Optimisation dans la Classification Multi-label

Réalisé par
Fofana Cheick Tidiane

Dans le cadre du cours
Label Recherche

Travail présenté à
M. Sylvain Salvati
M. Jean-Stephane Varre

Encadré par
Mme Julie Jacques
Mme Laeticia Jourdan
M. Sylvain Salvati

Du département d'Informatique
Faculté des Sciences et Technologie
Université De Lille
Cristal
1 Avril 2020

Table des matières

Introduction

La tarification à l'activité (**T2A**) est une méthode de financement des établissements de santé. Elle repose sur la mesure et l'évaluation de l'activité effective des établissements qui détermine les ressources allouées.

La mesure et l'évalution de l'activité effective des établissements passe par le processus de codage. Le codage correspond à l'association de codes **CCAM**(*Classification Commune des Actes Médicaux*) et **CIM-10**(*Classification Internationale des Maladies*) au séjour d'un patient dans un service hospilalier(*Unité Fonctionnel*).Il dépend des actes médicaux et diagnostics réalisés. Cette étape prend beaucoup de temps d'autant plus qu'il existe 50.000 codes. Un temps qui pourrai être consacré aux patients.

Avec la mise en place en France du **PMSI**(*standardisation des données pour le service public hospitalier*), les données récoltées sur les patients sont standardisées permettant leur exploitation. Ces données sont générées lors de l'admission d'un patient dans un hôpital. Elles contiennent des informations sur les actes médicaux réalisés et les diagnostics. Ces actes médicaux et diagnostics entretiennent potentiellement un certain degré de corrélation.

Dans le but d'automatiser le processus de codage l'une des approches est la classification *Multi-Label*(CML) un sous domaine du *Machine Learning*.

L'utilisation de cette méthode plûtot qu'une autre réside dans le fait que celle-ci permet de prendre en compte les corrélations entre les différents labels ici typiquement les codes associés.

La classification *Multi-Label* correspond à l'assignation d'un ou plusieurs labels à une instance(exemple de photos, document). Actuellement, il existe principalement deux types de méthodes discriminatoires pour la classification *Multi-Label* : la transformation de problèmes et l'adaptation d'algorithmes. La première consiste à transformer le problème de classification *Multi-Label* en plusieurs problèmes de classification *mono-label* plus classique.La seconde méthode constiste à adapter des algorithmes de classification *mono-Label* déjà existant afin de les adapter à la classification *Multi-Label*. L'avantage de cette seconde méthode réside dans le fait qu'elle prenne en compte les corrélations entre les labels, contrairement à la première méthode.

La méthode la plus adaptée au processus de codage est l'utilisation de l'adaption d'algorithmes déjà existant. L'implémentation de ces algorithmes peut se représenter sous la forme d'un problème d'optimisation.
Dans ce rapport nous traiterons la question suivante :
En quoi l'Optimisation permet-elle de résoudre les problèmes de classification Multi-Label ?

Pour ce faire, nous présenterons *l'Optimisation* et les différents procédés d'optimisation notamment les *Métaheuristiques*. Dans un second temps, nous aborderons la classification *Multi-label* et les principales techniques de résolution. Enfin nous présenterons deux méthodes de résolution de problème de classification *Multi-Label* avant de finir avec une proposition d'algorithme de classification *Multi-Label* utilisant *l'optimisation*.

1 Optimisation

L'**optimisation** est une branche des *mathématiques* cherchant à modéliser, à analyser et à résoudre analytiquement ou numériquement les problèmes qui consistent à minimiser ou maximiser une fonction sur un ensemble. Il joue un rôle important en recherche opérationnelle.

On distingue deux types de problèmes d'optimisation : les problèmes *combinatoire* ou *discrets* et les problèmes à variables *continues*. Parmi les problèmes combinatoires, on trouve le célèbre problème du voyageur de commerce : il s'agit de minimiser la longueur de la tournée d'un *voyageur de commerce*, qui doit visiter un certain nombre de villes, avant de retourner à la ville de départ. Un exemple de problème continu est celui de la recherche des valeurs afin de trouver le modèle optimal pour une régression linéaire ou polynomiale. Il existe aussi des problèmes mixtes qui se traduisent par des problèmes impliquant l'optimisation des types évoqués précédemment.

Pour résoudre des problème d'optimisation continue, il existe des méthodes dites d'optimisation globale qui sont souvent inefficaces. Dans le domaine de l'optimisation discrète, un grand nombre d'heuristiques ont été développée mais bien souvent trop nichées.

La notion de complexité(temps,mémoire) des algorithmes est primordiale. Il s'agit d'un facteur décisif dans la séléction d'une méthode de résolution d'un problème d'optimisation. On parle d'optimiation difficile, pour un problème discret lorsqu'on ne connaît pas d'algorithme *polynomial* pour le résoudre, et dans le cas d'un problème d'optimisation à variables continues,lorsqu'on ne connaît pas d'algorithme permettant de trouver un optimum global en nombres finis de calculs.

