Apprentissage automatique des réseaux d'interactions chez la fourmi Temnothorax nylanderi

Candice MOYET, Manon LEFEVRE

Avril 2022

Table des matières

1 Introduction

1.1 Introduction

Notre projet a pour but la création d'un outil permettant l'apprentissage et l'étude des réseaux d'interactions chez la fourmi Temnothorax nylanderi. Il s'agit en particulier d'identifier des changements de comportements au sein d'une colonie après qu'elle ait été confrontée à de la pollution. La première partie de ce projet consiste en la récupération de données de déplacements des fourmis : construction d'une méthode de différenciation des individus basée sur celles trouvées dans la bibliographie (utilisation de tags), acquisition et traitements des images du nid. Dans un second temps, nous procéderons à l'apprentissage des réseaux d'interactions grâce aux méthodes de machine learning supervisées et non supervisées. Cet outil sera ensuite utilisé dans le cadre d'une thèse en biologie sur l'étude de l'impact de la pollution sur les interactions des fourmis.

Pendant toute la durée du semestre nous avons été encadrées par Jean-Noël Vittaut, et nous étions en lien avec l'Institut d'Ecologie et des Sciences de l'Environnement de Sorbonne Université, par l'intermédiaire de Matthieu Molet et Marie Gressler. L'ensemble du code produit est disponible sur Git hub.

1.2 État de l'art

Les informations et les maladies circulent rapidement à travers toute une colonie de fourmis, plus rapidement au sein d'un groupe quand l'information vient d'une fourmi chargée d'aller chercher la nourriture [4] [5]. Dans le cas d'une exposition à un polluant comme le cadmium, les changements de comportements n'ont été que très peu étudiés. Le cadmium provoque chez les fourmis une augmentation du taux de mortalité des adultes, et une diminution de la taille et du taux d'émergence des ouvrières. Mais aucun effet des facteurs sociaux sur la résistance au cadmium n'a été trouvé, contrairement à ce qui a été trouvé chez d'autres insectes sociaux [2] comme les abeilles [6].

Il existe un certain nombre de systèmes de suivis d'individus utilisant des QR codes sous formes rectangulaires [1] ou circulaires [7]. Mais seulement peu d'entre eux ont été appliqués sur des fourmis.

Enfin, des logiciels ont été créés pour la prédiction de tâches et la détection d'interactions chez les animaux en utilisant des méthodes de machine learning classiques[3] ainsi que des réseaux de neurones convolutifs [8] ou des réseaux de neurones profonds [9] mais aucun d'eux n'ont été appliqués aux fourmis.

1.3 Contexte

Nous disposons d'une colonie de fourmis composée de 60 à 120 individus qui se déplacent dans et hors de leur nid. Nous récupérons des vidéos de cette colonie pour identifier au cours du temps la position des individus afin d'analyser leurs interactions et leurs rôles. Quatre rôles sont à distinguer au sein de cette espèce de fourmis ; les fourageuses qui sont chargées d'aller chercher la nourriture, les couveuses qui s'occupent des larves, les nettoyeuses qui gèrent les déchets du nid, et la reine [4][5]. Ce dernier rôle est un peu particulier car il n'y a qu'une seule fourmi qui le prend, alors que les autres rôles regroupent plusieurs fourmis

2 Acquisition des données

2.1 Génération des QR codes

Nous examinons tout d'abord la question de la génération des QR codes qui vont nous permettre d'identifier chaque individu de la colonie. Un certain nombre de contraintes et spécificités se présentent à nous : une colonie peut contenir entre 60 et 120 individus différents et la dimension finale des tags sera de 0.5×0.5 mm pour qu'ils puissent être collés sur les fourmis et tenir durant la durée de l'expérience. L'idée est donc d'optimiser le nombre de QR codes possible tout en gardant un nombre de pixel assez petit pour qu'ils soient visibles sur les fourmis et avec une taille suffisante pour minimiser les erreurs de détection. De même, pour réduire les erreurs, il faut que les QR codes soient suffisamment différents les uns des autres. Pour quantifier cette différence nous utilisons la distance de Hamming qui, pour deux suites de symboles (ici des bits) de même taille, donne le nombre de position où elles diffèrent. Nous avons testé différentes combinaisons possibles, indiquées sur la table 1, pour l'ensemble de ces contraintes et avons décidé de prendre le résultat proposant la meilleure optimisation, à savoir 8 bits d'identités avec une distance de Hamming de 3.

