Apprentissage automatique interactif pour les opérateurs du réseau électrique

Laure Crochepierre

UNIVERSITÉ DE LORRAINE

École doctorale IAEM Lorraine

Apprentissage automatique interactif pour les opérateurs du réseau électrique

THÈSE

présentée et soutenue publiquement le 27 Septembre 2022

pour l'obtention du

Doctorat de l'Université de Lorraine

(mention informatique)

par

Laure Crochepierre

Composition du jury

Président :	Marianne Clausel	Professeur, Université de Lorraine
Rapporteurs :	Laure Berti-Equille	Directeur de Recherche, IRD
	Antoine Cornuéjols	Professeur, AgroParisTech
Examinateurs :	Matthieu Geist	Professeur, Université de Lorraine, détaché à Google
	Christian Derquenne	Docteur, EDF R&D
Directrice de thèse :	Lydia Boudjeloud-Assala	Maître de Conférence, HDR, Université de Lorraine
Invités :	Vincent Barbesant	Ingénieur de recherche, Rte R&D
	Benjamin Donnot	Docteur, Rte Direction Valoration des données
	Antoine Marot	Ingénieur, Rte Direction Valoration des données

Laboratoire Lorrain de Recherche en Informatique et ses Applications — UMR 7503

Mis en page avec la classe thesul.

Remerciements

Je me dois de remercier toutes les personnes grâce à qui ces années se sont passées dans les meilleures conditions possibles et ont contribué de près ou de loin à la réussite de cette thèse. Je remercie toutes les personnes qui m'ont relue et questionnée, grâce à qui j'ai grandi, je me suis améliorée et j'ai constamment appris pendant ces trois années de thèse.

Je remercie en premier lieu Laure Berti-Equille et Antoine Cornuéjols d'avoir bien voulu être rapporteurs de cette thèse. Je les remercie pour tout le temps qu'ils ont consacré à la lecture de ce document. Leurs suggestions et remarques ont beaucoup contribué à son amélioration. Merci également à Marianne Clausel, Matthieu Geist, Christian Derquenne d'avoir accepté de participer à ce jury en tant qu'examinateurs.

Je voudrais, de même, vivement remercier mes deux encadrants de thèse, sans qui cette thèse aurait été une toute autre aventure. Tout d'abord un grand merci à ma directrice de thèse Lydia Boudjeloud-Assala pour sa disponibilité, sa gentillesse, sa rigueur et ses conseils. Sans son soutien au début de pandémie (et les suivants), ces mois incertains auraient été beaucoup plus difficiles. Un grand merci également à Vincent Barbesant pour son encadrement, son expertise et son écoute tout au long de la thèse. Grâce à son aide pour la définition d'une problématique industrielle réelle, j'espère avoir apporté ma petite pierre utile à l'édifice de la recherche au sein d'RTE.

Je remercie également Antoine Marot qui me suit depuis mon arrivée en tant que stagiaire au sein d'RTE, à l'initiative de qui j'ai pu commencer mon doctorat et qui m'a poussée, tout au long de ma thèse, à élargir mes perspectives. Merci à Remy Clément, Balthazar Donnon, Benjamin Donnot, Matthieu Dussartre, Clément Goubet, Alexandre Rozier, pour nos discussions fructueuses et les idées (parfois farfelues) qui en sont sorties. Merci également pour vos relectures pointues m'ont souvent amenée à mieux justifier et remettre en question mes hypothèses ! Merci à toute l'équipe Assistants Salle H24 (ex. Apogée-Itesla, ex. Apogée), et plus particulièrement Pauline Gambier-Morel, Camille Pache, Etienne Lesot, Sylvain Leclerc. J'ai énormément apprécié travailler avec vous tous et votre dynamisme m'a amené beaucoup de joie. Je souhaite également exprimer toute ma reconnaissance à Marie-Sophie Debry et Guillaume Trimbach pour leur aide et leur écoute tout au long de ma thèse. Je n'oublie pas de remercier Olivier Bretteville et Patrick Mercier pour votre prévenance et votre aide toujours à point nommé au coeur du système informatique ! Vous aviez toujours à coeur de m'aider lorsque je n'y mettais pas assez de coeurs et sans vos bons coeurs certaines de mes expériences n'auraient jamais vu le bout !

Sur un plan plus personnel, merci à la bande des Lazos et aux copains de Supélec pour votre amitié et vos encouragements. Votre enthousiasme votre joie de vivre sont toujours un grand soutien ! Maintenant cette étape de vie passée, je peux vous dire : à très vite pour de nouvelles aventures à vivre ensemble.

Merci aussi à la 2e compagnie d'arc de Nogent-Sur-Marne, grâce à qui j'ai pu garder un semblant de santé mentale... et de forme physique !

Enfin, je tiens à remercier du fond du coeur toute ma famille pour leur soutien et leurs messages d'encouragement réguliers. Je retournais travailler re-motivée après chacun de nos appels et échanges vidéos.. Les Skypes en famille du confinement ont été salvateurs et je garderai toujours en mémoire notre mardi gras fêté, tous déguisés, à distance. Ce manuscrit est le point final, à toutes mes années d'études : ça y est papa tu vas pouvoir souffler ! Merci de m'avoir soutenue et poussée pendant toutes ces années. Je n'aurais été en mesure de terminer cette thèse sans leur soutien émotionnel.

Un merci tout particulier à ma moitié, Thibault Buhet, qui m'aura soutenue et poussée tous les jours à donner le meilleur de moi-même. Merci pour ton aide autant émotionnelle, logistique, que mathématique et informatique. Merci d'essayer de me comprendre même quand mon raisonnement n'est pas toujours clair. J'espère que tu seras aussi fier de moi que je le suis de toi au quotidien. Merci enfin à ma belle-famille, pour votre gentillesse, votre amitié.

A mon père, pour son amour et son soutien indéfectible

Table des matières

Table des figures

Liste des tableaux

Glossaire

AC : Alternative Current, Courant Alternatif
AG : Algorithme Génétique
AR : Algorithme de Renforcement ou Apprentissage par Renforcement
BT : Basse Tension, niveau de tension
BNF : Backus Naur Form, format pour définir une grammaire
Consommation : elle représente un consommateur qui soustrait de l'électricité au réseau. Il peut s'agir d'un consommateur ou d'une agrégation de plusieurs consommateurs.
Calcul de répartition : Calcul de tension et phase aux noeuds électriques, et des transits sur les lignes. Il est aussi appelé *powerflow*
Contrainte : Dépassement du seuil de sécurité pour le transit d'une ligne électrique.
DC : Direct Current,
Dispatcher : Opérateur du réseau électrique.
DT : Decision Tree, ou arbre de décision
CIFRE : Convention Industrielle de Formation par la REcherche
EG : Evolution Grammaticale
EQM : Erreur Quadratique Moyenne
GNC : Grammaire Non Contextuelle
GRT : Gestionnaire de Réseau de Transport
HT : Haute Tension, niveau de tension 150, 90, et 63 kV
IEC : Interactive Evolutionnary Computation
IA : Intelligence Artificielle
iML : interactive Machine Learning, ou *apprentissage automatique interactif*
Injections : Ensemble des productions et des consommations
Leviers : Actions parmi lesquelles un opérateur peut choisir pour résoudre une contrainte
Ligne : Ligne électrique, reliant différents **noeuds** du réseau électrique
LSTM : Long Short Term Memory
MDP : Markov Decision Process
ML : Machine Learning
MT : Machine Teaching
Noeud : Point du réseau électrique où sont reliés les productions et les consommations
PG : Programmation Génétique
POMDP : Partially Observable Markov Decision Process
Production : Entité qui injecte de l'électricité dans le réseau électrique
RL : Reinforcement Learning, voir Apprentissage par renforcement (AR)
RS : Régression Symbolique
RTE : Réseau de Transport d'Electricité, l'entreprise en charge de la gestion du réseau de transport de l'électricité France
THT : Très Haute Tension, niveau de tension supérieur ou égal à 225 kV

TLF : Théorie des Langages Formels

Topologie : Description des connections physiques entre les différents composants du réseau électrique

Chapitre 1

Introduction

1.1 Introduction

Le réseau électrique est l'un des composants majeurs du fonctionnement d'un pays, et dont la tâche principale est d'acheminer l'énergie électrique depuis des centrales de production vers les consommateurs. Cependant, bien que l'accès à l'électricité soit aujourd'hui considéré comme un bien commun (Assemblée Nationale, 2020) sur lequel repose nos sociétés modernes, sa gestion n'en est pas pour autant acquise. En effet, l'installation de nouveaux moyens de production d'énergie renouvelable sur le réseau électrique, tel que le photovoltaïque et l'éolien, ainsi que la stratégie d'interconnexions des réseaux d'électricité avec les pays frontaliers, conduisent progressivement à des changements profonds dans le fonctionnement du réseau.

Face à ces évolutions, les opérateurs en charge de la gestion du réseau de transport d'électricité sont amenés à travailler dans un environnement aux dynamiques de plus en plus imprévisibles et rapides. Plus précisément, l'une des tâches importantes, et réalisée à intervalles réguliers par les opérateurs, est la synthèse d'informations issues de multiples outils pour connaître l'état de sûreté du réseau à un instant donné ou dans le futur. Elle a pour but d'identifier les éléments pouvant mettre en danger la sécurité l'approvisionnement en électricité. Or, cette tâche peut être longue à cause du grand nombre d'informations à synthétiser. Les opérateurs travaillent par ailleurs en temps de plus en plus contraints, cette synthèse d'informations doit être réalisée efficacement le plus rapidement possible. Comment assister les opérateurs dans cette étape de synthèse ? C'est cette question qui a poussé le Gestionnaire de Réseau de Transport (GRT) français, Réseau de Transport d'Electricité (RTE) à vouloir investiguer l'utilisation interactive de méthodes d'apprentissage automatique pour le réseau électrique, et plus particulièrement leur application à l'étude des transits électriques.

Bien que l'entreprise RTE dispose déjà d'outils permettant aux opérateurs de s'informer sur l'état du réseau, ceux-ci ne répondent pas entièrement aux futurs enjeux de synthèse d'informations. Les opérateurs utilisent notamment des indicateurs créés manuellement et fournissant une vision synthétique du réseau. Cependant, ces indicateurs ne sont pas mis à jour régulièrement car leur actualisation peut être longue et requiert une connaissance pointue du fonctionnement du réseau. Cette thèse vise donc à proposer des algorithmes d'apprentissage automatique capables d'intégrer et de prendre en compte de la connaissance experte pour construire et mettre à jour des indicateurs interprétables par les opérateurs.

Afin de détailler les différents éléments mentionnés ci-dessus, ce premier chapitre décrit le contexte général dans lequel s'inscrit cette thèse. Dans un second temps, nous introduisons les critères utilisés pour garantir la sécurité du réseau électrique, afin d'appréhender les défis liés

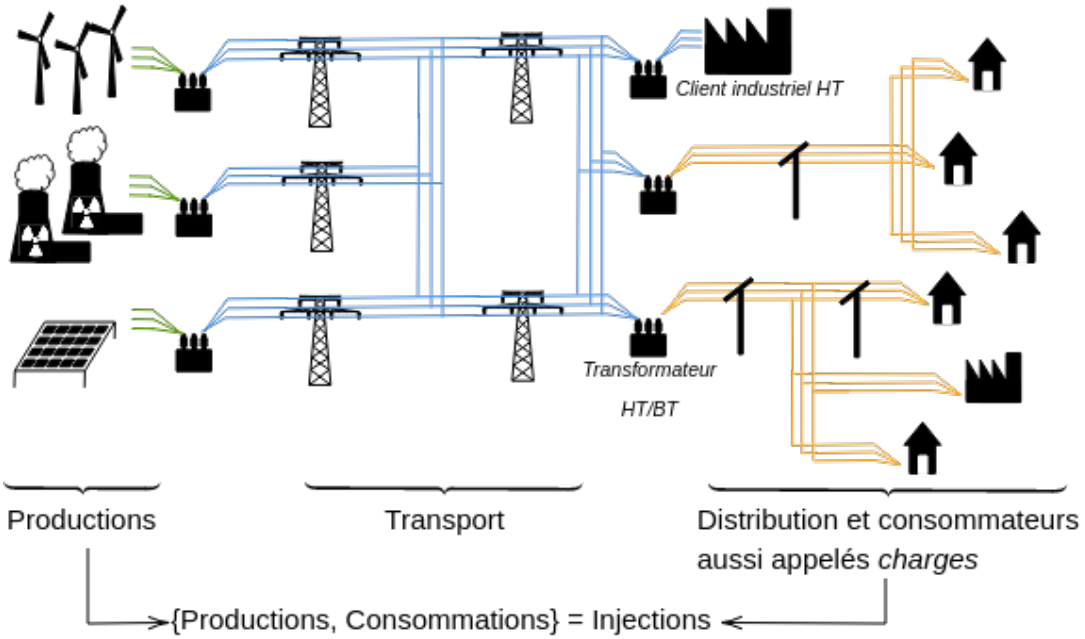


FIGURE 1.1 – Schéma synthétique du réseau électrique selon le découpage : production (en vert), transport (en bleu), distribution/consommation (en jaune). Selon le point de vue du GRT, le réseau de distribution et les consommations sont aggrégées au niveau des transformateurs, à l'interface entre transport et distribution (resp. bleu et jaune). Le réseau de transport est exploité en Haute Tension (HT) et Très Haute Tension (THT) et le réseau de distribution est en Basse Tension (BT). Par simplification, les changements de niveau de tension au sein du réseau de transport ne sont pas représentés.

avec la complétion de ces critères. Puis, nous présentons la manière dont le réseau est géré par les GRTs de nos jours, suivi des enjeux et des pistes d'évolution de sa gestion quotidienne. Dans un dernier temps, nous présentons la problématique et les contributions de cette thèse.

1.2 Contexte général

1.2.1 Quelques définitions

Production et consommation : les injections d'électricité

Le réseau électrique, tel que celui présenté dans la figure 1.1, est la structure matérielle permettant d'acheminer l'énergie électrique depuis des centrales de *production* vers les *consommateurs* finaux. Définissons dans un premier temps les termes de production et de consommation :

— La *production* est composée de différents moyens de génération d'énergie électrique parmi lesquels se trouvent les centrales thermiques (nucléaires, au gaz, mais également au charbon, au fioul, ...), les centrales hydrauliques ainsi que les autres moyens de production renouvelables (éolien, photovoltaïque, géothermique,...).

— La *consommation* comprend les consommateurs particuliers et industriels raccordés au réseau. Il s'agit d'une grandeur *imposée* au réseau électrique. En d'autres termes, les gestion-

naires de réseaux ne peuvent avoir une action directe sur la quantité d'électricité consommée par leurs clients et ne déconnecteront des consommations qu'en tout dernier recours.

Du fait de l'impossibilité actuelle de stocker l'électricité à grande échelle (par exemple au moyen de barrages hydrauliques à pompage-turbinage, ou de batteries), le système électrique requiert d'assurer en permanence l'égalité entre la quantité d'électricité produite et celle consommée pour assurer l'équilibre du réseau. Avec la définition de la consommation donnée plus haut, il est alors nécessaire de jouer sur d'autres paramètres du réseau, tel que les *productions*, pour en maintenir l'équilibre.

Ossature du réseau moderne en France

Le *système électrique* est généralement décomposé en trois éléments principaux : la *production*, le *transport*, la *distribution*. Comme synthétisé dans la figure 1.1, le réseau de transport inter-connecte toutes les principales productions du réseau pour faire transiter sur de grandes distances l'électricité produite vers le réseau de distribution au moyen d'un grand nombre de lignes électriques inter-connectées entre elles. Le réseau de transport est géré à Haute et Très Haute Tension (HT et THT), afin de limiter les pertes d'énergie :

— *Très Haute Tension* ou *THT* pour le transport sur de grandes distances et les interconnexions avec les pays voisins. La tension (qui a pour unité de Volt, noté V), est de 225 ou 400 kV pour les lignes THT.

— *Haute Tension* ou *HT* (150, 90, 63 kV) assurant la répartition électrique à l'échelle régionale, jusqu'aux réseaux de distribution, ainsi qu'aux industriels directement connectés au réseau haute tension.

La gestion du réseau de transport d'électricité en France est assurée par l'entreprise RTE (Réseau de Transport d'Electricité). Elle est en charge de l'exploitation, la maintenance et le développement du réseau électrique HT et THT. Il s'agit de l'un des plus grands GRT à l'échelle européenne qui assure à lui seul la gestion d'un peu plus de 10 000 lignes électriques, correspondant à près de 100 000 km sur l'intégralité du territoire français métropolitain. Le réseau de distribution pour sa part est exploité à moyenne (20 kV) et basse tension (ou BT), par le gestionnaire de réseau de distribution (tel que Enedis), et achemine l'électricité vers les consommateurs.

Nous adopterons dans la suite du document, le "point de vue" d'un GRT pour lequel distribution et consommation correspondent à une même catégorie d'acteurs que nous nommerons consommation ou *charge*, i.e. chaque *consommation* est ici considérée comme un agrégat de plusieurs consommateurs. Notons également que dans ce système, la production et la distribution correspondent respectivement aux principales entrées et aux sorties d'énergie du réseau de transport. Par la suite, nous nommerons l'ensemble des productions et des consommations sous le terme *injections*.

1.2.2 La transition actuelle du réseau français dans l'histoire moderne

La naissance d'un réseau centralisé et interconnecté

Le réseau électrique français n'a pas toujours été tel que nous le connaissons aujourd'hui, mais est le fruit d'évolutions successives, initiées à différentes époques pour assurer, puis conserver, une qualité d'approvisionnement électrique tout en accompagnant l'installation de nouveaux moyens de production et la demande des consommateurs. Initialement doté avant la Seconde Guerre Mondiale d'une structure locale de répartition de l'énergie, le réseau électrique a été uniformisé

au cours des années qui ont suivi, notamment par la nationalisation des quelque 86 et 1150 entreprises gérant respectivement le réseau de transport et de distribution (Barjot, 2013).

Une deuxième période d'évolution de structure du réseau a eu lieu avec la mise en place du parc nucléaire français. Impulsée par le plan Messmer, la capacité nucléaire a crû au cours des années 1970, pour atteindre environ 70% de la production d'électricité en 2019. Le réseau s'est progressivement étendu et densifié dans le même temps pour faire face à la demande croissante en électricité, passée de 171 TWh (térawatt-heure) en 1973 pour se stabiliser autour de 475 TWh dans les années 2010 (EDF, 2022).

Une autre transformation notable est survenue au cours de ces dernières années avec la libéralisation progressive du marché de l'énergie, sous l'impulsion des directives européennes votées dès 1996. Dans cette période, la France est passée d'un monopole avec une stratégie d'intégration verticale, à un découpage en activités production / transport / distribution, et où les activités de production sont ouvertes à la concurrence. Ces modifications ont conduit à un changement de paradigme dans la gestion des réseaux électriques. Les acteurs se sont alors démultipliés et ont modifié le paysage énergétique industriel français, offrant à la fois de nouvelles dynamiques d'investissements, mais également de nouveaux défis dans la gestion du réseau.

Aujourd'hui, dans un "nouveau" contexte de transition énergétique, le réseau de transport d'électricité est une nouvelle fois amené à changer, et ce, avec des contraintes différentes de celles précédemment citées.

Un réseau en transition

Bien que la quantité d'électricité consommée en France se soit stabilisée au cours des dernières années, les usages de l'électricité évoluent depuis toujours au gré des innovations technologiques (Beltran, 2004). Cette évolution est d'autant plus marquée dans l'histoire récente des réseaux, car elle tente de répondre aujourd'hui aux enjeux de la transition écologique pour la réduction drastique et rapide des gaz à effet de serre. Des changements significatifs sont attendus, par exemple, dans le domaine de la mobilité ou de l'habitat. Avec une prévision de 15 millions de véhicules électriques à l'horizon 2035 (Le Monde, 2021), pour la plupart rechargés à domicile, la consommation d'électricité pourrait de nouveau croître et connaître de nouvelles dynamiques intra-journalières. De même, la rénovation énergétique des bâtiments, l'arrivée progressive des objets connectés, ainsi que la mise à disposition de compteurs communicants et l'accroissement des projets d'autoconsommation, seront tous des facteurs de ces nouvelles dynamiques de consommation.

Par ailleurs, les moyens de production connaissent eux-aussi une transition profonde, comme détaillée dans la figure 1.2. L'installation massive de nouveaux moyens de production renouvelables décentralisés, tel que l'éolien et le photovoltaïque modifie significativement la manière dont l'électricité transite sur le réseau. Ainsi, plus de 30 000 sites de production d'énergies renouvelables sont déjà raccordés au réseau, et entre un à deux millions de nouveaux sites sont attendus d'ici dix ans (Le Monde, 2021). Ces changements déjà initiés sont amenés à se poursuivre et à s'intensifier au cours des prochaines décennies pour parvenir à l'objectif d'une neutralité carbone à horizon 2050 (Rte, 2021).

Le développement des interconnexions entre réseaux nationaux en Europe va également continuer à jouer un rôle significatif dans la mise en place à grande échelle de productions renouvelables intermittentes. Fruit d'efforts européens débutés dès 1950 (Bouneau, 2004), la stratégie d'interconnexion sur de grandes zones synchrones (c'est-à-dire exploitées à la même fréquence), permet aujourd'hui d'exploiter et de partager de grandes quantités d'énergie, dont le renouvelable, ainsi que d'améliorer la fiabilité du système.

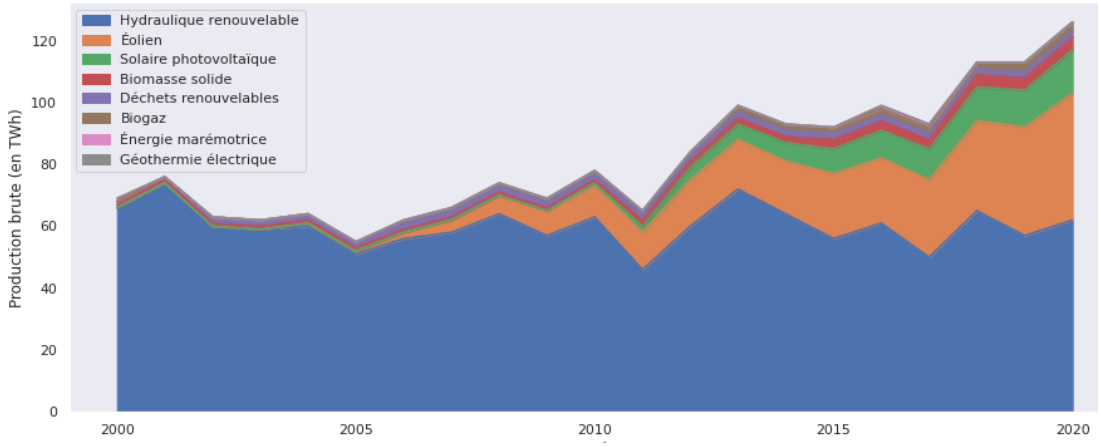


FIGURE 1.2 – Évolution de la production brute d'énergie renouvelable par filière en France de 2000 à 2020. (Ministère de la transition écologique, 2021)

Cependant, ces dernières années, l'installation de nouvelles lignes électriques pour faire face aux évolutions susmentionnées est freinée par leur coût élevé et la contrainte d'acceptation sociale relative à l'insertion de nouveaux composants sur le réseau. Pour faire face à ces contraintes, les GRTs sont aujourd'hui amenés à gérer le réseau plus près de ses limites d'exploitation. Là où l'acheminement de l'électricité était une tâche intrinsèquement complexe, cette complexité continue à croître, incitant par ailleurs les GRT à repenser leur gestion du réseau, pour faire face aux transformations anticipées dans un futur proche, en faveur de nouvelles modalités d'exploitation plutôt que du déploiement massif de nouvelles infrastructures. Détaillons maintenant plus précisément la structure actuelle du réseau électrique étudié dans le cadre de cette thèse.

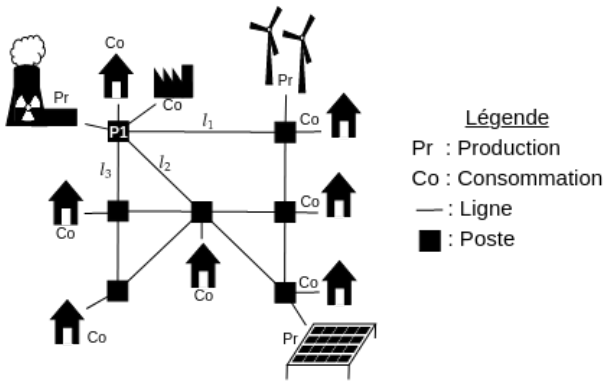
1.2.3 Structure et composants du réseau de transport

Le réseau de *transport* décrit dans cette thèse a donc pour objectif de faire transiter l'énergie sur de grandes distances depuis les sites de production, jusqu'aux consommateurs. Ce transport se fait au moyen d'un grand nombre d'ouvrages, parmi lesquels les lignes électriques jouent le rôle de support du transit électrique. Le réseau de transport, contrairement au réseau de distribution, est fortement maillé afin de disposer du plus de redondance possible dans le cas d'incident sur une ligne. Nous détaillerons l'importance de cette redondance dans la sous-section 1.3.2.

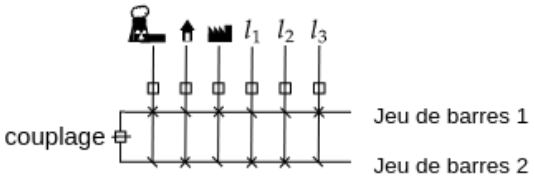
Les lignes sont connectées entre elles au sein de *postes électriques* où sont également connectées les injections, comme présenté dans la figure 1.3a. Plus précisément, il s'agit de l'endroit physique et géographique où il est possible d'effectuer des ré-aiguillages des transits électriques afin que l'électricité soit répartie sur les lignes en fonction de leurs capacités de transport.

Il existe donc plusieurs manières de connecter entre elles les lignes électriques au sein d'un poste. Nous appelons une *topologie* une possibilité de connexion d'un ensemble de lignes. Le terme topologie peut faire référence à la connexion entre lignes à l'échelle d'un poste ou à l'échelle d'une zone géographique. La topologie est un levier important dans la gestion du réseau, car il s'agit d'un paramètre à la main des opérateurs qui peuvent choisir pour passer d'une topologie à une autre.

Zoomons sur une topologie particulière du poste P1 (en haut à gauche de la figure 1.3a). Un exemple de connexion entre les lignes au sein du poste P1 (i.e. une topologie spécifique) est illustré dans la figure 1.3b. Dans le cas présenté ici, chacune des lignes électriques peut être



(a) Schéma synthétique du réseau de transport d'électricité. Les postes électriques sont reliés par des lignes. Plusieurs productions et consommations peuvent être connectées au même poste.



(b) Schéma du poste P1 pour une topologie donnée, notation RTE. Le poste comporte deux jeux de barres, ici découplés (i.e le couplage est ouvert comme représenté par le symbole ⫟). × représente la connexion d'une ligne sur une barre et \ la déconnexion. Le poste comporte alors deux noeuds électriques : le premier {, , l_3}, et le deuxième {, l_1, l_2}. Entre les deux noeuds ne transite pas directement d'électricité.

FIGURE 1.3 – Schémas synthétiques du réseau de transport d'électricité ainsi qu'un zoom sur la connexion électrique au sein d'un poste électrique (P1).

connectée soit au jeu de barres 1, soit au jeu de barres 2. Il est également possible (ou non) de relier électriquement les deux jeux de barres. Ces différentes combinaisons de connexions génèrent ici environ 2×6^6 topologies différentes, dont certaines sont électriquement équivalentes, ou sont opérationnellement non-utilisées. Dans le cas de la topologie étudiée, les deux barres sont déconnectées. Ceci est symbolisé par une barre horizontale au niveau du *couplage* entre les deux barres. Sur ce schéma, les connexions entre les lignes et les jeux de barres sont représentées par le symbole ×, et les déconnexions par le symbole \. Dans cas de la figure 1.3b, il existe deux *noeuds* électriques : les éléments {, , l_3} tous les trois reliés électriquement d'un côté forment un premier noeud, et les éléments {, l_1, l_2} un deuxième noeud. Cependant les deux noeuds ne sont pas reliés et l'électricité ne circule pas directement entre les deux : tous les éléments du noeud {, l_1, l_2} sont reliés uniquement au jeu de barres 1, tandis que ceux du noeud {, , l_3} sont uniquement reliés au jeu de barres 2 et les deux noeuds ne sont pas électriquement connectés. La topologie peut être modifiée par les opérateurs et varie donc au cours du temps. Dans cette thèse, nous considérons les informations de topologie comme des variables observées dans les postes électriques.

1.3 Sécurité du réseau électrique

Comme nous l'avons introduit plus haut, le réseau électrique est composé de câbles conducteurs nommés lignes reliant des moyens de production d'énergie, à des consommateurs appelés charges. Afin de maintenir le réseau en sécurité au cours du temps, il est nécessaire d'en connaître l'état en l'observant ou en le simulant.

1.3.1 Calcul d'un point de fonctionnement du réseau

Calcul de répartition

Afin de connaître régulièrement l'état du réseau au cours du temps, il est nécessaire d'en construire un point de fonctionnement estimé à partir des grandeurs connues au niveau des *injections*. Ce problème est connu sous le nom de *calcul de répartition* ou *powerflow*. Au niveau des injections, les grandeurs mentionnées peuvent être acquises de deux manières différentes et correspondent à deux modes d'utilisation :

— Le mode temps-réel : Le pas de temps étudié est alors situé dans le passé. Les grandeurs et la topologie sont issues de mesures réalisées sur le réseau. N'étant pas mesurées exactement au même instant, les différentes grandeurs sont figées, puis traitées à l'aide d'un processus appelé *estimateur d'état*.

— Le mode prévisionnel : Les grandeurs sont issues de processus métier de planifications, qui ont pour objectif de créer une situation de réseau prévisionnelle comportant des valeurs de production et consommation proches de celles qui doivent se réaliser.

Enjeux

La résolution du problème de powerflow et l'estimation de transit peut se faire au moyen de deux méthodes qui seront détaillées dans le chapitre 2 : le powerflow en Courant Alternatif (aussi noté powerflow en "AC"), ou son approximation en courant continu (powerflow en "DC").

La méthode en "AC" est une méthode précise, mais qui peut se révéler longue à calculer lorsqu'il est nécessaire de réaliser successivement un grand nombre de simulations, par exemple pour tester différentes variations de *prévisions* ou de variations de topologie. L'approximation en "DC" ou "Direct Current" est une linéarisation du problème en "AC" communément utilisée sur le réseau électrique. Bien que cette méthode présente des défauts du fait des simplifications qu'elle implique, elle présente aussi l'avantage d'être rapide à calculer et de toujours fournir une solution car elle ne diverge pas. A cause de ses simplifications, elle est également moins précise et, de fait, elle peut difficilement être utilisée pour des applications en temps réel qui requièrent des résultats précis. Elle est plutôt utilisée pour des études de développement où il est nécessaire de réaliser de grandes quantités de simulations. Les méthodes proposées et développées dans cette thèse viseront donc à proposer un compromis entre les deux méthodes ici présentées.

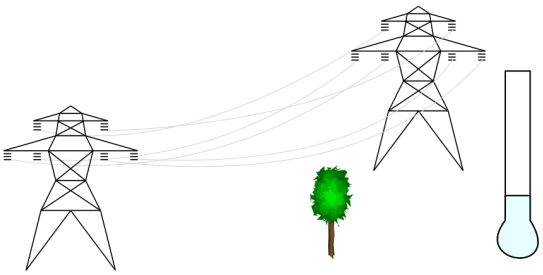
1.3.2 Critères de sécurité du réseau

Les deux méthodes de résolutions en "AC" et en "DC" fournissent des valeurs de transits électriques à partir desquels il est possible de vérifier si le réseau se trouve dans un état sûr. Nous allons maintenant détailler ci-dessous les critères de sécurité pour les transits électriques.

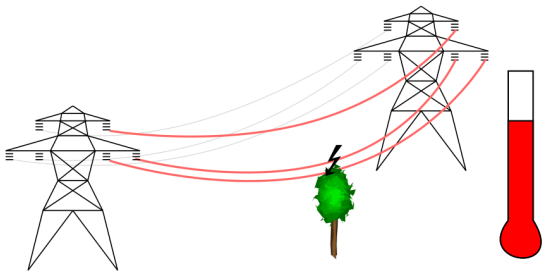
Limite thermique

L'objectif d'un GRT est de conduire l'électricité et de maintenir le réseau en sécurité *à tout instant*. Pour ce faire, il est nécessaire de s'assurer que chacun de ses composants est utilisé dans sa *plage de fonctionnement*. En ce qui concerne les lignes électriques, cela consiste à vérifier que la quantité d'électricité transitant sur chaque ligne, nommée *courant*, ne dépasse pas sa valeur maximale autorisée, choisie selon des critères opérationnels.

En effet, le passage d'un courant dans une ligne génère un échauffement des matériaux qui la compose. Comme illustré par la figure 1.4b, cet échauffement, appelé *effet Joule*, induit une



(a) Illustration de l'état d'une ligne en sécurité. Lorsque la quantité d'électricité circulant sur la ligne est en deçà de sa limite de fonctionnement, la ligne est suffisamment éloignée du sol.



(b) Illustration des conséquences d'une contrainte de transit. L'échauffement d'une ligne entraîne sa dilatation, son rapprochement du sol, pouvant lui-même entraîner des court-circuits.

FIGURE 1.4 – Illustration du comportement d'une ligne en sécurité (a) et d'une ligne en contrainte (b).

perte d'énergie sous forme de chaleur ainsi qu'une dilatation de la ligne. Cette dilatation réduit la distance entre la ligne et le sol, augmentant dans le même temps les risques de court-circuit mettant en danger les biens et les personnes situées à proximité. Afin de maîtriser ces risques, un ou plusieurs seuils d'intensité sont définis pour chaque ligne : ce sont les *limites thermiques*. Dans le cas où une ligne dépasserait sa limite à un instant donné, la ligne serait alors dite en *surcharge*.

Le non-dépassement de ces seuils, illustré par la figure 1.4a, garantit alors le respect des distances de sécurité et le bon fonctionnement du matériel. A partir de la définition de ces seuils, deux critères de sécurités sont communément utilisés par les GRTs pour assurer la sécurité du réseau : les critères de sécurités dits en "N" et en "N-1".

Critère de sécurité en "N"

Dans le cas où le transit sur une ligne dépasse sa limite thermique, on dit alors qu'elle est en *contrainte*, et celle-ci est automatiquement déconnectée, au bout d'un certain temps, par des protections situées aux extrémités de la ligne. Cependant, l'utilisation de ces protections n'est pas sans conséquence. En effet, lorsqu'une ligne est déconnectée, l'énergie qui transitait sur celle-ci est *reportée* sur les autres lignes du réseau. Ces autres lignes peuvent alors elles-mêmes entrer en contrainte et être à leur tour déconnectées. Ce mécanisme de déconnexions successives est aussi connu sous le nom de *surcharges en cascade* et peut conduire à un effondrement de la totalité du réseau électrique aussi connu sous le nom de *black-out* (Haes Alhelou *et al.*, 2019).

Afin de prévenir ce genre d'événement, le premier critère en "N" énonce qu'à tout instant, toutes les lignes (mais également d'autres ouvrages électriques tels que les transformateurs) ne doivent pas dépasser leurs limites thermiques. Le terme "N" signifie ici que toutes les N lignes du réseau prévues pour être en fonctionnement sont bien en service, en dehors de certaines lignes en cours de maintenance ou déconnectées de manière prévue.

Critère de sécurité en "N-1"

Cependant, le critère en "N" à lui seul n'est pas suffisant. Par exemple, à l'échelle du réseau français composé de 100 000 km de lignes, l'éventualité d'un *aléa* tel qu'un coup de foudre s'abattant sur une ligne, ou un vent fort générant un court-circuit par la chute d'un arbre sur une

ligne, n'est pas négligeable. Pour plus de robustesse, un deuxième critère est également introduit : le *critère de sécurité en "N-1"*. Ce critère stipule que "les éléments qui continuent à fonctionner à l'intérieur de la zone de contrôle d'un GRT après la survenue d'un aléa sont capables de faire face à la nouvelle situation sans enfreindre les limites de sécurité d'exploitation."(Commission de Régulation de l'Energie (CRE), 2018). En d'autres termes, le système électrique doit être capable de résister *à tout moment* à une défaillance ou à une panne inattendue d'un de ses composants pour présenter un niveau de fiabilité acceptable. L'évaluation de la fiabilité consiste généralement en une analyse des transits sur un grand nombre de simulations au cours desquelles on déconnecte à tour de rôle chacun des composants en service sur le réseau.

L'évaluation de ce critère nécessite donc de réaliser un grand nombre de calculs de répartition. Ainsi, pour une horodate donnée, un premier calcul de répartition est réalisé pour évaluer le critère en "N", puis encore N simulations supplémentaires sont nécessaires pour un réseau comportant N composants afin d'estimer l'impact de la déconnexion de chacun d'eux. Cela demande par exemple de réaliser environ 14 000 simulations à l'échelle du réseau français à chaque pas de temps et à chaque mise à jour des estimations d'injections. Ce grand nombre de calcul à réaliser peut dès lors être long à calculer et empêcher de réaliser des calculs exhaustifs. Pour autant, et bien que ce critère soit perfectible (par exemple par la prise en compte de la probabilité de l'aléa ou du coût des déconnexions), il permet aujourd'hui une bonne analyse des risques présents sur le réseau. Ainsi, les cas "N-2" qui correspondent à la survenue de deux aléas à un instant donné, ont quant à eux une probabilité d'occurrence plus faible et sont de fait souvent négligeables dans le cadre de la politique de risque de RTE. Ils ne seront pas pris en compte dans cette thèse.

Enjeux

Comme nous l'avons précisé dans la sous-section 1.3.1, deux méthodes sont communément utilisées pour calculer les transits électriques : les calculs de répartition en "AC" et en "DC". La première est lente mais précise tandis que la deuxième est rapide mais moins précise.

Or, avec l'augmentation des moyens de production renouvelables beaucoup moins prévisibles que les moyens de production historiques, il existe un besoin croissant de pouvoir simuler des variantes d'une situation *prévisionnelle* avant qu'elle ne se produise. Il est par exemple nécessaire d'estimer comment se comportera le réseau à un instant donné pour de nombreuses valeurs différentes de productions renouvelables car sa valeur réelle ne sera pas systématiquement connue avant que cet instant ne survienne. Ce nombre de simulations est d'autant plus important dans le cadre des études "N-1" où tous les ouvrages sont successivement déconnectés. Notons par ailleurs que parmi le grand nombre de situations simulées, seul un très petit nombre mènera à des situations qui ne sont pas en "sécurité" et où des contraintes pourraient survenir. Pour ne pas manquer celles qui mettraient le réseau en danger, et afin que les opérateurs puissent agir en *prévention* pour les éviter, il est donc nécessaire de réaliser un nombre de plus en plus grand de simulations.

Pour répondre à ces enjeux, cette thèse a pour objectif de proposer des méthodes permettant d'estimer, au moyen d'indicateurs, l'état du réseau en "N" et, in fine, en "N-1". Ces indicateurs doivent permettre d'inférer rapidement les contraintes les plus importantes du réseau pour réaliser de manière privilégiée les simulations les plus critiques pour la sécurité du réseau.

1.4 Éléments de gestion du réseau

Dans cette partie du chapitre, nous allons mettre en lumière des éléments importants de la gestion du réseau actuel, pour ensuite spécialiser comment sont gérés les critères de sécurité

FIGURE 1.5 – Une salle de conduite ou *dispatching* français. Chaque opérateur, dénommé *dispatcher*, dispose d'un pupitre composé d'un grand nombre d'écrans. Un écran synoptique sur le mur de la salle face pupitre est également partagé entre les opérateurs.

détaillés dans la sous-section 1.3.2 et enfin, présenter quelques pistes envisagées pour la gestion du réseau dans le futur et comment nos travaux pourraient s'y insérer.

1.4.1 Les dispatchers et les salles de conduite

Le réseau électrique est contrôlé dans de grandes salles de contrôle appelées *dispatching* semblables à celle présentée dans la figure 1.5. Par analogie avec les contrôleurs aériens dans les aéroports, le dispatching est la "tour de contrôle" assurant la surveillance du réseau électrique. Il a notamment pour charge la bonne répartition de l'électricité sur le réseau. Chaque salle assure la conduite du réseau sur un périmètre géographique donné, et chaque poste-opérateur du dispatching gère une zone spécifique de ce périmètre. Pour sa part, le réseau électrique français est doté de 7 dispatching régionaux qui se chargent de la conduite des réseaux par régions, ainsi que d'un dispatching national qui gère le réseau d'interconnexion à 400kV et les échanges avec l'étranger. Au total, 200 dispatchers se relayent 24 heures sur 24 et 7 jours sur 7, afin de maintenir le réseau à l'équilibre.

Nous avons présenté dans la sous-section 1.3.2 selon quels critères les opérateurs gèrent les surcharges sur réseau. Comme nous l'illustrons dans la figure 1.6 dans la colonne "intra-day", les opérateurs effectuent au quotidien des tâches variées dont certaines en lien avec la surveillance des transits (en rouge). Cette gestion des transits se fait de manière anticipée (en bleu dans la même figure), en amont du *temps-réel* avec l'aide d'autres équipes dédiées à la planification. Les aléas, mais aussi les événements planifiés tel que les opérations de maintenance du matériel, font donc l'objet de planifications minutieuses et leurs effets sur le réseau peuvent être anticipés de plusieurs jours à plusieurs mois avant qu'ils ne surviennent. Par ailleurs, notons que dans la gestion du réseau, en temps réel, les opérateurs n'ont pas uniquement comme tâche la gestion des transits, mais doivent surveiller d'autres grandeurs telles que la tension et la fréquence. Ces autres grandeurs ne sont pas l'objet direct de cette thèse, mais du fait des liens entre ces différentes

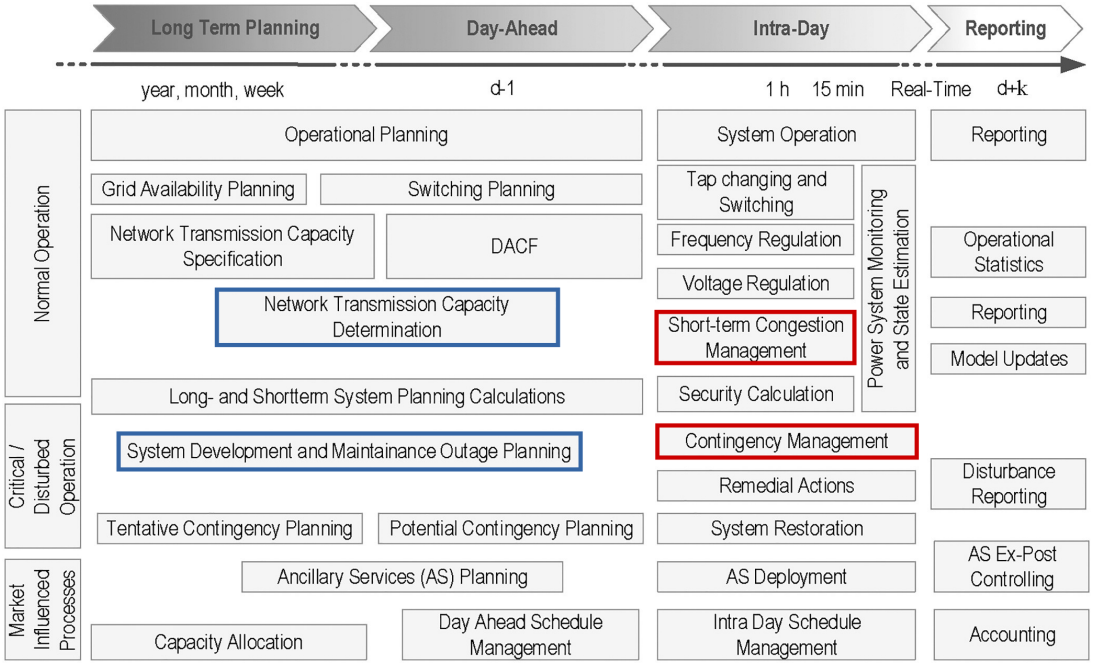


FIGURE 1.6 – Représentation des opérations de planification et de gestion du réseau. En rouge sont encadrées les activités des opérateurs en lien avec cette thèse. Les cadres bleus correspondent aux activités de planification réalisées en amont des activités des opérateurs. Image issue de (Prostejovsky *et al.*, 2019) représentant des résultats du projet GARPUR (Projet GARPUR, 2015).

activités, les travaux engagés dans cette thèse sur le sujet des transits pourraient par la suite être étendus à d'autres tâches, actuelles ou futures, menées par les opérateurs.

Afin de mener à bien chacune des tâches individuelles qui composent la gestion du réseau, de nombreuses applications on été créées pour répondre aux besoins spécifiques de chaque tâche. Ceci a mené au fil des années à la disposition des bureaux hémicirculaires présentés sur la figure 1.5 où une application sur un écran dédié est utile pour représenter un type d'information. Les opérateurs doivent donc faire la synthèse de toutes les informations présentes sur ces différents écrans afin de décider des meilleures actions à prendre sur le réseau. Les actions en questions nommées *leviers* sont principalement de deux types :

1. des changements de *topologie* : ce sont des leviers gratuits et pouvant être mis en oeuvre rapidement, sauf si des études sont nécessaires pour vérifier qu'ils garantissent toujours le respect des critères de sécurité ;
2. des modifications sur les *injections* : ce sont des leviers généralement moins rapides à mettre en oeuvre que les changements de topologie. Il peut s'agir de diminuer la production ou la consommation moyennant une contrepartie financière pour le producteur.

Enjeux

L'étape de synthèse d'information décrite ci-dessus est donc cruciale dans la gestion du réseau, mais peut être aujourd'hui longue à exécuter selon la situation à étudier. Or, les opérateurs travaillent en temps contraint et toutes les prévisions de la survenue d'aléas doivent être gérées.

FIGURE 1.7 – Visualisation de l'outil COMPARE. L'interface comprend une visualisation temporelle d'un indicateur donné, ainsi qu'un tableau de bord représentant un ensemble d'indicateurs à un instant donné.

D'autre part, avec les transformations prévues sur le réseau décrites dans la sous-section 1.2.2, RTE anticipe que les opérateurs devront faire face à un nombre croissant de situations complexes à analyser et sur lesquelles ils devront agir pour prévenir d'éventuels problèmes. Il parait donc nécessaire de pouvoir leur fournir de nouveaux outils permettant de construire des indicateurs fournissant une estimation synthétique de l'état du réseau afin de faciliter l'étape de synthèse d'information.

1.4.2 Indicateurs pour la gestion des transits électriques

Outil COMPARE : une supervision avec des informations agrégées

Des outils opérationnels permettent aux dispatchers de superviser l'évolution du réseau au cours du temps afin d'anticiper l'apparition de contraintes. Nous nous intéressons plus particulièrement dans cette section à l'outil appelé COMPARE à partir duquel les opérateurs peuvent suivre des indicateurs décrivant l'état d'une partie du réseau. Il s'agit d'un outil logiciel développé en interne au sein de RTE dans le langage Java. Mis en service au milieu de l'année 2014 et aujourd'hui en version 3.0, il est aujourd'hui utilisé par environ 200 opérateurs du réseau. Celui-ci est alimenté et actualisé avec des données historiques et mesures issues de capteurs placés sur le réseau. Plus précisément, dans cet outil, les indicateurs utilisés pour les études "N-1" permettent de palier la nécessité de réaliser un grand nombre de simulations, sur un grand nombre de pas de temps en amont du temps réel.

Les indicateurs suivis peuvent correspondre, par exemple, à l'agrégation d'une mesure physique (telle que la puissance active) à l'échelle d'une zone géographique donnée (telle qu'une ville ou agglomération). Ces indicateurs ont été créés par des opérateurs à partir de leur expertise métier combinant des connaissances sur le fonctionnement et la physique du réseau, et décrivent des concepts agrégeant plusieurs grandeurs prises en certains points du réseau. Citons quelques exemples d'indicateurs : la quantité d'électricité produite (ou consommée) dans une zone, l'électricité exportée vers un pays frontalier, ... etc.

Comme décrit dans la figure 1.7, les indicateurs peuvent ensuite être utilisés dans des visualisations temporelles (suivi de la valeur d'un indicateur au cours du temps), spatiales (suivi des indicateurs d'une zone géographique), ou encore sous forme de "tableau de bord" pour un instant donné (plusieurs indicateurs pris au même instant). Des seuils de gravité peuvent également être définis pour certains de ces indicateurs afin de générer des alarmes sonores sur un *poste-opérateur* (vision d'alerte). Les seuils ont pour but d'attirer l'attention des opérateurs sur des tâches à réaliser, seulement lorsque nécessaire, et ainsi leur dégager du temps pour le reste des tâches de conduite du réseau. Des pastilles de couleur (vert/orange/rouge) sont utilisées pour représenter la position des valeurs d'indicateurs par rapports aux seuils de gravité croissante. Ces indicateurs seront décrits plus amplement de manière statistique dans le chapitre 2.

Enjeux

Pour la gestion des transits, les indicateurs de l'outil COMPARE étaient originellement utilisés pour réaliser une analyse préliminaire des situations pouvant générer des contraintes, sans pour autant avoir à réaliser de coûteux calculs de répartitions. Lorsqu'ils génèrent des alertes, ces indicateurs présentent également l'avantage de permettre à l'opérateur d'identifier une origine ou cause potentielle de la contrainte. De plus, ces indicateurs étant des *expressions mathématiques*, ils constituent une information directement interprétable et compréhensible par un opérateur.

Cependant, parmi les 15 000 indicateurs existants, tous ne sont pas uniquement utilisés pour la surveillance des transits. Seuls certains d'entre eux sont directement utilisés pour l'étude "N" et "N-1". Par ailleurs, bien qu'ils donnent une vue globale de l'état du réseau, ces indicateurs peuvent difficilement être mis à jour. En effet, leur actualisation par des experts prend du temps et n'est, par conséquent, pas réalisée fréquemment. Les indicateurs ne sont donc pas exhaustifs et peuvent manquer des informations importantes sur l'évolution du réseau ou encore des liens entre les grandeurs caractéristiques du réseau (injection, tension ou topologie) et les transits.

Dans ce contexte, il semble alors pertinent de construire une nouvelle méthode générant automatiquement des indicateurs qui estimeraient de manière synthétique une ou plusieurs contraintes. Une des premières perspectives d'application de cette méthode serait alors de pouvoir compléter la liste des indicateurs COMPARE avec de nouveaux indicateurs combinant par exemple plusieurs grandeurs physiques. Au vu de l'expertise nécessaire pour construire ces indicateurs, il parait également nécessaire de pouvoir inclure itérativement *la connaissance* des opérateurs dans la création de ces indicateurs, par exemple au moyen d'*algorithmes interactifs*. Enfin, du fait que ces indicateurs puissent porter une information sur la source d'une contrainte, il parait également essentiel que l'approche choisie puisse générer des solutions *interprétables* par des experts. Ces différents besoins, directement issus de problématiques opérationnelles, ont été mis en évidence dès la genèse de la thèse et seront à garder en tête dans la suite du manuscrit vis-à-vis des solutions techniques retenues.

Cette problématique d'estimation synthétique des données du réseau s'insère dans un contexte plus général d'évolution des salles de conduites que nous allons présenter dans la prochaine sous-section.

1.4.3 Perspectives d'évolution des salles de conduites

Dans la sous-section 1.2.2 nous avons précisé que, pour faire face aux challenges à venir, les GRTs ne pourront que marginalement changer la structure de leurs réseaux, mais seront plutôt amenés à repenser leur manière de gérer le réseau.

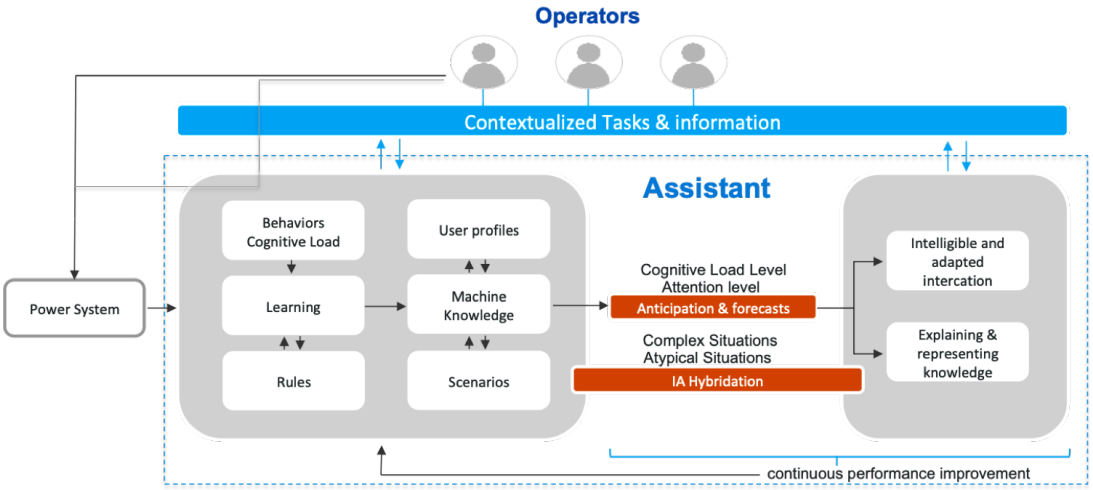


FIGURE 1.8 – Schéma descriptif d'un assistant pour les opérateurs du réseau électrique. Tiré de (Marot *et al.*, 2022b).