1.1 Les Méta-heuristiques

L'émergence des *méta-heuristiques* vise à résoudre des problèmes d'optimisation difficile, notamment avec des algorithmes de recherche stochastique itératifs qui progressent vers un optimum global(recherche d'une solution optimale dans un ensemble de solutions candidates).

Le principe d'un algorithme classique d'*amélioration itérative* est le suivant : on part d'une configuration initiale c_0. On essaie alors une modification élémentaire(mouvement), et l'on compare les valeurs de la fonction objectif, avant et après cette modification. Si le changement conduit à une diminution de la fonction objectif dans le cas d'une minimisation, il est accepté et la configuration c_1 obtenue, qui est *voisine* de la précédente, sert de point de départ pour un nouvel essai. Dans le cas contraire, on revient à la configuration précédente, avant de faire une autre tentative. Cet algorithme conduit à un minimum local c_n mais pas global c^*.

Les *méta-heuristiques* permettent de s'extraire des minimums locaux. Les *méta-heuristiques* de *voisinage*(recuit simulé, recherche avec tabou) autorisent la dégradation temporaire c'_n de la fonction d'objectif afin de s'extraire des puits locaux . Quant aux *méta-heuristiques* dit *distribués*(algorithmes génétiques et évolutionnaires), elles ont aussi ont des procédés pour éviter les minimums locaux(voir figure 1).

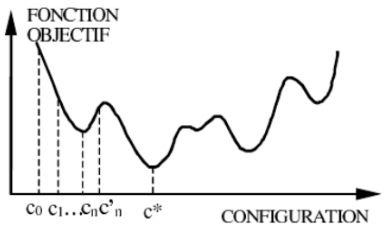
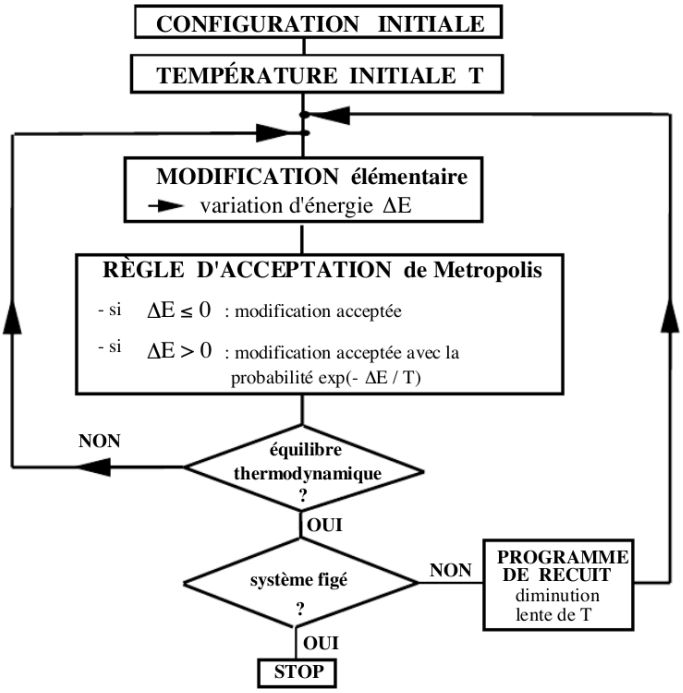


FIGURE 1 – Courbe d'un problème d'optimisation difficile

1.1.1 Les Méta-heuristiques de voisinage

Recuit simulé (*Simulated Annealing*)

Inspiré de la physique statistique et de la métallurgie(méthode du récuit), la technique du *Recuit simulé* est basé sur l'algorithme de *Metropolis* qui décrit le comportement d'un système en *équilibre thermodynamique* à température T. En partant d'une configuration donnée (par exemple, un placement initial de tous les composants dans un circuit électronique), on fait subir au système une modification élémentaire (on échange deux composants),si cette transformation améliore la fonction objectif du problème, elle est acceptée. Si elle provoque au contraire une augmentation ΔE de la fonction objectif, elle peut être acceptée tout de même, avec la probabilité $e^{-\Delta E/T}$. On itère ensuite ce procédé, en gardant la température constante, jusqu'à ce que l'équilibre thermodynamique soit atteint, concrètement au bout d'un nombre *suffisant* de modifications. On abaisse alors la température, avant d'effectuer une nouvelle série de transformations. La décroissance par paliers de la température est empirique, tout comme le critère d'arrêt du programme.



L'idée générale de l'algorithme est de ce permettre beaucoup de modification au début qui n'améliore pas forcement la fonction objectif, puis d'être de plus en plus vigoureux dans la sélection des voisins. On voit avec l'algorithme que plus ΔE est grand ou que plus la température T est petite, plus la probabilité d'être accepté est petite.Ce qui confirme le principe évoqué plus haut. L'avantage de cet algorithme réside dans sa simplicité d'implémentation et a sa généricité. Il engendre par compte beaucoup de calculs et de paramétrisation.