Bits identités	Hamming	QR-codes
6	2	32
8	2	256
9	2	512
6	3	16
8	3	256
9	3	512
6	4	16
8	4	64
9	4	128

TABLE 1 – Optimisation des contraintes sur les tags

Comme nous nous intéressons aux interactions des fourmis, il faut pouvoir différencier deux fourmis face à face qui interagissent ensemble et deux fourmis dos à dos qui n'interagissent pas ensemble. Il y a donc une notion d'orientation à ajouter dans la conception de ces tags. C'est pourquoi nous ajoutons aux bits d'identités des bits de contrôles qui nous permettront, d'une part, de calculer l'orientation du QR code par rapport à la tête de la fourmi et d'autre part, de détecter et corriger d'éventuelles erreurs. Nous utilisons la méthode présentée dans l'article BEEtag [1] ; les premiers bits du code d'erreur sont des contrôles de parité (1 (blanc) pour impair et 0 (noir) pour pair) de chacune des colonnes de la matrice identité. Les bits suivants sont générés en vérifiant la parité des lignes de la matrice identité. Ce contrôle d'erreur est ensuite répété et inversé (le premier bit du code d'erreur devient le dernier et inversement).

On ajoute enfin une bordure blanche et une bordure noire autour des QR-codes (toujours en suivant la méthode présentée dans l'article BEEtag[1]). Ces bordures nous serviront pour faciliter la détection des QR-codes. Pour finir, une marque blanche est intégrée dans le QR-code pour que l'orientation puisse être distinguable à l'oeil humain, lors de la pose des QR-codes sur les fourmis. Nous obtenons alors 256 QR codes comme ceux présentés sur la figure 1.

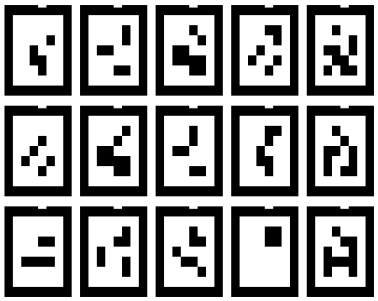

FIGURE 1 – Forme finale des QR-codes

L'étape suivante consiste à imprimer ces tags pour les coller sur les fourmis, une imprimante spéciale est nécessaire compte tenu de la taille des tags et celle que nous avions à disposition s'est trouvé en panne pendant plusieurs semaines. Nous avons donc travaillé sur la détection des QR codes en ajoutant sur les photos de la colonie de fourmis nos images de QR codes par retouche d'images.

2.2 Détection des QR codes

Cette détection a pour but de générer des séries temporelles composées d'une date, d'un identifiant d'individu et des coordonnées à partir d'une vidéo de la colonie pourvue de tags. Pour cela, nous sélectionnons une image par intervalle de temps fixé dans chaque vidéo.

Pour la détection nous avons utilisé la bibliothèque de traitement d'images python scikit-image [1]. Chaque image est binarisée, puis les régions d'intérêt sont extraites ; les bordures blanches et noires ajoutées au tags permettent de faciliter la détection par les méthodes de cette bibliothèque. Un exemple d'identification des régions d'intérêt est présenté sur la figure 2, nous pouvons observer, encadrées par deux croix vertes, cinq régions d'intérêts identifiées dont trois sont des tags. Nous précisons que cette image est un zoom de 50 % de l'image sur laquelle on observe l'entièreté

1. https ://scikit-image.org

du nid de la colonie. Une fois extraites, les régions d'intérêt sont ré-orientées et transformées en matrices binaires ; les bordures noires et blanches sont supprimées lors de la traduction en binaire (figure 3).