Au sein de RTE, la recherche de solutions pour une refonte des salles de conduite du réseau a été lancée par le projet d'"Anticipation du Poste Opérateur pour une Gestion Évoluée du système Électrique" (APOGEE). Ce projet, initié en 2015, a pour objectif de passer d'un pilotage majoritairement manuel où les opérateurs travaillent par *supervision* d'un ensemble de tâches, à un système de navigation par *hypervision* (Marot *et al.*, 2022a). En reprenant l'image du centre de contrôle aérien citée plus haut, l'hypervision permettrait ainsi de mettre l'accent sur l'anticipation via la configuration d'un plan de conduite (tel un plan de vol) et de proposer l'automatisation de certaines actions (tel le pilotage automatique d'un avion) pour que l'opérateur puisse concentrer son activité sur les tâches les plus complexes.

Pour ce faire, un concept a été dégagé : celui d'*assistant* pour les opérateurs du réseau électrique (Marot *et al.*, 2022b). Son fonctionnement est décrit dans la figure 1.8. A l'image d'un assistant personnalisé, celui-ci constituerait une interface unique d'information permettant de fournir à l'opérateur la "bonne" information contextualisée au "bon" moment afin que l'opérateur puisse prendre des décisions adéquates dans les temps impartis. L'assistant serait également capable d'identifier des situations à forte charge mentale pour les opérateurs afin de répartir les tâches au sein d'une équipe, ou bien encore de fournir le niveau d'information adéquat pour l'opérateur en fonction de son profil de connaissances et de son expertise. Enfin, pour être efficace, ce type d'assistant devrait être capable de prendre en compte l'expertise et les commentaires de ses utilisateurs afin de s'améliorer au fil du temps. L'assistant pourra automatiser des tâches routinières, à faible niveau de complexité, et l'épauler dans des tâches qui seront certainement de plus en plus complexes et avec de moins en moins de temps pour les réaliser. L'assistant n'aura cependant jamais vocation à remplacer l'opérateur et ce-dernier prendra toujours la décision finale.

Afin de concrétiser ces objectifs, des récents travaux font état de développements intéressants de l'Intelligence Artificielle (IA) pour les réseaux électriques (Kezunovic *et al.*, 2020; Duchesne *et al.*, 2020). Au sein de RTE, l'IA a été investiguée pour faire des recommandations d'actions à un opérateur grâce à un système expert interprétable (Marot *et al.*, 2018a), pour évaluer des risques (Donnot *et al.*, 2018), segmenter le réseau (Marot *et al.*, 2018b), pour de l'apprentissage par imitation (Donnot *et al.*, 2017), par renforcement (Marot *et al.*, 2020; Kelly *et al.*, 2020), ou par mimisation de l'erreur sur des lois physiques (Donon *et al.*, 2020) en prenant en compte la

structure de graphe du réseau (Donon *et al.*, 2019). Ces différents travaux constituent des pistes prometteuses. Enfin, des travaux sur le plan de l'interactivité, réalisés au début de cette thèse en collaboration au sein de l'équipe Apogée (Marot *et al.*, 2019), s'intéressent particulièrement à la compréhension de la courbe journalière de consommation française et proposent d'intégrer interactivement et itérativement des facteurs conditionnant la consommation (température, jours fériés, ...). L'intégration de ces informations sous forme de conditions dans des auto-encodeurs variationnels et conditionnels (CVAE)(Kingma *et al.*, 2014) a permis d'identifier, grâce à l'information résiduelle, les journées atypiques ne correspondant pas à ces facteurs préalablement connus et, in fine, découvrir de nouveaux facteurs impactant la consommation. Il s'agit là de quelques exemples montrant le potentiel de l'IA et intégrables sous la forme d'un assistant pour les opérateurs du réseau électrique. Ces différents éléments poussent le GRT RTE à explorer l'utilisation de méthodes d'IA, en complément des pratiques et méthodes usitées en entreprise, pour répondre aux différents *enjeux* décrits plus haut dans les sections 1.3.1, 1.3.2, 1.4.1 et 1.4.2 de ce chapitre.

1.5 Problématique

A partir de la description des méthodes de résolution et des critères de sécurité du réseau en section 1.3, de premiers enjeux combinatoires ont émergés, en lien avec la difficulté de réaliser un grand nombre de simulations du réseau électrique.

Par ailleurs, comme présenté dans la section 1.4, un grand nombre d'outils existent déjà pour aider les opérateurs dans leur conduite. Les opérateurs ont donc une tâche significative de synthèse d'information avant de pouvoir réaliser une action sur le réseau. Il existe notamment pour cela l'outil COMPARE permettant de suivre l'état du réseau au moyen d'indicateurs créés par les opérateurs grâce à leur expertise. Cependant, ces indicateurs ne sont pas exhaustifs et ne peuvent être actualisés régulièrement du fait du haut niveau de connaissances requis pour les mettre à jour.

Cette thèse vise à répondre à ces différents enjeux techniques de la manière suivante : afin de construire des *indicateurs* permettant de décrire les transits électriques de manière *interprétable* pour des opérateurs du réseau, nous souhaitons construire une méthode *interactive* capable d'*inclure de la connaissance experte* et de *prendre en compte des retours* de ses utilisateurs afin d'améliorer *itérativement* la pertinence de ces indicateurs.

Les indicateurs ainsi créés pourront, à terme, s'intégrer dans un outil tel que COMPARE. Également, à plus long-terme, les méthodes construites pourraient être intégrées dans un assistant pour les opérateurs du réseau, afin de leur fournir une information synthétique, leur donnant une estimation de l'état du réseau complet ou d'une portion de ce réseau. Dans le cadre d'un assistant, les opérateurs pourraient alors mettre à jour les indicateurs créés en utilisant leur connaissance, au moyen des méthodes développées dans cette thèse.

1.6 Contributions

Pour répondre à la problématique énoncée ci-dessus de construction d'indicateurs, nous proposons une résolution par *Régression Symbolique* (RS). Cette catégorie de méthodes a pour but de fournir une fonction sous forme d'expression symbolique décrivant les relations entre un ensemble d'observations d'entrées et une variable cible à estimer. Dans cette thèse, nous traiterons trois parties complémentaires du problème mentionné ci-dessus.

Nous nous intéressons, dans un premier temps, à savoir sous quelle forme prendre en compte de la connaissance experte qui serait représentable de manière structurée. Cette connaissance inclut par exemple des relations physiques et des principes métiers. Pour ce faire, l'apprentissage d'une fonction symbolique par RS est tout d'abord exploré par Evolution Grammaticale (EG), pour apprendre une fonction symbolique par un algorithme évolutionnaire, où la forme de la fonction est contrainte par des règles grammaticales. Plus précisément, la connaissance "structurée" des opérateurs est décrite sous forme de règles dans une Grammaire Non-Contextuelle (GNC). Dans ce cadre, nous proposons un premier algorithme interactif, noté EGI pour Evolution Grammaticale Interactive. Celui-ci est basé sur l'EG, et propose de mettre à jour une grammaire non-contextuelle, au fil des itérations d'apprentissage évolutionnaire, afin d'améliorer incrémentalement les solutions trouvées. L'utilisation d'une approche d'EGI est évaluée sur des données réelles issues de l'historique du réseau. Cette évaluation comprend notamment une évaluation des métriques d'apprentissage et de l'interprétabilité des solutions créées selon les modalités de (Doshi-Velez et Kim, 2017). Ce travail a été réalisé en collaboration avec un expert du réseau électrique et évalué sur des données réelles du réseau électrique français. L'effet de l'amélioration incrémentale de la grammaire est également évalué et des propositions sont faites pour construire des indicateurs généralisables, i.e. réutilisables d'une année sur l'autre.

Dans un second temps, nous explorons la possibilité d'améliorer l'apprentissage pour une tâche de régression symbolique contrainte par une grammaire. Pour ce faire, nous proposons un algorithme de renforcement pour la régression symbolique nommé RBG2-SR, pour "Reinforcement Based Grammar Guided Symbolic Regression", utilisant un algorithme d'apprentissage par renforcement pour chercher dans un espace délimité par une grammaire non-contextuelle afin de construire une expression symbolique pertinente pour des applications comportant des contraintes physiques. Cette méthode a été validée sur des jeux de données issus de l'état de l'art de la régression symbolique, puis évaluée sur un cas pratique avec jeu de données présentant des contraintes physiques pour réaliser une évaluation de l'interprétabilité.

Dans un dernier temps, l'interactivité entre un utilisateur et un algorithme d'apprentissage automatique est abordée pour guider l'algorithme pendant l'apprentissage. L'interaction est ici réalisée grâce à des connaissances implicites moins structurées, élicitées par les utilisateurs. Deux algorithmes sont étudiés. Un premier algorithme capitalisant sur l'EGI, propose une interface pour permettre à un utilisateur d'interagir avec l'algorithme pendant l'apprentissage. Cette première approche, appliquée à des données réelles, a notamment permis d'identifier des scénarios d'applications de l'interface proposé. Un deuxième algorithme interactif iRBG2-SR est proposé pour apprendre à partir de préférences de l'utilisateur, cette fois-ci dans le cadre d'un apprentissage par renforcement. Une interface d'interaction dédiée est proposée sur des données de la littérature de la régression symbolique ainsi que sur des données simulées du réseau électrique. Une évaluation de l'apprentissage interactif est réalisée par simulation d'interactions à partir de modèles de comportement des utilisateurs. Grâce à l'utilisation d'une approche par renforcement ouvrant sur l'usage de nouveaux algorithmes interactifs, ce travail exploite un second type d'interactivité, réalisé pendant l'apprentissage pour tirer parti de la connaissance exprimée sous la forme de préférence et capable d'englober l'apprentissage génétique et l'apprentissage par renforcement.

1.7 Organisation du manuscrit

Après avoir détaillé dans le chapitre 1 les enjeux de cette thèse, le chapitre 2 décrit une modélisation mathématique du problème étudié. Dans ce chapitre, des notions de physique du

réseau électrique ainsi qu'une présentation des indicateurs existants sont fournies. Ces différents éléments permettent de mettre en évidence les contraintes liées aux enjeux décrits dans ce premier chapitre, pour ensuite définir le problème industriel étudié comme un problème de régression symbolique.

Dans le chapitre 3, nous présentons les concepts fondamentaux manipulés dans le cadre de nos travaux, ainsi que l'état de l'art des thématiques en lien avec ces concepts : la représentation de connaissances, la régression symbolique et l'interactivité.

Les trois chapitres suivants détaillent ensuite les principales contributions de cette thèse. Le chapitre 4 présente l'algorithme d'Evolution Grammaticale Interactive (EGI) permettant d'inclure de la connaissance experte relative aux réseaux électriques dans une grammaire non-contextuelle, afin d'améliorer la construction de variables sous forme d'expressions symboliques pour l'estimation des transits sur les lignes électriques.

Puis, dans le chapitre 5, un algorithme de régression symbolique est présenté, nommé RBG2-SR pour "Reinforcement Based Grammar Guided Symbolic Regression". Celui-ci utilise un algorithme d'apprentissage par renforcement pour chercher dans un espace délimité par une grammaire non-contextuelle.

Le chapitre 6 se concentre sur l'interactivité pendant l'apprentissage. Cette approche est investiguée pour extraire de la connaissance implicite. Dans un premier temps, des travaux mettent un oeuvre un algorithme d'EGI où l'interactivité a également lieu pendant l'évolution. Un algorithme d'apprentissage par renforcement est ensuite proposé, iRBG2-SR, pour apprendre à partir de connaissances implicites, formulées sous la forme de préférences.

Enfin, le chapitre 7 clôt ce manuscrit en synthétisant les apports et conclusions tirés des expériences. Les perspectives sur lesquelles ouvrent ces travaux sont également présentées. Nous y tissons par exemple des liens avec d'autres travaux en cours sur l'étude du réseau électrique et présentons les pistes d'améliorations ainsi que les possibilités d'expériences à venir sur l'interactivité et l'utilisation de données réelles.

1.8 Conclusion du chapitre

Dans le contexte des transformations actuelles du réseau électrique français, ce chapitre introduit les grands enjeux dans lesquels s'insère cette thèse pour améliorer la synthèse d'information restituée aux opérateurs. Ces enjeux ont permis de définir la problématique à laquelle répond cette thèse, à savoir de proposer des algorithmes pour construire des indicateurs décrivant les transits électriques de manière interprétable pour des opérateurs, et où les algorithmes sont capables de prendre itérativement en compte la connaissance des experts. Nous proposons pour cela l'utilisation d'une grammaire pour représenter une partie de la connaissance ainsi que des algorithmes interactifs d'EG et d'AR pour apprendre sur cette structure de connaissances, ainsi que sur de la connaissance implicite.

Chapitre 2

Problématique industrielle

2.1 Introduction

Dans le chapitre précédent, les enjeux industriels dans lesquels s'insère cette thèse ont été mis en évidence. Il a notamment précisé l'enjeu de synthèse d'information et l'objectif de construire des indicateurs permettant de décrire les contraintes électriques de manière *interprétable* par des opérateurs. Ce chapitre présente le contexte industriel de cette thèse et propose une modélisation mathématique du problème industriel étudié. Le chapitre commence par introduire une description de la physique du réseau électrique, ses équations ainsi que les méthodes de résolutions du calcul de répartition en "AC" et en "DC" mentionnées dans le chapitre 1. Cette partie sert à décrire les contraintes physiques que devront respecter les indicateurs créés. Elle permet aussi d'introduire les notations utiles pour les études en "N" et en "N-1".

Dans un second temps, les indicateurs existants issus de l'outil COMPARE sont analysés. Cette analyse sert à extraire les caractéristiques, utiles aux opérateurs, qu'il est nécessaire de prendre en compte dans la modélisation. Enfin, une modélisation du problème industriel est présentée, à partir de ces éléments, ainsi que le choix du modèle finalement adopté, et des exemples de résolution. Ce modèle servira de référence dans les chapitres 4 et 6 pour l'estimation des transits électriques.

La section 2.2 a pour objectif de fournir une compréhension globale du fonctionnement du réseau d'un point de vue électrotechnique, des équations qui le régissent. Les non-linéarités présentes dans les équations détaillées dans ce chapitre sont notamment utiles la définition du problème. En sous-section 2.2.4, les méthodes mises en oeuvres pour l'étude du réseau ne sont pas essentielles à la compréhension du chapitre mais permettent de détailler les contraintes introduites dans le chapitre 1. Un lecteur à la recherche de plus de précisions sur le fonctionnement détaillé, la modélisation et la structure des réseaux électriques modernes pourra considérer la lecture du livre "Power System Stability and Control " de P. Kundur (Kundur, 1994).

2.2 Fonctionnement et équations physiques du réseau électrique

2.2.1 Structure du réseau

Le réseau électrique comporte un ensemble de noeuds notés N, reliés par des lignes L. D'autres composants sont également connectés au réseau, parmi lesquels les transformateurs déphaseurs qui servent à réguler le transit, les condensateurs et les capacités qui sont utilisés pour contrôler les tensions et les lignes HVDC (High Voltage Direct Current). Les composants mentionnés ci-

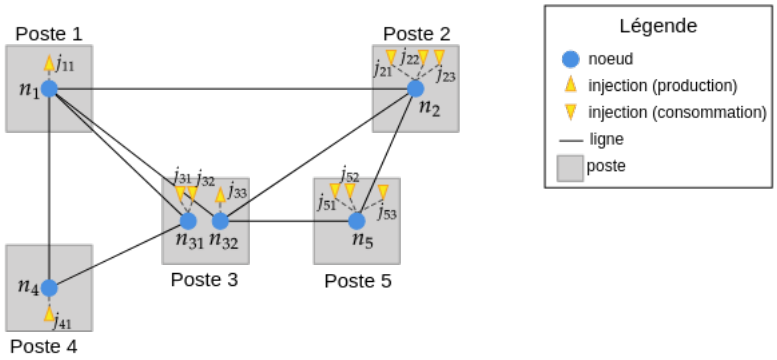


FIGURE 2.1 – Structure d'un réseau à 5 postes et 6 noeuds.

dessus ne sont pas tous explicitement pris en considération dans ce travail mais sont néanmoins cités ici, car ils sont utilisés dans la méthode utilisée pour le calcul de répartition de la sous-section 2.2.3.

Géographiquement, les noeuds N sont localisés dans des postes électriques. Plusieurs noeuds N peuvent être dans un même poste électrique, comme illustré dans la figure 2.1. A des fins de compréhensibilité, nous faisons référence ici à un poste électrique comme ayant un niveau de tension donné. Notons que deux postes ayant un niveau de tension identique seront alors reliés par une ligne de type *ligne*, tandis que deux postes ayant un niveau de tension différents sont connectés par une ligne de type *transformateur*. Les transits au travers des transformateurs seront étudiés de la même manière que les lignes et seront alors inclus dans l'ensemble L. Nous notons par la suite J l'ensemble des *injections* (productions et consommations).

Remarque sur l'évolution des infrastructures du réseau

Le matériel qui compose le réseau ne varie que légèrement au cours du temps, par exemple, lorsque de nouvelles lignes et de nouveaux postes sont construits. Sur la période de temps où nous étudions le réseau, peu de composants étant ajoutés au réseau sur la zone étudiée, nous pouvons considérer les ensembles N et L comme constants. Dans le cas des lignes ou noeuds ajoutés ou supprimés sur la période de temps de l'historique considéré, nous les intégrons dès le premier pas de temps dans les ensembles N et L, en les considérant comme déconnectés sur ces plages de temps où elles n'existaient pas encore. L est également de taille fixe.

2.2.2 La physique du réseau

Cette partie décrit les modèles, les unités et les équations physiques caractérisant le réseau électrique, puis détaille les outils de résolution permettant de déduire les grandeurs non directement observables du réseau à partir de celles connues, ainsi que leurs limites.

Modèle de lignes et grandeurs physiques du réseau

Le réseau électrique est un type de *circuit* électrique dans lequel des productions génèrent un signal électrique qui peut alimenter diverses consommations. Les grandeurs physiques principales de ce signal sont : l'intensité i qui quantifie le débit d'électricité circulant dans un circuit et se mesure en Ampère (A) ; et la tension v liée à la différence d'état entre deux points du circuit et se mesure en Volt (V). Ces signaux sont *alternatifs* et l'on parle de courant alternatif (noté AC

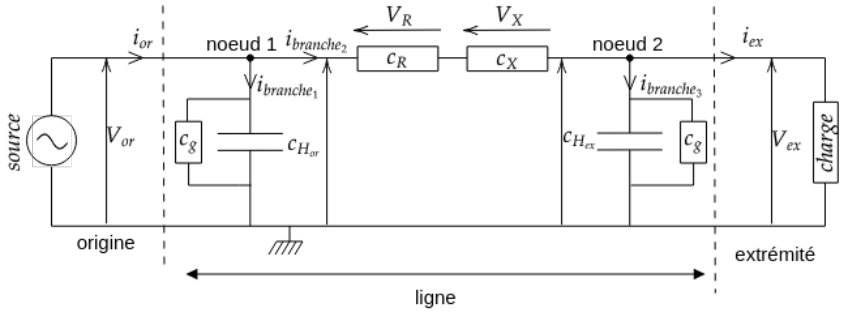


FIGURE 2.2 – Modèle en Pi d'une ligne électrique pour un circuit unifilaire entre une source et une charge.

ou *Alternative Current*) pour désigner un courant qui circule alternativement dans une direction du circuit, puis dans l'autre, à une fréquence de 50 Hz.

Le réseau, ainsi décrit, alimente une majorité des consommateurs avec un *circuit triphasé*. Il s'agit d'un système composé de trois signaux de même norme en notation complexe, mais décalées l'un par rapport à l'autre de 120° lorsqu'ils sont bien équilibrés. Par rapport à un système monophasé, l'usage du triphasé présente l'avantage de transporter l'électricité avec moins de pertes sur les lignes, tout en économisant du câble. Les lignes triphasées en régime équilibré sont généralement modélisées en électrotechnique par le schéma monophasé équivalent représenté en figure 2.2 appelé "modèle en Pi". Il s'agit d'un modèle valide pour décrire le transport d'électricité sur de moyennes distances, entre environ 80 et 250 km. Le modèle en Pi décrit une ligne électrique par plusieurs composants aussi appelés dipôles (résistances, condensateurs, inductances) possédant des caractéristiques électrotechniques notées c. En notation complexe, ces caractéristiques de lignes sont c_R sa résistance, c_X sa réactance, $c_{H_{or,ex}}$ ses demi-susceptances à l'origine et l'extrémité et c_g ses demi-conductances latérales. En notation complexe, la relation entre l'impédance, la résistance et la réactance est donnée par $\underline{c_Z} = c_R + jc_X$, avec j la racine carrée de -1. Elle permet également de définir l'admittance par $\underline{c_y} = \frac{1}{c_Z} = c_g + jc_b$ où c_b est nommée susceptance.

Les lois du réseau

Des lois de la physique relient les différentes grandeurs du réseau. Tout d'abord, la *loi d'Ohm* caractérise la tension aux bornes d'un dipôle et est proportionnelle au courant qui la traverse $\underline{v} = \underline{c_z} \cdot \underline{i}$. Les lois de Kirchhoff permettent ensuite de définir les relations entre les grandeurs de tension et intensité orientées du circuit.

— La première loi de Kirchhoff ou *loi des noeuds*, stipule que la somme algébrique des courants en un noeud du circuit est nulle. Par exemple, sur la figure 2.2, au noeud 1 $i_{or} = i_{branche_1} + i_{branche_2}$ et au noeud 2 $i_{branche_2} + i_{branche_3} = i_{ex}$. Par ailleurs les dipôles de caractéristiques c_R et c_X sont traversés par le même courant $i_{branche_2}$.

— D'après la deuxième loi de Kirchhoff, ou *loi des mailles* la somme algébrique des tensions le long d'une maille est toujours nulle, où une maille est définie comme un ensemble de branches d'un circuit qui forme une boucle. Ainsi, dans la figure 2.2 cette loi permet d'écrire notamment $V_{or} = V_R + V_X + V_{ex}$.

Les unités de puissance

Les unités de puissance peuvent être déclinées à partir de celles décrites ci-dessus. Tout d'abord, la puissance apparente d'un dipôle donné se définit par la formule, exprimée en Volt-Ampère (VA), ici en notation complexe :

$$\underline{s_{dipole}} = \underline{v_{dipole}} \cdot \underline{i_{dipole}}^* \tag{2.1}$$

Elle est notamment utile au dimensionnement des installations. Deux autres puissances sont reliées à celle-ci :

— La *puissance active* p exprimée en Watt (W) est la partie réelle de la puissance complexe. Elle est souvent décrite comme la puissance "utile" et correspond à la formule :
$p = v \times i \times \cos(\phi)$ avec ϕ le déphasage entre tension et intensité. L'opération de multiplication $\times$ est fréquemment omis dans la notation, et utilisé de façon implicite.

— La *puissance réactive* q, d'unité le Volt-Ampère réactif (VAr) est la partie imaginaire de la puissance complexe et correspond à la formule : $q = v \times i \times \sin(\phi)$. Bien que sa signification physique soit difficile à appréhender, elle a cependant un rôle clé dans le système et est générée par les éléments du réseau ayant une partie inductive ou capacitive. Cette partie réactive est modélisée par exemple par c_x dans le modèle en Pi des lignes électriques.

Les trois puissances sont reliées par la relation

$$s = \sqrt{p^2 + q^2}. \tag{2.2}$$

La modélisation présentée ci-dessus permet de décrire le fonctionnement d'une ligne électrique d'un point de vue électrotechnique. Elle présente les unités et lois physiques de l'électricité. Le réseau électrique pour sa part est composé d'un *grand nombre* de lignes électriques connectées entre elles pour former un réseau maillé. A partir de ces définitions, nous allons détailler comment décrire physiquement le réseau à un instant donné à partir de mesures et de calculs.

2.2.3 Les équations du réseau

Afin de connaître régulièrement les différentes grandeurs du réseau au cours du temps, et plus particulièrement les valeurs de ces grandeurs en régime permanent, il est nécessaire d'en construire un point de fonctionnement, calculé à partir des grandeurs connues au niveau des *injections*. Par opposition au régime transitoire, qui s'intéresse à l'état d'un système sur de courtes périodes de temps, un régime permanent est défini comme le régime d'un système stable observable après un certain temps, lorsque le régime transitoire s'est éteint. Ce problème est connu sous le nom de *calcul de répartition* ou *power flow*.

Données d'entrée

Pour l'étude d'un instant donné, à l'échelle de chaque noeud électrique, les injections en puissance active sont connues ainsi qu'une seconde grandeur : dans certains noeuds appelés "noeuds PQ", il s'agit de la puissance réactive tandis que dans les autres noeuds dits "PV" il s'agit de la tension. A ce même instant on considère également la topologie connue de manière exacte. Pour chaque noeud du réseau $n \in \{1, ...|N|\}$ avec N l'ensemble des noeuds, nous notons Θ_n la phase de la tension et $|v|$ le module de la tension au noeud. De la même manière, p_n correspond à la puissance active et q_n à la puissance réactive.

Caractéristiques des ouvrages

Chaque objet du réseau qui relie deux noeuds, comporte des *propriétés physiques*, modélisées par un nombre complexe : l'admittance. Entre deux noeuds $(n1, n2) \in N^2$, nous re-notons l'admittance d'une ligne $c_y^{n1,n2}$. Pour le réseau complet, l'admittance peut être représentée sous forme matricielle de taille $|N| \times |N|$:

$$c_Y = \begin{bmatrix} -\sum_{n \neq 1} c_y^{1,n} & c_y^{2,1} & ... & c_y^{1,N} \\ c_y^{1,2} & -\sum_{n \neq 2} c_y^{2,n} & ... & c_y^{2,N} \\ c_y^{1,N} & ... & c_y^{N-1,N} & -\sum_{n \neq N} c_y^{N,n} \end{bmatrix} \tag{2.3}$$

Les équations

Disposant des informations d'entrée du problème et des caractéristiques des objets présents sur le réseau, il est dès lors possible de chercher à calculer les informations manquantes : les puissances réactives et tensions (module et phase) aux noeuds où elles ne sont pas connues, ainsi que les transits sur les lignes. Puissances et tensions, en chaque noeud, sont reliées, dans le cas sans pertes, par le système d'équations 2.4 non-linéaires suivant (Kundur, 1994), avec $n \in N$:

$$\begin{array}{rcrcl} 0 & = & -p_n & + & \sum_{k=1}^{N} |v_n| \times |v_k| \times (c_g^{n,k} \times cos(\Theta_n - \Theta_k) + c_b^{n,k} \times sin(\Theta_n - \Theta_k)) \\ 0 & = & q_n & + & \sum_{k=1}^{N} |v_n| \times |v_k| \times (c_g^{n,k} \times sin(\Theta_n - \Theta_k) + c_b^{n,k} \times cos(\Theta_n - \Theta_k)) \end{array} \tag{2.4}$$

Ce système d'équations suppose une fréquence fixe en tension ainsi qu'un régime quasi-stationnaire équilibré. Grâce à ces équations et connaissant les grandeurs deux noeuds "PQ" et "PV", il est alors possible d'estimer les grandeurs inconnues en tous les noeuds ainsi qu'aux extrémités des lignes. Ainsi pour deux noeuds n_1 et n_2, les puissances active $p_{n_1 \to n_2}$, réactive $q_{n_1 \to n_2}$ ainsi que le transit $i_{n_1 \to n_2}$ circulant d'un noeud n_1 vers n_2 sont (Kundur, 1994) :

$$\begin{array}{rcrcl} p_{n_1 \to n_2} & = & & |v_{n_1}| \times |v_{n_2}| \times c_b^{n_1,n_2} \times sin(\Theta_{n_1} - \Theta_{n_2})) \\ q_{n_1 \to n_2} & = & - & |v_{n_1}| \times |v_{n_2}| \times c_b^{n_1,n_2} \times cos(\Theta_{n_1} - \Theta_{n_2})) \\ i_{n_1 \to n_2} & = & & \frac{\sqrt{p_{n_1 \to n_2}^2 + q_{n_1 \to n_2}^2}}{\sqrt{3}|v_{n_1}} \end{array} \tag{2.5}$$

Après voir donné les équations régissant le réseau électrique, détaillons dans les prochains paragraphes les méthodes de résolution en "AC" et en "DC" de ces équations.

2.2.4 Méthodes de résolution

Méthode de résolution en "AC"

Pour obtenir des valeurs de Θ_n et $|v_n|$ sur chaque noeud N du réseau ainsi que le transit i_l sur chaque ligne L, une méthode de résolution en "AC" est utilisée. Cette méthode de calcul de répartition est basée sur l'algorithme de Newton-Raphson (Sereeter *et al.*, 2019), noté NRPF, dont une version simplifiée est donnée dans l'Algorithme 1.

En plus des grandeurs des noeuds PQ et PV, le calcul de répartition suppose que le module de la tension $|v|$ et la phase de la tension Θ sont connus sur un noeud particulier appelé *noeud bilan*. La méthode de résolution décrite dans l'Algorithme 1 déroule les étapes suivantes : après avoir choisi des valeurs initiales ("suffisamment proches" de la solution finale), un processus itératif est répété plusieurs fois, soit jusqu'à convergence de l'algorithme ou jusqu'à atteindre un nombre

Algorithm 1 Méthode de résolution du calcul de répartition, basée sur l'algorithme de Newton-Raphson. Description simplifiée inspirée de (Donnot, 2019)

function NRPF SIMPLIFIÉ(c_Y, $Noeud\_bilan_{ID}$, $p_n \forall n \in N$, $q_n \forall n \in \{Noeud_{(P,Q)}\}$, $|v|_n \forall n \in \{Noeud_{(P,V)}\}$)

$v_n \leftarrow 104\% v_{nominal}$ et $\Theta_n \leftarrow 0$ ▷ Initialisation

while critère d'arrêt non-atteint **do** ▷ Boucle principale

$\Delta p_n = -p_n + \sum_{k=1}^{m} |v_n| \times |v_k| \times (c_g^{n,k} \times \cos(\Theta_n - \Theta_k) + c_b^{n,k} \times \sin(\Theta_n - \Theta_k))$

$\Delta q_n = -q_n + \sum_{k=1}^{m} |v_n| \times |v_k| \times (c_g^{n,k} \times \sin(\Theta j - \Theta k) - c_b^{n,k} \times \cos(\Theta_n - \Theta_k))$

$J = \begin{bmatrix} \frac{\partial \Delta p}{\partial \Delta \Theta} & \frac{\partial \Delta p}{\partial \Delta |v|} \\ \frac{\partial \Delta q}{\partial \Delta \Theta} & \frac{\partial \Delta q}{\partial \Delta |v|} \end{bmatrix}$ ▷ Calcul de la matrice Jacobienne J

$J^{-1} = \text{Inverse}(J)$ ▷ Inversion de J pour obtenir J^{-1}

$\begin{bmatrix} \Delta \Theta \\ \Delta |v| \end{bmatrix} = J^{-1} \begin{bmatrix} \Delta p \\ \Delta q \end{bmatrix}$ ▷ Calcul de l'écart pour $|v|$ et Θ

$|v_n| = |v_n| + \Delta |v_n|$ ▷ Mise à jour avec les nouvelles valeurs

$\Theta_n = \Theta_n + \Delta \Theta_n$

return $|v_n|$ et Θ_n

maximal d'itérations. Une fois les $|v|$ et Θ connus en tous les noeuds, les puissances actives, réactives et les transits sont calculés à partir de l'équation 2.5.

Notons que des situations de non-convergence de cet algorithme peuvent se produire, par exemple, lorsque l'initialisation choisie est trop loin de la solution à trouver. Un autre cas de non-convergence peut être lié aux données. Plus précisément, dans le cas de l'étude du réseau réel avec l'utilisation de données prévisionnelles, les valeurs prévues sur le réseau, utilisées en entrées du calcul de répartition, peuvent présenter une qualité insuffisante pour permettre la résolution du calcul de répartition.

Cette méthode est notamment celle utilisée dans cette thèse pour la génération des données du chapitre 4. Les données y sont générées avec le solveur propriétaire de RTE, nommé Hades2, basé sur cet algorithme. Les données utilisées par la suite dans les chapitres 5 et 6 utilisent la librairie Python Lightsim2grid (Donnot, 2020b), qui implémente également cet algorithme sur d'autres réseaux.

Méthode de résolution par approximation en "DC"

L'approximation en "DC" ou "direct current" (Hahn, 1931) est une approximation communément utilisée sur le réseau électrique (Stott *et al.*, 2009). Il s'agit d'une linéarisation du problème en AC qui suppose une tenue parfaite de la tension, et néglige l'effet de la puissance réactive ainsi que des pertes de transmission. Bien que cette méthode présente des défauts du fait des simplifications qu'elle implique, elle présente aussi l'avantage d'être rapide à calculer et de toujours fournir une solution car elle n'est pas itérative. Cette méthode repose sur les trois simplifications suivantes :

1. la résistance c_r d'une ligne est significativement plus petite que sa réactance c_x : $c_r << c_x$
2. la différence de phase $\Theta_1 - \Theta_2$ entre deux noeuds connectés par une ligne est faible : $\Theta_1 - \Theta_2 \approx 0$
3. l'amplitude de la tension $|v|$ est égale à une valeur nominale $|v|_{nominale}$.

Ces simplifications transforment le système d'équations 2.4 en :

$$0 = -p_n + \sum_{k=1,k\neq n}^{m} c_b^{n,k}(\Theta_n - \Theta_k) \tag{2.6}$$

$$0 = q_n + \sum_{k=1}^{m} c_b^{n,k} \tag{2.7}$$

L'équation 2.6 peut alors s'exprimer sous forme matricielle, ensuite soluble, par exemple, par inversion de matrice :

$$\begin{pmatrix} p_1 \\ ... \\ p_m \end{pmatrix} = c_Y \begin{pmatrix} \Theta_1 \\ ... \\ \Theta_m \end{pmatrix} \tag{2.8}$$

Comme précédemment une fois les valeurs de $|v|$ et Θ connues, les transits sont calculés à partir des équations 2.5. En conclusion, notons que le modèle "DC" néglige l'effet de la puissance réactive, ainsi que les variations de tension dans la prédiction du transit électrique. Ainsi, les modèles construits dans cette thèse pourront, en première approximation, se concentrer sur les variables correspondant aux puissances actives pour construire indicateurs décrivant le transit.

Les deux méthodes de résolution présentées ci-dessus, en plus de décrire comment ont été obtenues les données réelles utilisées dans les chapitres suivants, ont aussi permis d'extraire des éléments caractéristiques du problème étudié, à savoir :

— la forme des équations reliant les différentes grandeurs et variables du réseau ;

— l'ordre d'importance des grandeurs dans l'étude du réseau électrique et notamment l'importance de la puissance active.

2.3 Formalisme du problème

2.3.1 Notations

Avec l'application des méthodes de calcul de répartition, nous disposons des grandeurs de puissances actives et réactives en chaque injection $p_J = (p_j)_{j\in J}$, $q_J = (q_j)_{j\in J}$, les modules et phases des tensions en chaque noeud $|v|_N = (|v_n|)_{n\in N}$, $\Theta_N = (\Theta_n)_{n\in N}$. Par abus de notation, le module de la tension sera désormais noté $v_n = |v_n|$.

Les informations de connexion des lignes notées *topologies* τ et les intensités i sont également connus. Nous notons la topologie relative à une ligne, i.e. sa connexion à l'origine et l'extrémité de la ligne, $\tau_l = (connexion_{l_{or}}, connexion_{l_{ex}})$ et ses transits à l'origine et l'extrémité $i_l = (i_{l_{or}}, i_{l_{ex}})$ avec $l \in L$. Nous négligeons pour l'instant les pertes électriques par effet Joule sur la ligne, de sorte que $i_{l_{or}} \approx i_{l_{ex}}$. Nous notons cette grandeur i_l.

2.3.2 Études du réseau en "N"

Revenons maintenant aux études en "N" et en "N-1" introduites dans le chapitre 1. Pour rappel, l'étude du réseau en "N" consiste à analyser les "N" lignes du réseau électrique actuellement en service afin de vérifier, pour chacune, que leur transit $I_{\text{"N"}} = (i_l)_{l\in L}$ ne dépasse pas le ou les seuils de transit i_l^{seuil}, $l \in L$. Le critère de sécurité en "N" consiste donc à assurer que $i_l < i_l^{seuil}$, $\forall l \in L$. Selon la période de l'année (été / hiver / inter-saison) un ou plusieurs seuils peuvent être définis :

$$seuil \in \{seuils\}_{ete} \cup \{seuils\}_{inter\_saison} \cup \{seuils\}_{hiver}$$

Du fait des changements de seuils i_l^{seuil} au cours de l'année, l'origine d'un transit élevé pourra aussi différer. Nous considérons comme sources d'une variation de transit : les puissances au niveau des injections p_j et $q_j, j \in J$, la topologie τ, et dans une moindre mesure le module de la tension v et sa phase Θ. Dans le cas de l'étude d'une zone géographique et non d'un réseau complet, nous prenons également en compte les transits entrants $i_{frontiere}$ afin de travailler sur un système fermé.

Pour ce type d'études, nous cherchons une ou plusieurs variables (i.e. indicateurs) décrivant les relations entre transit en "N" notés $I_{\text{"N"}} = (i_l)_{l \in L}$ et les mesures (p, q, τ, v, Θ) de manière approximée, cette variable permettra d'expliquer et estimer l'état du réseau en "N". Afin de construire les approximations souhaitées, nous apprenons ces indicateurs sur des données appartenant à l'historique du réseau. Cette première catégorie d'études constitue un "bac à sable" dans lequel construire les méthodes qui seront ensuite utilisées pour les études en "N-1".

2.3.3 Extension aux études du réseau en "N-1"

Notons $\tau_{\neg ld}$ la topologie obtenue après déconnexion de la ligne $ld \in L$ en partant de la topologie initiale τ. Pour l'étude du réseau en "N-1", après déconnexion d'une ligne $l_d \in L$ choisie, de nouvelles tensions, phases et transits sont calculés à partir de $(J, \tau_{\neg ld})$ $\forall ld \in L$. L'étude en "N-1" vérifie ici que le transit après déconnexion d'une ligne $ld \in L$, noté $i_{l|\neg ld}$, reste toujours en dessous du seuil autorisé : $i_{l|\neg ld} < i_l^{seuil}, \forall t$. Pour faciliter l'étude en "N-1", nous cherchons à modéliser les relations entre les données (p, q, τ, v, Θ, $I_{\text{"N"}}$) en "N", et transits en "N-1" de manière synthétique.

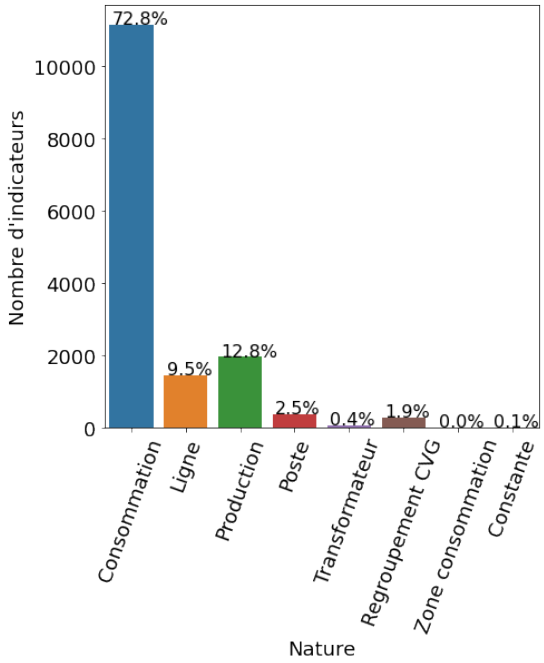
2.4 Description des indicateurs

2.4.1 Analyse des indicateurs existants

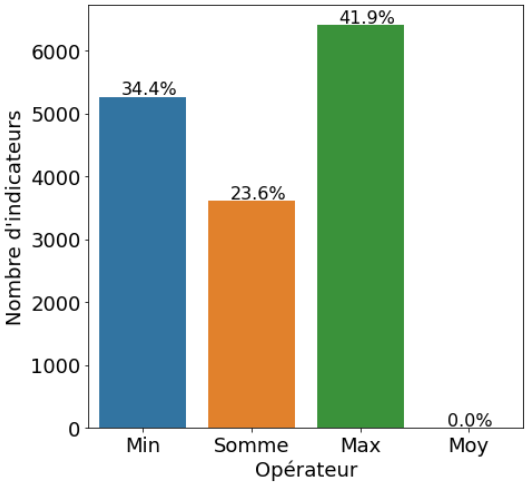
Les sous-sections 2.2.2 et 2.2.3 ont permis de décrire les équations physiques régissant le réseau électrique. Puis, la sous-section 2.2.4 a introduit des simplifications faites sur ces équations pour linéariser le problème en "AC". Cette linéarisation met notamment en évidence dans l'équation 2.6 l'importance des variables de puissance active et de phase.

Cependant, en pratique, les opérateurs ne s'appuient pas sur les valeurs de phase Θ pour analyser le réseau. Afin de trouver des solutions respectant les contraintes physiques, tout en s'appuyant des formules d'indicateurs déjà utilisées par les opérateurs, nous avons analysé ceux utilisés dans l'outil COMPARE. Pour cela, nous avons extrait les indicateurs présents dans l'outil en date d'octobre 2019 et en proposons d'identifier les principales variables et principales formules des indicateurs.

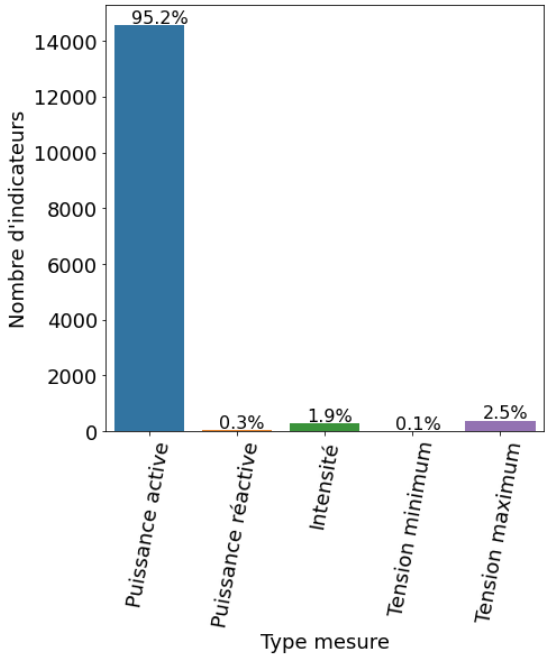
La figure 2.3 présente les statistiques que nous avons obtenus sur les 15 300 indicateurs présents dans l'outil au moment de cette étude. Ces statistiques portent sur la nature des objets présents dans les indicateurs, les opérations mathématiques utilisées pour les construire, les types de mesures physiques et les types de formules qui les constituent. D'après la figure 2.3a, les grandeurs étudiées sont principalement relatives à la *consommation* pour $72,8\%$ des indicateurs et relatives à la production pour $12,8\%$. $9,5\%$ des indicateurs créés présentent l'information de "lignes". Les indicateurs "lignes" peuvent, par exemple, être des indicateurs utiles dans l'étude "N-1" pour quantifier le *report de charge*, i.e. la quantité d'électricité qui se reporte sur une ligne en particulier lorsqu'une autre ligne spécifique se déconnecte. Il s'agit, par ailleurs, comme le montre la figure 2.3c, majoritairement de grandeurs de *puissances actives*. En outre, d'après la



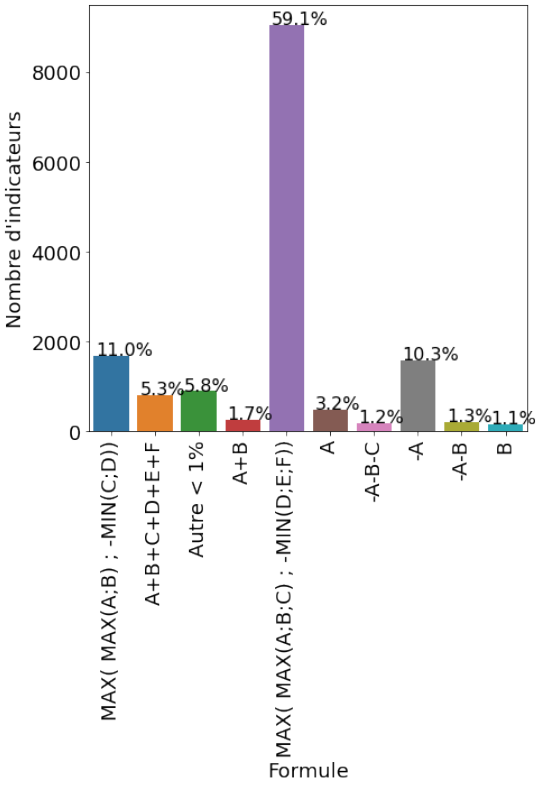
(a) Nature des objets présents.



(b) Opérations des indicateurs.



(c) Type de grandeur utilisée.



(d) Type de formule utilisée. Les lettres de A à F représentent des variables physiques prises parmi les variables de grandeurs (p, q, v, I).

FIGURE 2.3 – Description statistique des indicateurs COMPARE par : (a) nature des objets, (b) type d'opérations (c) type de grandeur et (d) type de formule utilisés dans les indicateurs.

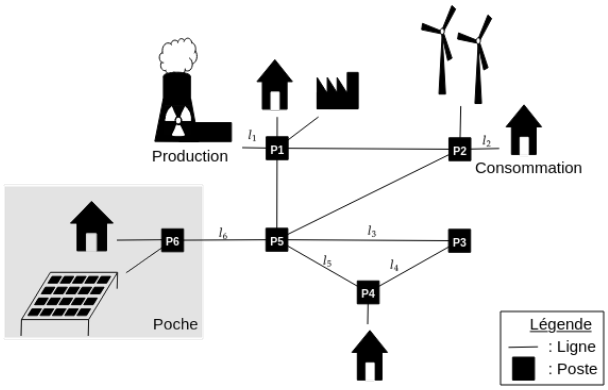


FIGURE 2.4 – Réseau utilisé pour illustrer les indicateurs.

figure 2.3b ces indicateurs agrègent des grandeurs au moyen d'opérations réparties entre *maximum* (41,9%), *minimum* (34,4%) et *somme* (23,6%) d'une liste de grandeurs. Enfin, comme présenté dans la figure 2.3d, les indicateurs sont majoritairement représentés, à 70%, par une formule comportant une fonction non-linéaire *max* et *min* (59,1% et 11% avec respectivement 6 et 4 variables).

Les ordres de grandeur de ces statistiques sont importants pour identifier les opérations fonctionnelles à utiliser lors de la création d'indicateurs, par les méthodes proposées et développées dans ce manuscrit.

2.4.2 Exemples d'indicateurs possibles

Nous avons vu ci-dessus les formules utilisées aujourd'hui dans les indicateurs de l'outil COMPARE. Illustrons maintenant, sur quelques exemples d'un réseau simplifié, des indicateurs qui pourraient être créés dans le futur pour l'estimation des transits. La figure 2.4 illustre sur un réseau simple des exemples d'indicateurs. Ce réseau comprend 6 postes, où chaque poste est relié par au moins une ligne au réseau. Le poste P4 représente une zone "isolée" du réseau, appelée *poche*, qui n'est relié aux autres poste que par une ligne.

Sur la ligne l_1 reliée au poste $P1$ qui évacue une production, le transit pourrait être estimé par :

$$i_{l1} = \frac{\sqrt{p_{P1}^2 + q_{P1}^2}}{v_{P1}}$$

De la même manière pour le transit sur une ligne l_2 reliant une consommation au réseau par le poste P2 :

$$i_{l2} = \frac{\sqrt{p_{P2}^2 + q_{P2}^2}}{v_{P2}}$$

Regardons maintenant le cas des lignes l_3, l_4 et l_5. Pour estimer le transit sur la ligne l_4, une condition de topologie pourrait être introduite sur la connexion de la ligne l_5 :

$$i_{l_4} \approx \text{if}\{\neg\ connexion_{l_5}\}\,\text{then}\{\text{coeff}\ \ p_{P_4}\}\,\text{else}\{i_{l_3}\}, \text{coeff} \in [0,1]$$

Enfin, pour le cas de la ligne l_6, celle-ci relie deux zones du réseau uniquement connectées par la ligne en question. Le transit i_{l6} peut alors être approché par la différence entre les puissances actives des deux zones : $i_{l6} \approx \sum_{poste \in poche} p_{poste} - \sum_{poste \notin poche} p_{poste}$.

Ces exemples permettent de mettre en évidence quelles fonctions non-linéaires pourraient être incorporées, tout en respectant les équations et relations détaillées en sous-section 2.2.2. Grâce aux éléments de physiques introduits dans la section 2.2, la prochaine section détaille la modélisation du problème de *construction d'indicateurs interprétables* introduit dans le chapitre 1.

2.5 Modélisation mathématique du problème

2.5.1 Choix du problème à modéliser

Après avoir décrit comment collecter des observations du réseau pour un pas de temps donné, nous étudions maintenant le réseau de manière statique sur une fenêtre temporelle fixe de taille M. Les données collectées pour l'estimation des transits électriques sont étudiées sans dépendance temporelle. En effet, les valeurs des transits électriques à un instant donné sont calculées à partir des grandeurs uniquement mesurées au même pas de temps, tel que décrit dans la sous-section 2.2.3. Pour chaque pas de temps m, les différentes grandeurs sont compilées, indépendamment du fait qu'elles soient définies aux noeuds ou aux lignes du réseau. Cette concaténation est faite dans un premier temps, sans tenir compte des unités physiques des variables observées. Nous obtenons alors un jeu de données de taille M, avec un ensemble d'observations du réseau par pas de temps sur D variables. Soit $X \in \mathbb{R}^{D \times M}$ un jeu de données comportant M observations décrites par D variables. X se décompose selon différentes catégories de variables : $X = (X_p, X_q, X_\tau, X_v, X_\Theta)$ avec respectivement :

- $X_p = (X_{p_j})_{j \in J}$ les variables de puissance active,
- $X_q = (X_{q_j})_{j \in J}$ les variables de puissance réactive,
- $X_\tau = (X_{\tau_l})_{l \in L}$ les variables de topologie,
- $X_v = (X_v)_{n \in N}$ les variables de module de la tension,
- $X_\Theta = (X_\Theta)_{n \in N}$ les variables de phase de la tension.

D'après l'analyse des indicateurs réalisée en section 2.4, nous pourrons dans un premier temps omettre X_q et X_Θ.

Soit $Y = Y_i = (y_l)_{l \in L}$ un ensemble de variables à estimer et $y_l^m \in \mathbb{R}$ l'observation $m \in M$ d'une variable de transit sur une ligne l. Les variables Y peuvent prendre plusieurs formes, décrites dans la tableau 2.1. Cette description et les différentes formulations possibles du problème ont été développées de manière itérative, au cours de la thèse, avec des experts du réseau. Dans le cas en "N", trois types de formulations du problème sont possibles : deux par régression et une par classification. Le premier cas vise à estimer directement le transit en N sur une ligne l, le deuxième cherche à estimer la distance, ou "marge", par rapport à un seuil de transit donné ; et enfin, le dernier a pour objectif de classifier les situations en sécurité ou non par rapport à un seuil de transit donné. Ce dernier cas de classification des situations, en contraintes ou non, correspond à la grandeur directement utile pour *déclencher des alertes* et avertir les opérateurs. Sachant que les seuils de transits peuvent varier au cours de l'année, nous privilégions l'estimation de $y_l = i_l^{\text{"N"}}$. L'estimation des deux variables suivantes peut être déduite de l'estimation de la première.

Les 3 premiers points en "N", dans le tableau 2.1, correspondent à un cas "bac à sable". L'application finale des méthodes développées dans cette thèse est l'étude de situations en "N-1". Les cas en "N" ont principalement vocation à construire et tester les méthodes permettant de construire les estimations en "N-1". Ils fournissent également des estimations de références

Type d'étude	Variables cibles $\forall l \in L, \forall ld \in L$	Type de Problème	Description
	$\boldsymbol{y_l = i_l^{\text{"N"}}}$	**Régression**	**Estimation du transit sur une ligne l**
"N"	$y_l = i_l^{\text{"N"}} - i_l^{seuil}$	Régression	Transit estimé par rapport à un seuil nominal, i.e. conditionnée par un seuil i_{seuil}. Problème proche du concept de "marge" par rapport à une capacité sur une ligne.
	$y_l = i_l^{\text{"N"}} > i_l^{seuil}$	Classification	Classification du dépassement d'un seuil, séparant les situations en sécurité ou non
"N-1"	$y_{l\|\neg ld} = i_{l\|\neg ld}^{\text{"N-1"}}$	Régression	Estimation du transit sur la ligne l après déconnexion de la ligne ld
	$y_{l\|\neg ld} = i_{l\|\neg ld}^{\text{"N-1"}} - i_l^{\text{"N"}}$	Régression	Différence de transit sur la ligne l avant/après déconnexion de la ligne ld
	$y_{l\|\neg ld} = i_{l\|\neg ld}^{\text{"N-1"}} - i_l^{seuil}$	Régression	Estimation du transit sur la ligne l après déconnexion de la ligne ld vis-à-vis d'un seuil de transit donné
	$y_{l\|\neg ld} = i_{l\|\neg ld}^{\text{"N-1"}} > i_l^{seuil}$	Classification	Classification du dépassement d'un seuil après déconnexion de la ligne ld

TABLE 2.1 – Description des différents problèmes d'estimation du transit, en "N" et en "N-1". Cette thèse s'intéresse au cas $y_l = i_l^{\text{"N"}}$ (en gras), les autres cas en "N" peuvent être déduits de cette formulation. En "N" comme en "N-1", les variables cibles peuvent être estimées par régression ou classification selon la modélisation.

par rapport auxquelles les transits "N-1" pourront être comparés (par exemple dans le deuxième cas en "N-1 du tableau 2.1). En "N-1", quatre types de formulations peuvent être données, avec cette fois-ci $Y = (y_{l|\neg ld})_{(l,ld)\in L^2}$: trois formulations correspondent à un problème de régression et la dernière est un problème de classification. De manière similaire au cas en "N", la variable à estimer peut être directement le transit sur la ligne l après déconnexion de la ligne ld, la différence ou la classification de ce transit par rapport à un seuil nominal. Un dernier cas est également sujet à étude en "N-1" : la différence du transit sur la ligne l entre avant et après déconnexion de la ligne ld.