FIGURE 2 - Organigramme du recuit simulé (Gauche)

Recherche avec Tabou (*Tabu Search*)

Inspiré de la mémoire humaine, il s'agit d'un algorithme simple qui repose sur un principe de mémoire à court terme. L'idée de la recherche tabou consiste, à partir d'une position donnée, à en explorer le voisinage et à choisir la position dans ce voisinage qui minimise la fonction objectif (dans le cas d'une minimisation). En partant d'une configration initiale qui est actualisée au cours des itérations, un mécanisme de passage d'une configuration s à une configuration t est mise en place. Il est divisé en deux étapes. La première consiste à contruire l'ensemble des voisins de s (ensemble des voisins accessibles en un seul mouvement). La seconde étape consiste à sélectionner dans l'ensemble précédent la configuration pour laquelle la fonction objectif est minimale(même si elle n'améliore pas l'objectif) soit s $\rightarrow$ t. C'est à partir de ce mécanisme que l'on sort d'un minimum local.

Le risque est qu'à l'étape suivante, on retombe dans le minimum local auquel on vient d'échapper. C'est pourquoi il faut que l'heuristique ait de la mémoire : le mécanisme consiste à interdire de revenir sur les dernières positions explorées d'où la mise en place d'une liste de mouvement interdit aussi appelé *liste tabou*. Dans cette liste, à chaque itération nous mémorisons après la sélection de la configuration adoptée (s → t), le mouvement inverse (t → s).Cette liste de tabou doit être actulisée afin d'éviter d'être bloqué dans la génération de l'ensemble des voisins. Deux mécanismes supplémentaires, nommés *intensification* et *diversification*, sont souvent mis en œuvre pour doter aussi l'algorithme d'une mémoire à long terme. L'intensification consiste à approfondir l'exploration de certaines zones de la courbe identifiées comme prometteuses. La diversification, au contraire concentre sa recherche de l'optimum sur des régions jusque là trop peu visitées.

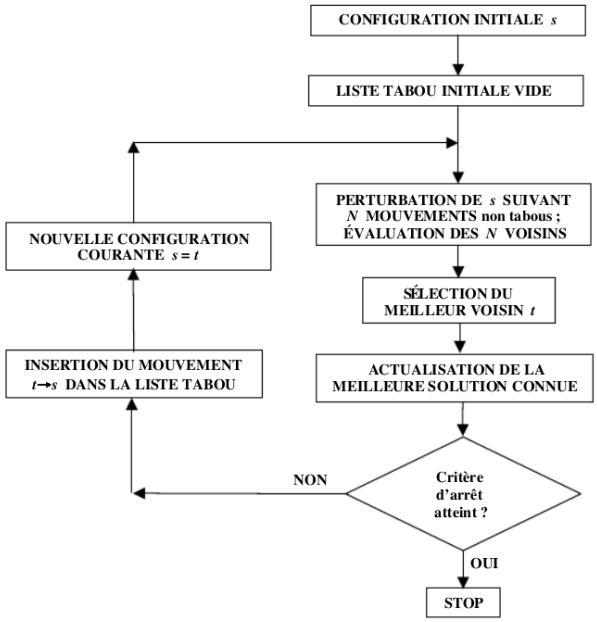


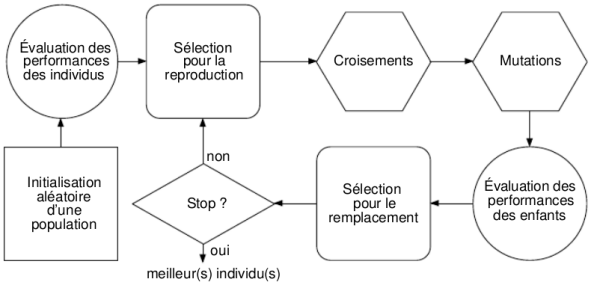
FIGURE 3 - Organigramme de l'algorithme tabou (Droite)

La complexité de cet algorithme est raisonnable par rapport au recuit simulé, en notant tout de même que l'intensification et la diversification rajoutent une complexité notable.

1.1.2 Les Méta-heuristiques distribuées

Algorithmes évolutionnaires (*Evolutionary Algorithms*)

Les algorithmes évolutionnaires sont inspirés de la théorie de l'évolution de C. Darwin. Ils sont généralement utilisés pour des problèmes contenant plusieurs optimum globaux ou encore des problèmes contenant plusieurs objectifs. Ils peuvent également fournir un jeu de solutions diverses. Il est aussi important d'évoquer leur important coût d'exécution, qui n'est plus un frein actuel avec l'augmentation de la puissance de calcul et des architectures de calculs parallèles.



Le principe est de faire évoluer progressivement, par générations successives, la composition d'une population, en maintenant sa taille constante. Au cours des générations, l'objectif est d'améliorer globalement la performance des individus.On part d'une population initiale N (choisit à priori aléatoire dans l'espace de recherche). Nous évaluons ensuite la performance de ces individus. Pour un individu x, dans le cas de la minimisation d'une fonction objectif f, x est d'autant plus performant que $f(x)$ est petit.