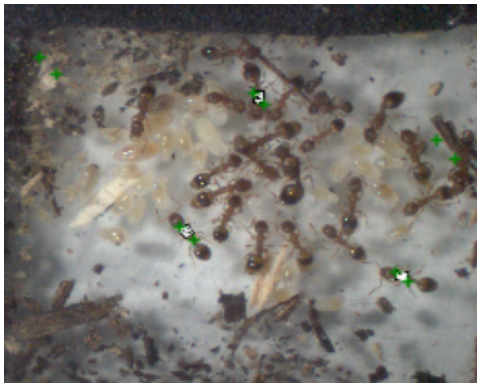

FIGURE 2 – Identification des régions d'intérêt

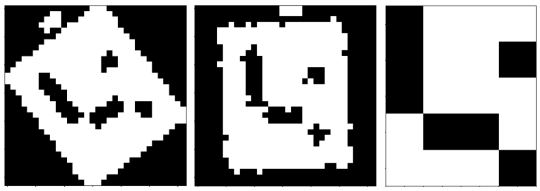

FIGURE 3 – Les étapes de détection d'un QR-code

Nous venons de présenter un QR-code dont la détection est parfaite, mais il arrive qu'il y ait des erreurs d'un ou plusieurs bits dans la détection. De plus, il nous faut aussi connaître l'orientation du QR-code. Ici nos tags sont rectangulaires donc deux orientations différentes sont possibles ; position initiale puis rotation de 180°. Pour chaque orientation, nous comparons la matrice identité avec la matrice d'erreur et d'orientation. Si elles correspondent, alors nous récupérons l'identifiant et les coordonnées du tag. Si elles ne correspondent pas, il y a deux possibilités : soit il y a des erreurs dans le tag, soit ce n'est pas la bonne orientation. Nous supposons qu'il est dans la bonne orientation et nous comptons le nombre d'erreurs. S'il y a une ou deux erreurs, nous appliquons une correction sur ces erreurs puis nous récupérons l'identifiant et les coordonnées du tag. Sinon nous essayons de tourner le QR code à 180°, s'il y a encore trop d'erreurs nous supposons qu'il ne s'agit pas d'un QR code. Sur la figure 4 est présenté un exemple de correction ; dans un premier temps le QR code est retourné puis comme il ne présente qu'une erreur (deuxième bit de la première ligne) il est corrigé. Nous pouvons ainsi obtenir des séries temporelles composées d'une date, d'un identifiant d'individu et de coordonnées.

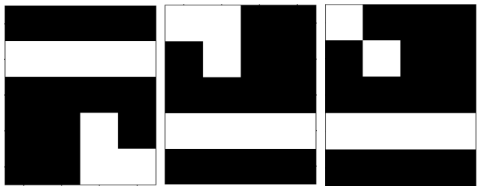

FIGURE 4 – Les étapes de correction et de détermination de l'orientation d'un QR-code

Il ne manque maintenant plus qu'à imprimer physiquement ces QR-codes, pour qu'ils puissent être collé sur le dos de chaque fourmis. Lors de cette phase d'impression, un certain nombre de problèmes techniques et matériels nous ont ralenti. Tout d'abord, l'imprimante spéciale pour ce type d'impression était en panne. Plusieurs semaines plus tard, une autre machine nous a été proposée. Mais le support utilisé pour l'impression est métallique et produit des reflets avec la lumière de la caméra, de plus, le contraste entre les deux couleurs du tag n'était pas assez fort. De même, les plaquettes ont tendance à être de couleur marron sur les photos et à être assez proche de la couleur des fourmis (figure 5). Cela ne permettait pas au modèle de détection de lire les tags.

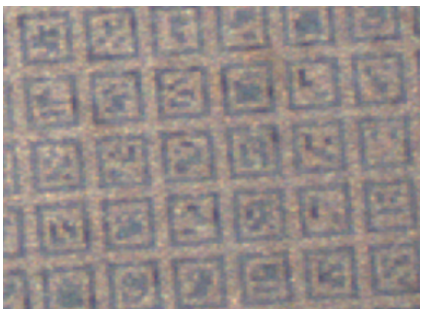

FIGURE 5 – Tags imprimés

Nous avons continué à travailler sur des pistes pour améliorer le contraste et trouver un meilleur support, par exemple en changeant la couleur de la plaque chimiquement par oxydation pour que le contraste entre la couleur des fourmis et des tags soit plus fort. De même, nous avons réfléchi à changer les tags ; l'idée est de les rendre plus petit afin qu'un bit de QR-code soit plus grand à l'image, et ainsi avoir une meilleure détection. Une autre solution peut être de réduire les bordures noires et blanches en les divisant par deux, ce qui permettrait d'augmenter la taille des bits d'informations. On pourrait aussi simplifier les tags en prenant moins d'individus et donc moins de bits d'informations, et s'intéresser à une nouvelle correction comme le code de Hamming ; le code correcteur le plus compact ayant la meilleure capacité de correction. Cela nous permettrait d'avoir moins de bits dans notre matrice de contrôle d'erreur et d'encore augmenter la taille des pixels sur les QR-codes.

