Aisée	x				
			Absence de plagiat !	x				
	Bibliographie	Références	Suffisantes (nombre, intérêt)	x				
			Pérennes	x				
			Complètes (auteurs, pages...)	x				
			Conséquentes (volume)	x				
		Références dans le texte		x				

Proposition de note haute	18,44
Proposition de note basse	13,44
Proposition de note du jury	

Projection	Organisation	Plan			X			
		Liaisons		X				
		Numérotation		X				
	Contenu	Informatif		X				
		Concis			X			
		Clair			X			
		Orthographe		X				
		Illustrations		X				
Oral	Présentation	Aisance			X			
		Tenue			X			
		Articulation, compréhension			X			
	Durée	Respect			X			
		Temps de parole équilibré		X				
	Réponses	Pertinence				X		
		Argumentation				X		

Proposition de note haute	15,56
Proposition de note basse	10,74
Proposition de note du jury	

Travail	Etude	Bibliographie	Adéquate		x			
			Suffisante		x			
		Cahier des charges	Clair	x				
			Formalisé		x			
		Hypothèses envisagées	Nombre			x		
			Pertinence		x			
		Validation	Analysée *a priori*			x		
			Tableau comparatif			x		
			Choix argumenté(s)		x			
			Faisabilité	x				
	Complexité	Temps consacré			x			
		Résultats obtenus		x				
		Difficulté	Intrinsèque		x			
			Vis-à-vis du binôme	x				
	Annexes	Fiches d'avancement	Régularité			x		
			Détaillées		x			
		Gantt	Prévisionnel et justifications	x				
			Effectifs, erreurs, corrections		x			

Proposition note haute	15,30
Proposition note basse	10,30
Proposition de note du jury	

Phase II : Conception détaillée, réalisation et recette

Rapport	Organisation	Plan	Equilibre	X				
			Cohérence	X				
		Fluidité	Introductions (partielles)	X				
			Transitions		X			
			Conclusions (partielles)	X				
		Tableaux, figures	Numérotés	X				
			Légendés	X				
			Référencés (non "en ligne")	X				
	Rédaction	Orthographe	Coquilles	X				
			Fautes évitables	x				
			Franglais, jargon		X			
		Rédaction	Aisée	X				
			Absence de plagiat !	X				

Proposition de note haute	19,09
Proposition de note basse	14,09
Proposition de note du jury	

Projection	Organisation	Plan		x				
		Liaisons			x			
		Numérotation		x				
	Contenu	Informatif		x				
		Concis		x				
		Clair		x				
		Orthographe		x				
		Illustrations		x				
Oral	Présentation	Aisance			x			
		Tenue		x				
		Articulation, compréhension			x			
	Durée	Respect		x				
		Temps de parole équilibré			x			
	Réponses	Pertinence			x			
		Argumentation			x			

Proposition de note haute	17,59
Proposition de note basse	12,59
Proposition de note du jury	

Travail	Conception	Générale	Clarté			x		
			Niveau de détail adéquat		x			
			Argumentation		x			
			Validation	x				
		Détaillée	Formalisation adéquate (algorithme...)	x				
			Validation	x				
	Réalisation	Développement	"Fonctionnalités"		x			
			Volume, environnement...		x			
		Tests (unitaires, d'intégration, fonctionnel, de performance...)			x			
	Recette	Livraison du paquetage		x				
		Redéploiement du paquetage		x				
	Complexité	Temps consacré			x			
		Résultats obtenus				x		
		Difficulté	Intrinsèque			x		
			Vis-à-vis du binôme		x			
	Annexes	Fiches d'avancement	Régularité		x			
			Détaillées		x			
		Gantt	Prévisionnel et justifications		X			
			Effectif, erreurs, corrections			X		

Proposition de note haute	15,00
Proposition de note basse	10,00
Proposition de note du jury	

FIGURE D.3 – Points à contrôler - final