FIGURE 4 - principe des algorithmes évolutionnaires (Gauche)

Ensuite nous procédons à la formation de la génération suivante en mimant les mécanismes de sélection de C. Darwin. Elle se décompose en 4 parties.Une phase de sélection où les meilleurs individus sont sélectionés. Ensuite une phase de reproduction où les individus choisis précédemment vont subir des modifications(croisement ou mutation).Après cette phase, les nouveaux individus seront évalués par rapport à un objectif fixe. Enfin la phase de remplacement qui consiste à créer la nouvelle génération à partir de la génération précédente et des nouveaux individus.Ce processus est itérer tant que la condition d'arrêt n'est pas atteinte, par exemple un nombre de génération fixé.

Les algorithmes de colonies de fourmis (*Ant Colony Algorithms*)

Les algorithmes de colonies de fourmis reposent sur le principe selon lequel une colonie de fourmis trouve le plus court chemin entre la fourmilière et la source de nourriture en se basant sur le fait que chaque fourmi suit le chemin où le plus de fourmis sont passés auparavant en déposant à son tour des phéromones sur son chemin qui s'évaporent avec le temps. Plus le chemin est long est plus la quantité de phéromones sur ce chemin est petite en raison de l'évaporation. il s'agit de la *stigmergie*.En cas d'obstacle, il y a une adaptation très rapide. Ainsi ces algorithmes contiennent plusieurs caractéristiques qui sont très intéressantes, tel que la flexibilité(adaption à son environnement), la robustesse(colonie survie même avec quelques individus manquant), et la décentralisation (il y a aucune autorité centrale).

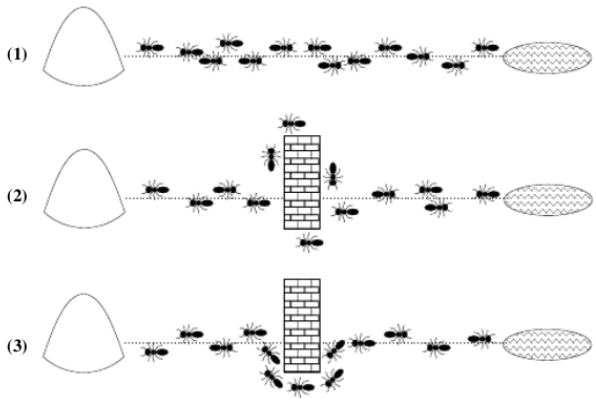


FIGURE 5 - Colonie de fourmis

2 Classification Multi-Label

La classification *Multi-Label* est une méthode d'apprentissage automatique qui correspond à l'assignation d'un ou plusieurs labels à une instance (Exemple de photos,Document). Formellement, elle porte sur l'apprentissage à partir d'exemples $\{(x_i,L_i)|\ 1 \leq i \leq \mathrm{N}\ \}$, où chaque exemple est associé à un ou plusieurs labels L_i,avec $L_i \subset$ L.Où L est l'ensemble fini de labels disjoints L= $\{y_1\ ,y_2,\ ...,y_Q\}$, $Q > 1$. Avec Q le nombre de classe et N le nombre d'exemples d'entrainement.Ces labels ne sont pas mutuellement exclusif, ce qui signifie que l'occurence d'un label n'empêche pas l'affection d'un autre label à cette même instance.

carac-1 (nb-roue)	carac-2 (hauteur)	carac-3 (puissance)	carac-4 (nb-place)	Label Associé
4 roues	2.70 mètres	700 chevaux	6	Voiture, Sport, SUV
2 roues	1.00 mètres	340 chevaux	2	Sport, Moto
4 roues	1.50 mètres	100 chevaux	4	Voiture

TABLE 1 – Exemple de classification Multi-label

Il existe principalement deux types de méthodes discriminatoires pour la classification multi-label : la transformation de problèmes et l'adaptation d'algorithmes présentées ci-dessous.

2.1 Transformation de problèmes

Le problème initial de classififcation *Multi-Label* se voit le plus souvent découpé en plusieurs petits problèmes de classification *mono-label* simple indépendant qui peuvent être résolus par des algorithmes de

classification *mono-label* déjà existant.Il ne faut pas faire la confusion avec la classification multi-classe qui consiste à attribuer à une instance un label parmi 3 classes ou plus. On peut avoir plusieurs procédés tel que :

Binary Relevant(BR)
Le principe de BR est de générer autant de classifieurs binaires que de label afin de les entrainer individuellement à reconnaitre une classe spécifique. Pour une donnée à classifier, elle passera ainsi par chacun des classifieurs et se verra attribuer tout les labels pour lesquelles elle aura été évaluée positive(pour une classe) par l'un des classifieurs. Cependant cette méthode ne prend pas en compte les corrélations entre les classes et peut éffectuer une mauvaise prédiction si seulement un classifieur est défaillant.