3 Analyse des données

Maintenant que nous avons des séries temporelles décrivant les déplacements des individus d'une colonie de fourmis, nous voulons extraire des informations sur le comportement des individus au sein de la colonie notamment sur les interactions entre individus, entre groupes, et le rôle de chaque individu. De façon à pouvoir comparer les comportements avant et après introduction de pollution dans l'environnement. Dans un premier temps, nous cherchons donc à identifier chaque individu et leur rôle au sein de la colonie puis nous étudierons les interactions et les potentiels changements de rôles des fourmis.

3.1 Identification des individus

3.1.1 Transformation des séries temporelles

A partir des séries temporelles nous voulons extraire plusieurs données : des matrices de comptages du nombre d'interactions entre les fourmis et un certain nombre d'évènements significatifs comme les sorties du nid, les interactions avec la reine, les visites au couvin et à l'aire de gestion des déchets. Pour construire les matrices d'interactions nous avons utilisé un premier jeu de données provenant de l'article [10] et présentant des séries temporelles avec l'identifiant d'une fourmi, des coordonnées et un numéro d'image de la vidéo que nous convertissons en donnée temporelle (nombre de secondes depuis le début de la vidéo). Nous définissons un temps t et une distance d. Une interaction entre deux individus i_1 et i_2 présents aux coordonnées (x_1, y_1) (resp. (x_2, y_2)) au temps t_1 (resp. t_2) est compté lorsque :

$$\sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2} \leq d \tag{1}$$

et

$$|t_1 - t_2| \leq t \tag{2}$$

Nous obtenons alors des matrices M de $I \times I$ où I est le nombre d'individus d'une colonie et le coefficient $m_{k,j}$ représente le nombre d'interactions entre l'individu k et l'individu j. Cette matrice est donc symétrique.

3.1.2 Présentation du jeu de données

Nous n'avons pas pu obtenir de jeu de données nous permettant d'extraire les données de sorties du nid, d'interactions avec la reine, de visites du couvin et de présence dans l'aire de gestion des

déchets. Mais il suffirait d'avoir les coordonnées de ces lieux d'importances et de compter pour chaque individus le nombre de fois où il se rend sur ces lieux. De la même façon pour l'interaction avec la reine, il faudrait identifier manuellement le tag correspondant à la reine (à l'aide d'un expert) puis récupérer la ligne de la matrice d'interaction correspondant à l'identifiant de la reine pour avoir le nombre d'interactions entre la reine et toutes le fourmis de la colonie.

Le premier jeu de données utilisé n'étant pas assez grand et détaillé pour exploiter correctement les matrices obtenues, nous utilisons dans la suite le jeu de données acquit dans l'article [11]. Celui-ci présente l'enregistrement des interactions de 6 colonies de (110, 128, 156, 98, 148 et 161 individus) sur 41 jours sous la forme d'une matrice du comptage du nombre d'interaction par jour par colonie. A partir de cette matrice nous avons pu créer des graphes d'interactions assez complexes. Sur le graphe de la figure 6 nous avons 110 sommets, un par fourmi de la colonie et chaque arête représente l'interaction entre deux individus. Les arêtes sont pondérées (les poids ne sont pas représentés sur la figure) par le nombre d'interactions entre les deux individus lors du jour 1.