Au début de cette thèse, nous avons comparé les différents points du tableau 2.1 afin d'identifier la modélisation la plus adaptée. Pour ce faire, les problèmes de classification et régression ont notamment été confrontés en utilisant des algorithmes tel que les arbres de décision (Breiman *et al.*, 1984), déjà utilisés dans le cadre de projets Européens ITESLA (Innovative Tools for Electrical System Security within Large Areas) (for Electrical System Security within Large Areas (ITESLA), 2012). Ces expériences initiales ont mis en évidence l'intérêt de privilégier un problème de régression pour d'obtenir un unique indicateur, ensuite ré-utilisable avec plusieurs seuils de transits. En effet, les indicateurs construits par régression peuvent être comparés à un ou plusieurs seuils et générer des valeurs binaires, correspondant aux états en sécurité / non-sécurité vis-à-vis de ces seuils. Il est donc possible de déduire une classification à partir du résultat obtenu par régression. Nous avons choisi d'étudier de manière privilégiée le cas en "N" avec $y_l = i_l^{\text{"N"}}$. Ce choix du travail en "N" a été motivé en premier lieu par la disponibilité des données. En effet l'historique "brut" du réseau correspond à une étude en "N". Les cas en "N-1", sont obtenus par simulation des déconnexions de lignes sur l'historique du réseau. Le choix de ne pas inclure de valeur des seuils dans les variables à estimer est quant à lui lié au fait qu'il puisse y avoir plusieurs valeurs de seuils possibles avec lesquelles comparer les transits. L'étude de $y_l = i_l^{\text{"N"}}$ est plus générique, et l'estimation de cette grandeur pourra par la suite être réutilisée pour l'étudier, par exemple, par rapport à un dépassement de seuil $i_{l|\neg ld}^{\text{"N-1"}} > i_l^{seuil}$.

2.5.2 Problème d'optimisation en "N"

Pour chaque variable y_l, $l \in L$ pour laquelle nous construisons un indicateur pour l'estimation du transit sur la ligne l, nous cherchons une fonction $f_l : \mathbb{R}^D \to \mathbb{R}$ permettant d'expliquer la relation entre X et y_l. Ainsi, pour une ligne $l \in L$, nous souhaitons trouver une fonction f_l tel que pour toutes les instances observées $(X^m, y_l^m)_{m\in M}$, nous ayons :

$$y_l^m = f_l(X^m), \forall m \in M, \text{ sous contrainte que } f_l \text{ soit } \textit{compréhensible par un humain} \tag{2.9}$$

Nous définissons la notion de *compréhensibilité* par deux éléments :

- le sens (physique ou métier) de la fonction f_l : étayée par le respect des contraintes physiques et opérationnelles dans la *forme de la fonction* ;
- la lisibilité de la fonction f_l : étayée par un critère de *complexité*.

Forme de la fonction

Étant donné les équations physiques et les combinaisons de variables utilisées dans les indicateurs de l'outil COMPARE, la construction d'une fonction f_l, $l \in L$ doit respecter certaines contraintes structurelles. Pour cela, nous nous donnons, dans un premier temps, un ensemble

restreint F d'opérations autorisées pour la construction de f_l. L'ensemble F pourrait être défini, par exemple, comme étant :

$$F = \{+,\ -,\ /,\ \times,\ \text{racine\_carre},\ \text{carre},\ \text{somme},\ \text{moyenne},\ \max,\ \min\}$$

L'ensemble décrit ci-dessus, comprend notamment les opérations utilisées dans les indicateurs COMPARE tel que décrit en section 2.4 (max, min, somme, moyenne, +, -), ainsi que d'autres opérations présentes dans les équations physiques de la sous-section 2.2.3 (tel que $\times$, racine_carre, carre ou /).

Une fois cet ensemble F défini à partir de contraintes physiques et métier, la construction de f_l passe par des sélections et combinaisons d'un nombre minimal d'opérations dans l'ensemble F, combinées avec certaines variables de X. Cette sélection et ces combinaisons de variables d'entrées et d'opérations permet alors de construire f_l sous la forme d'une *expression symbolique* (représentable de manière similaire aux indicateurs illustrés dans la sous-section 2.4.2) pouvant ensuite être évaluée sur les données X pour obtenir une estimation du transit. Les opérations issues de l'ensemble F peuvent être sélectionnées plusieurs fois.

Complexité de la fonction f_l

Comme nous l'avons précisé ci-dessus, nous voulons également que les fonctions $f_l\ \forall l \in L$ soient *lisibles* par un humain. Afin de quantifier la lisibilité d'une expression, nous utilisons un critère de *complexité*, que nous notons κ_l avec $\kappa_l(f_l) \in \mathbb{R}$. Ce critère de complexité a pour but principal de *quantifier* le nombre d'opérations issues de l'ensemble F, ainsi que le nombre de variables issues de l'ensemble X utilisées pour construire la représentation de f_l sous forme d'*expression symbolique*. Pour un $l \in L$ donné, un exemple de définition de κ_l pourrait être, par exemple, la somme du nombre de variables d'entrées de X et du nombre d'opérations de F utilisées : $\kappa_l(f_l) = \kappa_l^X(f_l) + \kappa_l^F(f_l) = count(f_l, X) + count(f_l, F)$, avec *count* une fonction comptant le nombre d'occurrences des éléments de X (ou F) présents dans f_l. Enfin, pour que f_l soit lisible par un humain, nous nous donnons un seuil de complexité κ_{seuil_humain} que devra respecter la complexité de la fonction. Avec ces différentes définitions, l'équation 2.9 revient alors à chercher une fonction optimale f_l^* tel que :

$$f_l^* = \underset{f_l}{\text{argmin}}\ (\text{erreur}(y_l, f_l(X))), \text{avec } \kappa_l(f_l) < \kappa_{seuil\_humain}. \tag{2.10}$$

Avec cette formalisation, nous pouvons maintenant donner quelques exemples de résolutions, avec des formes de solutions de plus en plus complexes.

2.6 Exemples de résolutions

2.6.1 Résolution linéaire à une unité physique

A chaque variable de X, notée x_d, $d \in D$ est associée une unité u_d^x (par exemple, la puissance active p). En découle $U^x = \{u_d^x\}_{d \in D}$ l'ensemble des unités distinctes de X. Nous définissons de même u_l^y l'unité de y_l. Nous définissons ainsi l'ensemble de toutes les unités de y : $U^y = \{u_l^y\}_{l \in L}$ ainsi que l'ensemble de toutes les unités $U = \{u | u \in U^x \cup U^y\}$.

Dans ce premier exemple de résolution, nous nous intéressons au cas où $U = \{u_p^x, u_p^y\}$ avec $u_p^x = u_p^y$, c'est-à-dire que toutes les variables sélectionnées x_d et y_l ont pour unité u_p, la puissance

active et $card(U) = 1$. Une forme de fonction f_l possible est la suivante :

$$f_l = \sum_{\substack{d \in D \\ u_d^x = u_p}} \alpha_d \, x_d \text{ , avec } \alpha_d \in \mathbb{R} \ \forall d \tag{2.11}$$

Dans ce cas précis, en reprenant les notations de l'équation 2.10, nous pouvons définir la fonction d'`erreur` par l'Erreur Quadratique Moyenne (EQM) et la fonction de complexité par le nombre de variables de X utilisées $\kappa_l(f_l) = \kappa_l^X(f_l) = \|\alpha\|_0$. En effet, l'ensemble F est ici restreint à $F = \{\times, +\}$, et κ_l^F peut être négligé car directement proportionnel à κ_l^X.

La recherche de ce type de fonctions peut se faire, par exemple, en utilisant des méthodes de type Lasso (Zou *et al.*, 2007) qui utilise une pénalisation en norme 1 sur les coefficients α et force sa sparcité.

2.6.2 Résolution avec topologie

Dans le cas de la résolution avec des variables de topologie, nous nous intéressons à la construction d'une expression conditionnée par une ou plusieurs variables de topologies, et dans laquelle pour chaque condition topologique, une sous-expression ne portant pas sur des variables de topologie est créée. Dans ce deuxième exemple, nous travaillons plus particulièrement sous la condition $card(U) = 2$ avec l'ensemble des unités $U = \{u_p^x, u_p^y, u_\tau^x\}$ et $u_p^x = u_p^y$. u_τ correspond ici à l'"unité" des variables booléennes, c'est-à-dire sans unité avec des valeurs binaires. Ici, une forme possible pour la fonction f_l est définie par :

$$f_l = \begin{cases} if\{\ condition_1(\tau)\ \} & then\{\ \sum\limits_{\substack{d \in D \\ u_d^x = u_p}} \alpha_d x_d\ \} \\ else\ if\{\ condition_2(\tau)\ \} & then\{\ \sum\limits_{\substack{d' \in D \\ u_{d'}^x = u_p}} \alpha_{d'} x_{d'}\ \} \\ \ldots & \\ else\{\ \sum\limits_{\substack{d'' \in D \\ u_{d''}^x = u_p}} \alpha_{d''} x_{d''}\ \} & \end{cases} \text{, avec } (\alpha_d, \alpha_{d'}, \alpha_{d''}) \in \mathbb{R}^3 \ \forall d,\ d',\ d'' \tag{2.12}$$

Les conditions $condition_1$, $condition_2$, ... sont des opérations booléennes sur les variables de topologie. La recherche de ces conditions peut se faire soit de manière jointe avec celles des alphas $(\alpha, \alpha', \alpha'')$, soit fournis comme information données a priori pour la recherche de f_l. L'identification des conditions pourrait être réalisée par des algorithmes dédiés, a priori et en amont de la recherche des alphas, par exemple, via des algorithmes de détection de communautés appliqués au réseau électrique tel que Infomap (Marot *et al.*, 2018b).

2.6.3 Résolution avec plusieurs unités physiques

A partir des cas linéaires, présentés ci-dessus, nous étendons maintenant le problème de recherche d'une fonction f_l avec un nombre supérieur d'unités, i.e. $card(U) \geq 2$. Par exemple, en prenant en compte toutes les variables du jeu de données initial, $X_p, X_q, X_\tau, X_v, X_\Theta$, nous obtenons $card(U) = 5$. Ici, avec $X_{\setminus X_\tau} = (X_p, X_q, X_v, X_\Theta)$, une forme possible pour la fonction f_l est alors définie par :

$$f_l = \begin{cases} if\{\ condition_1(\tau)\ \} & then\{\ sous\_indicateur_1(X_{\setminus X_\tau})\ \} \\ else\ if\{\ condition_2(\tau)\ \} & then\{\ sous\_indicateur_2(X_{\setminus X_\tau})\ \} \\ ... & \\ else\{\ sous\_indicateur_K(X_{\setminus X_\tau})\ \} & \end{cases} \tag{2.13}$$

Les conditions $(condition_k)_{k\in\{0...K-1\}}$ sont des opérations booléennes sur des variables de topologie τ, comme précédemment dans la sous-section 2.6.2. Les indicateurs $(sous\_indicateur_k)_{k\in\{0...K\}}$ réalisent quant à eux des opérations combinant les variables hors topologie $X_{\setminus X_\tau}$ uniquement par des opérations licites sur les différentes dimensions. Les formes de ces indicateurs peuvent être similaires à celles décrites en section 2.4 (à l'exception de l'indicateur décrivant i_{l3} car il comporte une variable de topologie).

Pour la recherche d'une telle fonction, la complexité peut être définie par le nombre de variables de X et d'opérations de F, condition par condition : $\kappa_l(f_l) = \sum_{k\in\{0...K\}} \kappa_{l,k}$ où $\kappa_{l,k}$ est, à nouveau, décomposé en comptant le nombre de variables de X et le nombre d'opérations de F utilisés pour construire $condition_k$ et $sous\_indicateur_k$.

Ce dernier exemple nous intéresse particulièrement dans cette thèse, car il vise à combiner les variables d'entrée par des opérations, potentiellement non-linéaires, tout en permettant de prendre en compte les variables de topologie. Il sera l'objet principal des travaux menés dans les chapitres suivants.

2.7 Conclusion du chapitre

Dans ce chapitre, nous avons décrit les contraintes physiques, relatives au réseau électrique, à prendre en compte dans cette thèse. Plus précisément, les contraintes physiques principales identifiées sont la forme des équations physiques et l'importance prioritaire de certaines grandeurs. Avec cette description, nous avons introduit les notations utiles pour la construction d'indicateurs utilisés dans l'étude en "N" et en "N-1". Puis, un tour d'horizon des indicateurs a été réalisé en montrant les indicateurs existants dans l'outil COMPARE, ainsi qu'en illustrant ceux qui pourraient être créés grâce aux méthodes développées dans cette thèse. Ce travail préliminaire permet d'identifier quelles sont les formes d'indicateurs utilisées par les opérateurs, ainsi que d'autres formes possibles exploitables dans le futur.

A partir de ces contraintes physiques et issues de l'expérience métier, nous proposons ensuite une modélisation du problème auquel répondra cette thèse. Le choix de la modélisation adoptée en "N" est détaillé, ainsi que son extension au problème "N-1". Enfin, des exemples de résolution de ce problème sont présentés pour illustrer le problème de recherche d'indicateurs.

Les pré-requis identifiés dans ce chapitre permettront, dans le prochain chapitre, de présenter l'état de l'art existant pour la résolution du problème modélisé.

Chapitre 3

Etat de l'art

3.1 Introduction

Le chapitre précédent, nous avons spécifié la problématique industrielle à laquelle vise à répondre cette thèse. Nous y avons défini le problème d'optimisation à résoudre ainsi que ses contraintes industrielles à prendre en compte, que sont le respect des lois physiques et les règles opérationnelles.

Ce chapitre vise à replacer les travaux de cette thèse dans le contexte des différents domaines qui y sont rattachés. Ces domaines sont principalement : la représentation de connaissance, la régression symbolique et l'interactivité. Dans un premier temps, nous présentons le domaine de la représentation de connaissances, et plus particulièrement, la représentation de connaissances pour l'apprentissage automatique. Puis, nous nous intéressons à la régression symbolique en détaillant différents types de méthodes de résolution tel que les algorithmes génétiques, l'apprentissage profond par réseaux neuronaux, ou l'apprentissage par renforcement. Enfin, dans un dernier temps, nous présenterons le domaine de l'interactivité, la taxonomie employée dans cette thèse, les moyens d'interaction efficaces et les algorithmes interactifs utilisés dans les différents domaines de résolution de la régression symbolique. Dans la section 3.3, les premières parties de chaque sous-section récapitulent brièvement le fonctionnement des algorithmes en question et sont destinées à donner une compréhension globale de leur principe aux personnes qui ne seraient pas familières de ces algorithmes.

3.2 Représentation de connaissances

3.2.1 Connaissance tacite et connaissance explicite

La représentation de connaissances vise à représenter des informations et des connaissances symboliques humaines aux ordinateurs, sous une forme utilisable par les différents partis. Elle est définie plus formellement par M. K. Bergman (Bergman, 2018) comme un moyen de "représenter des *informations* sur le monde sous une forme qu'un système informatique peut utiliser pour résoudre des tâches complexes." Il peut s'agir de connaissances scientifiques validées sur des expériences (lois physiques par exemple), de connaissances générales connues de tout le monde, ou de connaissances expertes détenues par un petit nombre d'initiés (von Rueden *et al.*, 2021). En gestion de la connaissance, ou *knowledge management* en anglais, la connaissance est conceptualisée sous deux formes principales, considérées comme opposées (Puusa et Eerikäinen, 2010) : la connaissance explicite et la connaissance tacite. Les caractéristiques (Dalkir, 2013) de ces deux

Connaissance tacite	Connaissance explicite
Adaptabilité, extensibilité à des situations nouvelles et exceptionnelles	Disséminable, reproductible et ré-applicable
Expertise, savoir-faire, compréhension des causes	Enseignable, éducative
Collaborative, vision et culture partagée	Organisable, systématisable, traductible en énoncés de mission et directives opérationnelles
Acquise par la pratique, par coaching/mentorat sur des expériences individuelles en face à face	Transmissible par des produits et des processus documentés

TABLE 3.1 – Propriétés des connaissances organisationnelles de types tacites et explicites. Tableau inspiré de (Dalkir, 2013).

formes de connaissances sont synthétisées dans le tableau 3.1. D'un côté, la connaissance explicite est facilement organisable et documentable. Elle est, de fait, plus facilement ré-utilisable et communicable à un système informatique. De l'autre côté, la connaissance tacite est, au contraire, moins formalisable. Elle est enseignable plutôt par la pratique ou sous forme de mentorat.

Ces deux formes de connaissances peuvent donc contenir des informations distinctes. Certaines connaissances non-explicites sont toujours uniquement détenues par les humains et transmissibles par eux. Ainsi, dans des domaines industriels, une partie de la connaissance de l'entreprise est, par exemple, détenue par les experts du domaine concerné. Dans le cadre de l'apprentissage automatique, apprendre sur de la connaissance humaine passe donc par une nécessaire prise en compte de cette double structure de connaissances (documentées ou non). Par conséquent, l'accès à de la connaissance tacite par un algorithme d'apprentissage requiert d'utiliser d'autres approches d'apprentissage que celles mises en oeuvre pour la connaissance explicite. Ce point sera détaillé dans la section 3.4.

3.2.2 Formalisation de la connaissance explicite

En informatique, différentes approches de formalisation de la connaissance explicite existent, tel que les *ontologies* (Studer *et al.*, 1998) ou les *grammaires formelles* (Chomsky, 1959). Une ontologie permet de structurer la connaissance d'un domaine spécifique par "une spécification formelle et explicite d'une conceptualisation partagée" (Studer *et al.*, 1998). Pour un domaine donné, elle est composée (Taye, 2010) de concepts qui structurent la connaissance, ainsi que des relations qui lient ces concepts. Les éléments de chaque concept sont représentés par des instances et des axiomes imposent des contraintes sur les valeurs des concepts ou des instances. Les ontologies sont notamment utilisées dans le domaine du web sémantique (Hepp *et al.*, 2007).

Dans le domaine linguistique, la Théorie des Langages Formels (Linz, 2011), noté TLF (ou en anglais, *formal language theory*) propose quant à elle de formaliser la connaissance de la structure des langues (syntaxe, grammaire, ...) par des grammaires formelles. Un langage y est décrit par un ensemble de *symboles*, appelé *alphabet*. Ces symboles sont ensuite combinés, sous formes d'ensembles finis, pour construire des chaînes de caractères formant des *expressions*. La structure d'un langage peut alors être décrite formellement par une *grammaire* explicitant des contraintes syntaxiques sur le langage. Plus précisément, une grammaire G est définie par un

tuple :

$$G = (\sigma_{start}, (\sigma_{nt})_{nt \in NT}, (\sigma_t)_{t \in T}, \rho) \tag{3.1}$$

avec σ_{start} le symbole de départ de la grammaire, $(\sigma_{nt})_{nt \in NT}$ un ensemble de symboles non-terminaux, $(\sigma_t)_{t \in T}$ un ensemble de symboles terminaux et ρ les règles de production utilisées pour combiner terminaux/non terminaux. La construction d'une expression à partir des règles ρ d'une grammaire sera détaillée dans la sous-section 3.2.3. Les règles de production sont centrales à la définition de la grammaire. Elles spécifient comment la grammaire transforme un ensemble de règles en *expressions* et définissent le langage associé à la grammaire. D'une grammaire G, découle alors le langage de G, noté $\mathscr{L}$(G), défini comme l'ensemble des expressions construites à partir de G.

Par analogie avec les ontologies, introduites au début de la section, nous pouvons également remarquer que les symboles (terminaux et non-terminaux) sont similaires aux "concepts" d'un langage, les expressions peuvent être assimilées aux "instances" et les règles de production sont comparables aux "relations" et "axiomes" qui contraignent les valeurs des concepts et des instances.

En TLF, la hiérarchie de Chomsky (Chomsky, 1959) propose d'ordonner les différents types de grammaires selon la capacité de restriction des règles de productions qui la compose. Cette hiérarchie comporte quatre niveaux de complexité croissante :

— les grammaires générales, sans restriction (type 0),

— les grammaires sensibles au contexte (type 1),

— les grammaires non-contextuelles (type 2)

— et les grammaires régulières (type 3)

Chaque famille de grammaire de type i est un sous-ensemble de la famille de type i - 1. Dans la suite de cette thèse, nous nous intéresserons plus particulièrement aux grammaires non-contextuelles (type 2). Celles-ci présentent notamment l'avantage d'être plus expressives que les grammaires régulières (type 3), tout en étant plus précise que les grammaires sensibles au contexte (type 1). Par ailleurs, elles sont, à ce titre, fréquemment utilisées pour représenter de la connaissance dans différentes applications telles que l'apprentissage évolutionnaire (Whigham *et al.*, 1995) et l'apprentissage par renforcement (Mandow *et al.*, 2020). L'usage de ce type de grammaire a également été étendu à d'autres domaines, au-delà de l'étude des langues, tels que la physique des particules (Cherrier *et al.*, 2019), la synthèse musicale (Loughran *et al.*, 2015) ou encore l'architecture (Mandow *et al.*, 2020).

3.2.3 Grammaires non-contextuelles

Définition

Chaque règle de production d'une Grammaire Non-Contextuelle (GNC) est de la forme :

$$A \leftarrow \gamma \tag{3.2}$$

avec $A \in (\sigma_{nt})_{nt \in NT}$ et $\gamma \in (\sigma_s)_{s \in (NT \cup T)^*}$, i.e. A est toujours un symbole non-terminal et γ peut être un symbole terminal ou non-terminal. Lorsque une règle de production $A \leftarrow \gamma$ est appliquée lors de la génération d'une expression, le symbole non-terminal A est remplacé par le symbole γ. Plus généralement, la construction d'une expression à partir d'une grammaire consiste en l'application d'un ensemble de règles de productions jusqu'à l'obtention d'une expression ne contenant *que* des symboles terminaux.

```
<exp> ::= <a> | <b>
<a> ::= <i> * <i> | <i> / <i> | <i> * ( <b> )
<b> ::= <i> + <i> | <i> - <i>
<i> ::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9
```

FIGURE 3.1 – Exemple de grammaire non-contextuelle pour la construction d'expressions symboliques. L'ensemble des symboles non-terminaux est {<exp>, <a>, <b>, <i>} et l'ensemble de symboles terminaux est {"*", "/", "+", "-", "0", "1", "2", "3", "4", "5", "6", "7", "8", "9"}. <exp> est le symbole de départ.

Notation

Pour écrire les règles de production des grammaires non-contextuelles, une convention est d'utiliser le format Backus-Naur Form (Backus *et al.*, 1963), noté BNF. Cette forme est essentiellement la même que la notation utilisée dans l'équation 3.2, mais présente une apparence différente. Nous illustrons cette notation dans la figure 3.1. Dans le format BNF, les symboles non-terminaux sont entourés de crochets triangulaires. Les symboles terminaux sont écrits sans marquage particulier. Le format BNF utilise également le caractère | pour séparer les différentes règles de production attachées à un même symbole non-terminal σ_{nt}. Enfin, l'élément : := est utilisé à la place de celui $\leftarrow$ de l'équation 3.2. Avec des notations, nous appelons la *production* d'un symbole non-terminal σ_{nt}, l'ensemble des règles de production de la grammaire associées à ce symbole. Cette production correspond à une ligne de la grammaire et commence par "$< \sigma_{nt} >$: :=" suivi des différentes règles utilisables pour remplacer le symbole σ_{nt}.

Construction d'une expression

Décrivons comment obtenir l'expression "9 + 1" à partir de la grammaire décrite dans la figure 3.1. Cette construction d'expression se décompose en quatre étapes :

1. Nous sélectionnons dans un premier temps une règle parmi l'ensemble des règles de productions du symbole de départ $< exp >$: <a> *ou* <b>. Nous choisissons la 2^{eme} règle <b>. L'expression à cette étape devient "<b>".
2. La 3^{eme} ligne de la grammaire définit des règles de productions du symbole $< b >$: <i> "+" <i> *ou* <i> "-" <i>. Nous sélectionnons la règle <i> "+" <i>. Nous créons une queue pour garder en mémoire les symboles à remplacer. Celle-ci contient deux éléments <i> et <i>. Nous remplaçons chacun d'entre eux en effectuant une recherche en profondeur itérant sur le premier symbole de la queue jusqu'à ce que celle-ci soit vide. L'expression à cette étape devient "<i> + <i>".
3. Le premier symbole de la queue est <i>. La production associée à <i> comporte les règles suivantes : "0" , "1" , "2", "3", "4", "5", "6", "7", "8", "9". Nous choisissons la règle "9". L'expression à cette étape devient "<9> + <i>".
4. Le dernier symbole de la queue est <i>. Nous choisissons à nouveau une règle parmi les règles de production du symbole <i>. La règle choisie est "1". L'expression finale est alors "9 + 1".

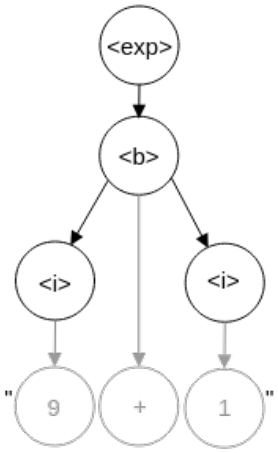


FIGURE 3.2 – Arbre syntaxique de l'expression "9+1", construite à partir de la grammaire de la figure 3.1. Chaque symbole utilisé est représenté dans un cercle. Les cercles noirs (respectivement gris) correspondent aux symboles non-terminaux (resp. terminaux) issus de l'application des règles mentionnées pour la construction de l'expression "9+1".

En résumé, l'application successive de quatre règles, issues de la grammaire de la sous-section 3.2.3, a permis de construire l'expression "9+1" :

<exp> : := <b>

<b> : := <i> + <i>

<i> : := "9"

<i> : := "1"

Une expression ainsi construite est alors représentable sous forme d'arbre syntaxique, aussi appelé *parse tree* en anglais, comme illustré dans la figure 3.1. Ce type d'arbres est composé de noeuds et de branches. Dans la figure 3.2, l'arbre d'analyse syntaxique comprend la structure entière, partant du symbole de départ <exp> et aboutissant à chacun des noeud de type branche et feuilles "9 + 1". Dans un arbre d'analyse syntaxique, chaque noeud est soit un nœud racine, soit un noeud branche, soit un noeud feuille. Dans l'exemple de la figure 3.2, <exp> est un noeud racine, <b>, <i> et <i> sont des noeuds branches, tandis que "9" et "1" sont des noeuds feuilles.

3.2.4 Grammaires probabilistes

Dans une GNC (Grammaire Non-Contextuelle), toutes les règles d'une production donnée sont implicitement *équiprobables*. Par exemple, en reprenant la grammaire définie dans la figure 3.1, les deux règles <a> et <b> associées au symbole <exp> sont aussi probables dans le language induit par cette grammaire. Afin de mieux représenter l'occurrence de certaines expressions dans un langage, des *Grammaires Non Contextuelles Probabilistes* (Sakakibara, 2017) proposent une nouvelle définition de la grammaire G avec le tuple suivant :

$$G = (\sigma_{start}, (\sigma_{nt})_{nt \in NT}, (\sigma_t)_{t \in T}, \rho, \Psi) \tag{3.3}$$

avec $\sigma_{start}, (\sigma_{nt})_{nt \in NT}, (\sigma_t)_{t \in T}, \rho$ de manière identique aux termes de l'équation 3.1 et en rajoutant le terme Ψ correspondant à la probabilité (ou poids) associée à chaque règle d'une même

```
<exp> ::= <a> | <b> || probs [0.1, 0.9]
<a> ::= <i> "*" <i> | <i> "/" <i> || probs [0.5, 0.5]
<b> ::= <i> "+" <i> | <i> "-" <i>  || probs [0.9, 0.1]
<i> ::= "0" | "1" | "2" | "3" | "4" | "5" | "6" | "7" | "8" | "9"
 || probs [0.05, 0.3, 0.05, 0.05, 0.05, 0.05, 0.05, 0.05, 0.05, 0.3]
```

FIGURE 3.3 – Exemple de grammaire non-contextuelle probabiliste. Les probabilités associées à chaque production sont listées à la fin de la ligne et séparés par l'élément "|| probs".

production. En d'autres termes, la somme des probabilités de règles de production d'un même symbole ont une somme de 1 (i.e. même ligne dans la grammaire au format BNF). La figure 3.3 détaille une version probabiliste de la grammaire présentée dans la figure 3.1. Les probabilités associées à chaque règle sont listées entre crochets et séparés des règles de la production par l'élément "|| probs".

Les grammaires non-contextuelles probabilistes ont été étudiées notamment en lien avec la Régression Symbolique (détaillée en section 3.3) et présentent l'avantage de pouvoir apprendre les poids associés à chaque règle (Sotto et de Melo, 2017; Cunha, 2021). Les poids associés aux probabilités peuvent être mis à jour à chaque génération en fonction règles présentes dans les meilleurs et les pires individus (Ratle et Sebag, 2001), en fonction de l'occurrence de chaque symbole (Kim et Ahn, 2015), ou progressivement avec une règle de mise à jour paramétrée par un taux d'apprentissage (méthode GB-LGP) (Sotto et de Melo, 2017). Les probabilités peuvent également être estimées par un réseau de neurones (Drori *et al.*, 2019).

3.2.5 Interprétabilité

Taxonomie

La notion de représentation de connaissances est fortement liée à celle d'interprétabilité. En effet, en apprentissage automatique, là où la représentation de connaissance permet de faire transiter de l'information depuis l'utilisateur vers la machine, les interprétations et explications permettent de faire le chemin inverse. Il est donc nécessaire de développer des algorithmes interprétables par les utilisateurs. Bien que l'interprétabilité ne soit pas l'objet central de ces travaux, elle joue un rôle important de manière plus générale dans le succès de l'approche d'apprentissage choisie. De plus, l'interprétabilité et l'interactivité (détaillée en section 3.4) sont deux domaines aujourd'hui perçus comme interdépendants (Ventocilla *et al.*, 2018). Il est donc nécessaire de définir la notion d'interprétabilité utile dans ce contexte pour choisir les algorithmes et les interactions adaptées.

La notion d'interprétabilité est décrite dans la littérature par différents termes en anglais considérés généralement comme équivalents et employés de manière interchangeable (Linardatos *et al.*, 2021; Ventocilla *et al.*, 2018) : interpretability, comprehensibility, understandability, explicability. En français nous utiliserons les termes d'interprétabilité, de compréhensibilité, d'intelligibilité et d'explicabilité.

Plusieurs distinctions sont faites entre les modèles interprétables (Du *et al.*, 2019). L'interprétation peut aussi être locale, i.e. s'appliquer à une prédiction seule, ou *globale* et interpréter plusieurs données. De plus, l'interprétation peut être réalisée *ad hoc* c'est-à-dire que le modèle est *intrinsèquement* interprétable, ou *post hoc* après l'apprentissage. L'interprétation de modèles a posteriori peut être faite par le biais d'importance de variables (Lundberg et Lee, 2017) ou d'explications contrefactuelles (Laugel *et al.*, 2019). Cependant, des travaux sur le paradigme

post-hoc met en garde sur le fait que les interprétations post-hoc peuvent être entachées d'erreurs ou résulter d'artefacts plutôt que d'une analyse de données (Rudin, 2019; Laugel *et al.*, 2019; Lipton, 2018).

Enfin, l'interprétabilité peut être *spécifique au modèle*, i.e. n'être pertinente que pour une certaine classe d'algorithmes, ou *agnostique au modèle* et s'appliquer à tout algorithme sans avoir besoin de connaître le modèle interne de l'algorithme. Dans le cadre de cette thèse, nous nous intéresserons aux algorithmes d'apprentissage ayant une interprétabilité intrinsèque afin de prendre en compte les contraintes industrielles détaillées dans le chapitre 2. L'interprétabilité est ici spécifique au modèle, et génère une interprétation globale.

A. Zytek et al. notent qu'un modèle est aussi interprétable que les variables qu'il utilise (Zytek *et al.*, 2022). L'interprétabilité peut donc être décrite à l'échelle des variables et leurs travaux proposent une taxonomie pour la décrire. Nous nous intéressons plus particulièrement à l'interprétabilité des variables construites à destination des *personnes en charge de la prise de décision*, généralement sans expertise en Machine Learning (ML) et qui utilisent des modèles et des explications de ML pour les aider à accomplir des tâches et à prendre des décisions. Les variables interprétables par ces utilisateurs doivent porter les caractéristiques suivantes (Zytek *et al.*, 2022) :

— *Readable* : les variables doivent être *lisibles* i.e. utiliser un langage et un vocabulaire commun et connu afin que les utilisateurs comprennent ce à quoi il est fait référence. Il s'agit du critère minimal à respecter pour être considéré comme interprétable. Les variables lisibles peuvent être des mots, des phrases simples et utiliser des opérations mathématiques connues de tous ou seulement par le public cible.

— *Human-worded* : les variables doivent être formulées en utilisant des mots humains. Par exemple, plutôt que d'utiliser une variable $has\_blue\_color \rightarrow True$ privilégier l'utilisation de $color \rightarrow blue$. Une variable *Human-worded* implique qu'elle est aussi *readable*.

— *Comprehensible* : les variables doivent faire référence à des mesures du monde réel, sur lesquelles les utilisateurs peuvent comprendre, en ayant pris en compte leur niveau de connaissance.

Ces caractéristiques de lisibilité, formulation par des mots humains et de compréhensibilité seront souhaitées dans l'interprétabilité des variables créées.

Algorithmes interprétables

Bien que l'interprétabilité et la précision d'un algorithme soient aujourd'hui considérées comme un compromis à trouver dans le choix du modèle, ce compromis n'est pas systématique, en particulier pour prendre des décisions à enjeux élevés (Rudin, 2019). C'est le cas, par exemple, lorsque les données sont suffisamment structurées pour fournir une représentation. De plus, certains domaines d'études, tel que celui présenté dans cette thèse, considèrent l'interprétabilité comme un *pré-requis* devant être satisfait par l'approche choisie. Ainsi, les travaux de cette thèse s'intéressent plus particulièrement à l'interprétabilité intrinsèque (ou *by design*) pour l'apprentissage d'un modèle de régression et consiste à apprendre des modèles fournissant une interprétabilité directe de la (ou des) solution(s) construite(s).

Dans la littérature, les modèles interprétables les plus fréquents en régression sont des modèles linéaires tel que la régression linéaire (Zou *et al.*, 2007) et RuleFit (Friedman et Popescu, 2008), des modèles non-linéaires tel que les arbres de décisions (Breiman *et al.*, 1984) et les k plus proches voisins (KNN)(Altman, 1992). Leurs caractéristiques respectives sont représentées

Algorithme	Linéarité	Monotonie	Interaction	Connaissance
Régression Linéaire (Zou *et al.*, 2007)	✓	✓	×	×
Arbre de décision (Breiman *et al.*, 1984)	×	Parfois	✓	×
RuleFit (Friedman et Popescu, 2008)	✓	×	✓	×
K-Nearest Neighbors (KNN) (Altman, 1992)	×	×	×	×

TABLE 3.2 – Résumé des algorithmes de régression interprétables fréquemment utilisés (Molnar, 2020).

dans le tableau 3.2. Ces approches peuvent prendre en compte des interactions entre plusieurs variables. Cependant, aucune de ces approches ne permet de prendre en compte de la connaissance représentée sous forme de règles et de contraintes sur la forme de la solution. De plus, certains de ces algorithmes présentent d'autres défauts. Par exemple, pour les règles de la forme IF-THIS-THEN-THAT, similaires à celles générées par les approches RuleFit ou des noeuds d'arbres de décision, des travaux (Huang et Cakmak, 2015) ont montré qu'elles ne sont pas systématiquement correctement interprétées. Il est également possible de sur-interpréter les coefficients d'une régression linéaire (Good et Hardin, 2012).

Face à ce constat, la modélisation du problème de régression par régression symbolique présente un avantage non négligeable d'un point de vue de l'interprétabilité et de la prise en compte de la connaissance. En effet, lorsqu'une approche de régression symbolique est combinée à l'utilisation d'une structure de connaissances (par exemple grammaticale) pour contraindre l'espace des solutions, la régression symbolique permet d'obtenir des solutions *compréhensibles*. Avec certaines structures grammaticales judicieusement choisies, il est également possible de construire par régression symbolique des variables *lisibles*, *formulées avec des mots* et donc, par conséquent, interprétables par des experts selon la définition de Zytek et al. (Zytek *et al.*, 2022).

3.3 Régression Symbolique

3.3.1 Définition du problème

Régression supervisée

En apprentissage automatique, le problème de régression (Hastie *et al.*, 2009) est formalisé de la manière suivante. A partir d'un ensemble de M observations de D variables, noté $X \in \mathbb{R}^{D \times M}$ et d'une variable cible $y \in \mathbb{R}^M$, l'objectif est d'apprendre, par supervision des données X et y, une fonction de modélisation $f : \mathbb{R}^D \to \mathbb{R}$, avec f appartenant à un espace fonctionnel $\mathcal{F}$, qui estime la relation f^* entre X et y :

$$y_m = f^*(x_m) \ , \ \forall (x_m, y_m) \in X \times y \tag{3.4}$$

Les observations X sont des données quantitatives qui peuvent prendre des formes variées, depuis des formats tabulaires, jusqu'aux images et aux graphes. L'espace fonctionnel $\mathcal{F}$ doit être choisi de façon à être suffisamment expressif pour bien représenter la relation de l'équation 3.4, sans avoir une taille trop grande pour éviter le sur-apprentissage (Hastie *et al.*, 2009).

En apprentissage supervisé, trouver la meilleure fonction f^* dans un espace fonctionnel $\mathcal{F}$ donné consiste en un objectif de *minimisation du risque empirique* (Vapnik, 1999), noté R_{emp} :

$$f^* = \underset{f \in \mathcal{F}}{\operatorname{argmin}} \ \mathrm{R}_{\mathrm{emp}}(f) \tag{3.5}$$

L'estimation du risque empirique est réalisée par une moyenne de l'*erreur*, calculée sur un ensemble de données (X, y) et où la métrique d'erreur, notée `metrique_erreur`, est choisie a priori :

$$\mathrm{R}_{\mathrm{emp}}(f) = \frac{1}{|M|} \sum_{m \in M} \texttt{metrique\_erreur}(f(x), y) \tag{3.6}$$

Bien que différents choix de métriques d'erreur sont possibles, l'Erreur Quadratique Moyenne (EQM) est la métrique la plus fréquemment utilisée (Hastie *et al.*, 2009). Pour $(x_m, y_m) \in X \times y$, elle est définie par :

$$\texttt{EQM}(y_m, f(x_m)) = ||y_m - f(x_m)||_2 \tag{3.7}$$

Par ailleurs, plus la taille du jeu de données considéré est grande, plus l'erreur mesurée sera proche du risque réel.

Régression symbolique

Tout comme en régression, la régression symbolique (RS) a pour objectif d'apprendre une fonction f vérifiant l'équation 3.4 en utilisant l'objectif d'optimisation défini dans l'équation 3.5 pour évaluer le modèle construit. Cependant, la Régression Symbolique (RS) se distingue de la régression définie ci-dessus, par le choix d'un espace fonctionnel spécifique $\mathcal{F}$ dans lequel une fonction f est représentée par une *expression symbolique*, notée e. La RS est donc un cas spécifique de régression où le modèle construit est contraint par $\mathcal{F}$, et la RS respecte également les équations (3.4 - 3.7). En RS, la fonction f est alors obtenue par évaluation de l'expression e sur les données X : $f(x) = \mathrm{eval}(e(x))$, $\forall x \in X$. Plus précisément, pour apprendre la fonction f, la RS se donne un espace fonctionnel $\mathcal{F}$ défini par un ensemble d'opérations mathématiques et de variables issues de X. Un exemple d'un tel espace serait, par exemple, l'ensemble de toutes les fonctions qu'il est possible de construire à partir d'une combinaison d'une base d'opérations F={+, -, x, /, abs, ... } et de variables $(x_d)_{d \in D}$: $\mathcal{F} = \{$"$x_0 + x_1$","$x_0 \times x_3$","x_d/x_{d-1}", "abs(x_2)" ...$\}$. La RS ne nécessite donc pas de connaître a priori la forme de la fonction, celle-ci est apprise à partir de l'*exploration* de l'espace fonctionnel $\mathcal{F}$.

Afin de contraindre plus spécifiquement l'espace fonctionnel spécifique $\mathcal{F}$ à un ensemble de fonctions autorisées (limitées, par exemple, par de la connaissance experte), la RS peut être *contrainte* et *guidée par une grammaire*. Dans ce cas, $\mathcal{F}$ est défini à partir du langage $\mathscr{L}$ induit par une grammaire G choisie, $\mathcal{F} = \mathscr{L}(G)$.

Détaillons maintenant les méthodes de résolutions du problème de régression symbolique, guidé ou non par une grammaire. Les méthodes peuvent être des approches : par programmation génétique, par réseaux de neurones profonds, par renforcement et par expansion de base.

3.3.2 Apprentissage par évolution grammaticale

La méthode la plus fréquemment utilisée dans la littérature pour résoudre le problème de régression symbolique est la *Programmation Génétique* (PG). Cette approche, dont les fondements ont été établis par J. Koza (Koza, 1990; Koza, 1992) est un type d'*Algorithmes Génétiques* (AG) faisant évoluer des fonctions, aussi appelés *programmes*. Cette catégorie d'algorithmes s'inspire des mécanismes biologiques présents dans l'évolution Darwinienne (Darwin, 1859). Plus précisément, la PG fait évoluer une population composée d'individus génétiques, au cours de plusieurs itérations, nommées générations. Chaque individu génétique est une fonction f correspondant à une expression symbolique construite à partir d'un espace fonctionnel $\mathcal{F}$ choisi. Les modèles de

PG sont intrinsèquement interprétables et cette caractéristique a été exploitée pour générer des approximations globales simples d'estimateurs boîtes-noires complexes (Evans *et al.*, 2019)

Bien que la PG permette d'apprendre une fonction f sous forme d'une expression symbolique, cette approche ne permet pas de prendre en compte de la connaissance formalisable. A ces fins, différentes stratégies d'intégrations de connaissances ont été proposées. Dans le cas de la prise en compte de l'expertise, la connaissance peut prendre la forme de variables pré-calculées avant l'apprentissage (Prieschl *et al.*, 2019). Pour la génération de programmes informatiques, l'approche *Strongly Typed Genetic Programming* (Montana, 1995) propose d'assigner un type informatique à chaque feuille et noeud de l'arbre syntaxique afin de réaliser des opérations respectant des types autorisés. Cette approche est, cependant, difficilement généralisable pour la prise en compte d'opérations complexes. Pour la prise en compte des unités physiques, l'approche de *Dimensionally Aware Genetic Programming* (Keijzer et Babovic, 1999) pénalise le score d'un individu par la justesse dimensionnelle de cet individu sous la forme d'une distance entre la dimension de son expression, par rapport à une formulation dimensionnellement correcte. Néanmoins, cette approche ne garantit pas d'obtenir des individus d'unités valides.

Pour prendre en compte de la connaissance telle que celle décrite sous forme de grammaire, la PG a été étendue à l'apprentissage sur des structures grammaticales au format BNF (Whigham *et al.*, 1995), sous le terme PG Guidée par la Grammaire, ou PG3. Cette approche permet par exemple de définir des unités physiques ainsi que les opérations autorisées sur ces dimensions, avec de meilleurs individus que lors de l'emploi de grammaires sans prise en compte des unités (Ratle et Sebag, 2000). L'emploi d'une grammaire pour décrire de la connaissance ouvre par ailleurs la possibilité d'appliquer la PG3 à des domaines aussi variés que le choix de pipeline en *Automatic Machine Learning* (AutoML) (Katz *et al.*, 2020), la modélisation financière (Brabazon et O'Neill, 2006), le design d'abris en architecture (O'Neill *et al.*, 2010), la prévision de quantité de pluie en météorologie (Dufek *et al.*, 2013) ou l'estimation de glucose sanguin (Contreras *et al.*, 2018).

Représentation d'un individu

En PG3, la structure de représentation sur laquelle sont réalisées les opérations génétiques est appelée génotype. Ce génotype est ensuite traduit, ici sous forme d'une expression symbolique textuelle : le phénotype. L'une des représentations intuitive du génotype en PG3 est celle sous forme d'arbre (Whigham *et al.*, 1995), tel que nous l'avons illustrée dans la figure 3.2. La PG3 a ensuite été étendue à l'utilisation d'un génome linéaire (Ryan *et al.*, 1998). Dans cette approche, nommée Evolution Grammaticale (EG), le génotype contient une série de codons, représentée par une liste d'entiers, pour encoder chaque symbole choisi dans la grammaire. Il s'agit de l'une des variantes les plus utilisées de la PG (White *et al.*, 2013).

En EG, la correspondance entre génotype et phénotype est réalisée, codon par codon, en sélectionnant une règle de production à partir de la valeur du codon selon l'opération suivante (O'Neill et Ryan, 2001) :

$$id\_regle = (valeur\_codon) \,\text{MODULO}\, (longueur\_production\_symbole\_courant) \tag{3.8}$$

Pour chaque codon, l'identifiant (id) de la règle prise dans la production du symbole courant (commençant à 0) est la valeur du codon, modulo le nombre de règles de cette production. L'opération modulo, aussi noté %, est utilisé pour adapter le nombre entier d'un codon au nombre de règles de production du symbole non-terminal courant. Avec cette représentation, la grammaire permet de générer uniquement des individus grammaticalement corrects et qui en respecte les règles grammaticales, contrairement à la PG.

Algorithm 2 Algorithme d'évolution grammaticale

Require: Observations X, Cible y, Espace fonctionnel grammatical $\mathcal{F} = \mathcal{L}(G)$

```
1: function ÉVOLUTION GRAMMATICALE(F)
2:     condition_d'arret_satisfaite ← False
3:     meilleur_individu ← None
4:     population ← initialisation_de_la_population(F)
5:     while not condition_d'arret_satisfaite do
6:         parents, meilleur_individu ← selection_des_parents(population, X, y)
7:         descendance ← croisement(parents)
8:         descendance ← mutation(descendance)
9:         population ← remplacement(population, descendance)
10:        condition_d'arret_satisfaite ← test_conditions(meilleur_individu)
       return meilleur_individu
```

En reprenant l'exemple de la figure 3.2 associé à la grammaire de la figure 3.1, un génotype pouvant conduire au phénotype "9+1" est [57, 224, 99, 121]. Les règles grammaticales successivement choisies sont :

— dans la production du symbole de départ <exp> comportant 2 règles, l'id est `57%2=1`, soit la règle numéro 1 de valeur "<b>" ;

— dans la production du symbole <b> comportant 2 règles, l'id est `224%2=0` c'est-à-dire la règle n°0 de valeur "<i>+<i>" ;

— dans la production du symbole <i> comportant 10 règles, l'id est `99%10=9`, c'est-à-dire la règle n°9 de valeur "9" ;

— dans la production du symbole <i> comportant 10 règles, l'id est `121%10=1`, c'est-à-dire la règle n°1 de valeur "1" ;

Algorithme d'apprentissage

L'EG est un algorithme dont les étapes d'apprentissage sont similaires à celles de la PG, à la différence près que l'EG utilise un espace fonctionnel $\mathcal{F}$ défini par une grammaire $G : \mathcal{F} = \mathcal{L}(G)$. Les itérations de l'algorithme d'EG sont détaillées dans l'algorithme 2 (O'Neill et Ryan, 2001). Cet algorithme prend en entrée un ensemble d'observations de variables X et d'une cible y, ainsi qu'un espace fonctionnel $\mathcal{F} = \mathcal{L}(G)$ construit à partir d'une grammaire G. L'algorithme commence par initialiser la population de la première génération de l'algorithme (ligne 4) en générant un ensemble d'individus génétiques à partir de G. Les algorithmes évolutionnaires sont sensibles à l'initialisation de la population et une grande variété de stratégies d'initialisation ont été proposées. Ces stratégies incluent l'initialisation aléatoire, l'initialisation avec profondeur et taille maximales fixées (par exemple, Probabilistic tree-creation version 2 (Harper, 2010)), l'initialisation avec indépendance de la position (Position Independant Grow PI_GROW) (Fagan *et al.*, 2016), ou encore l'initialisation aléatoire sans individus invalides ni doublon (Random valids no duplicates, RVD) (Nicolau, 2017).

Cette population évolue ensuite sur plusieurs générations, soit jusqu'à convergence de l'algorithme vers le score optimal, soit jusqu'à atteindre le nombre maximal d'itérations autorisées. L'évolution d'une génération donnée suit les étapes suivantes (Banzhaf *et al.*, 1998) :

- sélection : les parents de la prochaine génération sont sélectionnés parmi la génération courante selon une stratégie donnée. Un individu peut être sélectionné plusieurs fois. Les

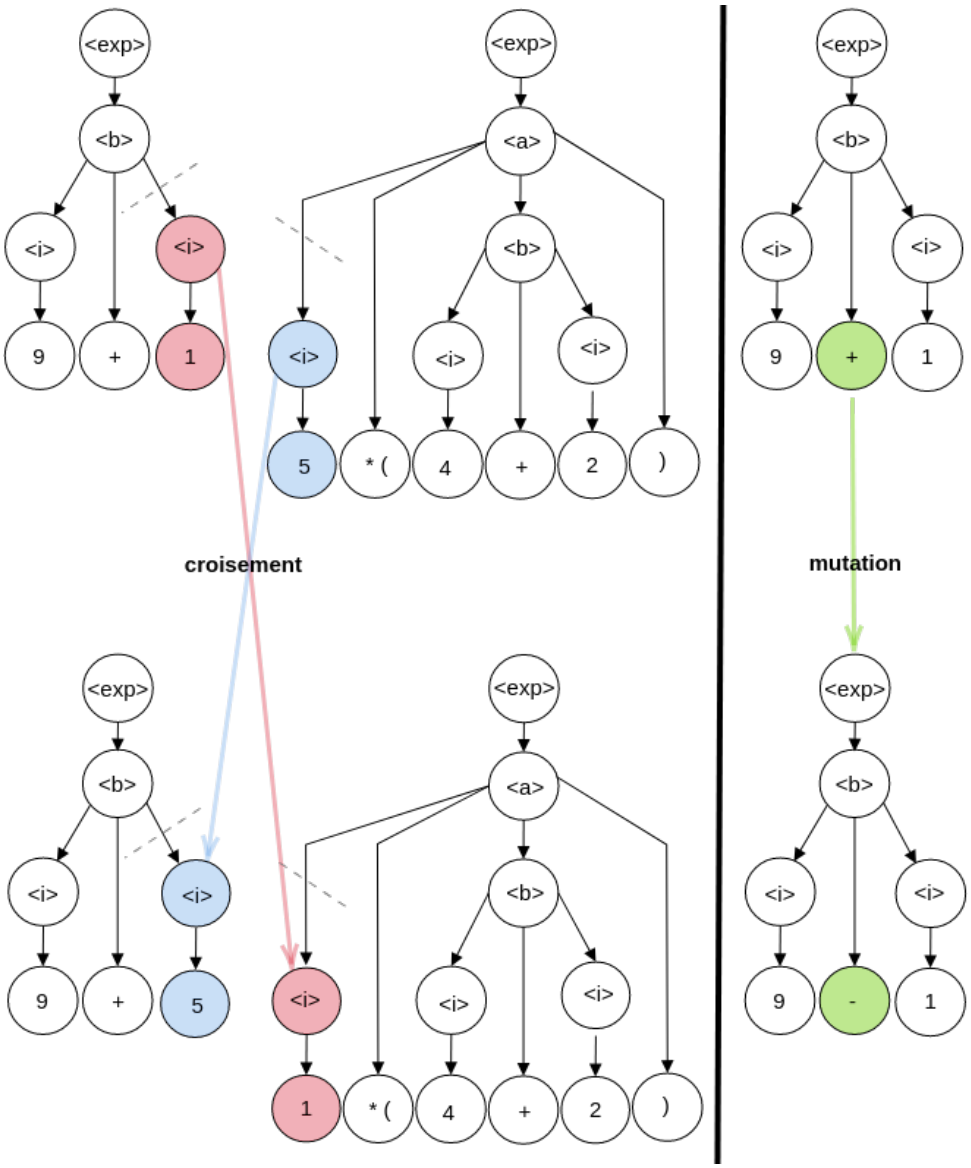


FIGURE 3.4 – Étapes de croisement et de mutation pour la programmation génétique guidée par la grammaire. L'étape de croisement recombine deux parents pour construire deux nouvelles expressions symboliques "enfants" portant une partie de symboles de chaque parent. L'opération de mutation modifie l'un des noeuds ou feuilles de l'arbre syntaxique, en le remplaçant par un nouvel élément pris parmi les règles autorisées dans la grammaire. Les individus ont été créés à partir de la grammaire figure 3.1.

stratégies classiques pour la sélection d'un groupe de parents incluent la sélection par classement en choisissant les individus ayant la meilleure fitness dans toute la population courante (Baker, 1985), par jeu de roulette proportionnellement à leur fitness (Hancock, 1994), ou encore en réalisant plusieurs tournois entre des individus choisis aléatoirement dans la population et en prenant l'individu de meilleure fitness dans chaque tournoi (Miller *et al.*, 1995).