Classifier Chain(CC)
Il s'agit d'une méthode pour transformer un problème de classification multi-label en plusieurs problèmes de classification binaire. Elle diffère de BR en ce que les labels sont prédites de manière séquentielle de sorte que la sortie de tous les classifieurs précédents soit utilisée comme caractéristique(*feature*) dans les classificateurs suivants. Cette méthode est une alternative à la méthode BR afin de mimer les corrélations entre les classes.

Label Powerset(LP)
Cette méthode consiste à construire toutes les combinaisons de labels possibles. Ensuite elle crée un classifieur binaire pour toute les combinaisons possibles de label. Ainsi nous faisons face à un problème de classification multi-classe.
Ex : Pour les A B C,les combinaisions potentielles sont [0 0 0], [1 0 0], [0 1 0], [0 0 1], [1 1 0], [1 0 1], [0 1 1], [1 1 1], où 1 signifie la préscence du label et 0 son abscence. LP prend en compte les corrélations existantes entre les différentes classes et à montrer de très bon résultats expérimentaux.

La principale limite des méthodes de transformation de problèmes est que la corrélation entre les labels n'est pas explicitement pris en compte.

2.2 Adaptation d'algorithmes de classification

Contrairement aux méthodes de transformation de problèmes, les méthodes d'adaptation d'algorithmes considérent explicitement les corrélations entre les labels. Il s'agit d'algorithmes adaptatifs qui résultent d'algorithmes de classification *mono-label* existant adaptés à la classification *Multi-Label*. Presque tous les paradigmes classiques de la classification classique(*mono-label*) ont été visités afin d'être adaptés aux données multi-labels.

Decisison Trees
Les méthodes de l'arbre de décision ont été principalement utilisées dans le génomiques. Il vaut la peine de citer ML-C4.5, l'adaptation du populaire C4.5.

Support Vector Machines (SVMs)
Dans [1], les auteurs proposent un algorithme de classement SVM appelé *Rank-SVM* qui a montré de meilleurs performances que BR en utilisant des SVMs.

Approches évolutionnaires
Les approches bio-inspirées ont été introduites avec l'algorithme *Multi-Label Ant-Miner*(MuLAM) proposé par [2]. Il s'agissait d'une extention de l'agorithme *Ant colony-based Ant-Miner* de [3]. En 2010, [4] ont aussi proposés GEP-MLC, une approche évolutionnaire afin de générer des règles de classification.

Nous citerons non exhaustivement les **Neural Networks**, **Ensembles**,**Associative classification** qui sont d'autres paradigmes adaptés à la classification Multi-label présentés dans [5].

2.3 Métrique d'évalution de performance

Il est nécessaire de définir comment évaluer les résultats de la classification *Multi-label*. Différent de la classification *mono-label*, où la classification d'une instance est soit correcte ou pas, dans les tâches *multi-label*, le résultat peut également être partiellement correct. Ce serait le cas où le classifieur ajoute/omet un des labels d'une instance.

D'après [6] On distingue deux types de métrique d'évaluation : *label-based metrics* et *example-based metrics*. L'idée derrière *label-based metrics* est de faire des calculs sur chaque label individuellement basé sur le nombre de *true positives*(tp), *true negatives*(tn), *false positives*(fp) et *false negatives*(fn), afin d'obtenir une valeur moyenne. Il existe ainsi deux métriques. Pour les *example-based metrics*, elles sont catégorisées en deux groupes, les métriques pour l'évaluation des classements des labels(non représentés dans le tableau) et celles pour l'évaluation de la classification. Soit $\{(x_i,Y_i)|\ 1 \leq i \leq t\ \}$ un ensemble de test de classificaiton *Multi-label* avec t instances. Pour une instance x donnée, Y représente l'ensemble des labels de cette instance et Z l'ensemble des labels prédits pour cette instance.

Métrique d'Evaluation de la classification	
Example-based metrics	
Subset accuracy ↑	$\frac{1}{t}\sum_{i=1}^{t}[Z_i = Y_i]$
Subset 0/1 loss ↓	$\frac{1}{t}\sum_{i=1}^{t}[Z_i \neq Y_i]$
Hamming Loss ↓	$\frac{1}{t}\sum_{i=1}^{t}[Z_i \Delta Y_i]$
Precision ↑	$\frac{1}{t}\sum_{i=1}^{t}\frac{\lvert Z_i \cap Y_i\rvert}{\lvert Z_i\rvert}$
Accuracy ↑	$\frac{1}{t}\sum_{i=1}^{t}\frac{\lvert Z_i \cap Y_i\rvert}{\lvert Z_i \cup Y_i\rvert}$
F1-Score ↑	$\frac{1}{t}\sum_{i=1}^{t}\frac{\lvert Z_i \cap Y_i\rvert}{\lvert Z_i\rvert+\lvert Y_i\rvert}$
Recall ↑	$\frac{1}{t}\sum_{i=1}^{t}\frac{\lvert Z_i \cap Y_i\rvert}{\lvert Y_i\rvert}$

TABLE 2 – Métrique d'évaluation

Ci-dessous ne seront présentés que les métriques principalement utilisée pour la classification soit *Hamming-Loss* et *Subset Accuracy* (voir TABLE 2).