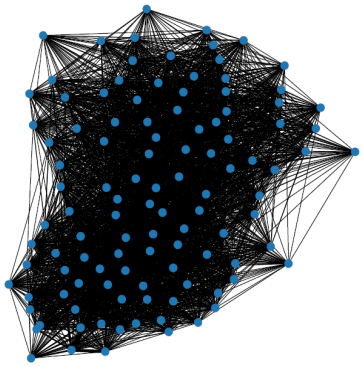

FIGURE 6 – Graphe d'interactions de la colonie 1 (1^{er} jour)

Est associé à ces matrices de nombreuses données sur les fourmis notamment celles du nombre de visites aux lieux significatifs identifiés plus haut et du nombre d'interaction avec la reine au long des 41 jours (une valeur par individu pour les 41 jours). Est aussi indiqué le rôle des individus (fourageuse, nettoyeuse, couveuse, reine), ces rôles pouvant évoluer au cours du temps, le jeu de données présente 4 étiquettes de rôle pour chaque individu évaluées aux jours 1, 12, 22 et 32. Avec ce deuxième jeu de données nous pouvons travailler sur l'identification des individus avec plusieurs méthodes : le spectral clustering fait à partir des matrices d'interaction, l'algorithme des k-moyennes, et finalement la classification supervisée grâce aux étiquettes de rôles.

3.1.3 Classification non supervisée

Dans un premier temps un spectral clustering est réalisé sur chaque matrice d'interactions. En effet, le spectral clustering semble particulièrement approprié à notre situation, puisque nous avons accès à un certain nombre de graphes complexes, comme vu précédemment. Nous cherchons à classer les fourmis dans les trois groupes : fourageuse, nettoyeuse et couveuse. Au préalable le rang et la colonne de la matrice correspondant à la reine est retiré s'il existe. En effet, le spectral clustering permettant de créer des groupes équilibrés et la reine étant le seul individu de son groupe, il n'est pas possible de la classer avec ce clustering. De plus, ce clustering nécessite un graphe connexe, nous rajoutons donc 1 à tous les coefficients de la matrice d'interactions donnée en entrée pour qu'elle produise un graphe connexe. Nous utilisions l'algorithme de spectral clustering de la bibliothèque python scikit-learn [2]. Pour ce clustering, score d'accuracy est calculé pour chaque clustering en comparant les groupes prédits aux vrais groupes. Pour cela, un label (fourageuse, nettoyeuse, couveuse) est associé à chaque groupe prédit puis le score d'accuracy est calculé. Ces opérations sont réitérées pour tester toutes les combinaisons de groupe et le meilleur score est gardé.

Nous appliquons également l'algorithme des k-moyennes, algorithme classique de clustering, à nos données. Ce n'est plus la matrice d'interactions que nous donnons en entrée au classifieur mais le nombre de fois où une fourmi est allée chercher de la nourriture, où elle est allée à l'endroit où sont les larves, à l'entrée du nid, à la pile de déchets et voir la reine. Les données d'entrée sont donc des matrices de $I \times 5$, où I est le nombre d'individus d'une colonie. La combinaison optimale

2. https ://scikit-learn.org

d'hyper-paramètres est calculée grâce à la méthode GridSearch de sklearn, notamment le nombre de réitérations de k-moyennes.

Pour ces deux classifieurs nous avons décidé de calculer plusieurs autres scores au vu du caractère spécial de l'évaluation d'un clustering. Nous utilisons les méthodes de la bibliothèque scikit-learn pour calculer un score d'information mutuelle (MI) (calculant la dépendance entre les vrais groupes et les groupes prédits), un rand index (donnant une mesure de similarité entre les groupes), un score d'homogénéité (chaque groupe ne contient que des fourmis ayant le même rôle), un score de complétude (toutes les fourmis ayant le même rôle sont dans le même cluster) et finalement un score de Fowlkes-Mallows (FMI) (équivalent à un score de précision et rappel). Les résultats sont présentés dans la table 2.

Classifieur	Accuracy	MI	Rand index	Homogeneity	Completeness	FMI
Spectral clustering	0.68	0.49	0.76	0.50	0.48	0.63
K-means	-	0.29	0.59	0.26	0.43	0.58

TABLE 2 – Scores de la classification non supervisée

Le spectral clustering donne des résultats satisfaisants contrairement à l'algorithme des k-moyennes qui ne semble pas adapté à notre problème. Il est possible que les données en entrée de ce classifieur ne permettent pas de classer les rôles des fourmis. En terme de classification non supervisée, nous pourrions essayer de classifier les rôles des fourmis avec des marches aléatoires dans les graphes produit à partir des matrices d'interactions. Les noeuds représentent les fourmis et les arcs la probabilité pour une fourmis d'interagir avec une autre.