- **croisement** : les génotypes des parents sont recombinés par paires et s'échangent une partie de leur génotype, selon une probabilité $\mathbb{P}_{croisement}$. Chaque paire de parents produit deux enfants qui portent des gènes (noeuds/feuilles) de leurs deux parents. En PG3, les croisements peuvent être réalisés en un point du génome (*one-point crossover*), en deux points (*two-points crossover*) ou encore de manière uniforme sur tout le génome (Poli et Langdon, 1998). Le principe de crossover en PG3 est illustré sous forme d'arbre dans la figure 3.4.
- **mutation** : après le croisement, les individus peuvent être modifiés aléatoirement, selon une probabilité $\mathbb{P}_{mutation}$, par remplacement des noeuds ou feuilles de l'arbre. Cette étape a pour but d'introduire de la diversité dans la population échantillonnée.
- **remplacement** : la génération courante est remplacée (entièrement ou non) par la nouvelle génération, générée par croisement/mutation des parents sélectionnés.

Enfin lorsque le critère d'arrêt est rencontré, le meilleur individu rencontré pendant l'apprentissage est retourné.

Gestion de la complexité

La gestion de la complexité des expressions générées par l'EG suscite un intérêt croissant sur le plan de l'explicabilité des modèles (Javed *et al.*, 2022). En effet, construire des expressions de taille plus faible permet de les rendre plus lisibles et interprétables. La complexité des expressions symboliques peut être définie par différents critères (Le *et al.*, 2016) : des critères structurels (le nombre de noeuds ou de niveaux de l'arbre, la longueur de description minimale, la somme des noeuds de tous les sous-arbres, ...), des critères fonctionnels (nombre de non-linéarités), ou des mesures statistiques (Akaike Information Criterion, Bayesian Information Criterion, Structural Risk Minimization). D'autres critères mentionnés comme ayant trait à la "complexité" permettent également de faire évoluer la structure des individus en spécialisant les noeuds les plus près du noeud racine avec les fonctions complexes et les fonctions les moins complexes vers les feuilles (Burlacu *et al.*, 2019). Or, tout comme la PG, l'EG peut souffrir du phénomène, appelé *bloat*, de croissance progressive de la taille du génome de ces individus au fil des générations sans amélioration de leur performance (Silva, 2008). L'origine de ce phénomène parait être multiple. Des travaux (Luke et Panait, 2006) en PG ont identifié des facteurs potentiels tel que la fitness, ou encore la présence de portions de génomes non-utilisées par le phénotype (introns). Le choix de la grammaire en EG a été également pointé comme origine de la taille croissante des individus (Nicolau et Agapitos, 2018).

Parmi les stratégies utilisées pour contrôler ce phénomène, l'une des approches les plus classiques (Koza, 1992; Luke et Panait, 2006) est l'utilisation d'un paramètre d'apprentissage sur la taille des individus, par exemple, en remplaçant tous les enfants dépassant une taille donnée par l'un de leurs parents (Koza, 1992), ou en répétant l'étape de croisement jusqu'à obtenir des individus valides (Martin et Poli, 2002). L'une des autres catégories d'approches est l'utilisation d'une contrainte de *parcimonie* sur la sélection des parents, généralement exprimée sous forme de terme additionnel dans la fitness.

Algorithm 3 Algorithme de descente de gradient stochastique

Require: Observations X, Cible y, Taux d'apprentissage λ

1: $\theta \leftarrow$ initialisation_des_paramètres_du_reseau()
2: **while** not condition_d'arret_satisfaite() **do**
3: $\quad M_{minibatch} \leftarrow echantillonnage\_minibatch(X, y)$
4: $\quad \mathrm{J}(\theta) \leftarrow \frac{1}{2M_{minibatch}} \sum_{m=0}^{M_{minibatch}} \texttt{metrique\_d\_erreur}(\mathrm{nn}_\theta(X^m), y^m)$
5: $\quad \theta \leftarrow \theta - \lambda \nabla_\theta \mathrm{J}(\theta)$

3.3.3 Apprentissage par réseaux de neurones

Définitions

Les réseaux de neurones profonds ont prouvé leur utilité pour la résolution d'une grande variété de tâches telles que l'analyse d'image (Krizhevsky *et al.*, 2012), le traitement des langues (Graves *et al.*, 2013) ou encore la planification et le contrôle (Silver *et al.*, 2016). En effet, les réseaux de neurones profonds sont de bons approximateurs de fonctions et des preuves de leur capacité d'approximateurs universels existent sous certaines conditions (Hornik *et al.*, 1989). Cependant, malgré leur expressivité (Raghu *et al.*, 2017) et leur succès empirique sur des applications diverses, les réseaux de neurones manquent souvent d'interprétabilité (Zhang *et al.*, 2021). Cet élément est aujourd'hui un frein à l'adoption et à la confiance des utilisateurs en de telles méthodes (Carvalho *et al.*, 2019), en particulier pour prendre des décisions fiables et/ou équitables à partir des résultats de ces algorithmes (Feuerriegel *et al.*, 2020; Zhang et Weiss, 2021).

Dans la littérature, plusieurs catégories de réseaux de neurones coexistent et sont aujourd'hui utilisés conjointement pour traiter aussi bien des données tabulaires, textuelles ou temporelles (Hochreiter et Schmidhuber, 1997) que des images (Russakovsky *et al.*, 2015). Schématiquement inspirés du fonctionnement des neurones biologiques, les réseaux de neurones artificiels proposent d'estimer la relation entre des données X et y par une succession de produits matriciels et de non-linéarités. Une opération de produit matriciel suivie d'une non-linéarité est appelée *neurone artificiel* (Rosenblatt, 1958) et suit la formule suivante en une dimension :

$$\mathrm{neurone}_k(x) = \Phi(W_k^T x + b) \tag{3.9}$$

avec $W_k = (w_{k,d})_{d \in D} \in \mathbb{R}^D$ la matrice des coefficients du neurone k et $b \in \mathbb{R}$. La fonction Φ est appelée *fonction d'activation* et correspond à une opération (généralement) non-linéaire telle que la fonction sigmoid, tangente hyperbolique ou d'Unité Linéaire Rectifiée (Rectified Linear unit ou ReLU en anglais) (Goodfellow *et al.*, 2016). Généralement assemblés par superposition de couches interconnectées de plusieurs neurones, les réseaux de neurones tirent leur expressivité du nombre de neurones utilisés pour réaliser l'approximation fonctionnelle entre X et y. Comme illustré dans la figure 3.5, où les neurones sont représentés par des cercles et les liens entre les couches par des traits, dans ces réseaux *denses* (ou feedforward) (Specht, 1991) les couches de neurones artificiels sont superposées et la sortie d'un neurone d'une couche est prise comme entrée de tous les neurones de la couche suivante. Les paramètres θ du réseau, i.e. les valeurs numériques des coefficients matriciels de chaque neurone, sont le plus souvent mises à jour par l'algorithme de descente de gradient stochastique (Goodfellow *et al.*, 2016) estimant le gradient de l'erreur par l'espérance d'un sous ensemble de données. Le gradient est estimé à partir de données et d'une métrique d'erreur préalablement choisie.

Les étapes d'apprentissage sont récapitulées dans l'algorithme 3. Après avoir initialisé les paramètres θ du réseau noté nn_θ, plusieurs itérations sur les données sont réalisées. Dans chaque

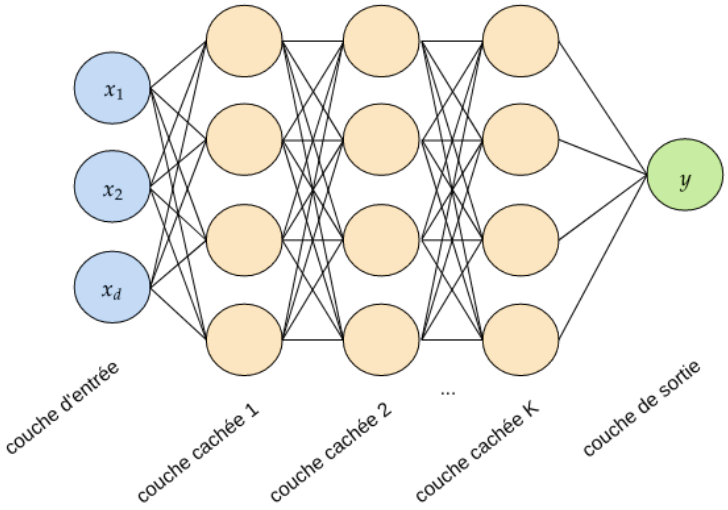


FIGURE 3.5 – Illustration d'un réseau de neurones profond à K=3 couches cachées. Un neurone artificiel est représenté par un cercle, les liaisons entre les sorties d'une couche et les entrées de la couche suivante sont symbolisées par des lignes. Le réseau présenté comprend 3 variables d'entrée et une variable de sortie. Chaque couche cachée comporte 4 neurones artificiels.

itération de l'algorithme, l'erreur sur les paramètres est estimée et utilisée pour ensuite mettre à jour les paramètres θ par rétro-propagation du gradient. Les lecteurs intéressés par plus de détails sur l'apprentissage par les réseaux de neurones pourront considérer la lecture du livre "Deep Learning" de I. Goodfellow, Y. Bengio et A. Courville (Goodfellow *et al.*, 2016).

L'utilisation des réseaux de neurones a été étudiée dans le cadre de la régression symbolique afin de rendre ces algorithmes neuronaux moins opaques. Les réseaux neuronaux ont été appliqués dans le but d'apprendre une expression analytique de la fonction construite par le réseau. Ce domaine dynamique se décompose aujourd'hui en trois catégories d'utilisation des réseaux de neurones : le codage de l'expression dans la structure du réseau, l'identification de simplifications et la génération d'expressions.

Codage dans la structure du réseau

Afin de tirer parti de la capacité d'apprentissage et l'expressivité des réseaux de neurones, une première approche nommée Equation Learner (EQL) (Martius et Lampert, 2017), propose d'apprendre un réseau neuronal feedforward dans lequel les fonctions d'activation classiques sont remplacées par des opérations telles que les fonctions sinus, cosinus, addition ou encore la fonction identité. Dans chaque couche, plusieurs activations différentes peuvent être utilisées. L'analyse de l'architecture du réseau permet, après apprentissage, de déduire une expression symbolique reliant X et y. Le choix des opérations à inclure comme fonctions d'activation joue néanmoins un rôle majeur la capacité l'apprentissage du réseau, notamment car l'utilisation de fonctions d'activation exponentielles et logarithmiques peut entraîner l'explosion des gradients, ainsi que des problèmes numériques. Pour faire face à ces problèmes, l'architecture du réseau d'EQL a été améliorée dans une nouvelle architecture EQL$^{\div}$ capable de prendre en compte des opérations de division dans les fonctions d'activation et de stabiliser l'apprentissage (Sahoo *et al.*, 2018).

Cette approche a par la suite été étendue à l'entraînement de réseaux de bout-en-bout (Kim *et al.*, 2021), c'est-à-dire en réalisant plusieurs tâches en plusieurs blocs d'un même réseau et

en apprenant les paramètres de toutes les étapes d'apprentissage de manière conjointe. L'architecture EQL a également été appliquée à des problèmes tel que la prédiction de vitesse du vent (Abdellaoui et Mehrkanoon, 2021).

Cette architecture a été également étendue en appliquant une relaxation continue à la structure du réseau par le biais de la re-paramétrisation de Gumbel-Max (Kingma et Welling, 2014). Dans ce réseau, nommé GMEQM (Chen, 2020), chaque entrée d'un noeud de fonction élémentaire est connectée à la sortie d'un autre noeud de la couche précédente selon une connexion stochastique.

Diviser pour régner

Une seconde catégorie d'approches, basées sur les réseaux de neurones, consiste à identifier les propriétés physiques respectées par la fonction symbolique f (telles que la symétrie ou la séparabilité) à l'aide d'un réseau neuronal. Les propriétés sont ensuite utilisées pour diviser récursivement le problème global de RS par un algorithme divide-and-conquer en sous-problèmes plus facilement solubles. La méthode, nommée AI Feynman (Udrescu et Tegmark, 2020) étendue en AI Feynman 2.0 (Udrescu *et al.*, 2020), applique pour cela de manière récursive différentes stratégies (analyse des dimensions physiques, tests basés sur des réseaux de neurones) jusqu'à obtenir des sous-problèmes suffisamment simples pour être résolus par des régressions polynomiales, ou une estimation "brute force" testant toutes les expressions symboliques d'un ensemble donné. Dans sa version 2.0 (Udrescu et Tegmark, 2020; Udrescu *et al.*, 2020), les gradients des réseaux de neurones sont utilisés pour identifier des propriétés de sub-division du problème global d'identification de f et de nouvelles propriétés sont étudiées. Des tests statistiques d'hypothèses sont également réalisés pour valider les propriétés. Notons que l'approche présentée ici peut être vue comme une forme d'intégration de la connaissance du domaine de la physique dans l'étude du problème de RS. L'algorithme AI Feynman (2.0) a par ailleurs été intégré dans des travaux cherchant à identifier des lois de conservation à partir des trajectoires de systèmes dynamiques (Liu *et al.*, 2022).

Génération d'expressions par Réseaux de Neurones

Enfin, une dernière catégorie d'approches propose d'utiliser les réseaux de neurones pour générer des expressions symboliques en utilisant des architectures de réseaux de neurones issues du domaine du traitement automatique des langues. Dans ces travaux, les réseaux sont entraînés sur des jeux de données $\mathscr{D} = \{\mathbf{y}_i, f_i\}_{i\in I}$, où pour chaque fonction f_i, M points $\mathbf{y}_i = (y_i^m)_{m\in\{1..M\}}$ sont obtenus par évaluation de f_i sur des données échantillonnées $X_i = (x_i^m)_{m\in\{1..M\}}$ sur un intervalle choisi.

Dans cette approche, les réseaux sont utilisés pour encoder une paire de données (X_i, y_i), puis les décoder sous la forme d'une expression symbolique textuelle f_i composée d'une séquence de symboles. Ces réseaux encodeurs-décodeurs prenant des séquences de données en entrée et en sortie (Seq2Seq) considèrent différentes catégories d'architectures neuronales. Dans des premiers travaux (Biggio *et al.*, 2020), un réseau convolutionnel de type fully-convolutional (Gehring *et al.*, 2017) est utilisée pour les étapes d'encodage et de décodage. Les travaux plus récents (Biggio *et al.*, 2021; d'Ascoli *et al.*, 2022; Valipour *et al.*, 2021) emploient des réseaux basés sur les Transformers par Generative Pre-Training (GPT)(Radford *et al.*, 2018) ou par les Set Transformers (Lee *et al.*, 2019) pour l'encodage/décodage. Les Transformers utilisent un mécanisme d'Attention (Vaswani *et al.*, 2017) qui a la particularité d'analyser une séquence d'entrée complète, pour décider à chaque étape de l'encodage/décodage quelles parties de la séquence sont impor-

tantes. Cette approche Seq2Seq a trouvé des applications dans le domaine des mathématiques, notamment pour l'intégration de fonctions et la résolution d'équations différentielles (Lample et Charton, 2020), ou encore pour identifier des relations de récurrence dans des données (d'Ascoli *et al.*, 2022). Contrairement à l'approche Seq2Seq où l'apprentissage est réalisé par descente de gradient, les poids des connexions du réseau peuvent également être optimisés par apprentissage évolutionnaire comme dans l'approche *Neuro-Encoded Expression Programming* (Anjum *et al.*, 2019).

On trouve dans les approches décrites ci-dessus une première piste pour utiliser les réseaux de neurones comme *algorithme calculatoire appris, pour générer des modèles symboliques intelligibles*. Contrairement à la PG, l'approche Seq2Seq présente aussi l'avantage de pouvoir s'améliorer au fil de l'apprentissage de plusieurs fonctions et ainsi d'apprendre à généraliser. Elle permet alors de réutiliser un même réseau, une fois appris, sur de nouvelles données. Cependant, la *découverte d'équations* sur des données non-rencontrées pendant l'entraînement ne peut donc se faire qu'après apprentissage et suppose par ailleurs de disposer de paires de données entrées-sorties suffisantes et prises dans une distribution adaptée pour entraîner le réseau de manière généralisable. L'accès à des données conditionne donc l'utilisabiltié de telles méthodes. Dans les scénarios applicatifs, comme le nôtre, dans lequel les équations ne sont pas connues exactement a priori, ces méthodes ne sont pas directement utilisables.

Parmi ces trois catégories d'approches de RS, aucune d'entre elles n'est aujourd'hui appliquée à des ensembles fonctionnels grammaticaux. Pour apprendre à générer des expressions symboliques dans un espace grammatical avec un réseau neuronal, le réseau Grammar Variational AutoEncoder (GVAE) (Kusner *et al.*, 2017) encode et décode des arbres syntaxiques décrits par une grammaire et où le décodage se fait par échantillonnage d'une représentation latente variationnelle (Kingma et Welling, 2014). La recherche dans l'espace latent grammatical peut être ensuite réalisée, par exemple, par un algorithme évolutionnaire (Lynch *et al.*, 2020). L'intérêt de l'utilisation conjointe d'une grammaire et d'un réseau de neurones pour la génération d'expressions symboliques est cependant étudié et prouve son utilité dans le domaine linguistique sous le terme Recurrent Neural Network Grammars (RNNG) (Dyer *et al.*, 2016).

3.3.4 Apprentissage par renforcement

Définitions et application à la régression symbolique

Dans le contexte de l'apprentissage automatique interprétable, les domaines de la Programmation Génétique (PG) et de l'Apprentissage par Renforcement (AR) sont aujourd'hui fortement liés. Par exemple, la PG a déjà trouvé des applications pour la génération de politiques de renforcement interprétables (Hein *et al.*, 2018; Videau *et al.*, 2022), ou pour estimer une fonction de valeur (Kubalík *et al.*, 2021). En RS, l'exploration de l'espace fonctionnel $\mathcal{F}$ a également été formalisée par des approches d'AR. Contrairement aux approches présentées précédemment qui explorent l'espace fonctionnel $\mathcal{F}$ par générations évolutionnaires (sous-section 3.3.2), ou qui apprennent en RS un modèle fonctionnel des relations entre des données X et y supervisé par l'erreur de prédiction de y (sous-section 3.3.3), l'AR propose de résoudre la RS en formulant le problème d'exploration de $\mathcal{F}$ par un processus de prise de décision séquentiel Markovien, dans lequel un agent suit une succession d'états distincts. Nous nous appuierons sur le livre de Sutton et Barto (Sutton et Barto, 2018) pour la description de cette catégorie d'algorithmes dans les prochains paragraphes. Nous fournissons ici un rappel sur l'apprentissage par renforcement dans le cadre général, en identifiant les spécificités de l'AR pour la régression symbolique.

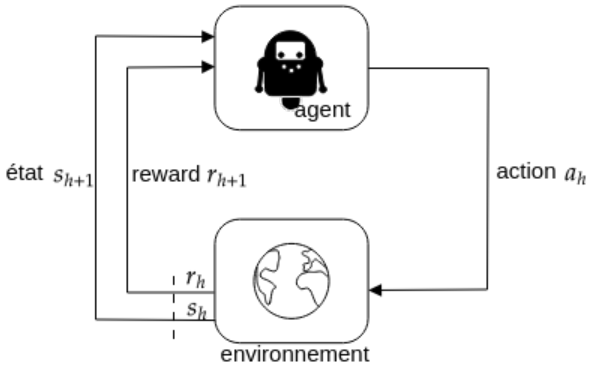


FIGURE 3.6 – Schéma, inspiré du livre de Sutton et Barto (Sutton et Barto, 2018), illustrant l'interaction entre un agent et son environnement. A un instant h, l'agent dans un état s_h choisi une action a_h. Une fois cette action réalisée dans son environnement, i.e. à l'instant $h+1$, l'agent passe dans un nouvel état s_{h+1} et reçoit un signal de récompense r_{h+1}.

L'AR vise à apprendre une politique de comportement d'un agent π, optimale pour la réalisation d'une tâche dans son *environnement*, sous la forme d'un problème de décision séquentiel. Plus précisément, ce problème est formalisé par un processus de décision de Markov (Sutton et Barto, 2018) (en anglais Markov Decision Process, noté MDP) définit par un tuple $(\mathcal{S}, \mathcal{A}, \mathcal{P}, r)$ avec :

— l'espace d'états de l'environnement $\mathcal{S}$,

— l'espace des actions possibles $\mathcal{A}$. Il peut être discret ou continu.

— la fonction relative à la probabilité de transition d'un état à un autre $\mathcal{P}$. Cette probabilité est notée $p(s'|s,a)$ et la transition d'un état s à un nouvel état s' par la réalisation d'une action a.

— la fonction de récompense (ou *reward*) r. La récompense associée à la réalisation d'une action a dans un état s est notée $r(s,a) \in \mathbb{R}$.

Avec ces définitions nous pouvons maintenant décrire les interactions entre ces différents éléments, tel qu'illustrés dans la figure 3.6. A chaque instant h d'une séquence appelée *trajectoire*, l'agent obtient une représentation de l'*état* de son environnement $s_h \in \mathcal{S}$ tel qu'il le perçoit et choisit une *action* a_h à partir de cette représentation. Une fois l'action réalisée, i.e. à l'instant $h+1$, l'agent reçoit un nouvel état s_{h+1}. L'objectif de l'agent est donc la maximisation de la récompense cumulée sur toute la trajectoire $R_h = \sum_{k=0}^{H} \gamma^k r_{h+k}$. Le terme $\gamma \in [0,1]$ est le facteur d'actualisation qui détermine le poids des récompenses futures. Dans le cas de la RS, la valeur $\gamma = 1$ est souvent prise par défaut, car une expression symbolique est difficilement évaluable avant la fin de la trajectoire et la récompense la plus importante est généralement celle obtenue lors de la complétion de la construction de l'expression symbolique. Pour la RS, nous nous intéresserons aux MDPs *épisodiques* considérant un horizon H fini et des trajectoires finies. A chaque instant $h \in \{0..H\}$ de cet horizon, une action a_h est prise réalisant la transition entre un état s_h et un nouvel état s_{h+1}, avec $(s_h, s_{h+1}) \in \mathcal{S}^2$.

L'apprentissage du MDP peut être réalisé par apprentissage direct d'une politique $\pi(a|s)$ ou par le biais d'une fonction d'estimation. La fonction d'estimation peut faire référence à une *fonction de valeur d'état* $V_\pi(s) = \mathbb{E}_\pi[R_h|s_h = s]$ qui estime l'intérêt d'un état, ou bien une *fonction de valeur état-action* $Q_\pi(s,a) = \mathbb{E}_\pi[R_h|s_h = s, a_h = a]$ qui estime la récompense sur le long terme, relative au choix d'une action a dans un état s en suivant une politique. Lorsqu'une fonction de

valeur est utilisée, l'action est sélectionnée à partir de la valeur de cette fonction. L'apprentissage d'une politique optimale π^* passe donc par le choix des actions supposées optimales par rapport aux données observées jusqu'à présent, tout en continuant à chercher de meilleures actions pour atteindre π^*. On parle, dans ce cas, de compromis entre exploration et exploitation. L'apprentissage peut être réalisé avec des méthodes tabulaires (Rummery et Niranjan, 1994; Watkins, 1989) où la fonction est exprimée dans un tableau indexé par des états. Lorsque le tableau est trop grand pour être géré directement, la politique et la fonction de valeur peuvent être approximées, par exemple par un réseau de neurones (Mnih *et al.*, 2015). Dans le cadre de la RS, où la taille de $\mathcal{F}$ croit exponentiellement avec H, une approche par approximation peut être envisagée.

Processus de Decision Markovien d'actions symboliques

En AR, les algorithmes considérés produisent généralement des politiques et des fonctions de valeurs complexes et peu interprétables. Tout comme dans les autres domaines de l'apprentissage automatique, l'explicabilité joue un rôle important dans l'acceptation et la confiance dans les modèles construits et a donc été étudiée. L'une des divisions parmi les méthodes d'apprentissage interprétable est faite entre les méthodes interprétables de manière *intrinsèque* ou *a posteriori*. Pour la régression symbolique, nous nous intéresserons plus particulièrement à la catégorie des méthodes de renforcement intrinsèquement explicables et dans lesquels l'élément fournissant l'explication est généré à partir d'un espace fonctionnel $\mathcal{F}$. Depuis plusieurs années, le renforcement est considéré comme une méthode de résolution adéquate pour le choix des noeuds/feuilles (sous forme d'actions) générant des arbres en PG (Downing, 2001).

Dans le cadre plus général de la construction de représentations symboliques, différents MDPs sont définis dans la littérature. Chacun de ces MDPs permet de représenter des contraintes et une forme de connaissances différente. En RS, l'espace fonctionnel $\mathcal{F}$ utilisé pour définir l'espace d'actions $\mathcal{A}$. Par exemple, lorsque $\mathcal{F}$ est défini par un langage $\mathcal{L}$ comportant un ensemble d'opérations et de variables, l'espace d'actions peut être défini par l'ensemble des symboles d'un langage $\mathcal{L}$ (Petersen *et al.*, 2021; Landajuela *et al.*, 2021). Pour générer des solutions symboliques, d'autres formats que la librairie d'opérations sont explorés telles que des *options symboliques* (Jin *et al.*, 2022), des règles IF-THEN (Huang *et al.*, 2020) ou des prédicats (Ma *et al.*, 2021).

Les grammaires non-contextuelles sont également utilisées. Dans ce cas, l'espace d'actions $\mathcal{A}$ peut être défini directement à partir de la grammaire G et une action correspond au choix d'une règle dans la grammaire. Celles-ci ne sont pas restreintes à une application à la régression symbolique. Comme décrit précédemment dans la sous-section 3.2.3, elles ont été utilisées pour la synthèse de programmes (Verma *et al.*, 2018), la construction de pipelines d'AutoML (Drori *et al.*, 2019) ou encore pour la génération d'abris (Mandow *et al.*, 2020). Ce type d'espace d'action $\mathcal{A}$ présente une caractéristique particulière. En effet, bien que $\mathcal{A}$ soit borné, toutes les actions ne sont pas systématiquement accessibles à chaque étape h. En effet, lors de la construction d'une expression (voir sous-section 3.2.3), pour une étape h, un état s_h est considéré et il contient l'information d'un symbole. Ce symbole est associé à une production de G et l'agent ne peut que choisir des actions (i.e. des règles de production) dans cette production.

Algorithmes de recherche

Différents algorithmes d'AR ont été considérés dans la littérature à la fois basés sur les fonctions de valeurs tels que Deep Q-Learning (Mnih *et al.*, 2015), ou des méthodes policy gradients tel que REINFORCE (Williams, 1992) ou Proximal Policy Optimization (Schulman *et al.*, 2017).

Contrairement aux autres approches d'AR qui cherchent à avoir le meilleur comportement moyen, la régression symbolique a pour objectif principal de maximiser les performances de la meilleure des trajectoires. La stratégie pour réaliser cet objectif consiste donc à maximiser le quantile (1-ϵ) de la distribution des récompenses de la politique actuelle (Petersen *et al.*, 2021). L'objectif correspondant est noté :

$$J_{risque}(\theta, \epsilon) = \mathbb{E}_{\tau \sim \pi_\theta}\Big[(R(\tau|\theta) - R_\epsilon) \sum_{h=0}^{H} log\ \pi_\theta(a_h|s_h) \ \Big|\ R(\tau|\theta) > R_\epsilon\Big] \tag{3.10}$$

Cette approche a montré ses performances pour la régression symbolique dans le cadre d'application à un espace fonctionnel défini par une librairie d'opérations. Elle a par la suite été appliquée au dimensionnement de composants électriques dans les convertisseurs de puissance à topologie fixe (Glatt *et al.*, 2021) et la découverte de politiques symboliques (Landajuela *et al.*, 2021).

Lorsqu'une structure grammaticale est considérée, l'intérêt d'une grammaire probabiliste a été mis en évidence sur un problème de découverte d'équations en régression symbolique (Brence *et al.*, 2021). Bien que ces travaux montrent de meilleurs résultats sur des grammaires probabilistes que sur celles non-probabilistes, l'utilisation d'une approche Monte-Carlo pour échantillonner des expressions symboliques dans cet espace fonctionnel ne semble pas suffisant. En effet, dans le cas des grammaires probabilistes, les poids de la grammaire sont fixes. *Pour un symbole donné*, l'implication sous-jacente est donc que les probabilités d'occurrences des règles de productions de ce symbole sont fixes, quels que soient les autres symboles de sa trajectoire. L'AR permet d'améliorer cela en prenant en compte l'information des symboles passés qui peuvent être décrits directement dans l'état s et/ou par l'observation d'un MDP Partiellement Observable (Partially Observable Markov Decision Process ou POMDP) (Kaelbling *et al.*, 1998). L'utilisation d'une modélisation par POMDP est partagée par plusieurs algorithmes (Verma *et al.*, 2018; Petersen *et al.*, 2021; Landajuela *et al.*, 2021). Elle présente notamment l'intérêt de pouvoir étendre la représentation du problème à un modèle de Markov caché, correspondant à un processus markovien de paramètres inconnus dans lequel les états sont partiellement connus.

Regardons maintenant quelques exemples d'algorithmes utilisés sur des espaces d'action grammaticaux. En AutoML formalisé par une grammaire, une approche d'agent neuronal est combinée avec un algorithme de Monte-Carlo Tree search (MCTS) qui prend des décisions avisées basées sur les sorties du réseau (Drori *et al.*, 2019). D'autres travaux, proche de la RS pour la génération de *programmes symboliques*, l'apprentissage est réalisé par imitation d'un "oracle" neuronal. Plus précisément, la phase de mise à jour a pour objectif de minimiser la distance entre une politique et un oracle, avec l'intuition qu'une distance est un objectif plus simple qu'une fonction de récompense (Verma *et al.*, 2018).

Approches Hybrides PG/AR

Enfin, du fait de la complémentarité entre les approches PG et d'AR qui peuvent apprendre sur des individus/trajectoires partageant la même représentation, des travaux proposent aujourd'hui de combiner les deux approches. La fusion des deux algorithmes en un seul permet d'espérer améliorer l'exploration et l'exploitation. Une méthode d'AR peut, par exemple, servir à générer une la population initiale d'un algorithme de PG et ensuite mettre à jour l'agent à partir des meilleurs individus/trajectoires (Mundhenk *et al.*, 2021). Lors de l'apprentissage d'un agent, la génération de trajectoires peut être réalisée directement à partir de l'agent, ou bien à partir des enfants d'une génération donnée (après crossover et mutation) (Zhang et Zhou, 2021).

Après avoir passé en revue différentes approches pour la régression symbolique, nous pouvons maintenant nous intéresser à l'interactivité et l'intérêt de la prise en compte de connaissance de l'utilisateur pendant l'interaction.

3.3.5 Apprentissage par expansion de bases

Modèles linéaires généralisés

En dehors des approches par PG, réseaux de neurones et AR, d'autres catégories de méthodes sont également utilisées pour la résolution du problème de RS. Les approches par expansion de bases en font partie. Il s'agit d'approches situées à la frontière entre la régression symbolique et la régression car elles utilisent un a priori *partiel* sur la forme de la solution. L'une de ces catégories s'intéresse plus particulièrement à la combinaison d'une base d'opérations préalablement choisies dans une équation de forme prédéfinie. L'idée de cette approche est alors d'augmenter préalablement les variables d'entrées X par un ensemble d'autres variables, définies comme transformations de variables de X par un ensemble d'opérations choisies. Les modèles linéaires généralisés (Nelder et Wedderburn, 1972) (ou *Generalized Linear Models* en anglais, noté GLMs) utilisent ainsi une fonction f ayant pour forme une moyenne pondérée de variables de X transformées. Plus précisément, un GLM est une fonction f prise sous la forme d'une combinaison linéaire d'une base d'opérations non-linéaires $(B_i)_{i\in I}$ décrite de la manière suivante :

$$f(x) = \alpha_0 + \sum_{i\in I} \alpha_i \times B_i(x),\ \alpha_i \in \mathbb{R} \tag{3.11}$$

La base B_i est un paramètre du modèle. Des bases d'opérations classiques sont par exemple les bases polynomiales $(x_{d1}^{i1} \times x_{d2}^{i2})_{(i1,i2)\in I^2,(d1,d2)\in D^2}$, les bases de fonctions *hinge* $(\max(0, constante - x_d), \max(0, x - constante))_{d\in D}$ (Friedman, 1991), ou les bases de *kernels* (Vovk, 2013). L'obtention de la fonction f et des coefficients α peut ensuite se faire, par exemple, par une méthode de régression linéaire des moindres carrés (Hastie *et al.*, 2009). Il est également possible d'ajouter une contrainte de régularisation sur le modèle construit en ajoutant une contrainte quadratique (régression Ridge (Hoerl et Kennard, 1970)) ou une contrainte linéaire (régression Lasso (Zou *et al.*, 2007)) sur les coefficients $(\alpha_i)_{i\in I}$.

Fast Function Extraction

Parmi des algorithmes utilisés pour construire des GLMs, l'algorithme Fast Function Extraction (FFX) (McConaghy, 2011), utilisant une technique de *pathwise regularized learning* (Friedman *et al.*, 2010), propose de combiner plusieurs opérations d'une base donnée afin de déterminer lesquelles doivent être incluses dans la fonction f finale. Pour ce faire, l'algorithme se décompose en trois étapes :

1. Génération d'un ensemble d'opérations à un et deux arguments (tel que F = {x, abs(x), log(x), x × log(x), ...}) ;
2. Calcul des coefficients $(\alpha_i)_{i\in I}$ avec différentes valeurs de contraintes de régularisation. Plusieurs fonctions sont obtenues, chacune avec un nombre différent de coefficients α nuls ;
3. Réduction de l'ensemble des fonctions à un front de Pareto non-dominé qui offre un compromis entre la complexité de la fonction (estimé par le nombre d'opérations de la base B utilisées) et l'erreur de prédiction.

En plus de minimiser l'erreur quadratique entre le modèle et les données d'entraînement, FFX présente donc la particularité d'utiliser un mécanisme de régularisation par une formulation de

type Elastic Net (Zou et Hastie, 2005) pénalisant conjointement les coefficients $||\alpha||^2$ et $||\alpha||^1$.

Cette méthode présente plusieurs avantages, par exemple en comparaison avec les approches génétiques. Contrairement aux AGs, FFX est une méthode déterministe et présente donc l'avantage d'obtenir des fonctions facilement reproductibles. Par ailleurs, il s'agit d'un algorithme ayant un temps d'exécution court, capable de traiter des données en grande dimension et générant un ensemble de fonctions f de complexités différentes. Il est donc possible de choisir une fonction de complexité souhaitée. Grâce aux avantages mentionnés, FFX a par exemple été appliqué à de l'analyse de fiabilité de circuits analogiques (Maricau *et al.*, 2012), à la récupération d'équations différentielles partielles linéaires (équations d'ondes et de chaleur) et non linéaires d'ordre supérieur (équations de Burgers, de Korteweg-de Vries et de Kawahara) (Vaddireddy et San, 2019).

Cependant, du fait de la restriction sur la forme de la fonction, comme présentée dans l'équation 3.11, l'algorithme FFX n'est pas en mesure d'utiliser un espace fonctionnel suffisant pour construire un modèle d'estimation sur toutes les données. Par exemple, FFX ne comprend pas la prise en compte de combinaisons d'opérations utilisées consécutivement (par exemple $\frac{\sqrt{(x_1)^2+(x_2)^2}}{x_3}$). Les bases d'opérations de FFX ne sont pas suffisamment générales pour donner à FFX une capacité d'approximation universelle (McConaghy, 2011). L'approche par FFX et plus généralement par expansion de base, est donc moins expressive que celle par EG et ne permet notamment pas de prendre en compte des contraintes sur les unités des variables de X. Au contraire, l'utilisation d'une grammaire ne restreint pas systématiquement la forme de la solution, tout en permettant de formaliser des contraintes spécifiques au problème étudié et notamment les unités et combinaisons d'unités.

Modèles Additifs généralisés

Parmi les méthodes basées sur un modèle, un modèle plus général que celui précédemment présenté, est celui des *Modèles additifs généralisés* (ou Generalized Additive Models en anglais, noté GAM) (Hastie et Tibshirani, 1986). Le modèle GAM réalise la somme de fonctions prises dans une base $(B_i)_{i\in I}$:

$$f(x) = \alpha + \sum_{i\in I} B_i(X_i) \tag{3.12}$$

L'objectif des GAM est de déduire les fonctions issues d'une base dont la composition agrégée se rapproche de la variable à prédire y. Les méthode de résolutions principales des modèles GAM sont l'algorithme de backfitting (Friedman et Stuetzle, 1981).

Notons que les modèles GAM, tout comme les GLM (Generalized Linear Models) et l'algorithme FFX ne permettent pas d'inclure de la connaissance sous forme grammaticale.

3.3.6 Synthèse sur les algorithmes de régression symbolique

Les approches présentées dans la section 3.3 permettent de résoudre le problème de régression symbolique défini dans la sous-section 3.3.1. Cependant, toutes ne portent pas les caractéristiques nécessaires pour répondre aux contraintes industrielles de cette thèse. En effet, afin d'obtenir des expressions utilisables par les experts, nous souhaitons pouvoir prendre en compte la connaissance du domaine tel que de la physique du problème étudié, ainsi que de la connaissance structurée, par exemple, représentée sous forme de règles d'une grammaire.

Le tableau 3.3 propose une synthèse des principales méthodes détaillées précédemment. Nous utilisons ici comme références la régression linéaire ainsi que les arbres de décision, déjà employés

Méthode	Interactions	Opérations choisies	Physique	Grammaire
Régression Linéaire (Zou *et al.*, 2007)	×	×	×	×
Arbre de décision (Breiman *et al.*, 1984)	✓	×	×	×
PG (Koza, 1990)	✓	✓	×	×
EG (O'Neill et Ryan, 2001)	✓	✓	✓	✓
GB-LGP (Sotto et de Melo, 2017)	✓	✓	✓	✓
$EQL^{\%}$ (Sahoo *et al.*, 2018)	✓	✓	×	×
Seq2Seq (Lample et Charton, 2020)	✓	✓	×	×
DSR (Petersen *et al.*, 2021)	✓	✓	×	×
AI Feynman (Udrescu *et al.*, 2020)	✓	✓	✓	×
FFX (McConaghy, 2011)	✓	✓	×	×

TABLE 3.3 – Résumé des principaux algorithmes de régression symbolique présentés dans ce chapitre.

dans le tableau 3.2. Dans un premier temps, nous pouvons remarquer que toutes les méthodes listées dans ce tableau, à l'exception de la régression linéaire, permettent de prendre en compte des interactions entre variables. Il s'agit ici d'une caractéristique nécessaire pour représenter des relations plus complexes que des relations linéaires, telles que, par exemple, celles généralement présentes dans les équations de la physique.

Néanmoins, cette caractéristique seule n'est pas suffisante pour construire des expressions cohérentes avec le domaine d'expertise. En particulier, les méthodes de RS énumérées ici se distinguent par la possibilité (ou non) d'intégrer de la connaissance dans l'algorithme. Seules trois méthodes permettent d'intégrer de la connaissance sur la physique du domaine : deux méthodes employant une grammaire (EG et GB-LGP) ainsi qu'une méthode cherchant à décomposer le problème en sous-problèmes plus simples selon des propriétés (AI Feynman). De plus, l'emploi d'une grammaire présente un intérêt particulier dans notre cas d'application car elle permet de représenter d'autres connaissances plus générales que celles de la physique, telles que de l'expertise métier. Ces deux méthodes employant une grammaire semblent donc particulièrement adaptées à nos contraintes et seront l'objet d'études dans les chapitres suivants.

3.4 Interactivité

3.4.1 Pourquoi mettre l'humain dans la boucle d'apprentissage ?

Différence entre apprentissage "passif" et "interactif"

En apprentissage automatique classique "passif", le modèle est généralement appris de manière automatique avec peu, ou sans, intervention humaine dans la phase d'apprentissage (Holzinger *et al.*, 2019). Bien que l'humain soit encore présent pour pré-traiter les données, ou choisir les meilleurs paramètres d'apprentissage, des domaines de recherche proposent aujourd'hui de s'affranchir de la présence de l'humain. Par exemple, en apprentissage automatique automatisé (en anglais automated ML, autoML), des librairies telles que Auto-WEKA (Kotthoff *et al.*, 2019), ou auto-sklearn (Feurer *et al.*, 2015) permettent de réaliser l'apprentissage de bout-en-bout de manière automatique. Les paramètres optimaux sont choisis automatiquement, ce qui rend les algorithmes de ces librairies accessibles à des personnes non initiées à l'apprentissage automatique. Une faible présence de l'humain est acceptable pour de nombreux algorithmes et catégories d'applications dans lesquels les informations contenues dans les données d'apprentissage sont suf-

fisantes pour construire un modèle avec un niveau de précision satisfaisant (Russakovsky *et al.*, 2015).

En pratique, le besoin de mettre l'humain dans la boucle est souvent sous-estimé (Doan, 2018) et les approches non-interactives présentent des désavantages (Holzinger *et al.*, 2019). Elles requièrent d'importantes ressources informatiques, des jeux de données de grandes tailles et la plupart d'entre elles sont considérées comme des boites noires complexes, difficiles à comprendre et à qui il est difficile de faire confiance. Sur le plan des données, il existe notamment des situations dans lesquelles la quantité de données à disposition n'est pas suffisante pour satisfaire les critères de précisions requis. Par exemple, en apprentissage semi-supervisé seule une partie des données est étiquetée et, lorsque le nombre d'étiquettes n'est pas suffisant, l'intervention d'un humain permet d'améliorer l'apprentissage avec seulement quelques étiquettes supplémentaires correctement choisies et fournies par l'humain (Settles, 2009). L'intervention d'un humain peut être également bénéfique d'un point de vue de la connaissance à partir de laquelle l'algorithme apprend. Comme nous l'avons décrit dans la section 3.2, la connaissance peut être divisée en connaissances tacites et explicites. Trouver la bonne manière de représenter la connaissance explicite est une tâche ardue, en particulier dans des domaines techniques pointus. Ici intervient l'intérêt de l'interaction, pour donner l'opportunité à un utilisateur d'enrichir et mettre à jour cette connaissance explicite (von Rueden *et al.*, 2021). Sur le plan de la connaissance tacite, l'interactivité d'un utilisateur avec un algorithme est un moyen d'éliciter de la connaissance implicite (Kerrigan *et al.*, 2021), difficilement accessible sans interaction. Enfin, comme mentionné plus haut, l'un des obstacles à l'utilisation des algorithmes d'apprentissages "classiques" est la complexité des résultats proposés par les algorithmes à leurs utilisateurs. Même lorsque ces algorithmes sont performants d'un point de vue des métriques d'apprentissage, le manque de compréhension par l'humain du processus d'obtention de ce résultat peut engendrer une méfiance dans les résultats proposés. Face à ce défi, l'interaction entre l'humain et l'algorithme permet d'améliorer sa confiance dans l'algorithme et ses résultats (Gutzwiller et Reeder, 2017). Ce dernier point est, par ailleurs, lié à l'explicabilité et l'interprétabilité des modèles créés comme présenté dans la sous-section 3.2.5.

Définition et objectifs

L'implication de l'humain dans le processus d'apprentissage peut prendre différentes formes (Robert *et al.*, 2016) : soit en utilisant l'humain pour remplacer l'algorithme (i.e l'humain générant le modèle directement à partir des données), soit en plaçant l'humain dans la boucle d'entraînement-évaluation (i.e. l'humain donnant des feedbacks à l'algorithme), soit en faisant travailler l'humain et l'algorithme main dans la main pour créer le modèle. Cette dernière forme d'interaction plus collaborative, est celle la plus fréquemment étudiée et adoptée aujourd'hui.

Plus généralement, les approches combinant humain et algorithme d'apprentissage sont regroupées sous le terme d'apprentissage automatique interactif (en anglais Interactive Machine Learning, iML). L'apprentissage automatique interactif (iML) est défini (Dudley et Kristensson, 2018) comme un paradigme d'interaction dans lequel un utilisateur ou un groupe d'utilisateurs construit et affine de manière itérative un modèle mathématique pour décrire un concept à travers des cycles d'entraînement et de révision. L'amélioration du modèle s'appuie sur les connaissances de l'utilisateur transmises à l'algorithme, sous une forme différente de celle des données d'apprentissage : tacite (équations algébriques, règles logiques, ..) et/ou implicite (autres commentaires humains). Dans l'iML, les itérations sont plus rapides, ciblées et incrémentales que dans l'apprentissage automatique traditionnel. L'interactivité permet ici une meilleure compréhension des résultats de l'algorithme car l'utilisateur a accès à plus d'informations, ce qui améliore finalement

la confiance dans le résultat proposé. L'inclusion de l'humain dans la boucle d'apprentissage a principalement pour but (Monarch, 2021) :

— d'améliorer la performance des modèles appris ;

— d'atteindre la performance algorithmique souhaitée plus rapidement ;

— de combiner l'humain et l'"intelligence" de la machine pour maximiser la performance ;

— d'assister l'humain dans les tâches qu'il doit réaliser pour améliorer son efficacité.

En plus des buts citées précédemment, d'autres objectifs pourraient être rajoutés à la liste définie par R. M. Monarch, tel que :

— d'améliorer la compréhension et la confiance dans les algorithmes ;

— de ne pas perdre en compétence à cause de l'automatisation de tâches, aussi appelé deskilling (Sambasivan et Veeraraghavan, 2022) ;

— de permettre à l'humain d'apprendre et de s'améliorer grâce à l'algorithme.

Ce dernier élément, encore peu pris en compte dans les algorithmes d'iML, a pour objectif d'obtenir une *intelligence hybride* (Dellermann *et al.*, 2019b; Dellermann *et al.*, 2019a). En effet, en apprentissage interactif les utilisateurs sont souvent utilisés comme des "enseignants" qui permettent à l'algorithme d'apprendre. Or, l'utilisation des algorithmes pourrait également être bénéfique aux humains. Ce concept vise donc à obtenir une interaction équilibrée, en utilisant les complémentarités de l'intelligence humaine (plutôt intuitive) et de la machine (plutôt analytique), pour obtenir des performances supérieures à celles que chacune des deux obtiendrait séparément. Ce concept définit l'humain et la machine comme deux systèmes coévoluant au fil de leurs interactions. Il s'agit ici d'un mécanisme d'interaction équilibré dans lequel l'algorithme et l'humain apprennent mutuellement l'un de l'autre et tirent tous les deux des bénéfices de leurs interactions.

3.4.2 Stratégies d'interaction

Recommandations générales

Différentes stratégies d'apprentissage interactif existent pour optimiser la manière dont l'utilisateur et l'algorithme interagissent. En effet, les algorithmes interactifs et les interactions qu'ils comprennent peuvent être très variés et mettre en place les bons mécanismes d'interaction n'est pas facile. Par exemple, des travaux ont identifié que l'interaction peut réduire la confiance de l'utilisateur dans le système qu'ils utilisent (Honeycutt *et al.*, 2020). Différents travaux proposent donc des recommandations générales à prendre en compte pour construire l'interaction entre humain et machine. Celles-ci portent principalement sur les caractéristiques de l'utilisateur et de la machine, ainsi que sur le choix de l'interaction.

Pour être performant, plusieurs caractéristiques communes aux utilisateurs doivent être prises en compte (Amershi *et al.*, 2014). Tout d'abord, les utilisateurs sont des humains, non des oracles, et l'exécution de tâches répétitives comme dire ce qui est bien ou mal à l'ordinateur pourrait être perçue comme ennuyeuse. Les utilisateurs veulent également démontrer comment l'algorithme devrait se comporter, par exemple en proposant des trajectoires d'actions à un algorithme de renforcement. En somme, les utilisateurs finaux veulent naturellement fournir plus que des étiquettes et pourraient faire des suggestions de différentes manières (par exemple, en suggérant de nouvelles fonctionnalités ou en ajustant leur importance). Ils devraient disposer d'autant de types d'interaction que possible. Les utilisateurs sont également fortement sensibles à leurs premières impressions du système avec lequel ils interagissent (Tolmeijer *et al.*, 2021). Ceci

influence donc leur perception de l'efficacité de l'algorithme et peut diminuer leur implication dans l'apprentissage. Cependant, bien que les premières interactions jouent un rôle important, l'effort requis par l'utilisateur se réduit considérablement à mesure que le modèle apprend, ce qui permet de réduire la charge de travail générée par l'interaction au fil du temps (Wallace *et al.*, 2012).

La prise de compte de ces caractéristiques humaines n'est possible que si les utilisateurs ont à leur disposition une interface soigneusement conçue. Il est donc nécessaire de disposer des bons outils pour construire l'interface appropriée. MARCELLE (Françoise *et al.*, 2021) est l'un de ces outils. Il propose une approche modulaire par composant permettant de séparer l'intégration de données, le calcul et l'interaction pour construire un processus interactif optimal. Différents modules peuvent être choisis pour gérer les sources et structures de données, les modèles et leur visualisation. Ce workflow et cette interface ont notamment été appliqués à la création d'une interface pour la détection de cancers de la peau en médecine.

Enfin, Dudley et Kristensson (Dudley et Kristensson, 2018) résument en six principes les objectifs que doit remplir un algorithme d'iML efficace :

— *Rendre les objectifs et les contraintes explicites* pour aider les utilisateurs à concentrer leurs efforts sur l'objectif et à améliorer la cohérence de leurs interactions.

— *Aider les utilisateurs à comprendre l'incertitude et la confiance des modèles* car les modèles probabilistes (contrairement aux données brutes) ne sont pas directement compréhensibles par les utilisateurs experts d'un autre domaine que l'apprentissage automatique.

— *Saisir l'intention plutôt que l'entrée* fournie par l'utilisateur car ces deux informations peuvent-être différentes et la prise en compte de cette distinction peut permettre d'améliorer l'apprentissage (Krening *et al.*, 2017).

— *Fournir une représentation efficace des données* pour utiliser au mieux les capacités de perception humaine. Ce principe passe donc par le choix d'une visualisation adaptée.

— *Exploiter l'interactivité et promouvoir des interactions diversifiées* pour permettre à l'utilisateur d'exprimer pleinement leur intention et d'appliquer leurs connaissances. Ce principe aide la compréhension de l'utilisateur.

— *Impliquer l'utilisateur* pour que celui-ci reste motivé.

Ces éléments décrits ci-dessus détaillent les caractéristiques d'un point de vue de l'utilisateur et de la machine. Nous allons maintenant regarder différentes stratégies d'interaction communément employées en iML. Bien que leur définition initiale ne les restreigne pas systématiquement à l'iML, nous les étudierons dans ce cadre. Nous nous intéressons plus particulièrement à l'Active Learning (AL), le Machine Teaching (MT) et le Curriculum Learning (CL).

Certaines stratégies peuvent être différenciées selon qui de l'utilisateur ou de l'algorithme initie l'interaction (Mosqueira-Rey *et al.*, 2021). En Active Learning, par exemple, l'algorithme réalise une requête auprès de l'oracle (l'utilisateur) pour obtenir des données étiquetées. Au contraire en Machine Teaching, l'utilisateur définit les concepts qu'il veut enseigner et les instances du jeu de données qui illustrent ce concept.

Active Learning

La stratégie d'Active Learning en iML considère les *humains* comme des oracles (Settles, 2009). Dans ce paradigme, l'algorithme d'apprentissage dispose d'un ensemble de données non-étiquetées et peut ensuite interactivement interroger un utilisateur utilisé comme oracle pour étiqueter n'importe quelle instance. L'AL a donc besoin d'algorithmes capables de sélectionner

de manière dynamique des exemples à étiqueter de sorte que la collecte et l'étiquetage de ces instances conduisent à un modèle ML de meilleure qualité.

L'AL est notamment utilisé en apprentissage semi-supervisé, où l'apprentissage est réalisé à partir de données partiellement étiquetées. L'Active Learning présente également un intérêt pour les scénarios d'apprentissage dans lesquels chaque supervision a un coût non-négligeable.

Cependant, les humains ne sont pas que des oracles (Amershi *et al.*, 2014) et ils peuvent fournir des connaissances au-delà de simples étiquettes de classes. Ceci ouvre donc aujourd'hui la possibilité de proposer à l'oracle d'autres interactions plus riches et plus informatiques telles que l'étiquetage de paires (Xu *et al.*, 2017). Cette approche possède de multiples applications lorsque l'obtention d'étiquettes directe est plus difficile que les comparaisons par paires.

Machine Teaching

Une autre stratégie d'apprentissage possible est le Machine Teaching (MT) (Zhu, 2015). Contrairement à l'apprentissage automatique traditionnel, qui fait apprendre des algorithmes par lot de données, le MT étudie un nouveau paradigme dans lequel l'étudiant (un algorithme) apprend grâce à un *enseignant* (un humain en iML) qui lui fournit des exemples de manière séquentielle et judicieusement choisis en fonction des performances actuelles de l'étudiant. Cette stratégie d'apprentissage est donc plus focalisée sur l'"enseignant". Le MT s'intéresse au problème du point de vue d'un enseignant souhaitant construire un jeu de données optimal (généralement de taille minimale) pour décrire un concept clé, de telle façon à ce qu'un étudiant puisse apprendre ce concept à partir des données sélectionnées.

Contrairement à l'Active Learning, le MT suppose que le où les enseignants humains contrôlent le processus d'apprentissage en délimitant les connaissances qu'ils ont l'intention de transférer au modèle d'apprentissage automatique. Dans le contexte du MT, un utilisateur qui agit en tant qu'enseignant suit alors le plan d'apprentissage suivant (Ramos *et al.*, 2020) :

1. *Planifier (et mettre à jour) le programme d'enseignement.* L'enseignant décide comment résoudre un problème et comment enseigner. Il choisit les exemples à utiliser pour l'enseignement, quand et comment évaluer l'apprentissage par rapport à l'objectif, quand et comment corriger/mettre à jour le modèle, décomposer ou fusionner des concepts, des variables, etc.
2. *Expliquer les connaissances pertinentes pour le domaine concerné* en utilisant les concepts et les exemples précédemment sélectionnés.
3. *Vérifiez les progrès de l'étudiant pendant qu'il assimile la connaissance donnée.* L'enseignement prend la forme d'un échange d'informations et la prochaine action de l'enseignant est influencée par l'état actuel de l'étudiant.