Hamming Loss
En terme plus simple, *Hamming-Loss* représente la fraction de label qui ont été mal prédit, c'est-à-dire le rapport du nombre de labels mal prédit sur le nombre total de labels.

Subset-Accuracy
Subset Accuracy représente le pourcentage d'exemples qui ont été classifiés correctement(sans ajout ni omission).

3 Optimisation et Classification Multi-Label

3.1 Algorithmes de classification Multi-Label

Multi-Label Ant-Miner (MuLaM)

MuLaM est un algorithme basé sur *ACO*(Ant Colony Optimization) proposé par *Chan* et *Freitas*[2] inspiré de *Ant-Miner* développer par *Parpinelli* et ses collègues.

Afin de mieux comprendre le présentation de **MuLaM**, il semble nécessaire de revoir *Ant-Miner*. *Ant-Miner* est algorithme de classification *Multi-Label* qui construit séquentiellement une liste des règles de classification, en découvrant une règle à la fois jusqu'à ce que la majorité des exemples de l'ensemble d'entraînement soit couvert par l'ensemble de règles. Conventionnellement une règle de classification est sous la forme *"IF <antécedent> THEN <conséquence>"* où l'antécedent contient entre 0 et plusieurs termes. En pratique une

règle contient largement moins de termes que d'attributs. *Ant-Miner* utilise la représentation de la logique propositionelle pour représenter un terme sous la forme d'un triplet <attribut, opérateur, valeur>. L'opérateur peut être [=,≠ , ≤, ≥], cependant l'algorithme n'utilise que l'opérateur "=". Donc chaque terme est sous la forme <attribut=valeur>.

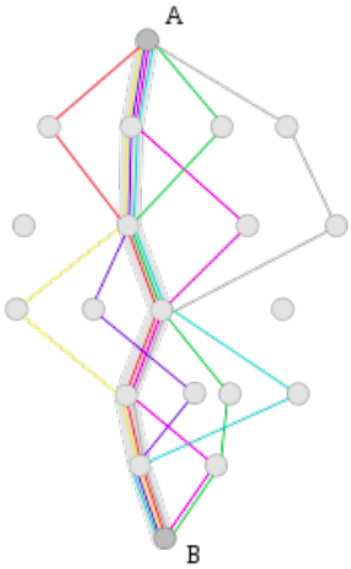


FIGURE 6 - Colonie de fourmis

La contruction des règles se fait à travers des colonies de fourmis. Chaque fourmi produit une règle qui est une solution candidate. Ainsi la solution est choisit parmi l'ensemble des règles produites par la colonie et ajouter à l'ensemble des règles de classification initialement vide. Les exemples du *trainingset* couvert par la solution sont retirés, et une nouvelle colonie de fourmis est mise en place afin de contruire une nouvelle règle pour les exemples restants. Ce processus de construction de règle est itéré jusqu'à ce que le nombre d'exemples non couvert atteigne une limite fixée.
Dans **MuLaM**, la principale différence réside dans le fait que chaque fourmi découvre non une seule règle mais un ensemble de règles couvrant les différents labels. L'espace de recherche est représenté conceptuellement sous la forme d'un graphe où chaque noeud représente une partie de la solution candidate et les arètes correspondent aux mouvements des fourmis dans l'espace de recherche. Le chemin suivit par une fourmis dans un graphe représente le processus de construction d'une solution candidante(Voir Fig.6).

Gene Expression Programming for Multi-label Classification (GEP-MLC)

Basé sur **GEP**(Gene Expression Programming), **GEP-MLC** est un algorithme de classification *Multi-Label* proposé par *S. Ventura , E. Gibaja et A. Zafra*. **GEP** est un algorithme évolutionnaire. Il propose une représentation des individus qui est simple à faire évoluer et permet à la fois de représenter des structures complexes. Dans **GEP**, chaque individu est représenté par un double codage. Le génotype composé d'un ensemble de gènes représenté sous forme de chaîne de caractères, et le phénotype associé à chaque gène est sous la forme d'un arbre d'expression. L'arbre d'expression(composé de fonctions et de valeurs) est généré à travers la chaîne répresentant un gène. Le phénotype d'un individu en fonction du problème peut-être un seul arbre généré à travers l'arbre de chaque gène de cet individu où l'ensemble des arbres de ses gènes. Le paradigme **GEP** définit un ensemble d'opérations génétiques adaptées à la double représentation des individus. Nous pouvons citer le *crossover*(une partie du génotype des parents est interchangée), *mutation*(changement aléatoire à une position aléatoire) et *transposition*(transfert d'un frangment de gène à une autre position).Plusieurs autres opérations dérivés existent.