3.1.4 Classification supervisée

Le jeu de données que nous utilisons nous permet de faire de la classification supervisée puisque nous avons une étiquette de rôle pour chaque individu. Si nous n'avions pas eu ces étiquettes de rôles, nous aurions pu demander à un expert de les réaliser. Nous avons utilisé plusieurs données en entrée de nos classifieurs (les mêmes que pour l'algorithme des k-moyennes) : le nombre de fois où une fourmi est allée chercher de la nourriture, où elle est allée à l'endroit où sont les larves, à l'entrée du nid, à la pile de déchets et voir la reine. Les données d'entrée sont donc des matrices de $I \times 5$, où I est le nombre d'individus d'une colonie. Nous cherchons toujours à classer les fourmis dans les trois groupes : fourageuse, nettoyeuse et couveuse. C'est le vecteur de label donné en entrée ; nous travaillons donc sur une classification multi-classe.

Nous avons choisis différents classifieurs de la bibliothèque scikit-learn en python. Tout d'abord nous avons commencé classiquement avec K-plus proche voisins et Naive Bayes. Pour Naive Bayes on utilise $MultinomialNB$ de sklearn qui est le classifieur bayésien adapté aux features discrètes et aux classes équilibrées ce qui correspond bien à nos données. Nous avons également appliqué un arbre de décision à nos données, en effet leur caractère descriptif nous paraît approprié à un arbre de décision. De plus, son caractère explicable nous permet de ressortir des informations intéressante pour l'étude du comportement des individus. Enfin nous avons testé avec un classifieur linéaire : un SVM en spécifiant en paramètres que les classes sont équilibrées.

Pour chaque classifieur nous avons effectué une recherche de la combinaison optimale des hyperparamètres. Puis ont été calculés les scores d'accuracy en entraînement et en test présentés dans la table 3 en validation croisée. Nous entraînons les classifieurs sur cinq colonies et testons sur la dernière.

Classifieur	Train accuracy	Test accuracy
K-NN	0.79	0.70
Naive Bayes	0.71	0.69
Decision Tree	0.78	0.68
SVM	0.75	0.70

TABLE 3 – Scores de la classification supervisée

Tous les classifieurs atteignent de bons scores assez similaires, il est donc possible d'identifier le rôle des fourmis en fonction de leurs activités. Le modèle d'arbre de décision nous retourne l'importance de chacune des caractéristiques présentées dans la table 4. Il apparaît que la caractéristique la plus importante est le nombre d'interactions avec la reine, le nombre de fois où un individu est

allé chercher de la nourriture et donc sorti du nid est également important. Au contraire, le nombre de visites aux larves, à l'entrée du nid et à la pile de déchet sont peu importants.

Caractéristiques	Importance
Recherche de nourriture	21%
Visites aux larves	4%
Visites à l'entrée du nid	7%
Visites à la pile de déchets	0%
Interactions avec la reine	67%

TABLE 4 – Importance des caractéristiques dans la classification par arbre de décision

3.2 Etude du comportement des individus

Maintenant que nous avons identifié le rôle de chaque individu dans la colonie de fourmis, nous cherchons à présent à ressortir des indices sur leurs interactions de façon à pouvoir étudier leur changement avant et après introduction de pollution dans l'environnement de la colonie. Nous allons, dans un premier temps, étudier les interactions inter-individus, inter-groupes et intra-groupes puis dans un second temps étudier le changement de rôle des fourmis au cours du temps.

3.2.1 Analyse des intéractions

A partir des matrices d'intéractions et en considérant les rôles de chaque individu nous avons généré des graphes d'interactions entre les groupes. Sur la figure 7 on peut voir un graphe $G = (V, E)$ où V est l'ensemble des sommets représentant les trois groupes fourageuses (F), nettoyeuses (C), et couveuses (N) ainsi que la reine (Q) et E est l'ensemble des arêtes. L'épaisseur des arêtes représente leur poids, c'est-à-dire que plus l'arête est épaisse plus son poids est élevé plus les deux groupes représentés par les sommets liés par l'arête interagissent fortement. Nous observons dans toutes les colonies une interaction plus forte entre les fourageuses et les nettoyeuses et une plus faible entre les fourageuses et les couveuses. Tous les groupes interagissent de la même façon avec la reine.