Le MT a par ailleurs été étudié dans le cas plus spécifique de l'apprentissage en ligne, ou *online* (Tegen *et al.*, 2020), dans lequel les instances de données arrivent par séquences. Dans les travaux de Tegen et al. différents scénarios en ligne on été étudiés sur des données réelles, à la fois avec l'AL et le MT. Les résultats montrent que les stratégies les plus performantes, en particulier au début de l'apprentissage, sont le MT déclenchée par une erreur (où le choix de fournir ou non une étiquette est basée sur l'erreur du modèle) et le MT déclenchée par un changement d'état (où l'utilisateur fournit une étiquette lorsque la classe de l'étiquette actuelle change).

Le principe du MT peut être étendu sous forme d'un processus itératif, appelé *Iterative Machine Teaching (Liu* et al.*, 2017)*, permettant d'apprendre sous la forme d'un dialogue et dans lequel l'enseignant peut observer et influencer l'étudiant par le choix d'instances d'apprentissage.

Curriculum Learning

Le *curriculum Learning* (Bengio *et al.*, 2009) est une stratégie d'apprentissage générale qui encourage l'utilisation d'exemples du jeu de données, des plus faciles aux plus difficiles. Cette stratégie, initialement utilisée en apprentissage supervisé (Bengio *et al.*, 2009), est également largement appliquée en apprentissage par renforcement (Narvekar *et al.*, 2020).

Enfin, bien que ces différentes stratégies soient initialement définies par des concepts distincts, elles sont permissives et il est possible de combiner plusieurs d'entre-elles pour construire de nouvelles stratégies.

3.4.3 Algorithmes interactifs

Dans le domaine de la RS, il est possible d'interagir avec les différents algorithmes présentés dans la section 3.3. Les interactions n'étant pas systématiquement réalisées avec des algorithmes utilisant des connaissances grammaticales, nous présentons ici plus généralement comment peut être réalisée l'interactivité pour chaque catégorie d'algorithme.

Apprentissage Génétique Interactif

Parmi les algorithmes évolutionnaires, l'interactivité est regroupée sous le terme d'Interactive Evolutionnary Computation (IEC). L'IEC permet d'inclure dans l'apprentissage des informations telles que des préférences humaines, des intuitions, des émotions, des aspects psychologiques (Takagi, 2001). Ce domaine peut être principalement approché de deux manières : par les algorithmes Evolutionnaires Interactifs (en anglais Interactive Genetic Algorithm ou IGA), ou les Algorithmes Génétiques Basés sur l'Humain (en anglais Human-Based Genetic Algorithm ou HBGA)(Kosorukoff, 2001). Avec les IGAs, l'algorithme évolutionnaire garde un rôle important et les interactions permettent à l'utilisateur d'attribuer un score de fitness aux individus de la population. Les IGAs sont notamment utilisés lorsqu'une fitness basée sur une métrique mathématique n'est pas pertinente pour comparer des individus évolutionnaires. Une fitness basée sur l'appréciation et le jugement d'un utilisateur est alors privilégiée. Cette catégorie d'approches interactives est notamment utilisée dans les domaines de la création artistique ou industrielle (Lv *et al.*, 2019; Yang et Tian, 2019), lorsque il est nécessaire de prendre en considération la perception et l'évaluation subjective de l'utilisateur concernant des solutions proposées par l'algorithme. Dans les HBGA (Kosorukoff, 2001), l'utilisateur a une charge plus importante pendant l'apprentissage. En plus de l'évaluation de la valeur de fitness, l'utilisateur est en charge d'effectuer également toutes les opérations évolutionnaires, depuis l'initialisation à la sélection en passant par la mutation et le croisement. L'interactivité est également fréquemment combinée avec les méthodes de PG et d'EG et trouve, par exemple, des applications dans le cadre de la synthèse et la génération musicale (Kaliakatsos-Papakostas *et al.*, 2012), où les utilisateurs sont en charge de l'évaluation d'individus, après écoute de leur morceau de musique respectif. L'interactivité en régression symbolique par PG est également étudiée dans le cadre applicatif pour extraire de la connaissance experte sur un domaine donné. Cette approche a notamment été appliquée à la modélisation du processus de production et stabilisation des bactéries lactiques (Chabin *et al.*, 2018). Dans ces travaux les utilisateurs participent à la création de modèles de relations entre des variables, en analysant itérativement des formules de différents scores et complexités et permettant, in fine, de générer un modèle global de l'ensemble du problème.

En plus de permettre d'apporter une évaluation subjective, l'IEC a d'autres bénéfices. De manière surprenante pour leur utilisateurs, les algorithmes d'IEC sont plus créatifs qu'attendu et

génèrent parfois des individus inattendus. Lorsqu'appliqués à de la génération d'images, l'IEC a, par exemple, permis de faire évoluer des images ressemblant à un alien, qui au fil des évolutions se sont transformées en voitures (Lehman *et al.*, 2020).

Cependant, de manière générale, l'un des problèmes majeurs de l'IEC est la fatigue de l'utilisateur induite par les requêtes répétées de l'algorithme. Ce phénomène est renforcé lorsque plusieurs individus ne peuvent être évalués simultanément (Kaliakatsos-Papakostas *et al.*, 2012). Des stratégies ont été proposées pour la réduire, comme l'utilisation d'une évaluation relative des individus (Wang et Takagi, 2005), l'interaction avec un sous-ensemble de la population (élite, centroïde de clusters, ...) et l'ajout d'information visuelle (Kaliakatsos-Papakostas *et al.*, 2012), ou l'estimation de la fitness de l'utilisateur (Lv *et al.*, 2019). Une autre solution consiste également à effectuer une évaluation automatisée et à demander occasionnellement une évaluation humaine à chaque n-ième génération (Kamalian *et al.*, 2005; Boudjeloud-Assala, 2005).

Apprentissage interactif par renforcement

L'interactivité est également étudiée en apprentissage par renforcement. En effet, l'interactivité a permis d'accomplir avec succès des tâches telles que la préparation de données (Berti-Équille, 2020), ou l'exploration de données en grande dimension (Personnaz *et al.*, 2021). Dans ce domaine, les retours et les commentaires des utilisateurs peuvent être intégrés (Cruz et Igarashi, 2020) : directement dans la récompense (*reward shaping*), dans la politique apprise π (*policy shaping*), ou dans les trajectoires générées. Ces commentaires peuvent prendre des formes variées aussi bien de valeur binaires ou numériques, de préférences dans un sous ensemble, de conseil sur l'action à choisir, de directives en anticipation. Cependant, certaines de ces méthodes ont aussi leurs inconvénients. Par exemple, le reward shaping peut souffrir de mauvais design de la récompense, induisant une politique avec des comportements non-desirés, bien que la récompense ait un score élevé. On parle alors de *reward hacking*. Par ailleurs, tout comme en IEC, la fatigue de l'utilisateur est un sujet de préoccupations car, au fil des itérations, la fatigue augmentant, la qualité des retours des utilisateurs tend à diminuer et l'utilisateur tend à donner des feebacks plus négatifs (Cruz et Igarashi, 2020).

En apprentissage par renforcement, une stratégie interactive basée sur une évaluation relative d'éléments est également possible : l'apprentissage sur les préférences (Fürnkranz et Hüllermeier, 2010). Il s'agit d'un paradigme d'apprentissage à partir de commentaires utilisateurs non numériques, qui présente notamment l'avantage de soulager le problème de design de la récompense et permet d'inclure des tâches réalisables par des personnes qui ne sont pas expertes en apprentissage automatique (Wirth *et al.*, 2017). Pour deux éléments e_1 et e_2, les préférences peuvent être (Kreps, 1988) :

- e_1 est strictement préféré à e_2, noté $e_1 > e_2$,
- e_2 est strictement préféré à e_1, noté $e_1 < e_2$,
- e_1 et e_2 sont équivalents, noté $e_1 \sim e_2$,
- e_1 est faiblement préféré à e_2, noté $e_1 \geq e_2$,
- e_2 est faiblement préféré à e_1, noté $e_1 \leq e_2$.

L'objectif est alors d'apprendre une politique π maximisant autant que possible les préférences, i.e. pour $e_1 > e_2$ les probabilités associées sont $\mathrm{P}_\pi(e_1) > \mathrm{P}_\pi(e_2)$. En général, l'apprentissage passe par la minimisation de la fonction de perte suivante : $\mathrm{Loss}(\pi, e_1 > e_2) = -(P_\pi(e_1) - P_\pi(e_2))$. Les préférences peuvent être données par l'utilisateur sur des trajectoires, des actions ou des états (Wirth *et al.*, 2017). Ces trois types de préférences imposent des challenges différents à l'utilisateur. Les préférences sur les états sont légèrement moins fatiguants que les préférences

sur les actions. Cependant, il est difficile d'estimer le résultat final à partir du choix d'une action. La préférence sur les trajectoires est la moins complexe, ce qui en fait l'une des approches les plus communément utilisées. Ce type de préférences présente néanmoins des défauts et l'algorithme apprenant sur ces préférences doit déterminer quels états ou actions sont utiles pour la résolution de la tâche.

Pour apprendre efficacement sur des préférences, l'utilisation d'oracles est utile pour simuler les préférences des utilisateurs et réaliser des benchmarks (Lee *et al.*, 2021). Ces benchmarks ont permis de mettre en évidence les baisses de performances rencontrées lorsque les utilisateurs ne fournissent pas les préférences attendues. Cet élément joue donc en la faveur d'utilisateurs "experts", plus à même de fournir des informations de qualité.

Dans le cadre d'applications à la régression symbolique, l'interactivité en apprentissage par renforcement n'a cependant pas encore fait l'objet de nombreuses études à notre connaissance. Un premier travail (Kim *et al.*, 2020) propose une plateforme interactive dans laquelle les utilisateurs peuvent ajuster dynamiquement les différents hyperparamètres d'apprentissage. L'interactivité principale est ici un mécanisme permettant à l'utilisateur d'approuver ou de rejeter les expressions trouvées par un algorithme de renforcement.

Autres approches

Bien que PG et AR soient les approches principales pour la régression symbolique interactive, d'autres travaux cherchent à développer des méthodes interactives, par exemple pour les méthodes par expansion de base. Pour les GAM, l'outil *GAM Changer* (Wang *et al.*, 2021) permet aux experts d'un domaine et aux scientifiques des données de modifier itérativement les poids du modèle afin d'aligner les comportements du modèle sur les connaissances du domaine.

3.4.4 Evaluation d'algorithmes interactifs

Évaluer des algorithmes interactifs en apprentissage automatique n'est pas une tâche facile et nécessite d'utiliser des procédures adaptées. En effet, leur évaluation nécessite de prendre en compte le comportement des algorithmes et de leurs utilisateurs, leur coopération et leur co-adaptation pendant l'apprentissage. A ces fins, deux types de validations sont nécessaires à la fois *centrée sur l'algorithme* et *centrée sur l'humain* (Boukhelifa *et al.*, 2018), évaluées avec des méthodes différentes par les communautés de l'apprentissage automatique et de l'interaction homme-machine (Dudley et Kristensson, 2018).

L'évaluation centrée sur l'algorithme implique l'utilisation de métriques de performance ou d'erreur (tel que l'EQM) et peut inclure d'autres métriques tel que le temps d'exécution.

Les évaluations centrées sur l'humain ont, elles, pour but d'analyser la perception subjective de l'utilisateur. Elles peuvent prendre la forme de questionnaires, d'entretiens, ou de commentaires informels. Il peut également s'agir d'observations d'utilisateurs durant l'entraînement de l'algorithme. Ces évaluations sont généralement réalisées après l'accomplissement de la tâche. Les questionnaires comme le Nasa-TLX (Hart, 2006), initialement développé pour estimer la charge de travail, permettent d'estimer la charge mentale ou temporelle, la performance perçue, l'effort requis ou la frustration de l'utilisateur. Afin de prendre en compte les spécificités des algorithmes d'iML, d'autres facteurs perçus par l'humain ont également été inclus dans des dérivés de ce questionnaire (Krening et Feigh, 2019) incluant la perception par l'utilisateur de la transparence et l'intelligence perçue par l'utilisateur, ou l'immediateté avec laquelle l'algorithme prend en compte les interactions de l'utilisateur.

Encore beaucoup d'évaluations se concentrent sur l'amélioration de la performance de l'algorithme, plutôt que sur l'utilisateur (Cruz et Igarashi, 2020). Or, mieux comprendre et modéliser le fonctionnement de l'interaction permettrait d'améliorer l'apprentissage. Dans ce sens, comprendre l'utilisateur et le modéliser permet d'assister et guider un utilisateur novice par des suggestions (Dabek et Caban, 2017). Pour mieux comprendre l'utilisateur, le modèle appris peut aussi être sauvegardé, à une fréquence donnée pendant l'apprentissage interactif. Ces modèles partiellement entraînés permettent ensuite d'attribuer un score à chaque participant, basé sur l'exactitude des justifications de chacun de ses feedbacks (Kulesza *et al.*, 2015).

L'évaluation de l'interprétabilité est également nécessaire dans le processus d'apprentissage. Doshi-Velez et Kim (Doshi-Velez et Kim, 2017) propose une procédure d'évaluation progressive en trois niveaux :

— *Niveau applicatif* : avec un humain dans la boucle, sur une tâche réelle ;

— *Niveau humain* : avec interaction. Il s'agit d'une simplification du niveau appliqué permettant de réaliser des expériences pour estimer l'efficacité de l'approche à moindre coût ;

— *Niveau fonctionnel* : sans humain, sur une tâche proxy servant à comparer l'algorithme avec d'autres algorithmes d'interprétabilité similaire.

Néanmoins, la construction d'un algorithme d'apprentissage interactif n'est pas une action ponctuelle mais doit être considérée comme un processus itératif comportant des révisions (Mathewson et Pilarski, 2022). Dans ce sens, l'utilisation d'oracles (ou utilisateurs) simulés est bénéfique au cours des premières étapes d'implémentation de l'algorithme interactif (Cruz et Igarashi, 2020). Tout en gardant tête que ces simulations peuvent différer du comportement réel de l'utilisateur, elles permettent d'avoir une première vision du fonctionnement de l'interaction.

3.5 Conclusion du chapitre

Ce chapitre a passé en revue les principaux concepts et méthodes d'apprentissage dans l'apprentissage automatique en lien avec le modèle décrit dans le chapitre 2. Nous y détaillons en premier le problème de représentation de connaissance, en distinguant la connaissance implicite et explicite. Nous synthétisons différents moyens de représenter la connaissance explicite dans le cadre de l'apprentissage automatique, en pointant vers certaines représentations adaptées à la régression symbolique. La création de variables par régression symbolique, qui est l'objet de cette thèse, est ensuite présentée en la comparant à la régression classique, puis différentes approches sont présentées. Celles-ci s'appuient sur l'apprentissage par les algorithmes génétiques, par des réseaux de neurones, par renforcement ou par des bases de fonctions. Une description détaillée de leur application à la régression symbolique est présentée, avec et sans prise en compte de connaissance experte.

Enfin, les forces et des enjeux de l'interactivité pour la régression symbolique sont également détaillés. En iML différentes stratégies d'apprentissage peuvent être mises en oeuvre. L'apprentissage interactif peut également être réalisé par le biais de différents algorithmes de RS et prendre en compte divers types de retours de l'utilisateur. Enfin, les méthodes d'évaluation en iML sont précisées, à la fois centrées sur l'humain et centrée sur l'algorithme.

Pour mettre en évidence les travaux connexes aux nôtres, ce chapitre aborde également certaines approches de RS récemment proposées. Les limites des travaux existants, qui constituent la motivation de plusieurs chapitres de cette thèse, sont également discutés :

— La représentation de connaissances expertes dans le cadre de la régression symbolique sur des données du réseau électrique,

— la prise en compte de la connaissance à la fois implicite et explicite,

— Le manque d'interactivité dans les approches de régression symbolique par renforcement.

Dans les prochains chapitres de cette thèse, nous mettrons en évidence quelles approches basées sur l'évolution grammaticale, le renforcement et l'interactivité pourront être utilisées pour résoudre ces problèmes.

Chapitre 4

Extraction de variables interprétables par Evolution Grammaticale Interactive

4.1 Introduction

Comme nous l'avons présenté dans le chapitre 3, de nombreux travaux de régression, symbolique ou non, proposent de créer des variables sous forme d'expressions symboliques par combinaisons d'une sélection de variables d'entrée, sans prendre en compte les spécificités physiques du problème telles que les grandeurs des variables considérées, la prise en compte de leurs unités ainsi que les relations qui les relient. De fait, l'explicabilité des résultats obtenus est d'autant plus complexe pour les experts du domaine que les expressions et variables obtenues ne respectent pas systématiquement les propriétés et contraintes physiques du problème étudié. Il apparaît alors nécessaire de pouvoir trouver une structure de données permettant de décrire ces contraintes.

Dans ce chapitre, nous présentons un premier algorithme interactif de création d'expressions symboliques dans lequel l'espace des solutions est contraint par un ensemble de règles grammaticales définies dans une grammaire non-contextuelle au format BNF, et où l'interactivité permet de mettre à jour la grammaire entre les apprentissages évolutionnaires afin d'améliorer les expressions symboliques trouvées. Les solutions construites sont alors interprétables par des experts, grâce au respect des règles de la grammaire. Nous décrivons dans un premier temps la construction d'une grammaire pertinente pour l'étude du réseau électrique dans la section 4.3. La création d'expressions symboliques est ensuite réalisée par un algorithme de Régression Symbolique (RS) basé sur l'Evolution Grammaticale (EG). Cet algorithme d'Evolution Grammaticale Interactive, noté *EGI*, permet de mettre à jour itérativement la grammaire pour en améliorer les règles et décrire de plus en plus finement le problème physique et les contraintes métier associées. Nous y comparons également ici plusieurs structures grammaticales : une première grammaire simpliste, puis une grammaire plus complexe dont la construction s'est faite incrémentalement grâce aux commentaires et retours d'experts. Suivant la terminologie utilisée par Doshi-Velez et Kim dans (Doshi-Velez et Kim, 2017), une première expérience a pour but spécifique d'évaluer l'interprétabilité de la méthode, d'un point de vue "fonctionnel" et "humain" en la comparant à d'autres méthodes interprétables et en faisant analyser les résultats par un expert. Une fois l'interprétabilité de la méthode validée, une deuxième expérience a pour but d'estimer le gain de performance obtenu par la mise à jour de la grammaire. Enfin, une évaluation de la robustesse des expressions créées est réalisée et deux pistes sont proposées pour permettre d'apprendre des indicateurs plus généralisables. Au cours des différentes expériences, la méthode est évaluée sur

un jeu de données réel constitué de mesures du réseau électrique français, au pas de 5 minutes de janvier 2014 à décembre 2017.

4.2 Problématique

Les grammaires non-contextuelles possèdent une structure modulable, permettant d'inclure des contraintes spécifiques au problème étudié au travers de règles adéquatement choisies. L'exploration de cette structure de données, historiquement, majoritairement, réalisée par des approches évolutionnaires (Ryan *et al.*, 1998), présente alors l'avantage de produire des solutions à la fois lisible par un humain, et où l'on peut forcer la structure à produire des solutions proches de celles utilisées par les experts du domaine.

Dans le problème étudié, à chaque variable observée ou mesurée du réseau électrique est associée une grandeur physique possédant une unité standardisée. Ainsi, une puissance active se mesure en Watt, là où une tension se mesure en Volt. Cependant lorsque l'on chercher à combiner ces variables pour en construire de nouvelles, certaines combinaisons sont physiquement illicites. Par exemple, additionner une puissance active avec une tension n'a pas de sens physique et de fait pourrait mener à la construction d'une variable non cohérente pour un expert, bien qu'elle ait un bon score. Dans ce chapitre, nous cherchons à tirer parti de la structure grammaticale pour notre application aux réseaux électriques.

Plus précisément, comme expliqué dans le chapitre 2, nous cherchons à trouver des *"explications"* sur la variable de transit électrique y_l, $\forall l \in L$. Plus formellement, étant donné un ensemble de variables observées $X \in \mathbb{R}^D$, nous voulons extraire pour chaque ligne $l \in L$ une fonction f_l réalisant des combinaisons pertinentes et potentiellement non-linéaires de la base de variables d'entrées X et de sorte que f_l soit pertinente pour expliquer et représenter y_l. Nous avons choisi ici de nous concentrer sur les méthodes de d'Evolution Grammaticale (EG) afin de proposer une grammaire personnalisée pour les données électriques.

Par ailleurs, en EG, les variables construites sont très fortement sensibles à la grammaire choisie. Le choix des règles dans la grammaire peut affecter notablement la vitesse de convergence de l'algorithme. C'est notamment le cas des grammaires "explosives" ayant plus de symboles non-terminaux que de terminaux (Harper, 2010). Bien que des recommandations aient été faites pour construire les grammaires (Nicolau et Agapitos, 2018), en construire une correcte dès la première itération parait complexe, en particulier lorsqu'elle sont appliquées à des données industrielles. Il semble donc pertinent de ne pas considérer la grammaire comme un élément statique, mais plutôt un élément dynamique que l'on puisse améliorer vis-à-vis du problème étudié. Dans l'objectif de mettre à jour la grammaire, des travaux proposent une approche, nommée Meta Grammar Grammatical Evolution (O'Neill et Brabazon, 2005), où une meta-grammaire d'entrée est utilisée pour spécifier la construction d'une autre grammaire syntaxiquement correcte. La mise à jour de la grammaire est faite pendant la phase d'apprentissage. D'autres travaux sur l'inférence grammaticale (De La Higuera, 2005; Pandey, 2021), proposent quant à eux d'apprendre une grammaire à partir de données telles que, dans notre cas, des expressions symboliques.

Cependant, les travaux présentés ci-dessus cherchent plutôt à optimiser la structure de la grammaire en la restreignant aux meilleures règles, plutôt que d'y inclure de nouvelles règles et de nouvelles connaissances qui n'étaient pas présentées dans les itérations précédentes. Par ailleurs, dans les travaux mentionnés ci-dessus, les grammaires utilisées sont modifiées automatiquement, sans laisser la possibilité à l'humain d'y rajouter des connaissances.

Nos travaux visent donc au contraire à proposer une approche permettant *d'enrichir* la grammaire avec de la connaissance experte en mettant l'humain dans la boucle d'apprentissage.

Nom de la grandeur physique	Notation	Unité	Abréviation
Puissance active	p	Watt	W
Puissance réactive	q	Volt-Ampère Réactif	VAR
Puissance apparente	s	Volt-Ampère	VA
Tension	v	Volt	V
Phase	Θ	Degré	$^\circ$
Intensité	i	Ampère	A
–	p/v	–	A
Puissance apparente	$\sqrt{p^2+q^2}$	Volt-Ampère	VA
Carré de la puissance active	p^2	–	W^2
Carré de la puissance réactive	q^2	–	VAR^2
Carré de la puissance apparente	p^2+q^2	–	VA^2

TABLE 4.1 – Description physique du problème

4.3 Approche proposée

4.3.1 Construction de la grammaire

La construction de la grammaire joue un rôle significatif dans les méthodes d'EG car elle définit les règles de recherche dans l'espace des expressions symboliques. En effet, une grammaire trop lâche avec peu de règles aurait un espace trop vaste à rechercher, tandis qu'une grammaire fortement contrainte par de nombreuses règles pourrait limiter à la recherche à des zones non pertinentes. Pour faciliter la compréhension, nous fournissons une version simplifiée de notre grammaire dans la figure 4.1. La version complète de la grammaire utilise toutes les variables décrites dans la section 4.4, y compris les variables topologiques, et une plus grande variété de fonctions sur chaque unité. Dans l'algorithme proposé dans ce chapitre, la grammaire est construite de manière itérative avec des experts dans la boucle qui peuvent fournir des commentaires permettant de retirer ou d'ajouter des contraintes telles que de nouvelles variables ou de nouvelles opérations.

Grandeurs et unités physiques du problème

Tout d'abord, avant de construire nos règles grammaticales, nous devons définir les unités physiques que nous souhaitons inclure dans la grammaire. Les unités physiques du problème étudié sont : la puissance active p (d'unité : watt W), la puissance réactive q (volt-ampère réactif VAR), la puissance apparente s (volt-ampère VA), l'amplitude de la tension v (tension V) ou l'intensité i (ampère A). À partir de là, il est alors possible de définir le carré de toutes les unités : p^2, par exemple, est la valeur au carré de p d'unité W^2. Certaines des unités précédemment citées sont définies par relation avec d'autres unités. La puissance apparente est par exemple définie à partir de la puissance active et réactive. Ces différentes unités sont récapitulées dans la tableau 4.1.

Description de la structure grammaticale

Pour illustrer la méthode utilisée pour construire notre grammaire, nous nous appuyons sur la grammaire présentée en figure 4.1.

Une fois les unités du problème connues, la première étape pour construire une grammaire est de définir quelle structure nous souhaitons autoriser en sortie, représenté ici dans la gram-

```
# 1) Création d'une expression unitaire (unités autorisées en sortie)
<expr> ::= <p> | <s> | <i> | <f> * <expr> | <p>/<v>

# 2) Définition des opérations licites sur chaque unité
<p>  ::= <p>-<p> | <pop>(<p>, <p>) | <sop>(<p>) | <p_var>
<q>  ::= <q>-<q> | <pop>(<q>, <q>) | <sop>(<q>) | <q_var>
<v>  ::= <v>-<v> | <pop>(<v>, <v>) | <sop>(<v>) | <v_var>
<p2> ::= <p>*<p> | square(<p>)
<q2> ::= <q>*<q> | square(<q>)
<s>  ::= sqrt(<p2> + <q2>) |<v> * <i>
<i>  ::= <s>/<v> | <pop>(<i>,<i>) | <sop>(<i>) | <i_frontière_var>
<f>  ::= <f>*<f> | <p>/<p> |  <q>/<q> | <v>/<v>

# 3) Définition des  opérations qui retournent une variable de même
unité que les variable
<pop>::= sum | minimum | maximum # Fonctions à deux arguments
<sop>::= abs | neg | pos # Fonctions à un seul argument
```

FIGURE 4.1 – Exemple de grammaire. Le symbole "|" est utilisé comme séparateur entre chaque règle utilisée pour remplacer le symbole placé au début de la ligne avant le "::=". Les terminaux correspondant aux variables d'entrée sont `<p_var>` `<q_var>`, `<v_var>` et `<i_frontier_var>`.

maire par le symbole `<expr>`. Pour rappel, l'élément remplaçant le symbole peut être l'une des règles séparées par le caractère "|". La définition de ce premier symbole permet d'appliquer des connaissances expertes sur l'unité de la variable de sortie. Dans cet exemple, `<expr>` peut être d'unité q, s, i ou p/v.

La deuxième étape consiste à *imposer* la cohérence des unités physiques de la variable de sortie, en définissant quelles opérations autorisées sur chaque unité ainsi que les combinaisons de deux unités. Pour passer d'une unité à l'autre, des lois de la physique telles que le triangle des puissances $s = \sqrt{p^2 + q^2}$ ou une variation de la définition de la puissance apparente $i = \frac{s}{v}$ ont été traduites sous forme de règles grammaticales. Les variables d'entrée sont définies ici à l'aide des observations `<p_var>`, `<q_var>`, `<v_var>`, et `<i_frontiere_var>` ($i\_frontiere$ correspondant au transit pour 9 lignes transfrontalières pour modéliser les interactions avec l'extérieur de la zone géographique étudiée) avec l'unité correspondante p, q, v, i. Elles peuvent être combinées pour produire une nouvelle variable ayant soit la même unité (par exemple `<p> - <p>` produit une nouvelle variable de unité de p), soit une unité différente (`<p>/<p>` est sans unité alors que `square(<p>)` a l'unité de p^2).

Enfin, nous définissons dans un dernier temps les opérations licites à effectuer sur une seule ou une paire de variables, qui sont répétées pour presque toutes les unités.

4.3.2 Création interactive d'expressions symboliques interprétables et physiquement cohérentes

Notre méthode, nommée EGI (Evolution Grammaticale Interactive), est basée sur les algorithmes d'EG dans lequel nous introduisons un mécanisme d'interaction avec des experts humains pour construire et améliorer à la fois la grammaire et les individus construits au fil des apprentissages. Tout comme les méthodes d'EG, notre méthode recherche dans l'espace via une population

Algorithm 4 Algorithme d'Evolution Grammaticale Interactive

Require: Observations X, Cible y, Grammaire initiale G_0

```
function Evolution Grammaticale Interactive(G_0)
    condition_d'arret_satisfaite ← False
    operateur_satisfait ← False
    meilleur_individu ← None
    G ← initialisation_de_la_grammaire(G_0)
    while not operateur_satisfait do
        population, meilleur_individu ← creation_de_la_population(G)
        condition_d'arret_satisfaite ← False
        while not condition_d'arret_satisfaite do
            parents ← selection_des_parents(population, X, Y)
            descendance ← croisement(parents)
            descendance ← mutation(descendance)
            descendance ← evaluation(descendance)
            population, meilleur_individu ← remplacement(population, descendance)
            condition_d'arret_satisfaite ← test_conditions(meilleur_individu)
        G ← mise_à_jour_de_la_grammaire(G)
    return meilleur_individu, G
```

en effectuant plusieurs fois des opérations génétiques de sélection, croisement, mutation sur une population d'individus, comme décrit dans l'algorithme 4. Les règles de la grammaire également sont toujours respectées par les individus.

Pour contraindre de plus en plus précisément l'espace de recherche au fil des apprentissages, nous demandons aux utilisateurs d'examiner les expressions symboliques proposées en sortie de l'algorithme, afin d'en extraire les variables, opérations et combinaisons pertinentes pour la création de nouvelles règles grammaticales. Les informations fournies par les utilisateurs peuvent prendre la forme de commentaires généraux sur les individus obtenus, de nouvelles relations physiques à intégrer, des unités ou des opérations additionnelles et des connaissances sur les unités des individus finaux. La grammaire ensuite est mise à jour avant le lancement du prochain apprentissage à partir des retours d'utilisateurs experts. Ces itérations nous ont par exemple permis de créer la grammaire décrite en figure A.2, obtenue grâce aux interactions avec des experts à raison d'une quinzaine d'interventions sur une période de 3 mois.

Dans le processus d'apprentissage ainsi décrit, la construction grammaticale est donc réalisée de manière itérative en même temps que l'amélioration des individus.

4.3.3 Fonction d'évaluation *fitness*

Nous utilisons la valeur absolue du coefficient de corrélation r comme mesure de l'adéquation ou *fitness* de chaque individu. Cette mesure, basée sur la corrélation linéaire de Pearson, est comprise entre 0 et 1, où une valeur r de 1 indique que les prédictions $\hat{y}$ correspondent parfaitement au comportement de la variable cible y. Par ailleurs, nous remplaçons par une valeur nulle les valeurs de corrélation ayant une p-valeur supérieure à un seuil de 0.05.

Si le coefficient de corrélation linéaire mesure principalement la force de la relation linéaire entre deux variables, il présente également l'avantage évident de pouvoir comparer le comportement de deux variables qui se situent sur des ordres de grandeur différents. Cette mesure semble particulièrement adaptée à notre cas d'étude car nous ne sommes pas intéressés ici à prédire

la valeur exacte du transit électrique mais plutôt à comprendre la relation sous-jacente globale entre les variables d'entrée X et le transit décrit par une sortie y. De plus, du point de vue de l'opérateur, il est aussi intéressant d'examiner une fonction f que sa valeur ré-échelonnée, par exemple $10 \times f$. Pour comprendre les avantages de l'utilisation de cette métrique, nous la comparons l'expérience de la section 4.5 également à la métrique de l'erreur quadratique moyenne (EQM) basée sur la distance.

4.4 Description des données réelles

4.4.1 Délimitation de la zone d'étude

Nous utilisons ici la description des données présentées dans le chapitre 2 avec un réseau à N noeuds et L lignes. Dans chaque noeud $n \in N$, les quantités présentes sont la puissance active p_n et la puissance réactive q_n, le module de la tension v_n et la phase de la tension Θ_n ; tandis que pour une ligne $l \in L$, nous utilisons des informations sur la connexion de la ligne $connected_l^{or}$, $connected_l^{ex}$ (avec l'origine or et l'extrémité ex) ainsi qu'une clé $neighbor\_id_l$ correspondant à la liste des lignes voisines (i.e. électriquement connectées) à ce pas de temps. Les variables situées dans les noeuds sont copiées dans chaque origine/extrémité de la ligne connectée à ce noeud. Bien que cette représentation implique une redondance dans les données, nous pouvons maintenant raisonner uniquement en termes de lignes électriques et oublier les objets noeuds.

A chaque extrémité de ligne, les transits i_l^{or} et i_l^{ex}, $\forall l \in L$ sont utilisées comme sorties de nos différentes méthodes et expériences. Nous pouvons ainsi considérer le système étudié comme un *système fermé* sans dépendance temporelle, puisque les variables cibles $i_l^{or,ex}$ fournies par notre simulateur ne dépendent que des mesures et du réglage des hyperparamètres experts, utilisés pour calibrer le calcul de puissance-transit.

Les variables décrivant les liens électriques sont $connected_l^{or,ex}$ (un booléen représentant la connexion de la ligne l à son origine ou à son extrémité), et $neighbor\_id_l$ (un id utilisé comme clé pour représenter toutes les lignes électriquement connectées à la ligne l). À partir de maintenant, nous désignons la base des variables initiale par $X = (X_l)_{l \in L}$ et les variables de transit cibles $y = (y_l)_{l \in L}$ avec :

$$\begin{aligned} \forall l \in L, \quad & X_l = ((p, q, v, \Theta)_{n\_or(l), n\_ex(l)}, connected_l^{or,ex}, neighbor\_id_l) \\ & y_l = (i_l^{or}, i_l^{ex}) \end{aligned}$$

Dans ces expériences, nous nous concentrons sur une zone spécifique du réseau gérée par RTE. En effet, le réseau électrique français dans son ensemble est un système très complexe, avec jusqu'à 6500 noeuds, 12000 lignes électriques, et de nombreuses interactions à la fois avec d'autres réseaux européens et en son sein. Une approche commune, pour contrôler la façon dont ces interactions influencent les études du réseau, consiste à diviser le réseau en sous-zones au sein desquelles les éléments ont une influence mutuelle élevée les uns sur les autres (Marot *et al.*, 2018b). Cette approche, historiquement réalisée par les GRT, permet à plusieurs opérateurs de travailler simultanément sur des zones séparées d'une taille acceptable qu'ils peuvent contrôler. Nous nous concentrons donc sur les données d'une vallée montagneuse sélectionnée, où l'escarpement contraint intrinsèquement les interactions avec le reste du réseau. Cette sélection restreint le périmètre d'étude à 69 noeuds et 92 lignes, où 9 lignes connectent la zone avec les autres parties du réseau.

A partir des mesures collectées de janvier 2014 à décembre 2017 et décrites dans un format propriétaire à RTE, nous exécutons des calculs de répartition en "AC" dont l'algorithme est donné

dans le chapitre 2. Les données de mesures et les résultats de simulations sont ensuite combinés et rassemblés. Nous obtenons ainsi 365 165 observations de pas de temps et des variables de transits cibles associées. Ces données sont enfin filtrées à l'échelle de la zone d'étude pour ne conserver que les variables relatives à la zone d'étude, soit 828 variables.

4.4.2 Identification de lignes électriques à étudier

Comme nous sommes uniquement intéressés par l'analyse des transits y sur les lignes *sensibles*, nous avons d'abord sélectionné un sous-ensemble de toutes les 92 lignes sur lesquelles réaliser la Régression Symbolique. Cette étape de pré-traitement permet de s'assurer que nous ne regardons pas uniquement des informations résiduelles ou des effets négligeables sur la variable cible. Pour ce faire, nous avons identifié les *lignes les plus fréquemment chargées* en sélectionnant celles qui dépassent un pourcentage donné $pourcentage\_i\_seuil$ (100, 90, 80, et 70%) de la limite thermique de la ligne i_seuil pendant un pourcentage du temps total $pourcentage\_temps$ ($0,05$ ou $0,1$%). La sélection finale est définie comme l'union des lignes identifiées comme chargées par une de chaque combinaison d'hyperparamètres ($pourcentage\_i\_seuil$, $pourcentage\_temps$). Cette méthode a été appliquée aux données de janvier 2014 à décembre 2017 décrites dans la sous-section 4.4.1 et nous a permis d'identifier 24 lignes.

4.5 Interprétabilité des expressions

4.5.1 Protocole expérimental

Cadre des expériences

Pour chaque ligne définie comme sensible, nous cherchons maintenant une variable $\hat{y}$ sous forme d'une expression symbolique pour décrire le transit électrique de sa variable cible y. Pour cela, nous faisons évoluer une grande population de 2 000 individus sur 200 générations en suivant les règles grammaticales définies ci-dessus. Une grande population s'avère nécessaire en raison du nombre élevé de contraintes définies dans la grammaire. L'ensemble de données initial contenant 365 165 observations est divisé selon un ratio de 80/20% entre les ensembles d'entraînement et de test. La recherche des expressions pour chaque ligne est lancée 30 fois avec une initialisation PI_Grow de la population, et seules les meilleures expressions sont conservées pour ensuite être inspectées manuellement par les opérateurs. Pour la reproductibilité des expériences, tous les hyperparamètres sélectionnés pour l'algorithme évolutionnaire (algorithme 4) sont fournis dans le tableau 4.2. Ceux-ci ont été choisis lors d'expériences préliminaires par validation croisée réalisée sur les différents paramètres du tableau.

Pour les deux métriques de fitness (corrélation et EQM) dans les deux expériences, nous insérons un terme de régularisation additionnel, relatif à la complexité des individus. Nous choisissons dans un premier temps de décrire la complexité κ d'un individu par la profondeur de son arbre syntaxique : nous le notons `profondeur_de_l'individu`. La profondeur est définie comme le nombre maximum de noeuds dans l'arbre de l'expression en question, de la racine à la feuille la plus éloignée. Cette contrainte a pour but d'empêcher l'explosion de la taille de l'arbre (appelée phénomène de gonflement ou *bloating*) (Purohit *et al.*, 2011) et est légèrement pondérée ($\epsilon = 10^{-8}$) pour ne supprimer que les noeuds redondants sans trop contraindre l'espace de recherche. Finalement, notre fonction de fitness devient :

$$fitness = metrique\_choisie + \epsilon \times \texttt{profondeur\_de\_l'individu} \tag{4.1}$$

Paramètre	Valeur
Génération	200
Taille de population	2000
Initialisation	PI Grow (Fagan *et al.*, 2016), profondeur initiale maximale de 10
Sélection	Tournoi avec une taille de 2
Crossover	Type variable un point (probabilité 0.9)
Mutation	Type int flip per codon (probabilité 0.1)
Élitisme	Top 10
Remplacement	Générationnel
Fitness	Corrélation de Pearson ou EQM

TABLE 4.2 – Paramètres évolutionnaires choisis pour l'expérience d'évaluation de l'interprétabilité.

Implémentation

Dans l'objectif de tirer le meilleur parti de leur implémentation parallélisée, nous avons utilisé l'implémentation open-source d'algorithme génétiques en Python PonyGE2 (Fenton *et al.*, 2017) comme code de référence. Grâce à la modularité de cette implémentation, nous y avons inséré de nouvelles métriques d'erreur basées sur la corrélation, un traitement de données spécialisé pour l'étude du réseau électrique, une nouvelle étape évolutionnaire personnalisée et la grammaire complète adaptée à notre problème. Le traitement de données personnalisé permet de prendre en compte les spécificités des données réelles utilisées dans ces expériences. Les données sont notamment chargées à partir d'un format "feather" adaptées aux grandes dimensions, les colonnes du jeu de données sont choisies pour correspondre à la grammaire et les lignes à la frontière de la zone sont définies. Ces définitions et sélections permettent ensuite de définir quelles variables du jeu de données sont utilisées dans l'évaluation. L'acquisition de données par les capteurs placés sur le réseau électrique pouvant produire des valeurs aberrantes hors distribution, nous avons également implémenté une étape de pré-traitement des données afin d'éliminer les éventuelles incohérences dues à une erreur de mesure.

4.5.2 Comparaison entre fitness

Dans la première partie de cette expérience, nous réalisons deux types d'apprentissages en parallèle, dans lesquels nous faisons varier uniquement la métrique de fitness utilisée pour évaluer pendant l'évolution les performances des individus, ligne par ligne. Le premier type d'apprentissage utilise l'Erreur Quadratique Moyenne (EQM) comme fitness, tandis que le deuxième type d'apprentissage est réalisé avec la valeur absolue de la corrélation de Pearson. Notre objectif est d'identifier la métrique la plus adéquate au problème, en choisissant entre des méthodes basées sur la distance, comme l'EQM, ou des méthodes basées sur la corrélation, comme la valeur absolue de la corrélation de Pearson. Pour chaque ligne *sensible* et pour les deux types de fitness, 30 apprentissages sont réalisés, soit 1440 exécutions. Les résultats obtenus sur l'ensemble de test sont résumés dans les Figures 4.2 et 4.3.

La figure 4.2 détaille les résultats des deux cas d'apprentissages (utilisant respectivement l'EQM ou Pearson comme fitness) pour 30 exécutions et les compare en utilisant la métrique d'EQM sur une échelle logarithmique (log-EQM). Dans cette figure, deux boîtes à moustaches sont associées à chaque ligne. Elles sont placées de chaque côté d'une ligne pointillée verticale :

— à gauche, la boîte à moustaches bleue correspond aux meilleurs individus trouvés pendant l'apprentissage évolués avec la fitness EQM ;

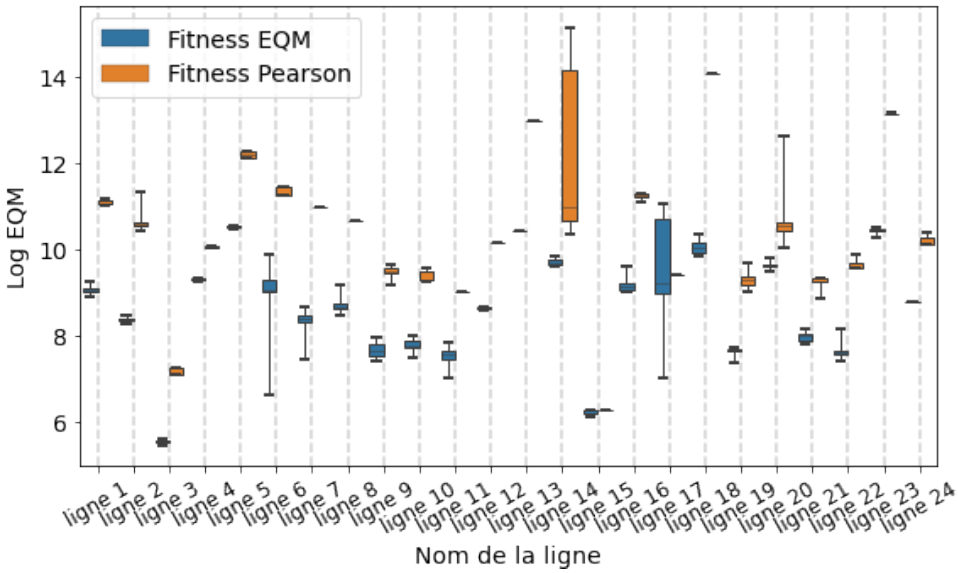


FIGURE 4.2 – Comparaison du score d'EQM en échelle logarithmique de deux populations respectivement évoluées avec une fitness basée sur l'EQM (en bleu) ou la corrélation de Pearson (en orange). Chaque boîte à moustaches résume le score de l'individu de fitness la plus élevée rencontré pendant l'apprentissage, pour chacune des 30 exécutions et pour une ligne donnée.

— à droite, la boîte à moustaches orange contient les scores des évolutions basées sur la corrélation.

De manière similaire, la figure 4.3 analyse les deux mêmes essais, comparés cette fois-ci en utilisant la valeur absolue de la corrélation de Pearson du meilleur individu rencontré pendant l'apprentissage.

En comparant, ligne par ligne, les évolutions basées sur la corrélation et celles basées sur la distance à l'aide des métriques d'EQM (figure 4.2) et de corrélation (figure 4.3), nous identifions que les deux évolutions produisent des individus notablement différents et presque opposés : les individus appris par une fitness d'EQM pendant l'évolution ont généralement des scores de corrélation très inférieurs à ceux des individus appris par une fitness de corrélation (et inversement sur l'échelle log-EQM). Alors que l'on pourrait attendre des individus appris avec une fitness d'EQM (et ayant une faible EQM) soient fortement corrélés avec la variable à prédire, ce n'est pas le cas pour la quasi-totalité des lignes apprises.

A l'exception de quelques lignes, où les deux métriques de fitness ont tout aussi bien fonctionné (tel que les lignes 3 et 15 sur l'échelle log-MSE en figure 4.2 ou les lignes 15 et 17 sur l'échelle de corrélation en figure 4.3 qui ont des scores proches quelle que soit la métrique de fitness d'évolution), les 2 métriques semblent explorer l'espace des expressions construites dans des directions opposées et ne peuvent pas être utilisées de manière équivalente.

De plus, en analysant les expressions produites par évolution basée sur la métrique d'EQM, nous identifions qu'elles ont tendance à combiner préférentiellement des variables de la base des variables d'entrées qui sont d'ordres de grandeur similaires à celle de la cible (bien que leur comportement soit différent). Contrairement aux individus créés avec la fitness de corrélation, les expressions obtenues avec la fitness EQM semblent de moindre intérêt physique/technique et utiliseraient principalement les variables d'intensité i. Ce point est critique pour l'utilisation des métriques d'évaluation basées sur la distance car nous ne pouvons pas normaliser nos données

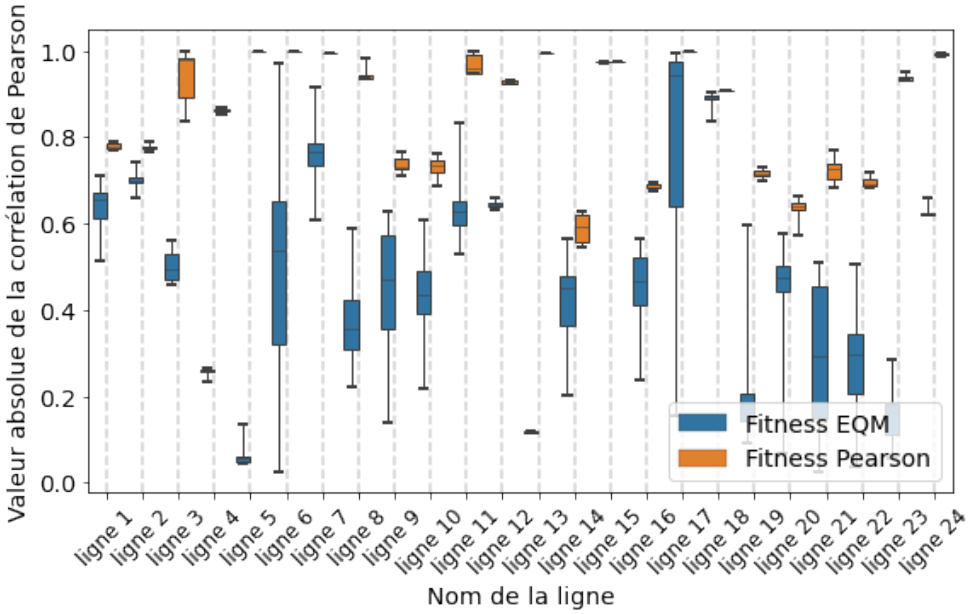


FIGURE 4.3 – Comparaison des deux stratégies d'évolution sur les mêmes 30 exécutions que celle de la figure 4.2. Chaque boîte à moustaches représente maintenant la valeur absolue de la corrélation de Pearson pour les meilleurs individus de chaque ligne. Les individus comparés sur les Figures 4.2 et 4.3 sont identiques.

pour qu'elles aient des plages similaires, car cela entraînerait l'impossibilité de respecter les lois physiques. Par exemple, après avoir normalisé chaque variable d'entrée de manière indépendante, les lois du circuit de Kirchhoff ne s'appliquent plus. Sur la base de ces observations, la seule stratégie acceptable est alors d'utiliser des métriques basées sur la forme physique, comme la corrélation de Pearson.

En ce qui concerne les expressions produites, cette première expérience a également montré que nous ne pouvons pas intuitivement construire moins d'expressions que le nombre de lignes sans perdre en performance, car les expressions extraites sont très différentes d'une ligne à l'autre.

4.5.3 Évaluation fonctionnelle

Dans la deuxième partie de l'expérience, nous comparons les individus obtenus précédemment par évolution avec une fitness de corrélation, avec les sorties d'algorithmes tels que LASSO Lars avec critère d'information de Bayes (Lasso Lars-BIC) (Zou *et al.*, 2007), arbre de décision (Breiman *et al.*, 1984) de profondeur limitée. Ces algorithmes ont été sélectionnés en raison de leur capacité à donner des sorties avec un niveau d'interprétabilité comparable, relevant ainsi de l'évaluation de type fonctionnelle ou "Functionally-grounded" (Doshi-Velez et Kim, 2017). Nous présentons également la variable la plus corrélée de l'ensemble de données original comme base de référence (étoile rouge dans la figure 4.4).

Afin de respecter la cohérence physique nécessaire à l'interprétabilité, une étape de prétraitement a été réalisée pour ces deux algorithmes. Pour ce faire, un premier entraînement a été réalisé pour chaque ligne en utilisant des données comportant toutes les unités physiques. Cette étape a mis en évidence que pour 10 des 24 lignes, les arbres de décision construits comportaient des nœuds de décision n'ayant pas le sens physique attendu. En effet, ces nœuds s'appuient sur

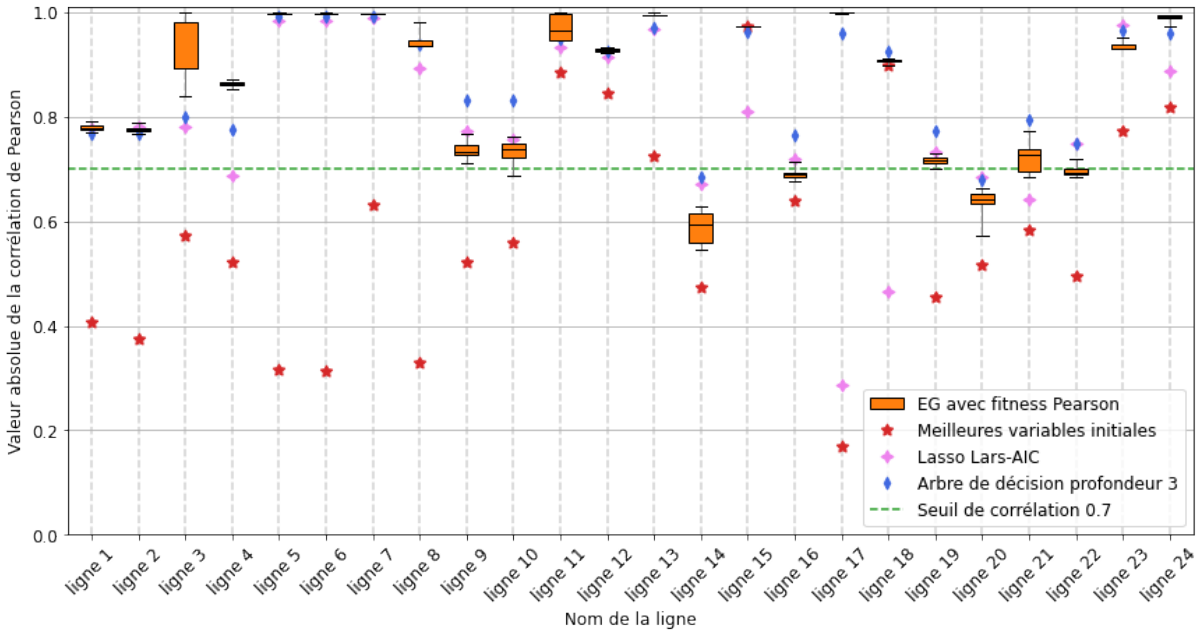


FIGURE 4.4 – Comparaison entre l'algorithme de EG and des algorithmes de l'état-de-l'art.

les variables de puissances réactives, qui ne sont pas utilisées directement par les opérateurs pour comprendre l'état du réseau, seulement dans un second temps. Une étape de pré-traitement a donc été ajoutée afin de filtrer ces variables avant l'apprentissage. De même, pour l'algorithme LASSO, une régression utilisant toutes les unités physiques produit des solutions additionnant des variables d'unités différentes, ce qui est illicite d'un point de vue physique. De fait, dans l'étape suivante, la régression LASSO a été apprise uniquement sur les variables de puissance active afin de produire des solutions d'un niveau d'interprétabilité comparable avec l'algorithme développé et ayant, en plus, une cohérence physique. Vis-à-vis de la méthode d'EG, il n'est cependant pas nécessaire de filtrer les variables car les relations physiques décrites dans la grammaire permettent de prendre en compte directement ces variables. Les résultats obtenus sont présentés dans la figure 4.4. Dans cette figure, les résultats associés à chaque ligne sont affichés le long d'une ligne verticale en pointillés et étiquetés en bas par le nom de la ligne correspondante. Ainsi, pour la ligne 24 à droite, nous avons de bas en haut : la variable la plus corrélée de l'ensemble de données initial (marquée par une étoile rouge) ; la corrélation avec la sortie Lasso Lars-BIC (rectangle rose) ; la corrélation avec la sortie d'un arbre de décision (diamant bleu) et la boîte à moustaches de notre méthode EG (boîte orange).