GEP-MLC[4] essaie d'apprendre un ensemble de fonctions discriminantes pour chaque label dans un problème de classification. Une fonction discriminante est une fonction mathématique qui reçoit un vecteur caractéristique de chaque modèle et renvoie une valeur réelle. En fonction d'un seuil(ici 0), cette valeur est interprété pour déterminer l'appartenance à une classe ou pas. Ainsi nous pouvons conceptuellement représenter cela sous cette forme : *IF (f(x) > seuil) THEN* $x \in$ *class ELSE* $\notin$ *class* ,avec *f(x)* la fonction discriminante.
Ainsi le classifieur consiste en un ensemble de règles(chacune associée à une fonction discriminante) sous

forme normale disjonctive(lié par l'opérateur logique OR).

Les individus sont représentés sous la forme de fonction discriminante. L'évaluation de l'arbre d'expression représente la valeur retournée par la fonction discriminante, ce qui permet ainsi de faire des prédictions. L'évaluation des individus est basé sur leurs prédictions. Elle est mésurée avec la métrique *F1-Score*(permet l'optimisation de deux métriques opposés, *Precision* et *Recall*) :

$$fitness = \frac{2 \times precision \times recall}{precision + recall}$$

Le fonctionnement de l'algorithme est le suivant. Nous partons d'une population d'individus diversifiés. Ensuite pour chaque classe, nous procédons à l'évaluation de la performance de prédiction des individus. Ensuite après cette évaluation, une correction individuelle du fitness pour chaque classe et chaque individu est mise en place. Il s'agit du *ToKen Competition*(distribution de jeton aux plus performants, ce qui améliore leur fitness après correction) qui permet d'émuler l'effet de niche présent dans un écosystème afin d'avoir des individus spécialisés dans la prédiction d'une classe bien particulière. Ce procédé favorise les plus performants et déforise les moins performants dans la prédiction. La formule utilisé pour la correction est la suivante :

$$fitness = \frac{original\_fitness \times jeton\_gagns}{nb\_jetons\_total}.$$

Après le *Token competition*, les meilleurs individus sont sélectionné afin de constituer la nouvelle génération en utilisant les opérations génétiques évoquées plus haut(crossover, transpostitions, etc...).

Quand l'algorithme est finit après un nombre de générations préfixé, il est facile de trouver les meilleurs individus possédant les meilleurs fonctions discriminantes afin de mettre en place les règles de classifications.

3.2 Proposition de résolution de classification multi-Label en problème d'Optimisation

Les deux algorithmes présentés ci-dessus malgré leurs résultats équivalents voire supérieurs aux algorithmes de l'état de l'art, peuvent voir certains de leurs points à améliorer.

La représentation des termes dans **MuLaM** n'utilise que l'opétateur '=', ce qui limite la puissance d'expression de ces règles. Ensuite dans **GEP-ML**, les fonctions discriminantes quant à elle présentent le désavantage de ne pas être facilement interprétables. Dans la méthode présenté ci-dessous, nous essayerons de remédier aux désavantages de ces méthodes.

L'idée générale de cet algorithme est basé sur la création d'un ensemble de règles de classification en utilisant un algorithme évolutionnaire. Les règles sont représentées sous la forme d'une suite de termes liées par l'opérateur logique AND. Les termes sont comme dans **MuLaM** à la seule différence qu'on peut utiliser toutes les opérations suivantes [=,$\neq$, $\leq$, $\geq$], ce qui permet d'augmenter la puissance d'expressions des termes. Pour chaque classe, l'algorithme évolutionnaire parcourera l'espace de recherche, soit un ensemble de termes formés à partir de données afin de trouver des règles représentant au mieux la classe courante. Ci-dessous vous trouverez un pseudo-code présentant cette méthode.

```
training_set = {ensemble des données}
ensemble_regles = {}
Pour Chaque Classe:
     current_data = {données réprésentant cette classe}
     Prepocessing(current_data)
     Generation_Espace_Recherche(current_data)
     Generation_population_initiale()
     Generation = 0
     Tant Que Generation < Max_Generation :
          /*Algorithme évolution classique*/
          Evaluation_Individus()
          Selection_Pour_Reproduction()
          Croisement()/ Mutations()
          Selection_Pour_Remplacement()
          Formation_Generation_Suivante()
          Generation += 1
     regles = Extraction_regles_individus()
     ensemble_regles =  ensemble_regles U regles
Generationclassifieur(ensemble_regles)
```

L'étape du pré-processing est décomposé en 2 étapes, la première consiste à discrétiser les valeurs numériques si elles sont continues. La seconde consiste à calculer pour la classe courante(X) son niveau de corrélation avec les autres classes(Y) avec la formule suivante(voir ci-dessous) : $\frac{nb-exemples-contenant-Y}{nb-exemples-contenant-X}$

Attr-1	Attr-2	Classes/Labels
3	true	A,B
7	true	A,B
2	false	B
1	false	C

TABLE 3 – *Donnée Brut pour les classes*[A,B,C]

Ex : Exemple calcul durant le pré-processing pour classe B
$CorellationB_A = 2/3 = 0.66$
$CorellationB_C = 0/3 = 0$

La génération de l'espace de recherche pour les individus de l'algorithme évolutionnaire se fera à travers les données et sera un ensemble de termes. Ainsi pour la Table 3(classe B) on aura : {Attr-1 *op* 3, Attr-1 op 7, Attr-1 *op* 2, Attr-2 = false, Attr-2 = true} avec *"op"* représentant une opération de la liste [=,≠ , ≤, ≥].