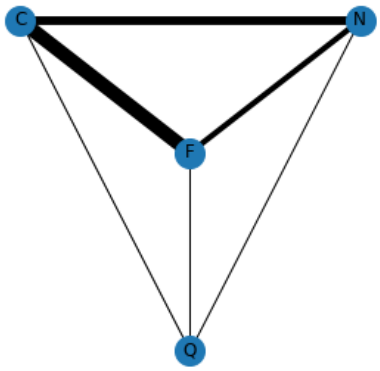

FIGURE 7 – Graphe d'interactions des groupes de la colonie 1 (12^{e} jour)

En plus de ces graphes nous calculons des indices qui permettent de mesurer les interactions au sein du nid et donc de repérer des éventuels changements après introduction de la pollution dans le nid. Nous calculons la part d'interactions entre chaque groupe par rapport à celles du nid entier (noté CF pour l'interaction entre les groupes C et F), le nombre total d'interactions au sein de chaque groupe (noté TT F pour le groupe F) ainsi que la moyenne du nombre d'interactions pour un individu au sein de chaque groupe (noté μ F pour le groupe F). Ces indices sont représentés dans la table 5 pour le 1er jour de l'expérience pour les trois premières colonies.

Colonie	C&F	C&N	C&Q	N&F	N&Q	F&Q	TT F	TT C	TT N	μ F	μ C	μ N
1	0.14	0.29	0.01	0.20	0.01	0.00	336	864	1265	62.9	62.5	63.7
2	0.11	0.23	0.01	0.25	0.02	0.01	130	162	839	27.4	17.7	33.4
3	0.14	0.32	0.01	0.17	0.01	0.00	156	594	875	36.5	44.3	43.3

TABLE 5 – Scores de la classification non supervisée

3.2.2 Analyse des changements de rôles

Le jeu de données que nous utilisons nous fournit quatre étiquettes de rôles pour chaque individu (1 tous les 10 jours). Nous pouvons donc étudier le changement de rôle des individus, cela peut être une information intéressante à étudier après introduction de pollution dans le milieu des fourmis. Pour cela nous avons décidé d'utiliser une chaîne de Markov. Nous construisons la matrice A de taille 4×4 où chaque coefficient $a_{i,j}$ est la probabilité pour un individu de passer de l'état i à l'état j. Les états sont les rôles déjà définis : fourageuse, nettoyeuse, couveuse, reine. Pour construire cette matrice nous procédons par simple comptage des changements de rôle puis chaque ligne est normalisée. Sur la figure 8 est représentée la chaîne de Markov ainsi construite pour la colonie 3. Nous pouvons observer assez logiquement que la reine ne change jamais et que, en général, les fourmis gardent leur rôle sauf pour les nettoyeuses qui peuvent devenir des fourageuses ou des couveuses. Il est aussi très clair que les fourageuses ne deviennent jamais des couveuses.