Comme le montre la figure 4.4, toutes les sorties de l'EG ont une corrélation plus élevée avec la cible que les variables de l'ensemble de données initial. En outre, en examinant uniquement les boîtes à moustaches, nous constatons que seules 3 des 24 valeurs les plus élevées des boîtes à moustaches sont inférieures au seuil de 0,7, en dessous duquel la corrélation n'est généralement pas considérée comme suffisamment forte pour qu'elle soit significative. Enfin, nous comparons notre approche en termes de corrélation avec des approches partiellement interprétables telles que Lasso Lars ou un arbre de décision de profondeur limité. Comme le montre la figure 4.4, nous pouvons voir que la méthode proposée fournit des résultats équivalent voire meilleurs que ceux des arbres de décision pour 14 des 24 lignes. Concernant la méthode LASSO, notre méthode obtient une corrélation plus élevée sur 14 des 24 lignes. Par ailleurs, pour deux d'entre-elles (lignes 1 et 2), les 3 méthodes sont équivalentes. Nous pouvons enfin noter que, la majorité des lignes

pour lesquelles les arbres de décision et LASSO ont des corrélations plus élevées correspondent à des lignes pour lesquelles les solutions générées par les 3 méthodes sont toutes proches du seuil de corrélation de 0.7 (deux d'entre elles sont en dessous du seuil : lignes 14 et 20). Grâce à ces expériences, nous pouvons mettre en évidence que les sorties de l'EG avec une corrélation significative sont au moins aussi fortement et parfois même plus corrélées à la cible que les sorties des autres méthodes. Dans le paragraphe suivant, nous examinons en détail les catégories de combinaison produites par notre approche.

4.5.4 Évaluation par l'expert

Enfin, la pertinence des principales expressions obtenues avec la méthode EG est évaluée techniquement par des experts des réseaux électriques afin de déterminer si les formules obtenues sont concluantes d'un point de vue technique et physique, si elles peuvent être utiles aux opérateurs et si elles ont un sens pour eux. Cette expérience a pour but de confirmer l'interprétabilité des sorties d'un point de vue humain en effectuant une évaluation "Human-Grounded" (basée sur l'humain) (Doshi-Velez et Kim, 2017). Pour ce faire, les expressions de scores les plus élevées de chaque lignes sont présentées aux experts.

Les premières conclusions sont que toutes les expressions de corrélation supérieure à 0,8 sont pertinentes, même si certaines d'entre elles pourraient être améliorées. Ainsi, 0,8 pourrait alors être utilisé comme un seuil d'acceptation en dessous duquel les expressions créées seraient rejetées. En effet, les expressions extraites dont le score est très élevé (supérieur à 0,9) sont souvent similaires entre plusieurs apprentissages successifs, et pourraient être utiles telles quelles. En revanche, pour les corrélations inférieures à 0,8, les experts auraient trouvé intéressant d'intervenir pendant le processus d'apprentissage en supprimant, remplaçant ou ajoutant des noeuds ou des feuilles dans les individus évolués pour augmenter leur score. Ce seuil de 0,8 fait écho au seuil statistique défini par J. D. Evans correspondant à une corrélation dite "très élevée" (Evans, 1996).

Après entraînement, les meilleures expressions pour chaque ligne étudiées sont présentées à un expert. Celui-ci a pour tâche de vérifier leur cohérence avec sa connaissance a priori de la zone étudiée, par exemple en s'assurant de la présence dans l'expression des variables d'entrées nécessaires à l'explication du transit sur la ligne (i.e. d'unités correctes et de bonne localisation géographique). En analysant les expressions littérales construites, des groupes d'expressions symboliques sont identifiés et dont les interprétations par les experts sont pertinente au regard du problème étudié :

— certaines expressions sont des variations autour de la formule de la puissance apparente $\sqrt{p^2+q^2}$ (par exemple $\frac{\sqrt{p^2+q^2}}{v}$ ou $\sqrt{(p1+p2)^2+(q1+q2)^2}$).

— D'autres expressions sont définies comme la somme, le minimum ou le maximum d'une liste de puissances actives, modifiées en utilisant la valeur absolue, les parties positives ou négatives. Dans certains cas, ces non-linéarités sont également utiles pour traiter les valeurs aberrantes provenant d'erreurs de capteur ou de simulation.

— Quelques expressions sont également une agrégation (somme ou différence) des transit transfrontaliers (par exemple, $i\_frontiere_1 - abs(i\_frontiere_2)$). Ces catégories de combinaisons semblent indiquer que la ligne cible correspondante est plus sensible à un phénomène *global* que les autres lignes. Ces expressions mettent en évidence quelles lignes sont sensibles à un transit élevé entrant ou sortant de la zone.

— En utilisant une version de la grammaire avec des variables topologiques, certaines de ces expressions incluent parfois plusieurs variables combinées en utilisant la topologie du réseau comme conditions, telles que :

```
si{ligne_1 est connectée} alors{surveiller egi_variable_1}
else{surveiller egi_variable_2}
```

4.6 Amélioration incrémentale de la grammaire

4.6.1 Protocole expérimental

L'expérience précédente nous a permis d'identifier la métrique de corrélation comme fitness pertinente pour l'apprentissage d'expressions symboliques des transits électriques. Une comparaison avec d'autres algorithmes interprétables nous a permis de réaliser une première évaluation de l'interprétabilité des expressions construites. Cette évaluation de l'interprétabilité nous confirme ainsi l'intérêt de l'utilisation d'une approche d'EG dans le cadre applicatif étudié.

Dans cette deuxième expérience, nous souhaitons maintenant évaluer l'effet de l'intégration des commentaires utilisateurs sur les expressions créées par l'algorithme génétique. Pour cela, nous comparons les expressions symboliques apprises avec deux grammaires : une première grammaire simpliste correspondant à celle choisie pour initialiser celle de l'Algorithme 4, notée G_0, et la deuxième grammaire G_{finale}, similaire à celle déjà testée dans la première expérience et construite incrémentalement à partir d'interactions successives et d'intégration des commentaires obtenus à la fin des apprentissages. Les deux grammaires se distinguent principalement par le fait que G_0 comporte moins de règles grammaticales que G_{finale}, certaines unités de sortie ont été interdites dans G_{finale} mais également d'autres fonctions et unités ont été rajoutées à G_0 pour construire G_{finale}. Les deux grammaires sont détaillées en annexe A. Notons que dans ces expériences, nous utilisons $< e >$ comme symbole de départ et ne prenons pas en compte les variables de topologie.

Comme données, nous utilisons pour l'apprentissage uniquement 75% des données de l'année 2014, sélectionnées aléatoirement. Le reste des données de l'année 2014 serviront de test, et celles des années suivantes auront pour but de voir la capacité de généralisation des meilleures expressions trouvées sur des années non-rencontrées pendant l'apprentissage. Les hyperparamètres utilisés dans cette expérience sont résumés dans le tableau 4.3. Dans cette seconde expérience, une deuxième méthode d'initialisation a été choisie après validation-croisée.

Dix apprentissages évolutionnaires sont réalisés pour chacune des 24 lignes identifiées comme *sensibles* : deux types de grammaires G_0 et G_{finale} sont testées 10 fois par ligne.

4.6.2 Comparaisons entre grammaires

Score de corrélation Comparons dans un premier temps les expressions obtenues vis-à-vis de leur corrélation de Pearson, représentées en figure 4.5. Sur cette figure, les résultats pour chaque ligne étudiée sont séparées sur l'axe des abscisses et représentés le long d'une ligne en pointillés verticale. Les meilleures expressions de chaque apprentissage sont testées sur les 25% restants des données de l'année 2014 et évaluées par leur corrélation de Pearson. Nous représentons par une boîte à moustache les résultats associés à chaque type de grammaire, avec pour une ligne donnée :

— en bleu du côté gauche de la ligne en pointillés le score des individus de meilleure fitness, rencontrés lors des apprentissages avec la grammaire initiale G_0

Paramètre	Valeur
Générations	2000
Taille de population	1000
Initialisation	Random Valid no Duplicates (RVD) (Nicolau, 2017)
Taille max du génome	50
Sélection	Tournoi avec une taille de 3
Croisement	Type variable 2 points (probabilité 0.9)
Mutation	Type int flip per codon (probabilité 0.1)
Élitisme	Top 10
Remplacement	Générationnel
Fitness	Corrélation de Pearson

TABLE 4.3 – Paramètres évolutionnaires de la 2ème expérience.

— en orange du côté droit de la ligne en pointillés le score des individus de meilleure fitness, rencontrés lors des apprentissages avec la grammaire finale G_{finale}

Dans un premier temps, notons, qu'il n'y a pas de dégradation significative du score pour les lignes déjà correctement prédites : notamment les lignes 5, 6, 13, 15 et 23. Par ailleurs, pour d'autres lignes (telles que les lignes 1, 10, 17 et 18), la grammaire finale permet d'améliorer significativement les expressions trouvées.

Néanmoins, nous pouvons constater une légère dégradation des performances obtenues pour les lignes 3, 4, 11, 14, 19, 20 et 24. Après analyse des expressions construites pour ces lignes, nous avons identifié que pour deux d'entre elles, les expressions sont basées uniquement sur des puissances réactives qui, bien qu'étant correctes physiquement, ne sont que très peu utilisées par les experts du fait de leur rôle moins important dans l'apparition des contraintes que la puissance active. En pratique pour la gestion du réseau, les mesures de puissance réactives sont donc souvent ignorées par les opérateurs, qui les voient comme un facteur du second ordre. Dans les travaux présentés dans ce chapitre, les puissances réactives ont par conséquent été supprimées de la grammaire pour construire la grammaire G_{finale}.

Les éléments présentés ci-dessus penchent en faveur de l'utilisation de plusieurs grammaires, plutôt qu'une grammaire généralisée pour toutes lignes. En effet, il est plus difficile de construire une grammaire fonctionnant aussi bien pour toutes les lignes, en particulier lorsque toutes les règles sont équiprobables. En effet, les améliorations apportées à la grammaire ont été faites en prenant en compte les problèmes de représentation pour les lignes les moins bien modélisées. De fait, l'espace fonctionnel généré par la grammaire finale est spécifié pour mieux modéliser ces lignes plus complexes.

Complexité des solutions Dans un deuxième temps, nous comparons dans la figure 4.6 la complexité des meilleures solutions de chaque apprentissage, ligne par ligne. Nous appelons ici *complexité* la partie "utile" du génome de chaque individu pour la traduction en phénotype, i.e. le nombre de règles grammaticale utilisées pour construire l'expression symbolique. La figure 4.6 se lit de manière similaire à la figure 4.5 avec pour chaque ligne, le long de la ligne en pointillé : à gauche en bleu la complexité des expressions apprises avec la grammaire initiale et à droite en orange celle des expressions apprises avec la grammaire finale. Pour la quasi-totalité des lignes, les expressions construites sont plus complexes avec la grammaire initiale qu'avec la grammaire finale. Il semblerait donc que l'ajout de nouvelles règles grammaticales, pertinentes pour le problème étudié, ait permis de construire des individus génétiques plus simples.

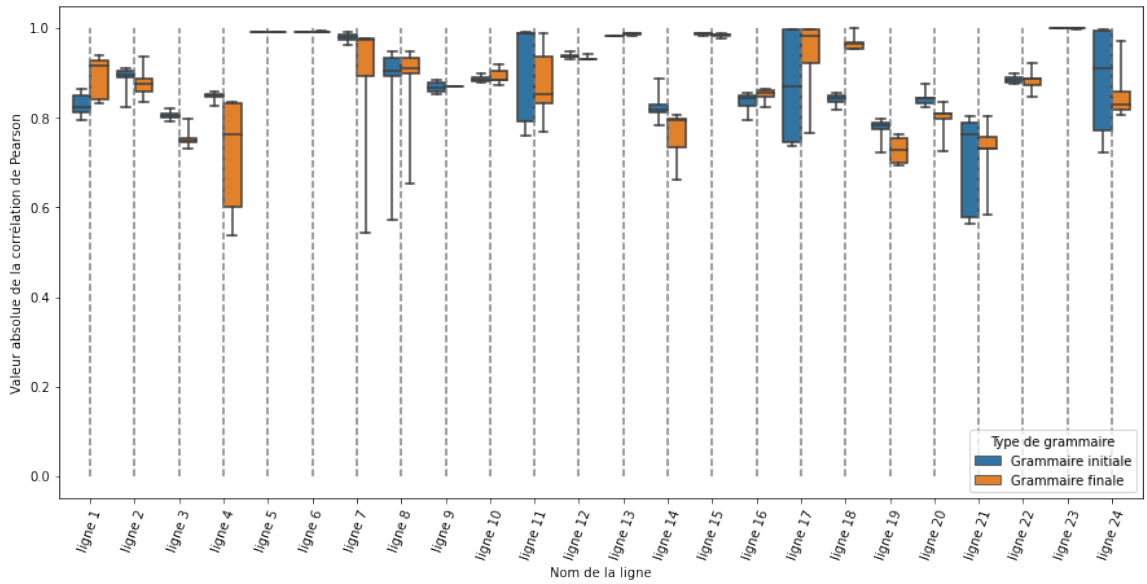


FIGURE 4.5 – Comparaison du score de corrélation de Pearson entre la première grammaire utilisée et la grammaire finale après mise à jour par les experts.

En combinant les résultats des figures 4.5 et 4.6, il semblerait également que contraindre la taille du génome à 50, comme réalisé dans l'expérience 2 ne nuise pas à la capacité de l'algorithme de construire des expressions pertinentes et d'une complexité acceptable.

Temps de calcul Enfin dans un dernier temps, la figure 4.7 compare le temps d'exécution de l'algorithme avec les deux grammaires testées, ligne par ligne. Elle se lit comme précédemment pour les figures 4.5 et 4.6 avec en bleu les résultats de la grammaire initiale G_0 et en orange ceux de la grammaire finale G_{finale}. Le temps de calcul moyen pour G_0 est de 97 minutes, là où celui pour la grammaire G_{finale} est de 89 minutes Pour la quasi-totalité des lignes (excepté les lignes 2, 6 et 18), le temps d'exécution moyen est plus court lorsque l'on utilise la grammaire G_{finale}. Cette réduction du temps de calcul peut aller jusqu'à 22 minutes, dans le cas de les lignes 16 et 19. L'apprentissage plus rapide avec G_{finale}, en partie liée à la diminution de la complexité des individus génétiques, est également un atout dans un cadre industriel pour permettre par exemple, à budget temporel fixé, de réaliser un plus grand nombre d'apprentissages, ou bien de faire évoluer une population sur un plus grand nombre de générations.

4.6.3 Évaluation de la généralisabilité des expressions

Pour évaluer la pertinence des indicateurs créés par apprentissage sur l'année 2014, ceux-ci sont évalués sur les années 2014 à 2017, afin de comparer leur performance au cours du temps et leur robustesse aux évolutions du réseau. Les résultats de cette analyse sont synthétisés dans la figure 4.8. Sur cette figure, pour chaque ligne, les individus de meilleure fitness trouvés sur 10 apprentissages avec la grammaire finale G_{finale} sont évalués sur les données des années 2014 à 2017. Des résultats similaires ont été obtenus avec les individus obtenu par la grammaire G_0.

Pour une ligne donnée, de chaque côté de la ligne en pointillés correspondante sont représentés : à gauche les années 2014 (bleu) et 2015 (jaune), et à droite les années 2016 (vert) et 2017 (rouge). Nous pouvons y constater trois phénomènes :

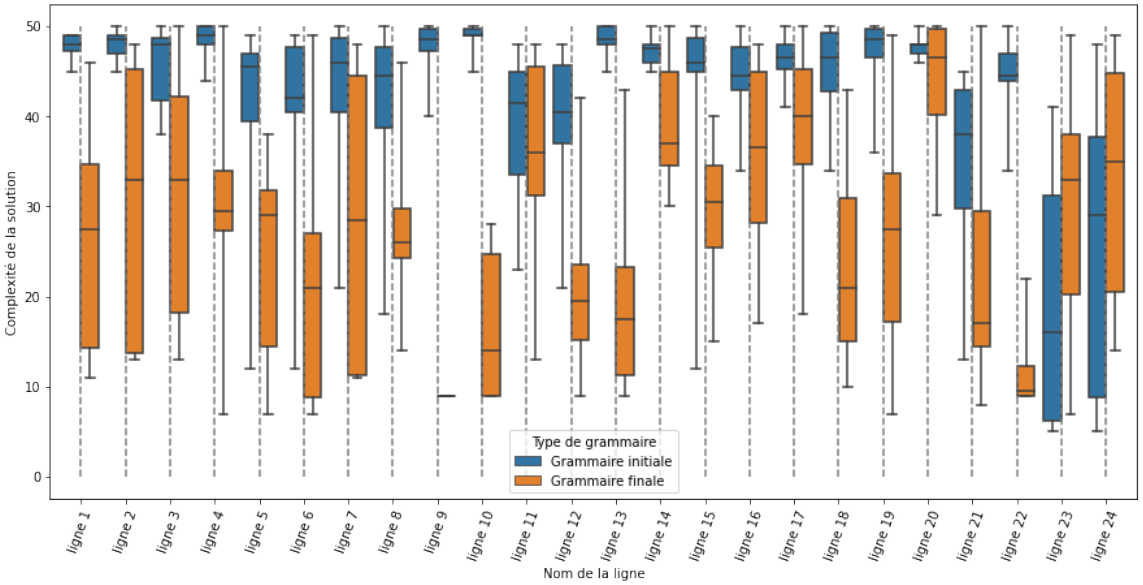


FIGURE 4.6 – Comparaison de la complexité des expressions construites pour les individus construits respectivement avec G_0 (bleu) et G_{finale} (orange).

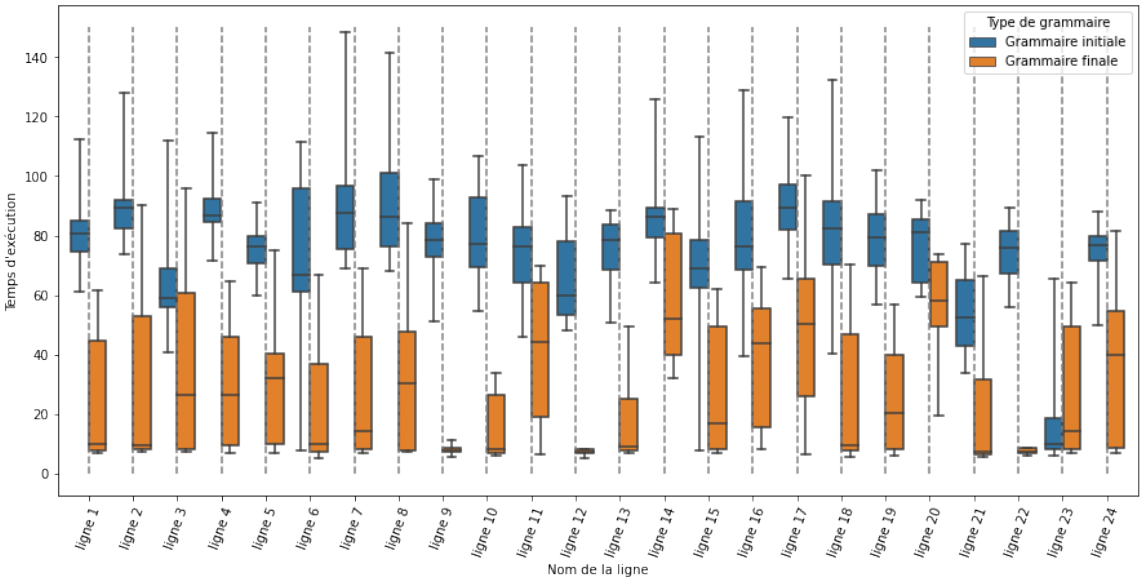


FIGURE 4.7 – Comparaison du temps d'exécution (en minutes) de l'algorithme entre la première grammaire utilisée et la grammaire finale après la mise à jour réalisée par les experts.

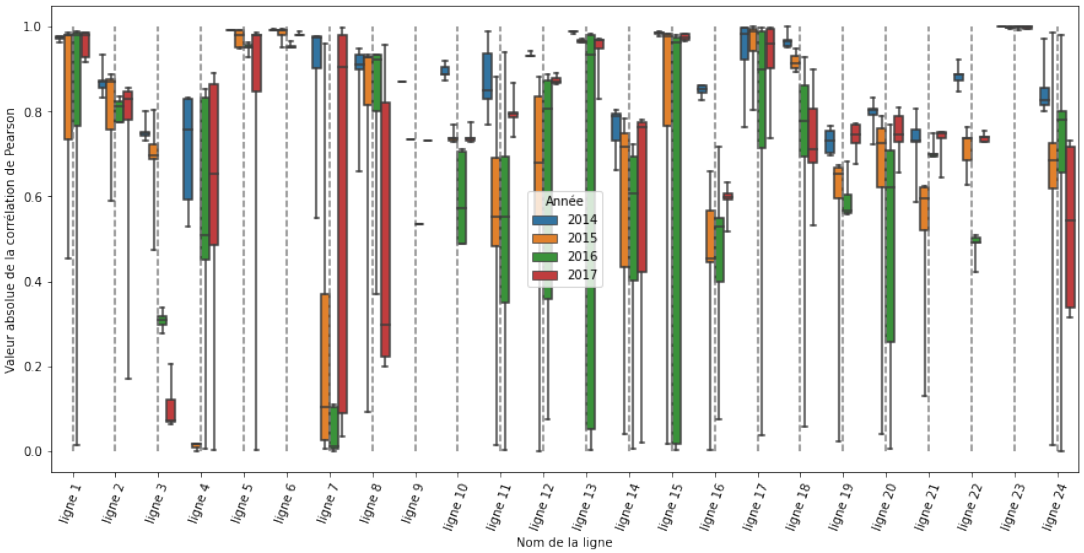


FIGURE 4.8 – Évaluation par corrélation de Pearson des individus génétiques, appris sur 75% des données de 2014 et testées sur 2014 à 2017.

— dans un premier temps, pour les lignes 1, 6, 15, 17 ou 23, nous pouvons voir que les que le score obtenu en médiane est toujours élevé et proche de 1, quelque soit l'année utilisée pour le test. L'expression utilisée comme indicateur reste pertinente et sont utilisation est généralisable aux années suivantes de l'historique.

— dans un second temps, pour des lignes telles que les numéros 3, 8, 18, ou encore 24, la performance de l'expression utilisée comme indicateur tend à diminuer, plus ou moins rapidement au fil des années. Cet élément semble mettre en évidence l'influence de l'évolution progressive des infrastructures du réseau électrique sur l'apprentissage. La distribution d'apprentissage (ici l'année 2014) est différente de celle de test et ceci pourra être important et intéressant pour apprendre des indicateurs sur des contextes particuliers : des indicateurs spécifiques pour une année de température "froide" ou une année "chaude", sachant que la température a un effet significatif sur le comportement des utilisateurs (Marot *et al.*, 2019). Cette dégradation implique par ailleurs, qu'il sera nécessaire de ré-évaluer régulièrement (par exemple tous les ans), les expressions créées avec de nouvelles données du réseau.

— enfin, un dernier type de comportement, illustré par les lignes 2, 4, 7, 9 à 14, 16, ou 19 à 22 semble montrer des phénomènes cycliques au cours du temps. En d'autres termes certains indicateurs pertinents pour l'année sur laquelle ils ont été appris, peuvent ne pas être aussi significatifs l'année suivante, mais rester intéressant pour l'étude de transits futurs. Par exemple, une expression utilisant un terme en lien avec la production hydraulique d'une centrale peut avoir une importance différente selon la quantité de pluie tombée cette année là. Ce type d'expressions ne sera pas à supprimer mais plutôt à compléter avec des termes additionnels. Ce phénomène cyclique pourrait par ailleurs mettre en évidence le besoin d'apprendre des indicateurs sur différentes années (un apprentissage sur plusieurs années comme dans la première expérience, ou bien plusieurs apprentissages chacun sur une seule année) pour plus de diversité dans les expressions créées.

Bien que ces résultats soient intéressants et permettent d'identifier comment générer une plus grande variété d'expressions utilisées comme indicateurs, il est également pertinent de chercher

à améliorer la stratégie d'apprentissage pour fournir des indicateurs plus généralisables. Dans ce sens, deux pistes ont été explorées quant à la l'amélioration de la généralisation des expressions :

1. l'utilisation d'un terme de pénalisation dans la fonction de fitness ;
2. l'apprentissage sur plusieurs années de données.

4.7 Amélioration de la généralisabilité des expressions

4.7.1 Pénalisation de la complexité dans la fitness

Afin d'améliorer la généralisabilité sur plusieurs années des expressions symboliques créées, nous avons souhaité tester une solution alternative à l'utilisation d'une petite taille de génome. Dans cette optique, la taille du génome est augmentée et un terme de pénalisation de la complexité des solutions est introduit, dans le but de supprimer autant que possible les termes "inutiles" des expressions. En effet, dans certaines des expériences précédentes, le manque de généralisabilité de l'expression était du à la présence de terme "fantômes". Il s'agissait de termes qui n'ont pas d'effet sur les données d'apprentissage, mais qui peuvent être néfastes sur de nouvelles données.

Nous avons souhaité tester une contrainte de pénalisation sur la complexité des expressions construites, plus forte que celle utilisée précédemment dans l'équation 4.1. Pour cela, nous rajoutons cette fois-ci un terme multiplicatif au terme de corrélation. Ce terme, noté $S_b(\hat{y})$, a pour but de pénaliser les expressions de grande taille (Bojarczuk *et al.*, 2004). Il se définit de la manière suivante :

$$S_b(\hat{y}) = \frac{\kappa\_seuil^k - 0.5 \times (\kappa(\hat{y})^k + 1)}{\kappa\_seuil^k - 1}, k \in \mathbb{N} \tag{4.2}$$

Le terme $\kappa(\hat{y})$ correspond à la complexité de $\hat{y}$ et est défini comme étant le nombre de règles grammaticales utilisées pour construire l'expression. Il équivaut à la taille de la partie "utile" du génome pour la construction de l'expression. k représente un paramètre de la fonction $k \in \mathbb{N}$. Nous utilisons dans notre cas $k = 3$ de manière à ne pas trop pénaliser les individus de petite taille. Ainsi, avec une taille maximale de génome de $\kappa\_seuil = 50$, nous conservons $S_b(\hat{y}) > 0.9$ tant que $\kappa(\hat{y}) < 30$. En s'appuyant sur la corrélation de Pearson r en valeur absolue, la fonction de fitness devient alors :

$$fitness(y, \hat{y}) = abs(r(y, \hat{y})) \times S_b(\hat{y}) \tag{4.3}$$

Afin de comparer l'apprentissage avec et sans ce terme de pénalisation, 4 apprentissages sont réalisés utilisant tous les paramètres définis dans la tableau 4.3 :

— un apprentissage avec une taille maximale de génome de 50 sans S_b. Cet apprentissage servira de référence.

— un apprentissage avec une taille maximale de génome de 50 avec S_b. Cet apprentissage permettra de comparer deux tailles apprentissages lorsque l'apprentissage est contraint par S_b. Il sera à comparer avec l'utilisation d'une taille plus grande de 100.

— un apprentissage avec une taille maximale de génome de 100 sans S_b. Cet apprentissage servira à identifier le score maximal que l'on peut obtenir avec une taille de génome de 100.

— un apprentissage avec une taille maximale de génome de 100 avec S_b. Sachant que la pénalisation de la complexité induit des individus moins complexes, cet entraînement sera à comparer avec les deux précédents pour évaluer l'effet de la pénalisation S_b.

Les résultats de cette expérience sont résumés dans la figure 4.9. Dans cette figure, 4 graphiques (un par année) représentent les valeurs absolues des coefficients de corrélation ligne

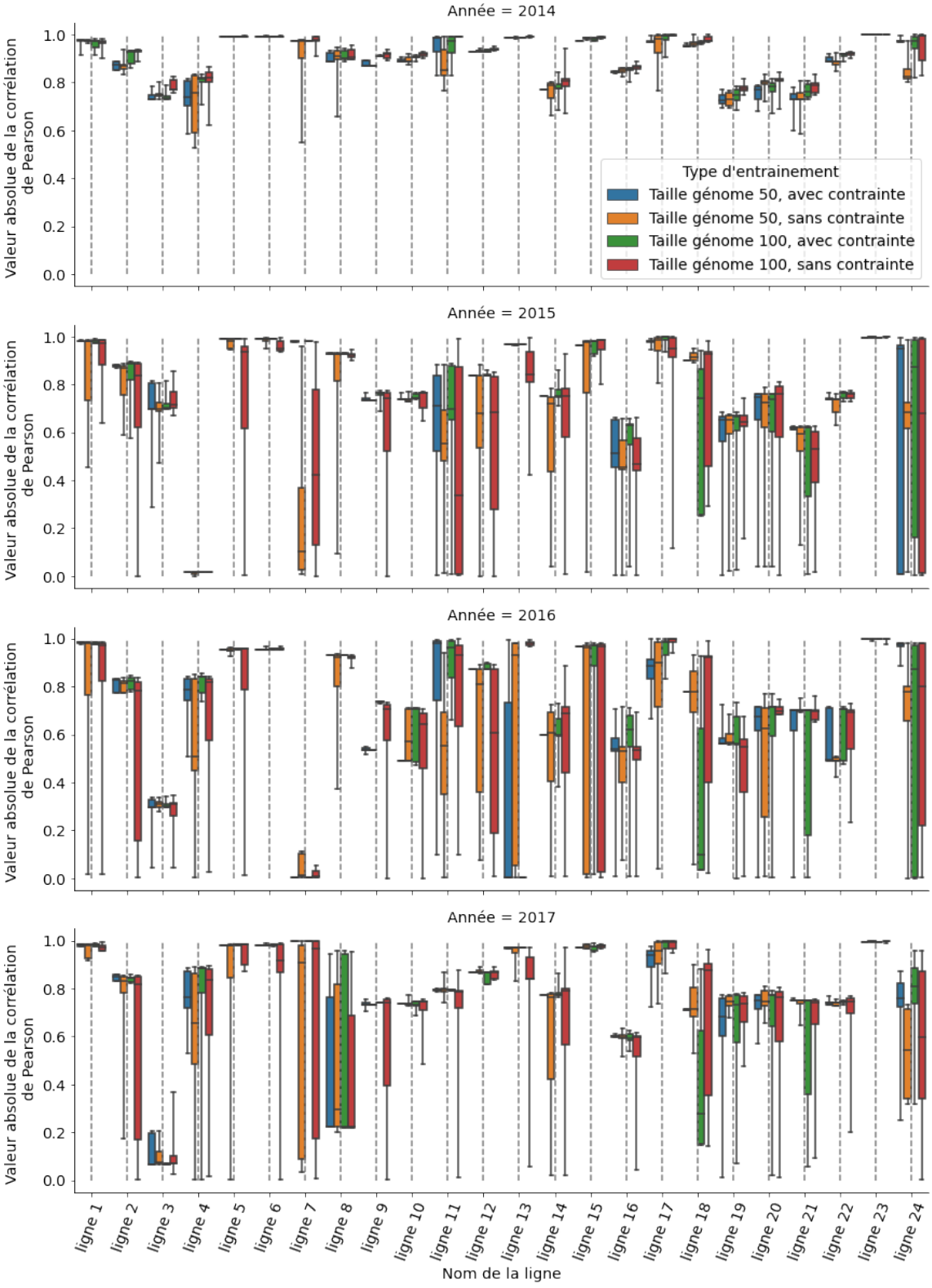


FIGURE 4.9 – Effet de la taille du génome et de la pénalisation sur l'apprentissage, évalué sur les données de 2014 à 2018. Apprentissage réalisé sur 75% des données de l'année 2014.

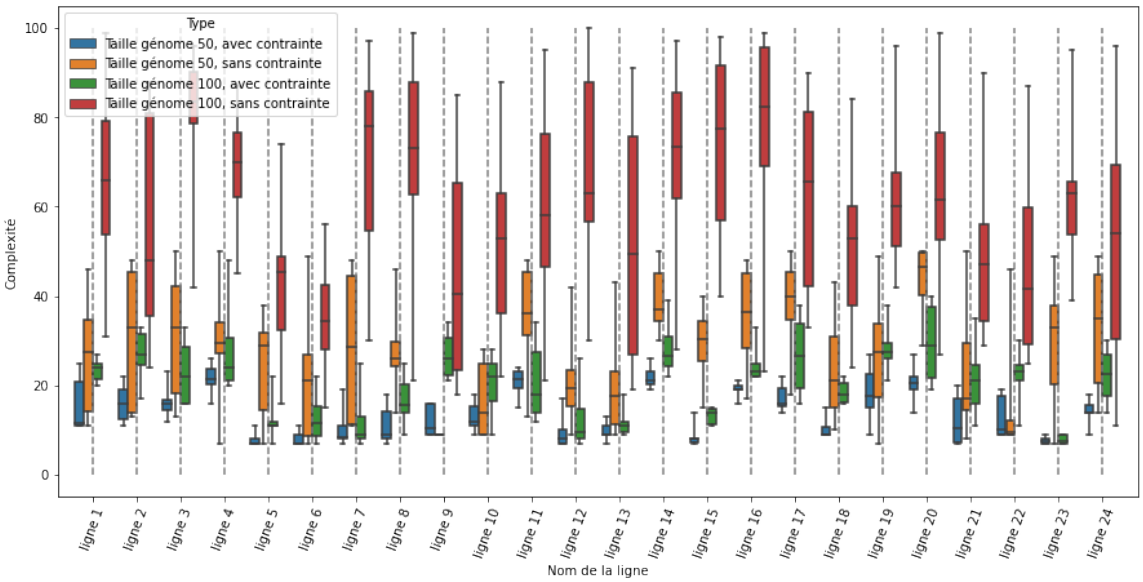


FIGURE 4.10 – Comparaison de la complexité des solutions en fonction de la taille du génome et de la pénalisation sur l'apprentissage, évalué sur les données de 2014 à 2018. Apprentissage réalisé sur 75% des données de l'année 2014.

par ligne et colorisés différemment pour chaque type d'entraînements. Tout d'abord, en s'intéressant aux résultats de l'évaluation sur l'année 2014, nous pouvons constater que les 4 types d'apprentissages obtiennent des résultats très similaires pour la quasi totalité des lignes, avec un léger avantage pour les entraînements réalisés sans pénalisation S_b et avec une taille de génome plus grande. Regardons maintenant les années 2015 à 2017, correspondant à des années non rencontrées pendant l'entraînement. Pour ces différentes années, et à l'exception des lignes 8, 11, 13, 18, 21, 24 (selon l'année) nous pouvons constater une forte diminution de la variance des scores lorsque le terme de pénalisation de la complexité est utilisé pendant l'apprentissage. A taille maximale de génome fixe de 100, les scores moyens sont également plus élevé lorsque l'apprentissage est fait avec le terme de pénalisation de la complexité.

Par ailleurs, en comparant la complexité des solutions obtenues avec les différents entraînements, représentés en figure 4.10 nous pouvons constater que l'apprentissage avec une de taille maximale de génome de 100 et une pénalisation de la complexité permet d'obtenir des solutions d'une complexité proche de celle obtenue avec l'entraînement sur une taille de génome de 50. Ces résultats montrent donc qu'il serait alors possible d'augmenter la taille maximale du génome, tout en conservant une complexité réduite conservant le critère de lisibilité des solutions. Ces résultats sur la complexité des solutions laisse par ailleurs présager qu'il soit possible d'augmenter encore la taille du génome.

Dans un second temps, nous souhaitons comparer la configuration d'apprentissage utilisant une taille de génome de 50 sans terme de contrainte ($config_1$), avec la configuration d'apprentissage utilisant une taille de génome de 100 avec contrainte ($config_2$). L'objectif final est d'identifier la configuration qui offre la meilleure généralisabilité en testant pour chaque ligne et chaque année quelle configuration obtient les meilleurs scores. Pour cela, nous utilisons le test U de Mann-Whitney (Mann et Whitney, 1947). Ce test statistique non-paramétrique est appliqué successivement chaque année et chaque ligne électrique pour comparer les distributions des scores de corrélation pour les configurations $config_1$ et $config_2$, en réalisant deux tests unilatéraux

		Type d'entraînement	
		Taille génome 50, sans contrainte	Taille génome 100, avec contrainte
Année de test	2014	4	5
	2015	1	7
	2016	2	5
	2017	3	6

TABLE 4.4 – Synthèse des résultats des tests U de Mann-Whitney sur la comparaison des apprentissages pour une taille de génome de 50 sans contrainte et une taille de génome de 100 avec contrainte. Pour chaque expression apprise, soit avec une taille de génome de 50 sans contrainte ou avec une taille de génome de 100 avec contrainte, l'expression est évaluée sur chaque année de données de 2014 à 2017. Puis, pour chaque année, les corrélations obtenues sur les deux types d'apprentissages sont comparées via le test de Mann-Whitney. Le tableau compte le nombre de lignes (sur les 24 étudiées) où chaque type d'apprentissage obtient de meilleurs résultats.

avec une p-value de 0.05. Dans le premier test unilatéral d'égalité des distributions (H0), nous considérons l'hypothèse alternative (H1) d'une distribution des résultats de la première configuration $config_1$ stochastiquement inférieure aux résultats de la deuxième configuration $config_2$. Un deuxième test unilatéral d'égalité des distributions (H0'), considère ensuite l'hypothèse alternative (H1') d'une distribution des résultats de la première configuration $config_1$ stochastiquement supérieure aux résultats de la deuxième configuration $config_2$. Les résultats de ces tests sont ensuite synthétisés dans le tableau 4.4 en comptant, par année, le nombre de ligne électrique pour lesquelles une configuration est meilleure que l'autre (sur 24). Lorsque H0 et H0' sont vérifiés, pour une ligne et une année donnée, les deux configurations sont équivalentes et ne sont pas comptées. En analysant les résultats du tableau 4.4 pour l'année 2014, nous pouvons remarquer que la configuration 2 est meilleure que la configuration 1 pour 5 des 24 lignes. Cette année là, la configuration 1 n'obtient de meilleures expressions que pour 4 lignes. De plus, la configuration 2 semble mieux généraliser sur les années suivantes car elle obtient de meilleurs scores sur les années 2015 à 2017 avec une amélioration des scores pour respectivement 7, 5 et 6 lignes (contre seulement 1, 2 et 3 lignes respectivement avec la configuration 1.

Par ailleurs, nous avons identifié dans la figure 4.10 que l'apprentissage avec une de taille maximale de génome de 100 et une pénalisation de la complexité permet d'obtenir des solutions d'une complexité proche de celle obtenue avec l'entraînement sur une taille de génome de 50. Ainsi, l'utilisation du terme de pénalisation de la complexité dans la fitness ne nuit que très peu à la performance des expressions trouvées, tout en offrant des solutions plus simples. Avec cette stratégie de pénalisation, il est donc possible d'augmenter la taille de génome pour obtenir de meilleurs résultats, sans accroître la complexité des solutions.

4.7.2 Augmentation du nombre d'années d'apprentissage

Dans l'objectif de construire des expressions pouvant être réutilisées d'une année sur l'autre, nous avons souhaité tester une seconde stratégie consistant à apprendre ces expressions sur un plus grand nombre d'années. A ces fins, testons l'apprentissage apprenant des expressions symboliques sur les 24 lignes selon deux types de données d'apprentissage : soit sur 75% des données de l'année 2014 uniquement, soit sur 75% des données des deux années 2014 et 2015. Nous utilisons ici dans les deux cas le même algorithme d'EG avec la grammaire G_{finale} et les paramètres définit dans la section 4.6, sans pénalisation de la complexité. Pour chaque ligne et

		Année d'entraînement	
		2014	2014 et 2015
Année de test	2014	6	1
	2015	0	17
	2016	1	3
	2017	1	10

TABLE 4.5 – Synthèse des tests U de Mann-Whitney en fonction du nombre d'années d'apprentissage et évalué sur les années 2014 et 2017. Pour chaque expression apprise, soit sur les données de l'année 2014 ou de l'année 2014-2015, l'expression est évaluée sur chaque année de données. Puis, pour chaque année, les corrélations obtenues sur les deux types d'apprentissages sont comparées via le test de Mann-Whitney. Le tableau compte le nombre de lignes (sur les 24 étudiées) où chaque type d'apprentissage obtient de meilleurs résultats.

chaque type de données d'apprentissage, 10 expressions symboliques sont créées. Nous évaluons ensuite ces expressions symboliques sur toutes les données, année par année, de 2014 à 2017. Les résultats sont synthétisés dans la figure 4.11 et le tableau 4.5.

La figure 4.11 représente, année par année les scores de corrélation des différentes expressions, en fonction du nombre d'année d'apprentissage. L'axe des abscisses représente le nom de la ligne et l'axe des ordonnées le score de corrélation. Le tableau 4.5 représente les résultats du test statistique U de Mann-Whitney (Mann et Whitney, 1947), utilisé pour comparer les distributions des deux entraînements à ligne et année identique, et tester si l'un des deux types d'apprentissage est statistiquement meilleur ou moins bon que l'autre. Le test est utilisé avec une p-value de 0.05 et testé successivement avec l'alternative d'une médiane plus grande et plus petite. Le tableau récapitule le nombre de fois, i.e. de lignes, où chaque type d'apprentissage obtient de meilleurs résultats.

Tout d'abord, en regardant l'année 2014, nous pouvons voir dans la figure 4.11 que les deux apprentissages obtiennent des scores de corrélation élevés. Cependant, la première ligne du tableau 4.5 souligne également que l'apprentissage sur l'année 2014 obtient un meilleur score pour 4 des 24 lignes. Bien que ceci soit attendu, (car nous testons sur une grande partie des données d'entraînement), notons que la différence est entre les deux méthodes est significative pour 4 lignes.

Dans le cas de l'année 2015, le tableau 4.5 indique cette fois-ci que l'apprentissage sur 2014-2015 est plus performant pour 17 des 24 lignes. La figure 4.11 met également en évidence une réduction de la variance des résultats pour quasi-totalité des lignes. Notons à nouveau ici que dans le cas de l'apprentissage sur 2 années de données, nous testons en partie sur les données d'entrainement et il semble donc normal d'avoir de meilleurs résultats.

Intéressons nous maintenant plus particulièrement aux années 2016 et 2017. Ces années correspondent à des données non-rencontrées pendant l'entraînement pour les expressions apprises sur 2014 ou 2014-2015. D'après la figure 4.11, pour 2016 et 2017, les scores de corrélation semblent être plus élevés et avoir une variance plus faible, dans le cas des expressions apprises sur deux années de données. Le tableau 4.5 confirme cette information en précisant que l'apprentissage sur deux années améliore les scores moyens pour 3 lignes en 2016 et 10 lignes en 2017. Pour toutes les autres lignes les scores moyens sont considérés comme similaires.

Grâce à la comparaison d'apprentissages sur 1 ou 2 années de données, nous avons pu mettre en évidence l'intérêt d'apprendre sur plus de données pour créer des expressions ou variables

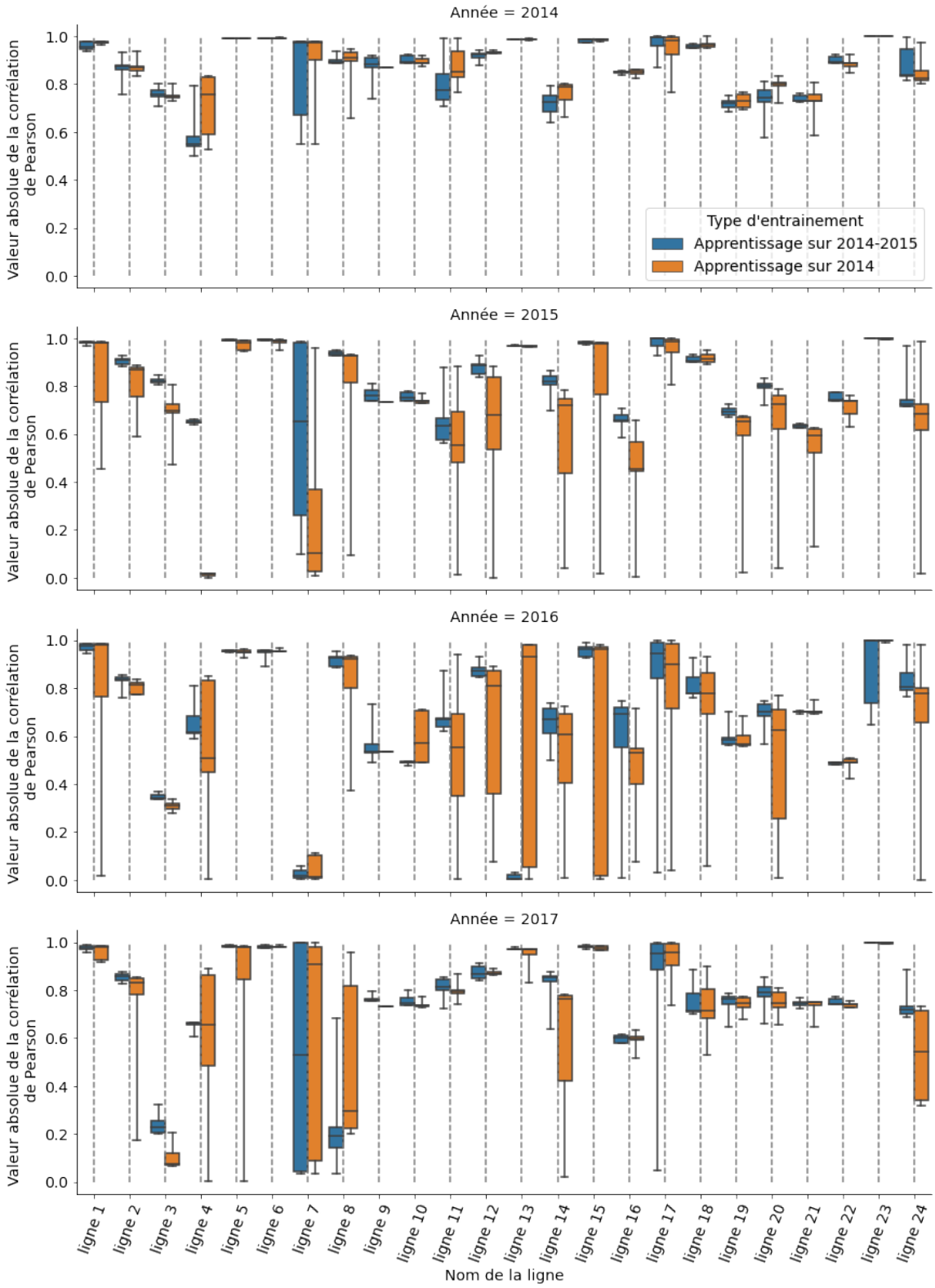


FIGURE 4.11 – Comparaison le capacité de généralisation des expressions apprises sur une ou deux années de données, évaluées sur les années 2014 à 2017.

qui se généraliseront mieux à de nouvelles données. Cependant, ce gain en généralisation est à mettre en regard du temps de calcul des deux types d'apprentissages. Dans nos expériences, le temps de calcul pour l'apprentissage sur une année de données était d'environ 85 minutes, là où l'apprentissage sur deux années de données nécessite en moyenne 250 minutes, multipliant presque par 3 le temps d'apprentissage.

4.8 Conclusion du chapitre

Dans ce chapitre, nous avons montré que la méthode d'évolution grammaticale interactive EGI nous a permis d'obtenir des résultats prometteurs sur *l'extraction de variables interprétables* sous forme d'expressions symboliques pour les transits électriques. Nous avons marqué la première étape avec la construction, puis la validation qualitative et quantitative d'une grammaire non-contextuelle personnalisée, capable d'inclure de manière interactive certaines connaissances du système électrique. Nos expériences fournissent également un aperçu de l'interprétabilité de notre méthode d'un point de vue "humain" et "fonctionnel", ainsi qu'une analyse quantitative de l'amélioration incrémentale de grammaire. Les résultats obtenus pointent également sur la nécessité de ré-actualiser à intervalles réguliers les expressions symboliques, tout en gardant en mémoire les expressions déjà construites pour qu'elles puissent éventuellement être réutilisées dans le futur. Deux pistes sont également explorées pour permettre d'apprendre des expressions généralisables : 1) par apprentissage avec contrainte de complexité ou 2) sur plusieurs années de données.

En introduisant l'expertise et les propriétés physiques dans les règles grammaticales, nous sommes parvenus à obtenir des expressions explicables d'un point de vue métier. Certaines d'entre elles ont également été jugées suffisamment pertinentes par les opérateurs de systèmes électriques pour être utilisables dans les outils existants. Cependant, pour quelques lignes, les variables cibles étaient encore difficiles à traiter avec l'approche proposée. En effet, pour ces quelques lignes, la création d'une seule variable n'est peut être pas suffisant pour prendre en compte la variabilité au cours du temps des valeurs d'injection et de topologie. Nous envisageons que la construction d'un espace multidimensionnel et conditionné par des éléments topologiques adéquatement choisis pourrait augmenter la représentativité des expressions construites. Dans cette optique, nous avons pour cela déjà inclus des variables topologiques dans la grammaire. Cependant, à cause du grand nombre de topologies possibles sur le réseau, ce n'est pas suffisant pour identifier les conditions les plus pertinentes. L'utilisation de méthodes dédiées à l'analyse de l'effet des topologies (Marot *et al.*, 2018b) sur les transits semble plus prometteur.

Comme nous l'avons précisé au début du chapitre, le choix d'une grammaire avant le début de l'évolution influence la qualité de l'apprentissage. Par exemple, dans le cas de grammaires explosives telle que celles utilisées dans ce chapitre, beaucoup d'individus invalides ou très complexes sont générés. Une solution pour palier cela est l'utilisation d'une grammaire probabiliste et dont les probabilités sont apprises itérativement (Sotto et de Melo, 2017). Nous avons donc identifié que l'apprentissage des probabilités associées à chaque règle grammaticale pourrait améliorer une exploration plus précise de l'espace, cette direction sera explorée dans le chapitre 5.

Par ailleurs, comme ces nouvelles fonctionnalités sont destinées à être utilisées par les opérateurs, une étape d'interaction pendant l'apprentissage semble pertinente. Plus fréquente, cette interaction consiste à introduire l'interactivité avec des non-experts en apprentissage automatique, directement au sein d'exécutions évolutionnaires (Amershi *et al.*, 2014). Nous leur permettrions ainsi de fournir des informations techniques qui pourraient aider de manière significative à

créer une représentation plus pertinente par exemple en sélectionnant/ajoutant/supprimant des individus. Cette voie sera explorée dans le chapitre 6.

Enfin, les travaux présentés dans ce chapitre, ont été testés sur des données réelles du réseau électrique dans le cas en "N". Les outils développés sur ces données pourront être par la suite évalués sur le cas et les données en "N-1" décrits dans le chapitre 2. Nous projetons également d'appliquer la méthode et la grammaire développées à des données d'une zone géographique plus large, uniquement sur les lignes électriques THT. Cette perspective nous rapproche de la réalisation d'une évaluation "applicative"(Doshi-Velez et Kim, 2017) de notre méthode avec des humains sur une tâche plus complexe. Enfin, afin de publier des résultats reproductibles sur des cas de test de systèmes électriques et de mettre notre code en libre accès, nous prévoyons in fine d'utiliser la librairie open-source Grid2Op (Donnot, 2020a) développé pour tester les stratégies d'apprentissage automatique pour les opérations du réseau électrique.

Chapitre 5

Apprentissage par Renforcement pour la Régression Symbolique Guidée par la Grammaire

5.1 Introduction

Comme nous l'avons mis en évidence dans le chapitre précédent, l'utilisation d'une grammaire en RS (Régression Symbolique) permet de représenter des contraintes physiques et métiers fortes, grâce auxquelles un algorithme d'EG (Evolution Grammaticale) peut ensuite créer des variables qui respectent ces contraintes. Cependant, toutes les règles de la grammaire ne sont pas utilisées de manière équivalentes dans la génération d'expressions. Il est alors intéressant de pondérer l'utilisation de certaines règles grammaticales en fonction de leur adéquation avec la variable cible à trouver. Partant de ce point, nous souhaitons disposer d'une méthode permettant de sélectionner successivement les meilleures règles afin de parcourir de manière privilégiée les zones utiles de l'espace de recherche.

Dans ce chapitre, nous présentons une méthode appelée "Reinforcement Based Grammar Guided Symbolic Regression", abréviée en, *RBG2-SR* permettant d'aborder la RS avec une approche d'apprentissage par renforcement profond utilisant un espace d'actions défini par une grammaire non contextuelle au format Backus-Naur Form (BNF) (Knuth, 1964). Une distribution sur l'espace des règles grammaticales est apprise, permettant de choisir pour chaque action la règle grammaticale la plus adaptée. La grammaire limite toujours l'espace des solutions aux seules solutions viables dans le domaine d'expertise. Du fait de la difficulté de travailler avec des données réelles complexes et bruitées, et afin de pouvoir comparer la méthode RBG2-SR avec d'autres approches de l'état de l'art de la RS utilisant des grammaires, notre méthode est testée sur un ensemble de jeux de données de référence dans le domaine de la RS. Nous montrons que notre approche est plus performante que les autres méthodes testées avec la même grammaire sur les benchmarks testés. Nous proposons également une analyse comparative de plusieurs méthodes avec et sans grammaire sur un jeu de données open-source présentant des contraintes physiques, pour montrer comment notre approche permet de s'appuyer sur une grammaire pour tirer parti de la connaissance experte.

Dans la section 5.2, nous présentons les motivations et les travaux existants qui justifient l'approche proposée. La section 5.3 définit ensuite l'environnement de renforcement et détaille le fonctionnement de l'algorithme utilisé. Enfin, la section 5.4 présente les deux expériences menées ainsi que les résultats.