Quant à la génération de la population initiale, elle se consiste à générer 100 individus de manière semi-aléatoire en se basant sur l'espace de recherche. Un individu est représenté sous la forme d'une ou plusieurs termes. On veillera à ce que les individus de la population initiale n'ai qu'un seul terme.

Ex : Exemple d'individus pour classe B
$IndividuB_1$ = [Representation : *"(Attr-1 = 3)"* fitness : 0.2]
$IndividuB_2$ = [Representation : *"(Attr-1 ≥ 2) AND (Attr-2 = false)"* fitness : 0.7]

L'évaluation des individus se fait à travers le **F1-Score** comme dans **GEP-ML** soit :

$$fitness = \frac{2 \times precision \times recall}{precision + recall}$$

L'opération de croisement est représenté par l'ajout des règles du parent 1 et 2 à l'enfant. La mutation est l'ajout,la supression ou le changement de valeur(ou opérateur) d'un attribut de l'individu. Ce changement de valeur intervient dans l'ensemble de définition de cet attribut. Ainsi pour les exemples précédents on pourra avoir :
$MutationIndividuB_1$ = [Representation : *"(Attr-1 = 7)"* fitness : 0.4]
$EnfantB_1B_2$ = [Representation : *"(Attr-1 ≥ 2) AND (Attr-2 = true)"* fitness : 0.6]

La sélection pour le remplacement est basé sur le fitness. Ensuite les règles sont extraites des individus de la dernière génération et seront rajoutées à l'ensemble des règles découvertes. Le classifieur sera constitué de toutes les règles découvertes.

Conclusion

Ce rapport présente la classification *Multi-label* qui se définit comme l'association d'un ou plusieurs label à une instance en utilisant différentes méthodes. Nous avons commencé par présenter les Méta-heuristiques, des algorithmes génériques qui permettent de résoudre des problèmes d'optimisation difficiles. Ensuite nous avons présenté les deux grandes catégories de techniques de résolution de problèmes de classification *Multi-label*. Enfin nous avons présenté les travaux de classification *Multi-Label* existants utilisant les méta-heuristiques avant de proposer une nouvelle approche de résolution s'inspirant de **MuLaM** et **GEP-ML**. Cette approche utilise un algorithme évolutionnaire afin de former un ensemble de règles de classification. L'avantage de cette méthode réside dans sa simplicité de compréhension et la puissance de ses règles. Une poursuite de ce travail serai l'implémentation de cette nouvelle approche et la comparer avec celles de l'état de l'art.

	Synthèse de Lecture				
	MuLaM	GEP-ML	GC	Rank-SVM	G3P-ML
Auteur	Chan & Freitas	Ventura et al.	Ventura et al.	T.Joachims et al.	Ventura et al.
Application	biologie	Image/biologie	*undefined*	biologie	biologie
Métrique utilisée	*undefined*	F1-Score	undefined	Ranking-Loss	Order Bias Distance Metric
Méta-heuristique	ACO	Algo. Gén.	Algo. Gén.	*undefined*	Algo. Gén.

TABLE 4 – Classification des articles

Références

[1] Elisseeff, A., Weston, J. : *A kernel method for multi-labelled classification.* In : Advances in Neural Information Processing Systems. (NIPS), vol. 14, pp. 681–687 (2001)

[2] Chan, A., Freitas, A.A. : *A new ant colony algorithm for multi-label classification with applications in bioinfomatics.* In : GECCO '06 : Proceedings of the 8th annual conference on Genetic and evolutionary computation, pp. 27–34. New York, USA (2006)

[3] Parpinelli, R., Lopes, H., Freitas, A. : *Data Mining with an Ant Colony Optimization Algorithm.* EEE Trans. On Evolutionary Computation 6(4), 321–332 (2002)

[4] Ávila, J.L., Gibaja, E.L., Zafra, A., Ventura, S. : *A Gene Expression Programming Algorithm for Multi-Label Classification.* Journal of Multiple-Valued Logic and Soft Computing 17, 183–206 (2011)

[5] Gibaja, Eva Ventura, Sebastian : *Multilabel Learning : A Review of the State of The Art and Ongoing Research.* Wiley Interdisciplinary Reviews : Data Mining and Knowledge Discovery.(2014)

[6] Tsoumakas, G., Katakis, I., Vlahavas, I. : *ata Mining and Knowledge Discovery Hand book*, Part 6, chap. Mining Multi-label Data, pp. 667–685. Springer (2010)