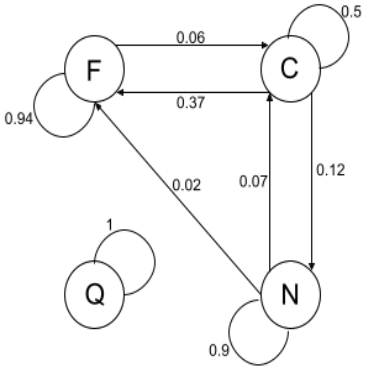

FIGURE 8 – Chaîne de Markov représentant les changements de rôle de la colonie 3

4 Conclusion

Si l'impression des QR codes n'a pas fonctionné, nous avons pu néanmoins mettre en place un système de détection de tags pour produire des séries temporelles. Nous avons appliqué différentes méthodes de machine learning de classification et de clustering sur des données réelles. Nous avons montré que les interactions avec la reine ainsi que le nombre de sorties du nid était important dans la détermination du rôles des fourmis. Enfin, nous nous sommes intéressées aux interactions entre les différents groupes et aux changements de rôle avec des chaînes de Markov. Ce travail n'a pas pu répondre entièrement au cahier des charges pour diverses raisons techniques. Dans la suite de ce projet, il faudrait imprimer les tags de manière à qu'ils puissent être lu par le modèle de détection, et que les modèles implémentés puissent être appliqués sur ces données dans le cadre de la recherche biologique. L'idée serait de l'expérimenter avant l'ajout de pollution et une fois l'environnement des fourmis pollué pour comparer les différents indices d'interactions. Ce modèle pourra servir pour d'autres espèces d'animaux.

Références

[1] James D. Crall, Nick Gravish, Andrew M. Mountcastle, and Stacey A. Combes. 2015. BEEtag : A Low-Cost, Image-Based Tracking System for the Study of Animal Behavior and Locomotion. PLOS ONE 10, 9 (September 2015), e0136487. DOI : https ://doi.org/10.1371/journal.pone.0136487

[2] L. Jacquier, C. Doums, A. Four-Chaboussant, R. Peronnet, C. Tirard, and M. Molet. Urban colonies are more resistant to a trace metal than their forest counterparts in the ant Temnothorax nylanderi. Urban ecosystems 2021 24, 3 , 561–570. DOI : https ://doi.org/10.1007/s11252-020-01060-9

[3] Mayank Kabra, Alice Robie, Marta Rivera-Alba, Steven Branson, and Kristin Branson. 2012. JAABA : Interactive machine learning for automatic annotation of animal behavior. Nature methods 10, (December 2012). DOI : https ://doi.org/10.1038/nmeth.2281

[4] Danielle P. Mersch, Alessandro Crespi, and Laurent Keller. 2013. Tracking Individuals Shows Spatial Fidelity Is a Key Regulator of Ant Social Organization. Science 340, 6136 (April 2013), 1090–1093. DOI : https ://doi.org/10.1126/science.1234316

[5] Thomas O. Richardson, Jonas I. Liechti, Nathalie Stroeymeyt, Sebastian Bonhoeffer, and Laurent Keller. 2017. Short-term activity cycles impede information transmission in ant colonies. PLoS Computational Biology (May 2017). DOI : https ://doi.org/10.1371/journal.pcbi.1005527

[6] Nathalie Stroeymeyt, Anna V Grasse, Alessandro Crespi, Danielle P Mersch, Sylvia Cremer, and Laurent Keller. 2018. Social network plasticity decreases disease transmission in a eusocial insect. Science 362, 6417 (November 2018), 941–945. DOI : https ://doi.org/10.1126/science.aat4793

[7] Fernando Wario, Benjamin Wild, Margaret Jane Couvillon, Raúl Rojas, and Tim Landgraf. 2015. Automatic methods for long-term tracking and the detection and decoding of communication dances in honeybees. Frontiers in Ecology and Evolution (September 2015). DOI : https ://doi.org/10.3389/fevo.2015.00103

[8] Benjamin Wild, David M. Dormagen, Adrian Zachariae, Michael L. Smith, Kirsten S. Traynor, Dirk Brockmann, Iain D. Couzin, and Tim Landgraf. 2021. Social networks predict the life and death of honey bees. Nature Communications 12, 1110 (February 2021). DOI :http ://dx.doi.org/10.17169/refubium-30015

[9] Benjamin Wild, Leon Sixt, and Tim Landgraf. 2018. Automatic localization and decoding of honeybee markers using deep convolutional neural networks. (February 2018). DOI : https ://doi.org/arXiv :1802.04557

[10] Evlyn Pless, Jovel Queirolo, Noa Pinter-Wollman, Sam Crow, Kelsey Allen, Maya B. Mathur, Deborah M. Gordon. 2015. Interactions Increase Forager Availability and Activity in Harvester Ants. PLOS ONE (November 2015). DOI : 10.1371/journal.pone.0141971

[11] Anshuman Swain, Sara D. Williams, Louisa J. Di Felice and Elizabeth A. Hobson. 2021. Interactions, information and emergence : Exploring task allocation in ant colonies1 using network analysis. DOI : https ://doi.org/10.1101/2021.03.29.437501