Algorithme Génétique (AG)	Apprentissage par Renforcement (AR)
génération	itération
population	batch
individu	trajectoire
gène	action
fitness	récompense

Table 5.1 – Tableau de concordance entre AG et AR

5.2 Problématique

La PG (Programmation Génétique), par exemple via l'EG (Évolution Grammaticale), était jusqu'à récemment l'une des méthodes de résolution privilégiées de problèmes de RS. Cependant, l'EG fait l'objet de certaines critiques. L'un des problèmes mis en évidence est la sensibilité des opérations de croisement et de mutation à l'endroit de l'expression où elles sont réalisées. Les croisements et mutations survenant aux premières positions d'un génotype peuvent notamment grandement réduire (mais également augmenter) le score de l'individu (Castle et Johnson, 2010).

Par ailleurs, il existe de nombreux parallèles entre les AGs (Algorithmes Génétiques) et les algorithmes d'Apprentissage par Renforcement (AR). Nous en récapitulons certains aspects dans la tableau 5.1. Plus précisément, d'une part, un AG fait évoluer, au fil des génération, une population constituée d'un ensemble d'individus possédant des gènes dont le phénotype optimise une fonction de fitness. De l'autre, les AR cherchent à apprendre la politique d'un agent en itérant sur une succession d'actions formant des trajectoires apprises de façon à maximiser un score de récompense (reward), et pouvant être appris par ensemble de trajectoires appelé *batch*. Ainsi, conditionnellement au choix d'une représentation *commune* aux deux méthodes de résolution et en passant si besoin par une mise à l'échelle de l'objectif optimisé, il est possible de résoudre avec du renforcement certains problèmes étudiés avec les algorithmes génétiques.

Face à ce constat, et à la faveur du développement de méthodes d'AR pour la RS (Petersen *et al.*, 2021), nous avons souhaité explorer la RS guidée par la grammaire avec des méthodes de renforcement pour étudier les variables physiques proposées en sortie et les comparer avec des approches avec / sans grammaire, et avec / sans renforcement. Il n'existe aujourd'hui que peu de travaux combinant AR et utilisation de la grammaire pour la RS et la création de variables. L'utilisation de grammaires en renforcement ont, par exemple, permis de réaliser du design 2D de logements (Ruiz-Montiel *et al.*, 2013), d'apprendre des politiques d'agents interpretables (Verma *et al.*, 2018) ou de modéliser les règles suivies par l'expert, qui seront ensuite utilisées pour apprendre une politique de renforcement (Araki *et al.*, 2022). Ainsi, en amont de cette thèse, seules quelques approches utilisent des algorithmes de renforcement pour résoudre des tâches de RS sans grammaire (Petersen *et al.*, 2021; Mundhenk *et al.*, 2021) tandis que d'autres approches utilisent des grammaires comme structure d'un environnement de renforcement pour d'autres tâches que la régression symbolique (Mandow *et al.*, 2020; Verma *et al.*, 2018; Drori *et al.*, 2019). Nos travaux visent donc à explorer une grammaire non-contextuelle (GNC), définie pour une tâche de RS, avec un algorithme d'AR.

5.3 Approche proposée

5.3.1 Définition de l'environnement d'apprentissage par reinforcement

Dans cette approche, nous employons un processus de décision de Markov (MDP) (Sutton et Barto, 2018) pour modéliser le problème de RS. Plus précisément, nous choisissons de considérer un processus de décision de Markov partiellement observable (POMDP)(Kaelbling *et al.*, 1998) dans un cadre épisodique fini avec un horizon maximal H. Cette section est consacrée à la définition des principales composantes du POMDP dans un cadre d'apprentissage par renforcement, à savoir : espace d'états *state*, espace d'*actions* et récompense *reward*. Dans la section 5.3.1, nous définissons les espaces d'états et d'actions pour la RS et donnons la définition de la récompense ainsi que ses propriétés. A partir des définitions l'ensemble du POMDP est détaillé. Enfin, la sous-section 5.3.1 détaille comment apprendre une politique utilisée pour générer les probabilités d'actions étant donné l'état actuel.

Espace d'états et d'actions dans un environnement Grammatical

Considérons un cadre d'apprentissage par renforcement où la tâche globale est de trouver une fonction symbolique optimale f^* de sorte que $f^* = \text{argmin}_{f \in F_G} ||y - f(X)||$, avec F_G l'espace de fonctions accessible à partir d'une grammaire donnée G. La grammaire est définie par le tuple $(\sigma_{start}, (\sigma_{nt})_{nt \in NT}, (\sigma_t)_{t \in T}, \rho, \Psi)$ avec σ_{start} le symbole de départ de la grammaire, $(\sigma_{nt})_{nt \in NT}$ un ensemble de non-terminaux, $(\sigma_t)_{t \in T}$ un ensemble de terminaux, ρ les règles de production pour combiner terminaux/non terminaux, et Ψ la probabilité associée à chaque règle de production. Les figures 5.1 et 5.4 de la section 5.4 montrent des exemples de grammaire BNF inspirés des travaux de Sotto et Melo (Sotto et de Melo, 2017).

Avec cette notation, nous proposons de définir la construction de f^* comme un problème de décision séquentiel où un agent choisit itérativement une règle parmi l'ensemble des règles de la grammaire pour construire la fonction f. Dans cet espace grammatical, nous spécifions d'abord le nombre maximal d'étapes pour créer f, appelé l'horizon maximal H. Nous définissons ensuite pour chaque étape $h \in [0, ..., H]$ une *action* a_h comme la *sélection d'une règle* dans la règle de production accessible depuis l'état courant. Nous proposons également de définir l'état ou *state* s_h à l'étape h par $s_h = (a_h^{past}, a_h^{parent}, a_h^{siblings}, d_h, \sigma_h, m_h, \eta_h)$, avec $a_h^{past} = [a_0, ..., a_{h-1}]$ toutes les actions précédemment sélectionnées, a_h^{parent} l'action prise par le parent dans l'arbre de dérivation, $a_h^{siblings}$ l'action prise par chaque soeur ou frère déjà calculé dans l'arbre de dérivation, d_h la profondeur de l'arbre d'expression à l'étape h, σ_h le type de symbole courant à trouver à l'étape h, m_h un masque sur les actions accessibles du symbole d'état courant σ_h et η_h des informations cachées sur l'état courant. Nous appelons noeuds frères et soeurs les noeuds ayant la même profondeur que le noeud actuel, et noeud parent le noeud situé au-dessus du noeud actuel dans l'arbre construit à partir des actions. D'autres définitions de l'espace d'états seront testées dans la sous-section 5.4.2. Initialement, nous définissons l'état par le symbole de départ σ_{start}, les actions précédemment sélectionnées sont une liste vide, m_0 masque les actions inaccessibles du symbole initial, et l'information cachée η_0 est initialisée aléatoirement. Une *trajectoire* d'actions $\tau_k = (a_0^k, ..., a_h^k), h \in [H]$ est associée à chaque fonction symbolique f_k. Nous considérons le cas où la grammaire est choisie et construite de manière à ce qu'il y ait toujours au moins une action accessible à chaque étape.

L'algorithme 5 détaille la procédure d'échantillonnage des épisodes. Un exemple visuel de cet échantillonnage d'expressions est fourni dans la figure 5.1. À partir d'une grammaire donnée

Algorithm 5 Échantillonage d'un épisode, retourne une fonction f par episode.

Require: Taille maximale de l'horizon H, politique π_θ, grammaire G

function ECHANTILLONAGE_D'UN_EPISODE(H, π_θ, G)
  $queue, a^{past}, a^{parents}, a^{siblings}, f \leftarrow$ vide
  $d \leftarrow 0$
  $\sigma \leftarrow$ `get_symbole_de_depart`(G)
  $m \leftarrow$ `get_masque`(σ, G)
  $\eta \leftarrow$ `initialisation_aleatoire`$()$
  for h in H **do**
    $state \leftarrow ((a^{past}, a^{parent}, a^{siblings}, d, \sigma, m, \eta)$
    $action\_probs, \eta \leftarrow \pi_\theta(state)$ ▷ π_θ décrit dans la figure 5.3(a)
    $action \leftarrow$ `echantillone`$(action\_probs)$ ▷ Cadre bleu "Action Sampling" de la figure 5.3
    $a_{past} \leftarrow$ `append`$(a_{past}, action)$
    $\sigma_{NT}^{child}, \sigma_T^{child} \leftarrow$ `get_symboles_enfants`$(action, G)$
    $f \leftarrow$ `traduire`$(f, \sigma_{NT}^{child}, \sigma_T^{child})$
    $queue \leftarrow$ `etendre`$(\sigma_{NT}^{child}, queue)$ ▷ Place σ_{NT} au début de la queue
    $\sigma \leftarrow$ `pop`$(queue)$
    $a^{parent}, a^{siblings} \leftarrow$ `get_parents_et_freres_soeurs`(σ, a_past)
    $m \leftarrow$ `get_masque`(G, σ)
    $d \leftarrow d + 1$
  return f

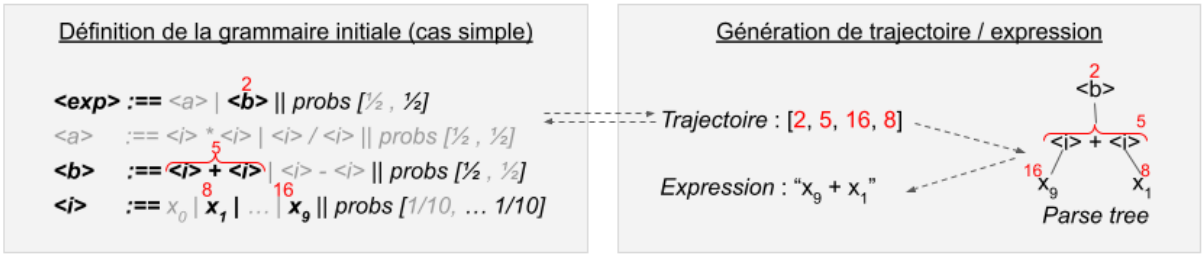


FIGURE 5.1 – "$x_9 + x_1$" Génération d'expressions et de trajectoires à partir d'une grammaire donnée. Une grammaire simpliste est donnée (à gauche), avec des actions numérotées de 1 à 16 et le symbole de départ $< exp >$. A droite, une *trajectoire* est échantillonnée à partir de cette grammaire, à partir de laquelle nous définissons l'*arbre de dérivation* (ou parse tree) et l'*expression* symbolique correspondante.

avec 16 actions, numérotées de 1 à 16, et <exp> comme symbole de départ, nous générons une trajectoire d'actions. La construction de la trajectoire se déroule comme suit :

1. Nous sélectionnons, dans un premier, temps la première action parmi l'ensemble des actions accessibles à partir du symbole de départ $< exp >$: ici, actions n°1 ou n°2. A partir des probabilités associées à chacune de ces actions, nous réalisons un échantillonnage pondéré par les probabilités listées dans `probs`. L'action n°2 est tirée et la règle lui correspondant est "$< b >$". Étant donné que cette règle contient un symboles non terminal $< b >$, il faut le remplacer par une règle grammaticale "accessible" pour le symbole $< b >$.
2. La 3^{eme} ligne de la grammaire définit des actions accessibles à partir de $< b >$: les actions n°5 ou n°6. Étant donné les poids, nous tirons l'action n°5, "$< i > + < i >$". Celle-ci contient cette fois-ci deux symboles non-terminaux ($< i >$ et $< i >$) qu'il faudra remplacer lors des prochaines étapes. Nous créons une queue pour garder en mémoire les symboles à remplacer. Nous remplaçons chacun d'entre eux en effectuant une recherche en profondeur en bouclant sur le premier symbole de la queue jusqu'à ce que celle-ci soit vide.
3. Le remplacement des symboles non-terminaux s'effectue un symbole à la fois, par l'algorithme de parcours en profondeur, en itérant sur le premier symbole, jusqu'à ce qu'il atteigne un symbole terminal. Nous itérons cette procédure jusqu'à ce que la taille maximale de trajectoire soit atteinte, ou bien jusqu'à ce qu'il ne reste plus de symboles non-terminaux non remplacés par une valeur terminale.

Définition de la fonction de récompense

La métrique de référence à minimiser en RS est l'erreur quadratique moyenne (EQM). Pour correspondre aux définitions d'AR où la fonction de récompense *reward* est une fonction monotone croissante, nous utilisons la fonction $\frac{1}{1+x}$ pour mettre à l'échelle le score entre 0 et 1 :

$$r_h = \begin{cases} 0 & if\ h < H \\ R = \frac{1}{1+EQM(y,\hat{y})} & if\ h = H \end{cases} \tag{5.1}$$

Comme la fonction f ne peut être évaluée qu'à la fin de l'épisode, lorsque la fonction est complète, la récompense r est égale à 0 jusqu'à l'étape finale et vaut $\frac{1}{1+EQM(y,\hat{y})}$ à l'étape H, où $\hat{y} = f(X)$ comme indiqué dans l'équation 5.1. Nous notons également la récompense cumulative attendue R et soulignons que $R = \sum_h r_h = r_H$. La propriété de sparsité est utilisée dans la figure 5.3.2 pour simplifier la fonction de perte de l'algorithme REINFORCE (Williams, 1992).

Processus de décision markovien partiellement observable

Compte tenu des définitions précédentes de l'espace, de l'action et de la récompense, le POMDP que nous considérons ici est défini par un tuple $(S, A, r, P, H, \Omega, O)$ avec S l'espace d'états, A l'espace d'actions, $r : S \times A \to [0, 1]$ la fonction de récompense, $P : S \times A \to [0, 1]$ le noyau de transition, $\Omega = (o_1, o_2, ..., o_K)$ un ensemble d'observations et O un ensemble de probabilités conditionnelles d'observation $O(o|s', a)$. Nous écrivons comme $P(s'|s, a)$ la probabilité d'avoir une transition vers l'état $s \in S$ quand on prend l'action a dans l'état s. Un cadre POMDP est ici considéré pour atténuer le fait que les effets des actions sont incertains jusqu'à ce que l'état final soit atteint ainsi que l'observabilité partielle de l'état. Chaque épisode k construit une trajectoire d'actions $\tau_k = (a_k^h)_{h\in[H]}$ qui se termine soit lorsque l'horizon maximum est atteint, soit lorsque la fonction f_k construite par τ_k est complète et peut être évaluée.

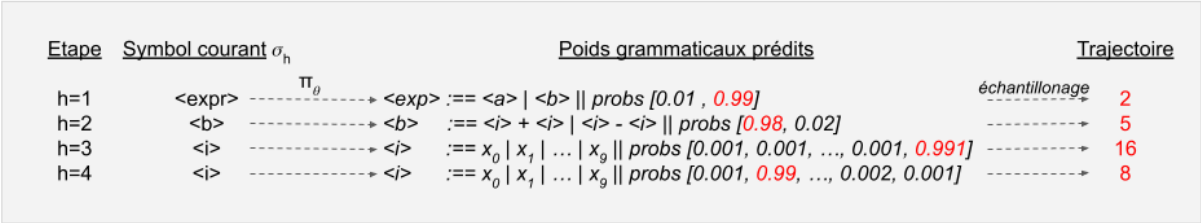


FIGURE 5.2 – Génération de poids et de trajectoires avec la grammaire de la figure 5.1 après entraînement. Les éléments en rouge définissent les résultats de l'échantillonnage sur la grammaire. Les poids de la grammaire sont mis à jour en recherchant l'expression cible "$x_9 + x_1$"

Optimisation de la politique

Étant donné l'espace d'actions grammatical défini dans la sous-section 5.3.1, il existe une politique optimale π^* capable de générer une trajectoire aussi proche que possible de la fonction symbolique f^*. La grammaire étant une entrée de notre algorithme, selon le choix de la grammaire choisie en entrée, il est même possible de générer une trajectoire optimale correspondant exactement à f^*.

Pour rechercher cette fonction symbolique optimale, nous proposons d'apprendre une politique π_θ, paramétrée par un vecteur θ pour générer les poids des règles d'actions grammaticales à chaque étape $h \in [H]$ de la trajectoire, à partir de laquelle on échantillonne l'action suivante. Plus précisément, pour construire f, la politique stochastique π_θ attribue un vecteur de probabilité aux actions accessibles à partir d'un état donné. A chaque étape h, l'action est alors échantillonnée selon le vecteur de probabilité donné par $\pi_\theta(s_h, o_h)$. En utilisant la récompense définie dans la section 5.3.1, nous mettons à jour itérativement les paramètres de π_θ pour échantillonner, dans le futur, des trajectoires plus pertinentes par rapport à la récompense définie.

En utilisant le même exemple de grammaire de la figure 5.1, nous montrons dans la figure 5.2 comment cette grammaire pourrait être mise à jour lorsqu'elle est entraînée sur la recherche de l'expression "$x_9 + x_1$". Après l'entraînement, toutes les règles grammaticales inutiles ont maintenant une probabilité faible ou nulle, et il est plus facile de générer l'expression cible correcte avec cette grammaire.

5.3.2 Implémentation sur l'apprentissage d'une politique et la découverte d'une fonction.

Modélisation d'un POMDP avec un réseau de neurones récurrent

Afin de trouver une fonction optimale f^*, nous proposons d'utiliser la politique π_θ comme un outil d'exploration, entraîné pour mettre l'accent sur l'exploration des régions les plus pertinentes de l'espace grammatical. Pour ce faire, nous apprenons π_θ avec l'algorithme "policy gradient" appelé REINFORCE (Williams, 1992) sur l'architecture de réseau neuronal décrite dans la figure 5.3. Dans la figure 5.3(a), nous décrivons (à l'aide de la légende de la figure 5.3(b)) l'architecture récurrente permettant de prédire une action a_h à partir d'un état donné s_h à l'étape h. Pour gérer la récurrence, nous choisissons les réseaux LSTM (Long Short-Term Memory) (Hochreiter et Schmidhuber, 1997). Il s'agit d'une structure spécialement conçue pour capturer les dépendances temporelles à long terme et ils sont également capables de modéliser l'observabilité partielle du MDP susmentionné (Wierstra *et al.*, 2007).

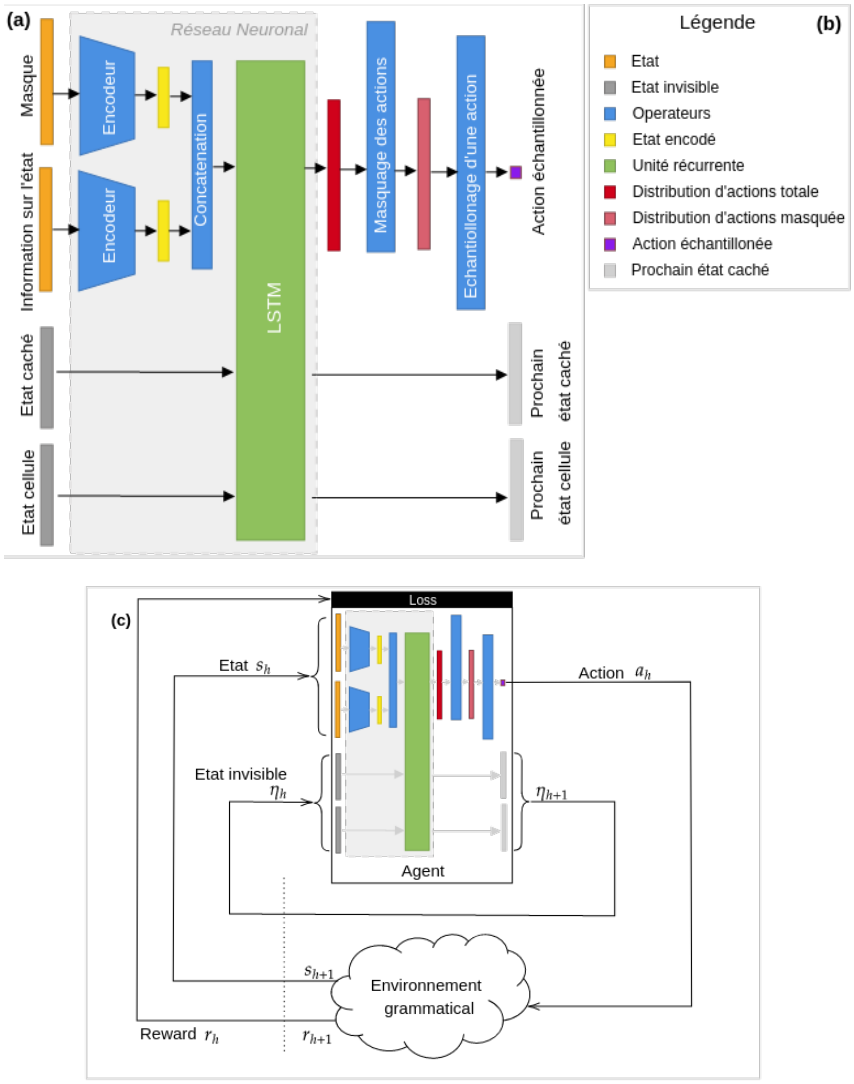


FIGURE 5.3 – Architecture de réseau neuronal pour apprendre π_θ (en couleur). **(a)** montre l'architecture récurrente permettant de prédire une action à partir d'un état donné, comme décrit dans l'algorithme 5 lignes 9 et 10. **(b)** est la légende des couleurs que nous utilisons dans **(a)** et **(c)**. **(c)** représente l'interaction entre agent et environnement POMDP.

Le réseau neuronal prend en entrée l'état s_h (en orange), et l'état caché η_h (en gris). Toutes les caractéristiques d'entrée de l'état sont codées à l'aide de convolutions ou de couches feedforwards, en fonction de leur forme. Les segments d'états codés sont ensuite concaténés et transmis à une cellule LSTM avec l'état caché η_h. Cette cellule récurrente produit à la fois une estimation des observations pour le prochain état caché η_{h+1}, et un encodage de toutes les actions de la grammaire. Ce codage des actions est ensuite masqué à l'aide du masque d'état m_h (voir la figure 5.3) et produit la distribution sur les actions *accessibles* de l'état actuel. Ensuite, nous échantillonnons l'action s_h selon la distribution sur les actions accessibles.

Masque sur les actions invalides

Pour gérer les contraintes grammaticales sur l'espace des actions, nous proposons d'utiliser à chaque étape h un masque sur les actions inaccessibles m_h dans l'état courant s_h. De manière similaire à ce qui est fait par (Huang et Ontañón, 2020), les actions invalides sont masquées par une multiplication par éléments avec une grande valeur négative et suivie par la fonction `softmax`.

Fonction de coût favorisant l'exploration

La fonction objective de l'algorithme REINFORCE (Williams, 1992), simplifiée avec l'équation 5.1, est $J_\theta = \mathbb{E}_\pi\big[R \sum_h \ log\ \pi_\theta(a_h|s_h)\big]$ à partir de laquelle nous spécifions la politique optimale $\pi^*_\theta = \text{argmax}_\theta J_\theta$. Comme les méthodes policy gradient telles que REINFORCE ont tendance à avoir une grande variance (Chung *et al.*, 2021), il est courant de soustraire une référence au batch, comme par exemple une moyenne mobile des récompenses entre les batchs. Cependant, la RS ne s'intéresse qu'à la construction d'une politique qui maximise les trajectoires les plus performantes trouvées pendant l'apprentissage, et ces stratégies peuvent avoir une convergence lente car de nombreuses trajectoires échantillonnées ne sont pas pertinentes et ont une récompense finale faible ou nulle. Sur la base de ces commentaires, B. K Petersen (Petersen *et al.*, 2021) propose une approche policy gradient favorable au risque (*risk-seeking policy gradient*), qui calcule uniquement la fonction de coût basée sur le quantile supérieur ϵ des récompenses attendues R_ϵ, c'est-à-dire les trajectoires les plus pertinentes sur l'ensemble du batch :

$$J_\theta^{risque}(\epsilon) = \mathbb{E}_{\tau\sim\pi_\theta}\Big[(R(\tau|\theta) - R_\epsilon)\sum_{h=0}^{H} \ log\ \pi_\theta(a_h|s_h) \ \Big|\ R(\tau|\theta) > R_\epsilon\Big] \tag{5.2}$$

Ils ont également ajouté un terme d'entropie $\mathcal{H}$, pondéré par $\lambda_\mathcal{H}$, pour encourager l'exploration :

$$J_\theta^{entropie}(\epsilon) = \mathbb{E}_{\tau\sim\pi_\theta}\Big[\mathcal{H}(\tau|\theta) \ \Big|\ R(\tau|\theta) > R_\epsilon\Big] \tag{5.3}$$

Finalement, notre fonction de coût finale devient :

$$\begin{aligned} J_\theta^{tot}(\epsilon) &= J_\theta^{risque}(\epsilon) + \lambda_\mathcal{H} J_\theta^{entropie}(\epsilon) \\ &= \mathbb{E}_{\tau\sim\pi_\theta}\left[\begin{array}{l} (R(\tau|\theta) - R_\epsilon)\sum_{h=0}^{H} \ log\ \pi_\theta(a_h|s_h) \\ +\lambda_\mathcal{H}\mathcal{H}(\tau|\theta) \end{array}\ \middle|\ R(\tau|\theta) > R_\epsilon\right] \end{aligned} \tag{5.4}$$

Compte tenu de cette fonction de coût, l'architecture du réseau neuronal est apprise à l'aide de l'algorithme 6.

En résumé, la méthode RBG2-SR que nous proposons, réalise une tâche de RS sous contrainte dans laquelle une structure grammaticale restreint la création d'expressions symboliques. Nous adoptons un cadre POMDP avec un horizon fini H, pour lequel la récompense associée r_h est

Algorithm 6 Procédure d'apprentissage

Require: X, cible y, horizon maximal H, taille du batch B, nombre d'itérations d'apprentissage N, quantile seuil ϵ

$\pi_\theta \leftarrow \pi_{\theta_0}$
for n in N **do**
  $episodes \leftarrow$ `echantillonage_des_episodes`(H, π_θ, G, B) ▷ Algorithme 5 utilisé B fois
  $recompense \leftarrow$ `evaluation_des_episodes`$(episodes, X, y)$
  $top\_episodes\_et\_recompenses \leftarrow$ `filtrage_des_episodes`$(episodes, recompense, \epsilon)$
  $\pi_\theta \leftarrow$ `mise_a_jour_de_la_politique`$(\pi_\theta, top\_episodes\_et\_recompense)$

nulle jusqu'à ce que l'expression complète f ait été générée. Une grammaire définit un ensemble de règles de contraintes utilisées pour la construction de l'expression, qui masque les actions non-accessibles à chaque étape $h \in [H]$. Les poids des actions accessibles au pas de temps courant sont générés par π_θ, un réseau neuronal appris avec l'algorithme REINFORCE, en utilisant une fonction de perte avec un terme relatif à la prise de risque ainsi qu'un terme d'entropie.

5.4 Expériences et résultats

Dans cette section, nous décrivons deux expériences. La première compare la méthode proposée avec plusieurs méthodes de RS issues de l'état de l'art (Whigham *et al.*, 1995; Sotto et de Melo, 2017) sur des benchmarks de référence (Uy *et al.*, 2011; Keijzer, 2003; Vladislavleva *et al.*, 2009; Pagie et Hogeweg, 1997). Dans la deuxième expérience, nous présentons un cas d'utilisation d'exploration de variables où nous montrons une application de notre approche sur un ensemble de données du monde réel avec une relation inconnue à découvrir.

5.4.1 Validation de l'approche sur des jeux de données de références

Protocole d'évaluation

Afin d'évaluer notre méthode et de la comparer à d'autres travaux de l'état de l'art, nous considérons 34 fonctions issues dans les benchmarks Nguyen (Uy *et al.*, 2011) (notées N1 à N10), Keijzer (Keijzer, 2003) (K1-15), Vladislavleva (Vladislavleva *et al.*, 2009) (V1-8) et Pagie (Pagie et Hogeweg, 1997) dont la génération de données est synthétisée dans le tableau 5.2. Ces différentes fonctions présentent différents niveaux de difficulté. Le benchmark Nguyen est connu pour être plus facile car il utilise principalement une variables d'entrée et ne nécessite pas d'optimiser les valeurs constantes dans les expressions. Les benchmarks de Keijzer et Vladislavleva sont plus complexes car ils contiennent des fonctions avec jusqu'à 5 variables d'entrée et nécessitent également de représenter des constantes dans les terminaux de l'arbre de dérivation. Nous avons également utilisé la fonction de Pagie (P1) (Pagie et Hogeweg, 1997), qui a la réputation d'être plus difficile (McDermott *et al.*, 2012). Nous avons appliqué plusieurs directives identifiées pour l'évaluation comparative de la régression symbolique, telles que fournies par (McDermott *et al.*, 2012). Notre procédure de génération de données utilise leurs fonctions et intervalles d'échantillonnage pour les ensembles d'entrainement et de test (McDermott *et al.*, 2012).

Les fonctions symboliques sélectionnées sont comparées aux méthodes basées sur la grammaire suivantes :

— *Evolution Grammaticale* (EG) qui utilise un algorithme génétique basé sur la grammaire implémenté avec la librairie Deap en Python. (Fortin *et al.*, 2012)

TABLE 5.2 – Génération de données pour le benchmarks de RS : Nguyen (Uy *et al.*, 2011), Keijzer (Keijzer, 2003), Vladislavleva (Vladislavleva *et al.*, 2009), et Pagie (Pagie et Hogeweg, 1997). (respectivement notés N1-10, K1-15, V1-8 et P1). Les variables (colonne *Vars*) sont x, y, z, v, w et leur représentation correspondante dans la grammaire est $x[1]$ à $x[5]$. $U[a, b, c]$ est un échantillonnage uniforme comportant c échantillons pris entre a et b. $E[a, b, c]$ échantillonne dans une grille de points uniformément espacés avec un intervalle de c, de a à b. Le tableau 5.2 est une version étendue des informations de génération présentées par (McDermott *et al.*, 2012).

Name	Function	Vars	Train set	Test Set
N1	$x^3 + x^2 + x$	1	$U[0, 2, 20]$	$U[0, 2, 20]$
N2	$x^4 + x^3 + x^2 + x$	1	$U[-1, 1, 20]$	$U[-1, 1, 20]$
N3	$x^5 + x^4 + x^3 + x^2 + x$	1	$U[-1, 1, 20]$	$U[-1, 1, 20]$
N4	$x^6 + x^5 + x^4 + x^3 + x^2 + x$	1	$U[-1, 1, 20]$	$U[-1, 1, 20]$
N5	$sin(x^2)cos(x) - 1$	1	$U[-1, 1, 20]$	$U[-1, 1, 20]$
N6	$sin(x) + sin(x + x^2)$	1	$U[-1, 1, 20]$	$U[-1, 1, 20]$
N7	$ln(x + 1) + ln(x^2 + 1)$	1	$U[0, 2, 20]$	$U[0, 2, 20]$
N8	$\sqrt{(x)}$	1	$U[0, 4, 20]$	$U[0, 4, 20]$
N9	$sin(x) + sin(y)$	2	$U[0, 2, 100]$	$U[0, 2, 100]$
N10	$2sin(x)cos(y)$	2	$U[0, 2, 100]$	$U[0, 2, 100]$
K1	$0.3xsin(2\pi x)$	1	$E[-1, 1, 0.1]$	$E[-1, 1, 0.001]$
K2	$0.3xsin(2\pi x)$	1	$E[-2, 2, 0.1]$	$E[-2, 2, 0.001]$
K3	$0.3xsin(2\pi x)$	1	$E[-3, 3, 0.1]$	$E[-3, 3, 0.001]$
K4	$x^3e^{-x}cos(x)sin(x)(sin^2(x)cos(x) - 1)$	1	$E[0, 10, 0.05]$	$E[0.05, 10.05, 0.05]$
K5	$\frac{30xz}{(x-10)y^2}$	3	$x, z : U[-1, 1, 1000]$ $y : U[1, 2, 1000]$	$x, z : U[-1, 1, 10000]$ $y : U[1, 2, 10000]$
K6	$\sum_i^x \frac{1}{x}$	1	$E[1, 50, 1]$	$E[1, 120, 1]$
K7	$ln(x)$	1	$E[1, 100, 1]$	$E[1, 100, 0.1]$
K8	$\sqrt{(x)}$	1	$E[0, 100, 1]$	$E[0, 100, 0.1]$
K9	$arcsinh(x)$	1	$E[0, 100, 1]$	$E[0, 100, 0.1]$
K10	x^y	2	$U[0, 1, 100]$	$E[0, 1, 0.01]$
K11	$xy + sin((x - 1)(y - 1))$	2	$U[-3, 3, 20]$	$E[0, 1, 0.01]$
K12	$x^4 - x^3 + \frac{y^2}{2} - y$	2	$U[-3, 3, 20]$	$E[0, 1, 0.01]$
K13	$6sin(x)cos(y)$	2	$U[-3, 3, 20]$	$E[0, 1, 0.01]$
K14	$\frac{8}{2+x^2+y^2}$	2	$U[-3, 3, 20]$	$E[0, 1, 0.01]$
K15	$\frac{x^3}{5} + \frac{y^3}{2} - y - x$	2	$U[-3, 3, 20]$	$E[0, 1, 0.01]$
V1	$\frac{e^{-(x-1)^2}}{1.2+(y-2.5)^2}$	2	$U[0.3, 4, 100]$	$E[-0.2, 4.2, 0.1]$
V2	$e^{-x}x^3cos(x)sin(x)(sin^2(x)cos(x) - 1)$	1	$E[0.05, 10, 0.1]$	$E[-0.5, 10.5, 0.05]$
V3	$e^{-x}x^3cos(x)sin(x)(sin^2(x)cos(x) - 1)(y - 5)$	2	$x : E[0.05, 10, 0.1]$ $y : E[0.05, 10.05, 2]$	$x : E[0.05, 10, 0.1]$ $y : E[-0.5, 10.5, 0.5]$
V4	$\frac{10}{5+(x-3)^2+(y-3)^2+(z-3)^2+(v-3)^2+(w-3)^2}$	5	$U[0.05, 6.05, 1024]$	$U[-0.25, 6.35, 5000]$
V5	$30\frac{(x-1)(z-1)}{y^2(x-10}$	3	$x : U[0.05, 2, 300]$ $y : U[1, 2, 300]$	$x : E[-0.05, 2.1, 0.15]$ $y : E[0.95, 2.05, 0.1]$
V6	$6sin(x)cos(y)$	2	$U[0.1, 5.9, 30]$	$E[-0.05, 6.05, 0.02]$
V7	$(x - 3)(y - 3) + 2sin((x - 4)(y - 4))$	2	$U[0.05, 6.05, 300]$	$U[-0.25, 6.35, 1000]$
V8	$\frac{(x-3)^4+(y-3)^3-(y-3)}{(y-2)^4+10}$	2	$U[0.05, 6.05, 50]$	$E[-0.25, 6.35, 0.2]$
P1	$\frac{1}{1+x^{-4}} + \frac{1}{1+y^{-4}}$	2	$E[-5, 5, 0.4]$	$E[-5, 5, 0.4]$

```
<e> ::= (<e><dop><e>) | (<etw1><dopw1><etw1>) | <sop>(<e>) | <et> || probs [1/4,1/4,1/4,1/4]
<et> ::= (- x[x.columns<varidx>]]) | x[<varidx>] || probs [0.5, 0.5]
<etw1> ::= ( - x[<varidx>]) |  x[<varidx>] | 1 || probs [1/3,1/3,1/3]
<dopw1> ::=  + |  - || probs [1/2, 1/2]
<dop> ::= + | - | * | / || probs [1/4, 1/4, 1/4, 1/4]
<sop> ::= cos | sin | exp | log || probs [0.25, 0.25, 0.25, 0.25]
<varidx> ::= 1... nvar || probs [1/nvar ... 1/nvar]
```

FIGURE 5.4 – Exemple de grammaire inspirée de (Sotto et de Melo, 2017). <e> est de le symbol de départ, $T = \{$<e>, <et>, <etw1>, ..., <varidx>$\}$ NT=$\{$`x[], +,-,*,/,cos,sin, exp, log,1,..,nvar`$\}$

— *Probabilistic Model Building Genetic Programming* (GB-LGP) (Sotto et de Melo, 2017). Cet algorithme met à jour la distribution de probabilité de la grammaire en utilisant une sélection des individus d'une population génétique via un algorithme de descente de gradient.

L'implémentation de ces méthodes n'étant pas directement disponible avec notre représentation d'expression, nous avons réimplémenté les deux méthodes. Le code suivant est disponible sur notre dépôt Github [1].

Une recherche d'hyperparamètres est réalisée pour les trois algorithmes sur grille de 3 à 5 valeurs par paramètre et testée sur des données générées à partir des fonctions du benchmark Nguyen (Uy *et al.*, 2011). A l'issue de cette recherche de paramètres, nous choisissons les paramètres les plus performants suivants pour RBG2-SR : $\lambda_H = 0,005$ et un taux d'apprentissage $\alpha = 0,001$. Toutes les méthodes sont comparées sur un horizon maximal de 50 actions et chaque exécution est effectuée sur une population/un batch de 1000 expressions avec un total de 2000 millions d'expressions testées au maximum (ce qui correspond à 2000 itérations : taille du batch $\times$ nombre training steps$= 1000 \times 2000 = 2M$). Les différentes méthodes sont comparées sur la même grammaire, présentée en figure 5.4.

Résultats sur les benchmarks

Pour chaque méthode et pour chaque fonction des benchmarks, 30 apprentissages indépendants sont réalisés et la meilleure expression de chaque apprentissage est conservée. Les expressions trouvées sont ensuite comparées à l'aide des moyennes de l'erreur quadratique moyennes (EQM) et des écarts types entre l'expression exacte à découvrir et la meilleure solution en cours d'exécution de chaque algorithme. La dernière ligne du tableau 5.3 correspond aux résultats du test U de Mann-Whitney pour échantillons indépendants (Mann et Whitney, 1947). Cette catégorie de test est employée sur chaque paire de méthode ($m1$, $m2$) et pour les 34 fonctions, en réalisant deux tests unilatéraux. Dans le premier test unilatéral d'égalité des distributions (H0), nous considérons l'hypothèse alternative (H1) d'une distribution des résultats de la première méthode $m1$ stochastiquement inférieure aux résultats de la deuxième méthode $m2$. Un deuxième test unilatéral d'égalité des distributions (H0'), considère ensuite l'hypothèse alternative (H1') d'une distribution des résultats de la première méthode $m1$ stochastiquement supérieure aux résultats de la deuxième méthode $m2$. Pour les deux types de test une p-value de 0.05 est utilisée. Lorsque H0, et H0' les deux méthodes sont dites équivalentes. Lorsque H1 (resp. H1'), $m1$ est dite moins performante (resp. plus performante) que $m2$.

1. https://github.com/laure-crochepierre/reinforcement-based-grammar-guided-symbolic-regression

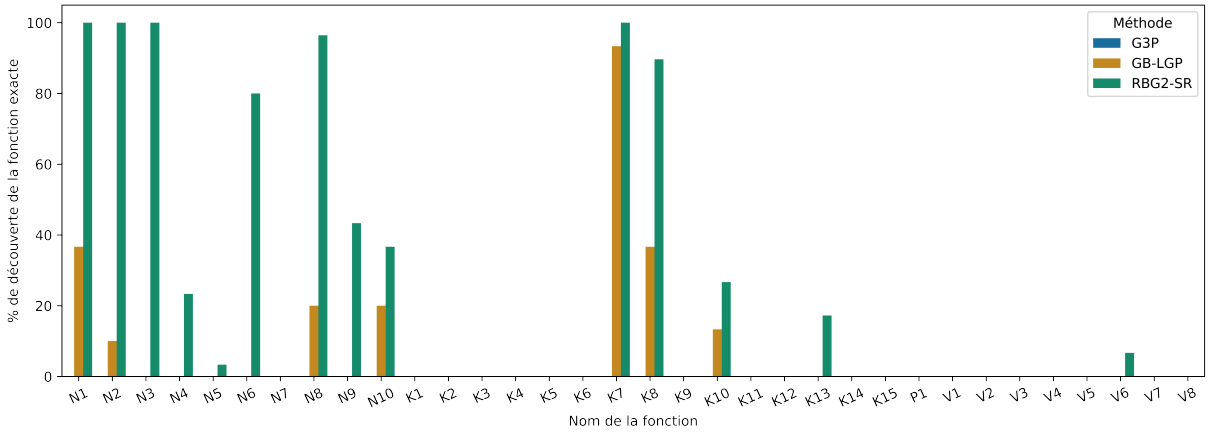


FIGURE 5.5 – Pourcentage de solutions exactes trouvées sur les quatre benchmarks pour les trois méthodes RBG2-SR (vert), GB-LGP (orange), EG (bleu). 100% signifie que la solution exacte a été découverte lors des 30 exécutions, 50% signifie que 15 exécutions sur les 30 ont été capables de trouver les bons résultats.

Les résultats du test U sont résumés en comptant le nombre de fois où chaque méthode est plus performante (symbole +), équivalente (∼) ou moins performante (−) que les deux autres méthodes. Par "équivalent", on entend le cas où deux méthodes (ou plus) sont aussi bonnes l'une que l'autre et ont les mêmes meilleurs résultats. On dit qu'une méthode est "moins bonne" si au moins une des deux autres méthodes est "meilleure" ou "équivalente". Les résultats de ces benchmarks sont présentés dans le tableau 5.3 avec les meilleurs résultats par fonction en gras et les scores équivalents en italique.

Comme le montre l'antépénultième ligne du tableau 5.3, notre méthode réduit l'EQM moyenne d'un ordre de grandeur et est statistiquement plus performante que les autres méthodes sur 22 des 33 benchmarks selon le test U de Mann-Whitney. Le benchmark K5 n'est pas pris en compte dans les tests car toutes les méthodes ont de mauvaises performances et produisent une erreur EQM élevée sur ce benchmark (supérieure à 10^{14}). Nous notons également que nos performances sont similaires à celles des autres méthodes pour sept benchmarks, conduisant à une erreur au moins similaire dans plus de 90% des benchmarks testés (30 sur 33 fonctions).

Plus précisément, en examinant le benchmark Nguyen, nous montrons que la méthode proposée surpasse les autres méthodes pour toutes les fonctions, sauf pour N4, N6 et N9, où l'erreur est similaire entre RBG2-SR et EG. Ensuite, pour les benchmarks Keijzer et Vladislavleva, notre méthode propose des solutions avec une erreur inférieure (resp. équivalente) pour 10 benchmarks sur 15 (resp. 1/15) et 5 benchmarks sur 8 (resp. 3/8). Nous soulignons également que notre méthode semble compléter la méthode évolutionnaire d'EG, en particulier sur le benchmark Keijzer, car sur les quelques benchmarks où notre méthode n'obtient pas les meilleurs résultats, l'EG a l'erreur la plus faible d'entre les trois méthodes testées.

Analysons maintenant la capacité de notre algorithme à retrouver l'expression symbolique équivalente à l'expression cible exacte, à la précision numérique près. Il est intéressant de noter que notre méthode atteint une erreur nulle pour plusieurs benchmarks (N1-3 et K7), ce qui signifie que nous récupérons tout le temps une expression équivalente à la solution exacte. Plus de détails sur ce résultat sont décrits dans la figure 5.5. Dans cette figure, nous représentons pour chaque benchmark et méthode le nombre de fois (en pourcentage) où nous pouvons récupérer la solution exacte jusqu'à la précision numérique. Tout d'abord, nous voyons que deux des trois méthodes

TABLE 5.3 – Erreur quadratique moyenne et écart-type pour les méthodes évaluées, moyennée sur 30 exécutions (les meilleurs résultats sont en gras). Le symbole − est utilisé lorsqu'il est impossible de calculer une solution ou lorsque l'erreur de solution est supérieure à 10^{10}. Sur les deux dernières lignes, est indiqué d'abord l'EQM moyennée sur toutes les exécutions valides (sur 30 exécutions), et tous les benchmarks, puis le nombre de fois où chaque méthode est plus performante (symbole +), équivalente (∼) ou moins performante (−) que les autres, en utilisant le test U de Mann-Whitney.

Name	GB-LGP (Sotto et de Melo, 2017)	EG (Whigham *et al.*, 1995)	RBG2-SR (Notre méthode)
N1	5.71×10^{-2} ($\pm 7.6 \times 10^{-2}$)	2.35×10^{-3} ($\pm 2.6 \times 10^{-3}$)	**0.00 (±0.0)**
N2	9.29×10^{-2} ($\pm 1.7 \times 10^{-1}$)	2.19×10^{-2} ($\pm 5.8 \times 10^{-2}$)	**0.00 (±0.0)**
N3	2.24×10^{-1} ($\pm 2.7 \times 10^{-1}$)	1.82×10^{-2} ($\pm 2.4 \times 10^{-2}$)	**0.00 (±0.0)**
N4	2.24×10^{-1} ($\pm 4.1 \times 10^{-1}$)	$\mathbf{1.48 \times 10^{-2}}$ ($\mathbf{\pm 1.6 \times 10^{-2}}$)	$\mathbf{1.62 \times 10^{-2}}$ ($\mathbf{\pm 1.6 \times 10^{-2}}$)
N5	6.16×10^{-3} ($\pm 1.2 \times 10^{-2}$)	1.31×10^{-3} ($\pm 1.9 \times 10^{-3}$)	$\mathbf{5.87 \times 10^{-4}}$ ($\mathbf{\pm 8.6 \times 10^{-4}}$)
N6	1.92×10^{-2} ($\pm 1.4 \times 10^{-2}$)	1.70×10^{-3} ($\pm 1.4 \times 10^{-3}$)	$\mathbf{2.78 \times 10^{-4}}$ ($\mathbf{\pm 7.9 \times 10^{-4}}$)
N7	1.48×10^{-2} ($\pm 1.5 \times 10^{-2}$)	$\mathbf{3.97 \times 10^{-4}}$ ($\mathbf{\pm 3.0 \times 10^{-4}}$)	$\mathbf{3.30 \times 10^{-4}}$ ($\mathbf{\pm 3.1 \times 10^{-4}}$)
N8	2.11×10^{-2} ($\pm 4.1 \times 10^{-2}$)	5.64×10^{-2} ($\pm 2.9 \times 10^{-1}$)	$\mathbf{1.24 \times 10^{-4}}$ ($\mathbf{\pm 6.6 \times 10^{-4}}$)
N9	1.41×10^{-1} ($\pm 1.2 \times 10^{-1}$)	$\mathbf{3.93 \times 10^{-2}}$ ($\mathbf{\pm 1.1 \times 10^{-1}}$)	$\mathbf{1.66 \times 10^{-2}}$ ($\mathbf{\pm 2.3 \times 10^{-2}}$)
N10	1.81×10^{-2} ($\pm 3.1 \times 10^{-2}$)	6.67×10^{-3} ($\pm 1.2 \times 10^{-2}$)	$\mathbf{1.38 \times 10^{-3}}$ ($\mathbf{\pm 1.5 \times 10^{-3}}$)
K1	3.38×10^{-2} ($\pm 5.2 \times 10^{-3}$)	1.11×10^{-2} ($\pm 9.7 \times 10^{-3}$)	$\mathbf{2.35 \times 10^{-3}}$ ($\mathbf{\pm 3.0 \times 10^{-3}}$)
K2	4.48×10^{-2} ($\pm 1.1 \times 10^{-3}$)	3.72×10^{-2} ($\pm 4.5 \times 10^{-3}$)	$\mathbf{2.28 \times 10^{-2}}$ ($\mathbf{\pm 7.4 \times 10^{-3}}$)
K3	4.50×10^{-2} ($\pm 1.1 \times 10^{-4}$)	4.03×10^{-2} ($\pm 6.5 \times 10^{-3}$)	$\mathbf{3.15 \times 10^{-2}}$ ($\mathbf{\pm 7.7 \times 10^{-3}}$)
K4	8.71×10^{-2} ($\pm 2.0 \times 10^{-2}$)	2.44×10^{-2} ($\pm 1.7 \times 10^{-2}$)	$\mathbf{1.19 \times 10^{-2}}$ ($\mathbf{\pm 8.9 \times 10^{-3}}$)
K5	−	−	−
K6	2.56×10^{-2} ($\pm 7.8 \times 10^{-2}$)	$\mathbf{1.46 \times 10^{-3}}$ ($\mathbf{\pm 2.0 \times 10^{-3}}$)	2.32×10^{-3} ($\pm 1.9 \times 10^{-3}$)
K7	$\mathbf{4.70 \times 10^{-4}}$ ($\mathbf{\pm 1.8 \times 10^{-3}}$)	6.59×10^{-7} ($\pm 3.0 \times 10^{-6}$)	**0.00 (±0.0)**
K8	5.05×10^{-1} (± 1.0)	1.94×10^{-1} ($\pm 2.1 \times 10^{-1}$)	$\mathbf{2.92 \times 10^{-2}}$ ($\mathbf{\pm 8.9 \times 10^{-2}}$)
K9	1.03×10^{-3} ($\pm 3.8 \times 10^{-3}$)	$\mathbf{3.11 \times 10^{-6}}$ ($\mathbf{\pm 4.3 \times 10^{-6}}$)	4.08×10^{-6} ($\pm 3.4 \times 10^{-6}$)
K10	2.45×10^{-3} ($\pm 2.3 \times 10^{-3}$)	6.94×10^{-4} ($\pm 7.5 \times 10^{-4}$)	$\mathbf{1.79 \times 10^{-4}}$ ($\mathbf{\pm 2.0 \times 10^{-4}}$)
K11	8.49×10^{-1} (± 1.9)	$\mathbf{5.52 \times 10^{-1}}$ ($\mathbf{\pm 2.8 \times 10^{-1}}$)	2.42 (± 9.5)
K12	$3.40 \times 10^{+2}$ ($\pm 4.4 \times 10^{+2}$)	$6.33 \times 10^{+4}$ ($\pm 3.5 \times 10^{+5}$)	**2.36 (±1.2)**
K13	−	3.04 (± 3.6)	$\mathbf{5.23 \times 10^{-1}}$ ($\mathbf{\pm 1.2}$)
K14	5.65×10^{-1} ($\pm 7.5 \times 10^{-2}$)	4.21×10^{-1} ($\pm 1.9 \times 10^{-1}$)	$\mathbf{1.49 \times 10^{-1}}$ ($\mathbf{\pm 1.7 \times 10^{-1}}$)
K15	2.41 (± 1.5)	$2.03 \times 10^{+8}$ ($\pm 1.1 \times 10^{+9}$)	$\mathbf{9.22 \times 10^{-1}}$ ($\mathbf{\pm 1.6 \times 10^{-1}}$)
P1	2.14×10^{-1} ($\pm 2.5 \times 10^{-1}$)	$\mathbf{1.66 \times 10^{-1}}$ ($\mathbf{\pm 1.2 \times 10^{-1}}$)	$\mathbf{1.11 \times 10^{-1}}$ ($\mathbf{\pm 9.2 \times 10^{-2}}$)
V1	$\mathbf{5.74 \times 10^{-2}}$ ($\mathbf{\pm 2.7 \times 10^{-2}}$)	−	$\mathbf{5.13 \times 10^{-2}}$ ($\mathbf{\pm 1.7 \times 10^{-2}}$)
V2	$\mathbf{8.39 \times 10^{-2}}$ ($\mathbf{\pm 1.9 \times 10^{-2}}$)	−	1.47×10^{-1} ($\pm 6.4 \times 10^{-1}$)
V3	−	4.52 ($\pm 1.7 \times 10^{+1}$)	$\mathbf{8.31 \times 10^{-1}}$ ($\mathbf{\pm 2.9 \times 10^{-1}}$)
V4	$\mathbf{3.83 \times 10^{-2}}$ ($\mathbf{\pm 3.6 \times 10^{-3}}$)	$\mathbf{3.87 \times 10^{-2}}$ ($\mathbf{\pm 4.7 \times 10^{-3}}$)	$\mathbf{3.71 \times 10^{-2}}$ ($\mathbf{\pm 5.3 \times 10^{-3}}$)
V5	2.77×10^{-1} ($\pm 1.2 \times 10^{-1}$)	1.54×10^{-1} ($\pm 9.1 \times 10^{-2}$)	$\mathbf{4.10 \times 10^{-2}}$ ($\mathbf{\pm 3.3 \times 10^{-2}}$)
V6	4.76 (± 5.3)	−	$\mathbf{8.64 \times 10^{-1}}$ ($\mathbf{\pm 7.6 \times 10^{-1}}$)
V7	$2.60 \times 10^{+1}$ ($\pm 7.6 \times 10^{+1}$)	$1.13 \times 10^{+1}$ (± 3.8)	**9.94 (±1.0)**
V8	4.12 ($\pm 2.9 \times 10^{-1}$)	3.93 (± 1.9)	$\mathbf{2.06}$ ($\mathbf{\pm 5.7 \times 10^{-1}}$)
EQM Moyenne	$1.14 \times 10^{+1}$	1.70	$\mathbf{6.09 \times 10^{-1}}$
U test	$+1 / \sim 3 / -29$	$+3 / \sim 5 / -25$	$\mathbf{+22 / \sim 7 / -4}$

récupèrent des fonctions principalement sur les benchmarks Nguyen et Keijzer. Comme nous l'avons détaillé ci-dessus, contrairement aux autres méthodes pour la plupart des benchmarks, la méthode RBG2-SR peut souvent récupérer la solution exacte pour 9 des 10 expressions du benchmark Nguyen. Sur le même benchmark, EG ne récupère que des solutions approximatives, et GB-LGP ne peut trouver que quatre fonctions avec un pourcentage de réussite inférieur à celui de notre méthode.

Alors que GB-LGP retrouve de manière extacte 3 expressions sur 15 dans le benchmark Keijzer, notre méthode RBG2-SR récupère 4 fonctions avec un taux de récupération plus élevé. La méthode RGB2-SR récupère aussi parfois la fonction V6 sur le benchmark Vladislavleva. Remarquons également que certaines fonctions des benchmarks (N7, K1-K6, K11, K12, K14, P1, V1-V5, V6 et V8) ont un pourcentage de découverte de la fonction exacte de 0%, signifiant que ces fonctions ne sont pas retrouvées de manière exacte par aucune des trois méthodes. Ces fonctions sont donc approximée par une fonction dont l'EQM moyenne est détaillée dans le tableau 5.3. Notons que ces résultats sont liés à la grammaire utilisée. Les méthodes GB-LGP et EG pourraient potentiellement trouver des solutions plus exactes en choisissant une autre grammaire. De plus, parmi les autres fonctions non trouvées, certaines sont difficiles, voire impossibles à construire avec la grammaire choisie. Ceci est particulièrement vrai pour les fonctions telles que K1-3 ou V1 qui nécessitent de trouver des constantes décimales, ce qui n'est pas supporté dans cette version de la grammaire.

5.4.2 Etude par ablation

Afin d'identifier les éléments de la méthode RBG2-SR qui sont essentiels au succès de la recherche d'expression, nous avons effectué une étude d'ablation sur les éléments de la définition d'état s_h et sur l'algorithme. Dans l'algorithme lui-même, nous essayons de supprimer l'objectif de recherche de risque J_θ^{risk} (en gardant toutes les trajectoires) et le terme de perte d'entropie $J_\theta^{entropy}$. En ce qui concerne la définition de l'état, nous comparons l'état défini avec et sans : le symbole courant σ_h, le masque courant m_h, la profondeur courante d_h, le noeud parent a_h^{parent}, les noeud frères et soeurs $a_h^{siblings}$ et les actions précédemment sélectionnées a_h^{past}. Cette étude d'ablation utilise les dix fonctions du benchmark Nguyen et a été exécutée sur 10 apprentissages indépendants pour chaque combinaison ablation/fonction. Les résultats de la meilleure expression de chaque apprentissage, sont comparés au cas de référence *baseline* où aucun élément n'est occlus. Les résultats sont résumés dans la tableau 5.4.

Tout d'abord, en examinant l'apprentissage de l'algorithme lui-même, nous montrons dans le tableau 5.4 que la combinaison des deux termes de perte est pertinente pour notre problème. L'EQM augmente de manière significative lorsque l'on supprime l'un de ces termes. La politique de recherche de risque est cruciale pour ce type d'apprentissage : la suppression de la politique de recherche de risque augmente l'erreur de 3200% par rapport à la méthode proposée avec recherche de risque. Le terme d'entropie est également d'une grande importance, avec une augmentation de l'erreur de 860%. Nous avons effectué un deuxième type d'ablation sur la définition de l'état. Dans cette partie de l'étude d'ablation, nous avons essayé de retirer des éléments des entrées *state* du réseau neuronal (le bloc orange appelé "information d'état") : soit l'information d'actions du parent a^{parent}, l'information d'actions des frères et soeurs $a^{siblings}$, les actions sélectionnées précédemment a^{past}. Les résultats du tableau 5.4 indiquent que ces informations sur les frères et soeurs, les actions passées, la profondeur et les symboles sont très bénéfiques à la recherche d'expressions puisque la suppression d'un de ces termes augmente l'erreur d'au moins 63%. Les informations sur la profondeur et les symboles semblent être d'une importance presque égale pour une recherche d'expression réussie. En ce qui concerne les informations sur les parents, il

TABLE 5.4 – Etude par ablation. les scores fournis sont les moyennes des scores EQM (± écart-type) et pourcentage de variation sur 10 exécutions du benchmark Nguyen. Tous les résultats doivent être comparés au cas de référence : $Variation(\%) = 100\ \frac{reference - ablation}{reference}$

Type	Ablation	EQM	Variation(%)
	Référence	1.58×10^{-3} ($\pm 4.92 \times 10^{-3}$)	-
Algorithme	Sans entropie	1.51×10^{-2} ($\pm 4.41 \times 10^{-2}$)	860%
	Sans prise de risque	5.16×10^{-2} ($\pm 8.68 \times 10^{-2}$)	3200%
État	Sans parent	1.51×10^{-3} ($\pm 4.68 \times 10^{-3}$)	−4%
	Sans frères/soeurs	2.57×10^{-3} ($\pm 8.75 \times 10^{-3}$)	63%
	Sans actions passées	2.97×10^{-3} ($\pm 1.01 \times 10^{-2}$)	87%
	Sans profondeur	3.73×10^{-3} ($\pm 1.18 \times 10^{-2}$)	140%
	Sans symbole	3.83×10^{-3} ($\pm 1.23 \times 10^{-2}$)	140%

semble que leur suppression tende à réduire l'erreur d'un faible 4%. Cependant, en examinant plus attentivement chaque benchmark, la suppression des informations sur les parents n'est bénéfique que pour la recherche du benchmark N9. À l'exception de N1-3 et N7 (où les deux configurations récupèrent toujours l'expression exacte), l'algorithme complet proposé (*Baseline*) surpasse l'ablation du parent. Pour les autres benchmarks, l'augmentation correspondante se situe entre +50 et +55%.

5.4.3 Analyse de l'interprétabilité sur des données physiques

L'objet de ce chapitre est de décrire un algorithme qui fournit des solutions symboliques interprétables directement lisibles par des humains disposant de connaissances physiques et/ou des experts du domaine étudié. A partir des résultats de la première expérience, nous voyons également une application potentielle de notre approche à des ensembles de données plus complexes avec une relation inconnue entre un ensemble d'observations X et une variable cible y, où les variables X et y peuvent être d'unités physiques différentes. Dans ce scénario, l'utilisation d'une grammaire est particulièrement importante pour limiter les sorties à des solutions réalistes en termes d'unités physiques. Par exemple, il n'est pas physiquement possible d'additionner une vitesse (mesurée en mètres par seconde) avec une distance (en mètres).

Pour atteindre cet objectif d'interprétabilité physique, nous avons conçu une deuxième expérience sur le jeu de données Airfoil Self-Noise (Brooks *et al.*, 1989), pour comparer les solutions proposées par notre algorithme avec celles données par quatre méthodes de l'état de l'art (Whigham *et al.*, 1995; Sotto et de Melo, 2017; McConaghy, 2011; Petersen *et al.*, 2021). Le jeu de données est accessible sur le Machine Learning Repository d'UCI (Dua et Graff, 2017) [2]. Les méthodes testées sont GB-LGP, EG de l'expérience précédente, et deux autres méthodes non basées sur la grammaire *Fast Function Extraction* (FFX) (McConaghy, 2011) et *Deep Symbolic Regression* (DSR) (Petersen *et al.*, 2021), décrits dans le chapitre 3.

2. `https://archive.ics.uci.edu/ml/datasets/airfoil+self-noise` (Consulté le 27 septembre 2022)

```
<exp>      ::=  <unit> * const  | <no_unit> * const
           | const-10*log10(<no_unit>/const)*<no_unit>
           | const-10*log10(<unit>/const)*<no_unit>
           ||probs [0.25, 0.25, 0.25, 0.25]
<unit>     ::= <distance> | <velocity> | <time> |(<no_unit> * <unit>)
           || probs [0.25,0.25,0.25,0.25]
<no_unit>  ::= <no_unit>*<no_unit>| cos(x.alpha) | sin(x.alpha) | <distance>/<distance>
            | <velocity>/<velocity> | <time>/<time> ||  probs [0.16,0.16,...,0.16]
<velocity> ::= (<velocity><dop><velocity>) | (<distance>/<time>) | x.U_infinity
            || probs [0.33,0.33,0.33]
<distance> ::= (<distance><dop><distance>) | (<velocity>*<time>) | abs(<distance>)
            | x.delta | x.c || probs [0.2,...,0.2]
<time>     ::=(<distance>/<velocity>)| (1/x.f) || probs [0.5,0.5]
<dop>      ::= - | + || probs [0.5,0.5]
```

FIGURE 5.6 – Grammaire utilisée dans la deuxième expérience sur le jeu de données Airfoil. Le symbole de départ est <exp>.

Description des unités physiques

Le problème étudié dans cette expérience est la prédiction du niveau de pression acoustique à l'échelle (SSPL) sur des profils NACA 0012 de différentes tailles. L'estimation est réalisée à l'aide des variables suivantes : fréquence (unité Hz), angle d'attaque (degré °), longueur de corde (mètres m), vitesse du flux libre (mètres par seconde $m.s^{-1}$), épaisseur du déplacement par aspiration latérale (mètres m). Les mesures sont obtenues à l'aide d'ailettes observées pour différentes vitesses et angles d'attaque en soufflerie. L'envergure de l'aile et la position de l'observateur sont fixes pour toutes les mesures. L'ensemble de données est réparti entre 70 % dans l'ensemble d'entrainement et 30 % dans l'ensemble de test.

Construction d'une grammaire

La première étape de pré-traitement, avant la comparaison des méthodes, est la définition d'une grammaire contrainte qui contient les premières connaissances du domaine étudié, telle que celle utilisée dans la figure 5.6. Le symbole de départ est <exp>. Ce symbole décrit les dimensions (unités) et les structures fonctionnelles que l'algorithme autorise en sortie. D'après les études précédentes (Lau *et al.*, 2009) sur ces données, nous voulons privilégier une expression de la forme : $\texttt{exp} = \texttt{constant} - 10 * \texttt{log10}(\texttt{child\_expression})$. Nous ajoutons également plusieurs autres structures pour laisser à l'algorithme la liberté d'explorer.

Pour résumer la grammaire de la figure 5.6, les trois premières lignes décrivent ce que sont les unités et les non-unités (composition d'unités) du problème. Dans les quatre dernières lignes, la grammaire contraint les opérations sur chaque dimension (ou unité) à des combinaisons physiquement cohérentes. Ces lignes définissent également comment passer d'une unité à une autre en utilisant des propriétés physiques (comme la loi de vitesse). Cette partie de la description grammaticale est particulièrement importante pour décrire l'expertise et les connaissances que nous voulons inclure pour contraindre l'espace de recherche pendant la résolution de la RS.

Résultats et analyse

Dans cette expérience, tous les algorithmes sont exécutés sur 10 apprentissages indépendants (excepté FFX (McConaghy, 2011) car il s'agit d'un algorithme déterministe qui fournit plusieurs solutions), et la meilleure expression de toutes les exécutions est indiquée dans le tableau 5.5.

TABLE 5.5 – Analyse du jeu de données Airfoil Self-Noise. Les meilleures expressions trouvées sont présentées avec leurs scores EQM, leur coefficient de détermination $\mathcal{R}^2$, et leur complexité κ.

Méthode	Expression	EQM	$\mathcal{R}^2$	κ	$\kappa - H$ (< 0)
DSR	$-\alpha + U_{infinity} - \frac{U_{infinity}}{\sin(\log_{10}(U_{infinity}))}$	239.15	−4.07	10	−40 (<<)
GB-LGP	$127.36 - 10 \times \log_{10}\left(\frac{c \times \delta \times f}{\frac{U_{infinity} \times \delta}{c} + 2 \times U_{infinity}}\right) \times \cos^2(\alpha)$	35.6	0.24	24	−26 (<)
$\text{FFX}_{\kappa<H}$	$128 - 53.2 \times \log_{10}(f) \times \delta - 7.64 \times \log_{10}(f) \times c - 1.92 \times \log_{10}(\delta) - 0.529 \times \log_{10}(f)$ $+0.0264 \times U_{infinity} - 0.00401 \times \alpha^2 - 0.000819 \times f + 0.000391 \times U_{infinity}^2$	20.1	0.57	43	−7 (<)
EG	$101.38 - 10 \times \log_{10}\left(\frac{1}{f} + \frac{\delta \times f^2 \times c^2}{U_{infinity}^3}\right)$	19.5	0.58	19	−31 (<)
RBG2-SR (ours)	$\mathbf{83.85 - \frac{10 \times \log_{10}\left(\frac{U_{infinity}}{c \times f}\right) + \frac{U_{infinity}^2}{f^2 \times c \times \delta}}{1 + \frac{c \times \delta \times f^2}{U_{infinity}^2}}}$	13.0	0.72	**32**	**−18 (<)**
$\text{FFX}_{\kappa_{max}}$	$0.001 \times U_{infinity}^2 - 0.026 \times U_{infinity} + 14.4 \times \alpha \times \delta - 0.492 \times \alpha$ $+12.8 \times c^2 + ... + \log_{10}(f) + 18.8 \times \log_{10}(\delta) - 71.3 \times \log_{10}(f) + 272$	10.3	**0.78**	144	86 (>>)

Dans le cas de l'algorithme FFX, deux expressions sont retenues, celle obtenant l'EQM la plus faible sans limite de complexité de la solution $\text{FFX}_{\kappa_{max}}$, ainsi que celle permettant d'obtenir l'erreur la plus faible sous contrainte de respecter la limite de complexité, notée $\text{FFX}_{\kappa<H}$.

Les résultats obtenus sont comparés sur l'ensemble de test en fonction de l'erreur EQM, du coefficient de détermination $\mathcal{R}^2$ et de la complexité κ. Afin d'obtenir un terme calculable de la même manière pour toutes les méthodes testées, la complexité κ est une complexité fonctionnelle (ou comportementale (Le *et al.*, 2016), définie par la somme des opérations et des variables d'entrée utilisées dans la formule. D'après le tableau 5.5, nous pouvons noter que l'algorithme le plus performant sur ce jeu de données est $\text{FFX}_{\kappa_{max}}$, avec une erreur d'EQM de $10,3$. Cependant, cette méthode produit également l'expression présentant la complexité la plus élevée (environ 140), au-dessus de l'horizon maximal H de 50. Comme on le voit dans la colonne *Expression*, la solution $\text{FFX}_{\kappa_{max}}$ est complexe, n'est pas directement lisible par un humain et combine des variables avec des unités différentes. Elle risque donc de sur-apprendre sur les données utilisées. Lorsque l'on cherche à respecter la limite de complexité imposée l'expression $\text{FFX}_{\kappa<H}$ obtient une erreur plus élevée de 20.1, supérieure à celle de la méthode RBG2-SR que nous proposons. Parmi les quatre autres méthodes, nous soulignons que la méthode RBG2-SR que nous proposons produit la deuxième erreur la plus faible et le coefficient de détermination le plus élevé de $0,72$, tout en conservant une complexité acceptable de 36, proche du seuil H mais inférieure. En outre, il convient de noter que toutes les méthodes basées sur la grammaire ont utilisé une expression qui utilise le format : $\texttt{constant} - 10 \times \texttt{log10}(\texttt{child\_exp})$. Par exemple, la méthode DSR trouve une solution simple, largement en dessous du seuil H. Cependant, cette solution est en réalité trop simpliste et ne respecte pas la cohérence dimensionnelle. Ces résultats tendent à plaider en faveur de l'utilisation de contraintes grammaticales pour la découverte d'équations. L'expression trouvée par notre méthode, pourrait, par exemple, être utilisée pour estimer la valeur de la `constant` dans l'équation mentionnée ci-dessus.

Enfin, en ce qui concerne la cohérence en terme d'unités ou dimensions physiques, toutes les méthodes non basées sur la grammaire ont construit des combinaisons interdites de différentes unités. Ceci permet de mettre en évidence qu'elles ne sont pas encore capables de rivaliser avec les méthodes basées sur la grammaire pour construire des expressions physiquement pertinentes.

5.5 Conclusion du chapitre

Ce chapitre présente un nouvel algorithme (RBG2-SR) pour la régression symbolique guidée par la grammaire en utilisant une approche de recherche par apprentissage par renforcement. L'utilisation d'une grammaire pour contraindre l'espace de recherche, permet ici l'inclusion de connaissances experte dans le processus d'apprentissage et dans le format des solutions. Nous décrivons une modélisation sous forme de POMDP de la tâche de RS dans un espace d'actions grammatical. La méthode proposée est comparée à des algorithmes de l'état de l'art sur des données de référence et montre des améliorations significatives par rapport aux autres algorithmes en ce qui concerne la métrique d'erreur et la découverte d'expressions exactes. Nous avons réalisé une étude par ablation sur des blocs de notre algorithme et sur la définition des états. Les résultats montrent que les informations sur les parents, les frères et soeurs, les actions passées, la profondeur et les symboles sont tous des éléments importants de la définition de l'état. Dans la deuxième expérience, nous montrons également comment l'utilisation d'une approche basée sur la grammaire pourrait être utile et interprétable lorsque l'on travaille sur un jeu de données avec des contraintes physiques entre les caractéristiques d'entrée.

Ce chapitre est une première étape réalisant la validation scientifique de l'approche sur des données de l'état de l'art. Les résultats obtenus ouvrent sur différentes catégories de perspectives. La première d'entre elles consiste en l'application de la méthode développée sur des benchmarks de RS et des données de RTE ayant des contraintes physiques. Des expérimentations similaires à celles réalisées dans le chapitre 4 sont envisagées pour exploiter l'algorithme RBG2-SR sur des problèmes industriels. Par ailleurs, en comparant les résultats d'EG et de RGB2-SR, nous envisageons également des améliorations en faisant de l'apprentissage croisé (Zhang et Zhou, 2021) entre l'EG, pour encourager l'exploration, et notre méthode, pour l'apprentissage et l'échantillonnage. De plus, comme le processus de construction de la grammaire peut être une tâche chronophage, nous pourrions nous inspirer des techniques de construction automatique d'ontologies (Emani *et al.*, 2019) pour créer et améliorer automatiquement la grammaire. Des perspectives d'application de notre méthode pourraient, par exemple, être de trouver des politiques interprétables qui suivent des règles grammaticales expertes. Dans la deuxième expérience, nous voulons par la suite explorer le comportement de notre algorithme lorsqu'il traite des horizons plus longs jusqu'à 100 actions pour créer des expressions plus expressives, ou des horizons plus courts pour des expressions plus concises et interprétables.

Enfin, notons que ce chapitre a également permis de mettre en évidence qu'il n'est pas nécessaire d'apprendre sur l'intégralité des trajectoires avec la méthode RBG2-SR. Un sous-ensemble sélectionné de manière pertinente, en filtrant les meilleures trajectoires ayant une récompense supérieure au quantile ϵ, permet apprendre de manière pertinente sur un faible pourcentage de des trajectoires générées. Cet élément est particulièrement intéressant dans l'objectif de créer un algorithme interactif : il sera possible, par exemple, de présenter à l'utilisateur seulement les meilleures trajectoires. Cette approche sera adoptée dans le chapitre suivant.

Dans le chapitre suivant, nous présenterons les travaux engagés dans le cadre de l'interactivité avec les algorithmes présentés dans les chapitres 4 et 5.

Chapitre 6

Régression Symbolique Interactive Guidée par la Grammaire

6.1 Introduction

Dans le chapitre 4, nous avons montré comment représenter des contraintes physiques et des connaissances métiers relatives du domaine de l'électricité par les règles d'une grammaire non-contextuelle pour apprendre sur ces règles par EG (Evolution Grammaticale). Le chapitre 5 a ensuite présenté une méthode "RBG2-SR" permettant d'apprendre sur ce même espace grammatical au moyen d'un algorithme par renforcement.

Cependant, bien que ces deux chapitres présentent une première catégorie d'interaction réalisée avec la grammaire entre deux apprentissages pour en améliorer les règles, il n'est cependant pas encore possible, dans ces travaux, d'interagir au cours de l'apprentissage avec les algorithmes ainsi présentés. Dans ce chapitre, nous présentons les travaux engagés pour proposer des interactions aux utilisateurs pendant l'apprentissage. Deux travaux ont été menés dans cette direction : un premier travail mettant en oeuvre un algorithme d'EG tel que celui du chapitre 4 et un second travail étudiant l'interactivité avec l'algorithme de renforcement RBG2-SR introduit dans le chapitre 5.

La section 6.2 présente les travaux préliminaires proposant à des experts du réseau électrique d'interagir avec les solutions d'un algorithme génétique pour les améliorer grâce à leur expertise. Ce premier travail, utilisant des données réelles du réseau électrique a notamment permis d'identifier des scénarios d'utilisation de l'interactivité pour une tâche de régression symbolique sur les transits électriques. La structure de la suite du chapitre est la suivante. Après avoir détaillé dans la section 6.3 les motivations des travaux sur l'interactivité de l'algorithme RBG2SR, nous présentons dans la section 6.4 le système interactif développé, puis les cas d'utilisation implémentés sont détaillés dans la section 6.5. La section 6.6 décrit ensuite les expériences menées avec l'interface développée. Celles-ci sont composées d'interviews menés à la suite d'expériences utilisateurs et à partir desquelles certains comportements utilisateurs ont été modélisés. A partir de ces modélisations, des simulations ont ensuite été réalisées pour analyser l'apprentissage en fonction de différentes fréquences d'interactions. Enfin, dans la section 6.7 nous proposons des pistes de généralisations de cette approche à un usage multi-algorithmes d'une interface unique.

6.2 Travaux préliminaires

Ces premiers travaux avaient pour objet d'inclure l'interactivité avec des experts pendant l'apprentissage d'algorithmes évolutionnaires, que nous nommerons *micro-interactivité*, afin d'identifier des cas d'utilisation pertinents et nécessaires sur des données du réseau électrique. Après avoir résumé les motivations de cette première approche dans la sous-section 6.2.1, nous détaillerons dans la sous-section 6.2.2 l'algorithme et dans la sous-section 6.2.3 l'interface développé pour enfin préciser les cas d'utilisation identifiés dans la sous-section 6.2.4.

6.2.1 Motivations

Nous nous concentrons particulièrement sur l'explication des transits sur les lignes électriques à partir de variables mesurées et simulées. Dans le chapitre 4, la méthode développée fournissait pour la majorité des lignes électriques, des résultats intéressants mais perfectibles sur la construction de variables symboliques représentant les transits électriques. Parmi les pistes investiguées, une première solution visant à apprendre par renforcement la distribution de probabilité sous la forme d'une politique stochastique a été présentée dans le chapitre 5. Une deuxième piste était de donner la possibilité à l'utilisateur d'enrichir les individus-expressions par son expertise pendant l'apprentissage sous forme de *micro-interactions* pendant l'apprentissage. Afin de relever les défis mentionnés ci-dessus, nous avons développé une plateforme interactive où des experts humains peuvent directement analyser les résultats intermédiaires, proposer des solutions, et fournir des commentaires si nécessaire. Comme dans le chapitre 4, la grammaire peut être mise-à-jour de manière interactive. Ce type d'interactivité sera noté *macro-interactivité*. La combinaison des deux types d'interactivité doit, ici, permettre aux experts du domaine de guider plus finement la recherche vers ce qui est considéré, selon eux, comme la région la plus pertinente de l'espace de recherche, réduisant ainsi l'effort de calcul nécessaire pour inférer des solutions pertinentes.

6.2.2 Description du système et détails de l'algorithme évolutionnaire

La figure 6.1 donne un aperçu haut-niveau de la plateforme interactive. La procédure d'entraînement est divisée en deux composantes : l'apprentissage évolutionnaire à gauche, et l'élicitation interactive des connaissances à droite. Entre les deux se trouve l'interface d'interaction (voir la sous-section 6.2.3). L'interactivité est ici multi-niveaux ce qui permet à l'utilisateur de distiller ses connaissances à différentes étapes. L'utilisateur interagit avec l'algorithme de PG (Programmation Génétique) à la fois pendant l'exécution par micro-interactions (par la mise à jour des paramètres, l'inspection des individus et l'insertion), et entre deux exécutions successives par macro-interactions (principalement en mettant à jour la grammaire). Plus précisément, pendant l'exécution de l'évolution, l'utilisateur est sollicité pour l'initialisation et la mise à jour de la population. Les apprentissages successifs sont effectués jusqu'à ce que l'utilisateur soit satisfait de la solution proposée.

Grammaire probabiliste Nous utilisons une version pondérée de l'EG, dérivée de la Programmation Génétique Guidée par une Grammaire Probabiliste (Sammut et Webb, 2011), et notée EGP (Evolution Grammaticale Probabiliste) dont la grammaire est capable de contenir des informations sur le problème pour ensuite générer des individus portant cette connaissance. En EGP, par rapport à la EG, un poids est associé à chaque règle grammaticale. Nous utilisons ici ces poids pour la sélection des individus lors de l'initialisation de la population.

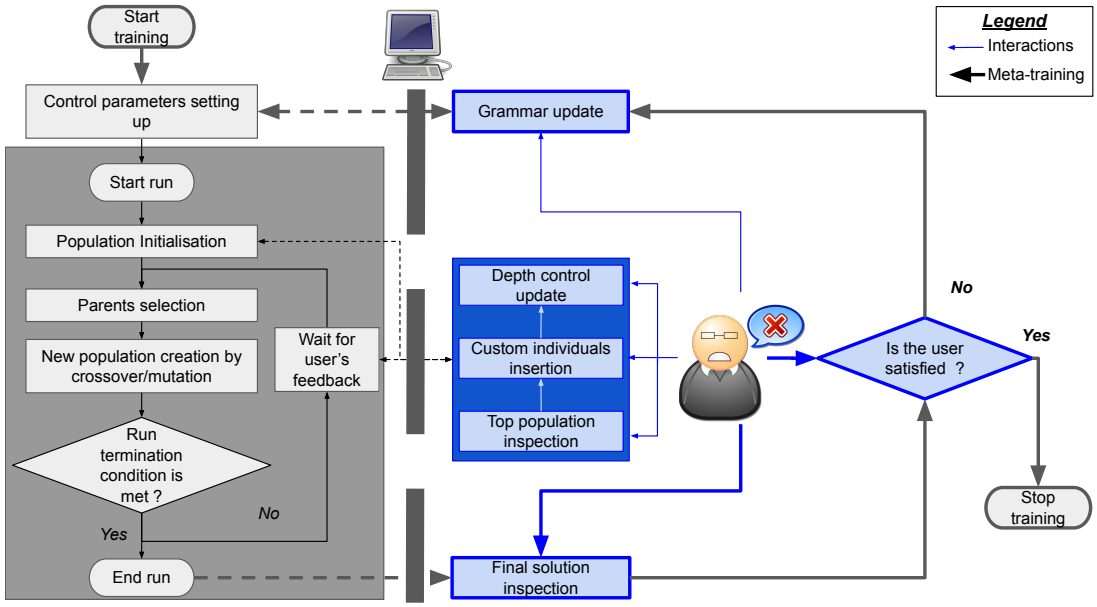


FIGURE 6.1 – Vue d'ensemble de la plateforme proposée avec : à gauche l'évolutionnaire, et à droite les détails de l'interaction multi-niveaux. "Start training" est le point d'entrée du processus.

Fonction fitness Nous utilisons des métriques basées sur la corrélation (Savic *et al.*, 1999), capables d'identifier des relations linéaires (Pearson) ou non-linéaire (Spearman) entre les données avec une corrélation r modifiée $S_c(y, \hat{y}) = max(0, r(y, \hat{y}))$ qui restreint la population aux individus positivement corrélés.

Pour éviter le gonflement ou *bloating*, nous ajustons la fitness pour pénaliser les expressions de grande taille avec $S_b(\hat{y})$ déjà utilisée dans le chapitre 4 (équation 4.2) :

$$S_b(\hat{y}) = \frac{\kappa\_seuil^k - 0.5 \times (\kappa(\hat{y})^k + 1)}{\kappa\_seuil^k - 1}, k \in \mathbb{N}$$

Au final, la fitness devient alors : $fitness = S_c \times S_b$.

Données Dans ces travaux, les données utilisées sont identiques à celles du chapitre 4. Parmi, les 24 lignes électriques (ou variables) sélectionnées y dans le chapitre 4 en raison de leur difficulté et leur intérêt opérationnels. 13 sont déjà bien analysés sans interactivité (avec un score de corrélation de Pearson supérieur à 0.85). Dix autres sont plus difficiles à traiter sans interactivité (score dans $[0.65, 0.8]$). L'ensemble de données est séparé en ensembles d'entraînement et de test. L'algorithme EGP utilise l'ensemble d'entraînement, et l'ensemble de test est affiché dans l'interface de visualisation.

Implémentation L'algorithme a été codé à partir du code open-source PonyGE2 (Fenton *et al.*, 2017), étendu aux grammaires probabilistes et amélioré pour permettre l'utilisation des métriques d'erreur basées sur la corrélation, et la modification de la population.

6.2.3 Platforme interactive

Dans ce travail, nous voulons tirer parti des connaissances des experts en interagissant avec eux. Pour cela, nous proposons une plateforme web construite à l'aide des frameworks Plotly

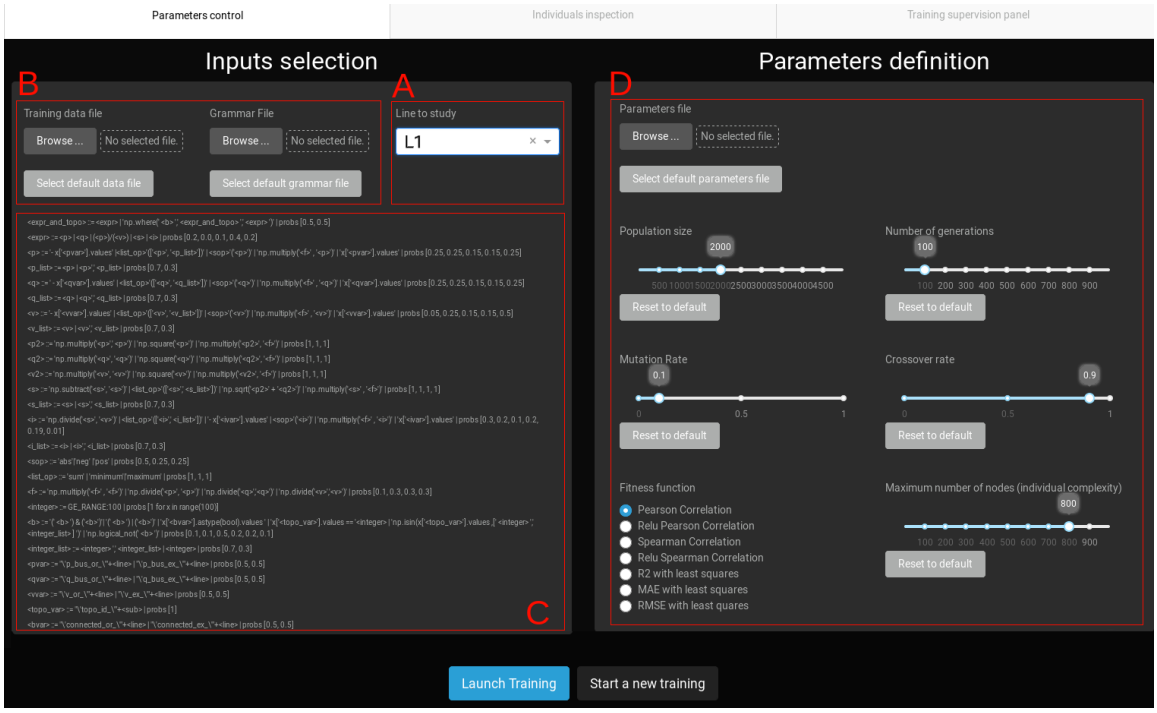


FIGURE 6.2 – Onglet de contrôle des paramètres. La sélection obligatoire de la cible se trouve dans la zone A. Les zones B et C sont consacrées à la définition (facultative) d'une grammaire et d'un ensemble de données personnalisés. D'autres paramètres d'apprentissage de l'algorithme EGP peuvent également être modifiés dans la zone D. Dans le cas où l'utilisateur ne spécifie pas de grammaire ou de paramètres particuliers, des valeurs par défaut sont prises.

et Dash (Hossain *et al.*, 2019) en Python. La plateforme comporte deux éléments : l'interface web côté client où l'utilisateur interagit avec l'algorithme dans le navigateur, et l'algorithme côté serveur. L'utilisateur est sollicité à quatre étapes clés de l'algorithme :

- avant de lancer l'apprentissage, pour la calibration des hyperparamètres ;
- à l'initialisation, pour inclure les individus définis par l'utilisateur ;
- entre les itérations spécifiques, pour proposer de nouveaux individus basés sur l'analyse des individus les mieux classés de la génération précédente ;
- à la fin de chaque exécution, pour tirer certaines conclusions des expériences concernant à la fois la qualité des solutions et la pertinence des hyperparamètres sélectionnés.

L'interface web est séparée en trois onglets : un onglet de contrôle des paramètres avant l'apprentissage ; un onglet d'inspection de la population pour fournir un feedback aux prochaines générations ; et un onglet de supervision pour suivre l'évolution de paramètres d'apprentissage.

Onglet de contrôle des paramètres La première action de l'utilisateur est le calibrage des hyperparamètres dans un onglet dédié. Cela consiste en la définition d'une variable cible y, de données et d'une grammaire (chargés ou utiliser un fichier par défaut). La grammaire peut y être visualisée et les personnes familières avec la EGP peuvent y configurer des paramètres de l'algorithme.

Expression	Fitness	Pearson	Spearman	r2	mae	mse
maximum([x['p_bus_or_l1'], x['p_bus_or_l2']])	0.811	0.815	0.823	-4.558	117.592	16304.189
maximum([x['p_bus_ex_l1'], x['p_bus_or_l3'], x['p_bus_or_l4']])	0.767	0.769	0.784	-4.553	117.158	16291.798
maximum([x['p_bus_ex_l1'], x['p_bus_or_l3'], x['p_bus_or_l4']])	0.76	0.771	0.794	-4.437	116.093	15950.959
maximum([x['p_bus_ex_l3'], x['p_bus_or_l1'], x['p_bus_or_l5']])	0.76	0.771	0.792	-4.437	116.093	15950.959
maximum([x['p_bus_ex_l3'], x['p_bus_or_l6'], - x['p_bus_ex_l4'], x['p_bus_ex_l2'], x['p_bus_or_l1'], - x['p_bus_ex_l8']])	0.755	0.766	0.799	-4.414	115.661	15882.871
maximum([x['p_bus_ex_l8'], x['p_bus_ex_l7'], x['p_bus_or_l1']])	0.753	0.764	0.798	-4.416	115.705	15889.992
(x['p_bus_or_l1']) / (x['v_or_l8'])	0.723	0.725	0.807	-5.189	123.53	18156.849
(x['p_bus_or_l1']) / (x['v_or_l4'])	0.722	0.725	0.806	-5.213	123.67	18226.018
(x['p_bus_or_l1']) / (x['v_ex_l5'])	0.722	0.725	0.806	-5.213	123.67	18226.018
(x['p_bus_or_l1']) / (x['v_ex_l3'])	0.722	0.725	0.806	-5.213	123.67	18226.025
x['p_bus_or_l1']	0.693	0.694	0.727	-4.884	122.209	17261.815
maximum([x['p_bus_ex_l3'], x['p_bus_ex_l8'], pos(- x['p_bus_ex_l9'])])	0.687	0.695	0.743	-4.496	116.39	16122.446

FIGURE 6.3 – L'onglet d'inspection pour l'analyse des individus. L'utilisateur peut inspecter, mettre à jour, tester et visualiser les individus dans la zone A. Les boutons de la zone B permettent d'augmenter ou diminuer la complexité maximale des expressions dans les prochaines générations. La zone C compare plusieurs expressions au cours du temps par rapport à la cible. Dans la zone D, l'utilisateur peut sélectionner d'autres expressions à analyser.

Onglet d'inspection et de feedback Pendant l'apprentissage et dès l'initialisation, l'utilisateur est invité à proposer des individus personnalisés. L'interface correspondant est présenté en haut de la figure 6.3 et montre les individus de la génération actuelle, classés par score. Cette étape d'inspection-réaction est également proposée de manière similaire à certaines générations.

En haut (zone A de la figure 6.3), les utilisateurs visualisent les meilleurs individus. À droite, une vue temporelle fournit un résultat qualitatif de la qualité de la solution. À gauche, l'individu est représenté sous forme d'arbre où les noeuds sont des fonctions et des feuilles des variables. L'utilisateur peut saisir des solutions personnalisées sous forme de texte pour les comparer à l'individu actuel par leurs scores. Une fois que l'utilisateur est satisfait de ce nouvel individu, il peut l'insérer dans la population actuelle afin qu'il soit utilisé dans la prochaine génération. Après inspection, l'utilisateur peut augmenter ou diminuer le paramètre de complexité, puis passer à la génération suivante (zone B de la figure 6.3). Deux graphiques supplémentaires permettent également de comparer temporellement et par des métriques (R2, EQM, ...) plusieurs individus. Du fait du coût élevé du temps d'un expert ainsi que pour limiter la fatigue des utilisateurs, nous proposons de sélectionner dans la population les dix meilleurs individus et d'ajouter dix individus pris au hasard dans le reste de la population. Par ailleurs, l'inspection manuelle est uniquement réalisée à des générations spécifiques (par défaut 1/10).

À la dernière génération, l'utilisateur analyse la population finale et tire des conclusions de l'entraînement pour améliorer les règles grammaticales des prochains apprentissages.

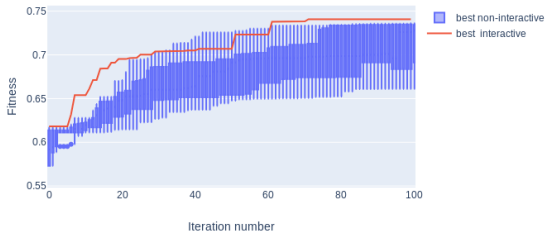
Onglet de supervision Un dernier onglet affiche des statistiques générales sur l'apprentissage : l'évolution de la fitness, complexité des individus. Sur chaque graphique, les statistiques de l'apprentissage en cours sont superposées à celles de 40 entraînements non-interactifs. Ces graphiques peuvent être utilisés pour mettre-à-jour les paramètres choisis lors des prochains entraînements ou en effectuant un arrêt précoce de l'apprentissage.

6.2.4 Scénarios d'utilisation dégagés

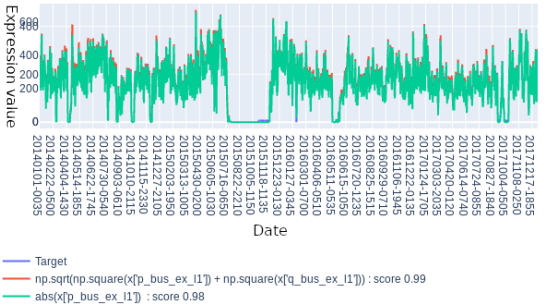
A partir de la plateforme décrite ci-dessus et des tests préliminaires réalisés avec quelques utilisateurs, des cas d'utilisation utiles pour les experts ont été dégagés. Cette section décrit les deux scénarios d'utilisation de cette plateforme : pour améliorer le score des résultats ou pour raffiner une solution de fitness élevée. Plutôt que de décrire les scores détaillés nous nous concentrons ici sur la description haut-niveau des tâches.

Distillation de connaissances sur les lignes mal décrites Dans le cas où la EG seule n'est pas suffisante pour construire une expression, nous comparons dans la figure 6.4a les apprentissages interactif et non-interactifs sur une ligne où la meilleure fitness était inférieure à 0,8 sans interactivité. La boîte à moustaches bleue représente l'évolution de la fitness sur 40 apprentissages non interactifs, et la ligne rouge montre la meilleure fitness avec interactivité.

Alors que les apprentissages non micro-interactifs ont tendance à stagner après quelques itérations et peuvent s'améliorer après la 50^{eme} génération, le modèle interactif continue à s'affiner après la 70^{eme} génération. Ces éléments penchent en faveur de l'interactivité pendant l'apprentissage pour éviter les optima locaux. De plus, la version interactive obtient les meilleurs résultats à partir de la 60^{eme} génération. Cependant, celle-ci ne surpasse pas celle non interactive dès la 1^{ere} interaction. Il semble que l'expert ne puisse pas distiller ses connaissances dans les individus trop simples du début ou bien que l'algorithme a besoin de quelques itérations pour se calibrer.



(a) Comparaison entre l'exécution micro et macro-interactive et 40 exécutions non micro-interactives sur une variable cible difficile. Les essais non interactifs sont représentés par un diagramme en boîte bleu et l'essai interactif par un graphique linéaire rouge.



(b) Vue temporelle des individus à complexité variable, sur un cas simpliste. Les deux solutions (lignes rouges et vertes) correspondent visuellement à la variable cible (en bleu) et ont un score similaire. L'expertise est nécessaire pour identifier la solution la plus pertinente.

FIGURE 6.4 – Description des deux cas d'utilisation identifiés.

Réduire la complexité des solutions déjà efficaces Le deuxième cas d'étude est celui où une évolution sans interaction pendant l'apprentissage donne déjà de bons résultats. La figure 6.4b montre un exemple d'une telle situation. Dans ce cas, les évolutions non-interactives convergent en quelques itérations vers une solution proche de la solution optimale, et l'algorithme propose plusieurs solutions de scores similaires mais de complexités diverses. Les connaissances d'un expert seraient utiles ici pour identifier le compromis précision/complexité le plus pertinent. Dans ce cas, deux solutions de score similaire et élevé sont souvent visuellement quasiment identiques. La solution la plus simple est-elle suffisante ou trop extrême ? Seul un expert pourrait privilégier une solution en se référant à ses connaissances.

6.2.5 Bilan de l'expérience préliminaire

Dans ce travail préliminaire, nous proposons une approche interactive pour la compréhension des phénomènes physiques à partir d'une plateforme interactive multi-niveaux conçue pour l'élicitation des connaissances des experts. Les experts peuvent y fournir des informations sur leur compréhension du problème à différentes étapes du processus, à la fois dans chaque exécution et entre deux exécutions. Ce travail nous a également permis d'identifier deux scénarios d'interactions correspondant aux deux côtés du spectre, à la fois sur les lignes mal décrite par l'algorithme présenté dans le chapitre 4, mais également sur les lignes pour lesquelles les variables créées étaient pertinentes.

Le temps d'un expert étant coûteux, nous n'avons pas pu jusqu'alors faire des études approfondies auprès des utilisateurs. Par ailleurs, certains mécanismes d'interaction proposés ici, particulièrement celui basé sur la proposition de solution pourraient être complétés par d'autres interactions. En effet, bien qu'il s'agisse d'une tâche à haute valeur ajoutée sur laquelle les utilisateurs sont prêts à passer du temps (Pommeranz *et al.*, 2012), c'est également un tâche exigeante et pouvant être perçue comme difficile. Nous avons donc cherché dans la suite de ces travaux à proposer un algorithme permettant toujours ce mécanisme d'interaction, mais qui ne serait plus désormais le mécanisme d'interaction principal à partir duquel apprend l'algorithme.

6.3 Problématique

Comme nous l'avons mentionné plus haut dans la sous-section 6.2.5, l'iML (interactive Machine Learning) présente l'avantage de permettre à l'utilisateur d'intervenir pendant l'apprentissage d'un algorithme. Durant cette phase d'interaction, l'utilisateur est alors en mesure de guider l'apprentissage en utilisant sa connaissance. En effet, la connaissance d'un métier implique pour un expert qualifié de juger et de raisonner à partir d'un ensemble d'informations et d'un contexte. L'avis et le raisonnement d'un expert, forgés par l'expérience, sont majoritairement heuristiques et ne sont souvent pas accessibles de manière directe par des observations (Shadrick *et al.*, 2005). De fait, une partie de cette connaissance est "difficile à décrire, à examiner et donc à réutiliser" (Ribeiro, 2013). Prendre en compte cette connaissance pendant l'apprentissage peut dès lors difficilement se faire à partir d'observations et l'utilisateur doit pouvoir éliciter cette connaissance pendant l'apprentissage.

Dans la section 6.2, nous avons identifié différents scénarios où l'interactivité présenterait un intérêt dans le cadre applicatif étudié. Néanmoins, les interactions mises en oeuvres dans ces premiers travaux réalisés à partir d'interactions exhaustives, nécessitent un temps élevé d'interactions avec l'expert. Ceci n'est pas facilement compatible avec l'application finale visée pour le réseau électrique. Également, l'un des freins déjà identifiés à l'utilisation d'AG (Apprentissage Génétique) est la fatigue de l'utilisateur qui s'accroît au fil des itérations (Takagi, 2001). Nous avons donc souhaité explorer d'autres mécanismes d'interactions permettant de contrebalancer le phénomène de fatigue de l'utilisateur. Dans ce cadre, nous avons par ailleurs déjà envisagé dans la section 6.2 ne donner la possibilité à l'utilisateur de ne pas interagir à toutes les générations de l'EG (Evolution Grammaticale), mais plutôt de choisir une fréquence d'interaction avec l'algorithme pour soulager l'effet de fatigue. Nous avons souhaité poursuivre dans cette direction en étudiant l'effet de la fréquence d'interaction sur l'apprentissage. De même, pour rendre l'expérience de l'utilisateur plus agréable et réduire le nombre d'interactions requises, nous avons décidé de mettre en oeuvre plusieurs directives (Amershi *et al.*, 2014) telles que la reconnaissance du fait que les utilisateurs veulent donner plus que de simples "labels", et qu'ils aiment démontrer comment l'algorithme devrait fonctionner. Dans la section 6.4, nous avons donc souhaité proposer plusieurs modalités d'interaction différentes, que l'utilisateur pourra utiliser ou non à chaque étape d'interaction.

Par ailleurs, d'après les résultats obtenus dans le chapitre 5, l'utilisation d'AR (Apprentissage par Renforcement) semble être une approche pertinente pour la RS (Régression Symbolique) guidée par la grammaire. Ceci ouvre aussi sur la possibilité d'utiliser d'autres catégories d'algorithmes interactifs que ceux utilisés en AG et EG. En effet, l'AR rend possible l'utilisation d'autres catégories d'algorithmes interactifs tel l'apprentissage sur des préférences (Wirth *et al.*, 2017). Dans ces algorithmes, l'utilisateur peut éliciter ses préférences sous forme d'un signal numérique de récompense, d'un feedback oridinal, ou encore de règles `if_then_else` en langage "naturel" (Maclin *et al.*, 2005). Les préférences sont ensuite apprises par la politique par différents moyens : soit par apprentissage direct d'une politique via des préférences, soit par apprentissage par une fonction de substitution (surrogate) tel qu'un modèle des préférences de l'expert ou une fonction numérique. Dans ces trois cas, il en résulte un nouvelle politique π permettant d'échantillonner de nouvelles trajectoires correspondant de plus en plus aux préférences des utilisateurs. Nous nous intéresserons dans nos travaux à l'apprentissage direct d'une politique sur des préférences du fait de la compatibilité de cette approche avec les travaux menés dans le chapitre 5. Par ailleurs, les préférences peuvent être données à l'échelle d'une action, d'un état ou bien d'une trajectoire complète. Cependant, réaliser des analyses à l'échelle des trajectoires demande moins d'effort de la part de l'utilisateur que de le faire sur les actions ou états. Ainsi, afin de

respecter notre objectif de ne pas sur-solliciter l'utilisateur, nous avons choisi de nous intéresser particulièrement aux algorithmes travaillant sur des préférences à l'échelle de trajectoires. Notons également que l'élicitation de préférences sur des paires peut être facilitée par l'utilisation de catégories de préférences telles que des groupes ordonnés "Bon" (>) "Moyen" (>) "Mauvais". En effet, en répartissant des trajectoires dans différentes catégories selon ses préférences, l'utilisateur est alors capable de générer un grand nombre de paires de préférences à partir de l'ordre des catégories (Kuhlman *et al.*, 2018).

Enfin sur le plan de la RS interactive utilisant l'AR des travaux récents (Kim *et al.*, 2020) proposent une interface interactive pour contrôler l'algorithme d'AR par des mises à jour dynamiques des hyperparamètres et la sélection/suppression d'expressions symboliques dans le batch. Cependant, ces travaux n'explorent pas la possibilité d'utiliser d'autres métriques d'apprentissage sur des préférences.

Les différents éléments présentés ci-dessus ont donc motivé la construction du système interactif où une politique est apprise à partir de paires préférences sur des trajectoires et dans lequel les paires de préférences peuvent être générées en utilisant différentes interactions.

6.4 Description du système

La procédure d'entraînement utilisée est présentée dans la figure 6.5. L'interface interactive est ensuite détaillée dans la sous-section 6.4.2. L'algorithme est basé sur celui présenté dans le chapitre 5, une approche par renforcement pour la RS, où des règles grammaticales sont utilisées pour contraindre la construction d'expressions symboliques. Comme dans le chapitre 5, nous considérons ici un Processus de Décision Markov Partiellement Observable (POMDP) (Kaelbling *et al.*, 1998) avec un horizon fini et un signal de récompense éparse. Le système présenté ici implémente les éléments suivants :

— Nous proposons une plateforme interactive pour effectuer une RS guidée par renforcement grammatical interactive (RBG2-SR interactive, aussi notée iRBG2-SR) en apprenant directement à partir de préférences humaines sur des paires d'expressions.

— Nous proposons plusieurs façons d'obtenir des préférences : 1) avec *catégories de préférence*, en étiquetant les expressions comme "Bon", "Moyen", ou "Mauvais", ce qui permet de générer de nombreuses paires de préférences en moins d'interactions (Kuhlman *et al.*, 2019) ; 2) avec une définition directe de *préférence sur les paires* ; 3) ou avec *suggestion de solution* en essayant d'améliorer une expression proposée par l'algorithme avec une expression créée par l'utilisateur.

— Nous intégrons plusieurs tâches de RS tirées de benchmarks de l'état de l'art avec une complexité croissante, ainsi que leur grammaire ; et un jeu de données de RS industriel à partir duquel nous pouvons effectuer une RS exploratoire plus complexe.

6.4.1 Algorithme d'apprentissage

La recherche d'expressions symboliques se décompose selon les étapes suivantes :

Procédure P1 : Lancement de l'entraînement Pour lancer l'entraînement, l'utilisateur doit sélectionner un fichier de jeu de données, un fichier de grammaire au format BNF (Knuth, 1964) et une fréquence d'interaction avec l'algorithme. Le jeu de données spécifie la variable cible à modéliser y et fournit des instances X pour évaluer l'expression. Une *action* est définie comme la sélection d'une règle dans la grammaire, tel que présenté dans le chapitre 5.

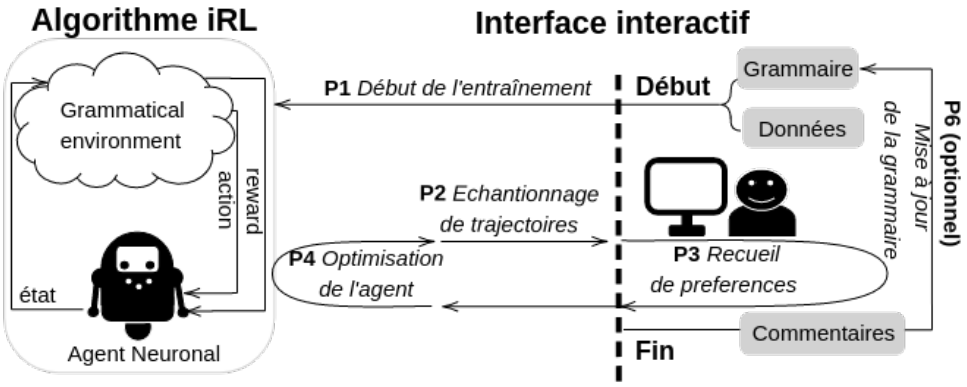


FIGURE 6.5 – Aperçu du processus d'apprentissage interactif.

Procédure P2 : Échantillonnage de trajectoires À chaque étape de la création d'une trajectoire, l'agent neuronal π génère une distribution sur les actions grammaticales accessibles P_θ (les règles peuvent être masquées en fonction de l'état actuel). L'action est ensuite échantillonnée à partir de cette distribution. Une séquence d'actions construit une trajectoire τ, ensuite traduite en une expression symbolique f_τ. Cette procédure est vectorisée pour générer un batch de trajectoires. Comme dans le chapitre 5, les trajectoires sont évaluées par la *récompense cumulative* R_τ de l'équation 5.1 qui utilise une erreur quadratique moyenne (EQM) mise à l'échelle .

Procédure P3 : Collecte des préférences Les meilleures expressions du batch sont affichées à l'utilisateur. Le mécanisme de collecte des préférences est détaillé dans la sous-section 6.4.2.

Procédure P4 : Optimisation de l'agent L'agent neuronal π est entraîné à l'aide de l'algorithme REINFORCE (Williams, 1992) avec un comportement de recherche de risque (Petersen *et al.*, 2021). Pour deux trajectoires τ_1, τ_2, nous désignons par $\tau_1 > \tau_2$ la préférence de τ_1 sur τ_2. Les préférences et l'équivalence entre les trajectoires sont insérées dans la fonction objectif, afin de maximiser la probabilité d'occurrence des trajectoires préférées tout en évitant les trajectoires non préférées selon une perte de désaccord par paire pondérée (Duchi *et al.*, 2010). Si deux trajectoires sont jugées pertinentes, elles sont toutes deux maximisées.

$$J_\theta = \mathop{\mathbb{E}}_{\substack{(\tau_2,\tau_1)\sim\pi \\ \tau_1>\tau_2}} [log(P_\theta(\tau_1))(R(\tau_1)) - log(P_\theta(\tau_2))(R(\tau_2)) \,\big|\, (R(\tau_1) > R_\epsilon \; et \; R(\tau_2) > R_\epsilon)] \tag{6.1}$$

Entre les exécutions où des interactions se produisent, les trajectoires peuvent être re-simulées, et la préférence réutilisée. Les procédures 2 à 4 sont répétées dans cet ordre jusqu'à ce qu'un critère de fin soit satisfait : soit la solution exacte trouvée, soit le nombre maximal d'itérations d'apprentissage atteint.

Procédure P5 : Mise à jour de la grammaire. En option, à la fin de l'apprentissage, l'utilisateur peut commenter et affiner les contraintes grammaticales comme précédemment dans le chapitre 4. Une fois transposés dans de nouvelles règles, ces commentaires amélioreront la grammaire utilisée lors des prochaines sessions d'entraînement.

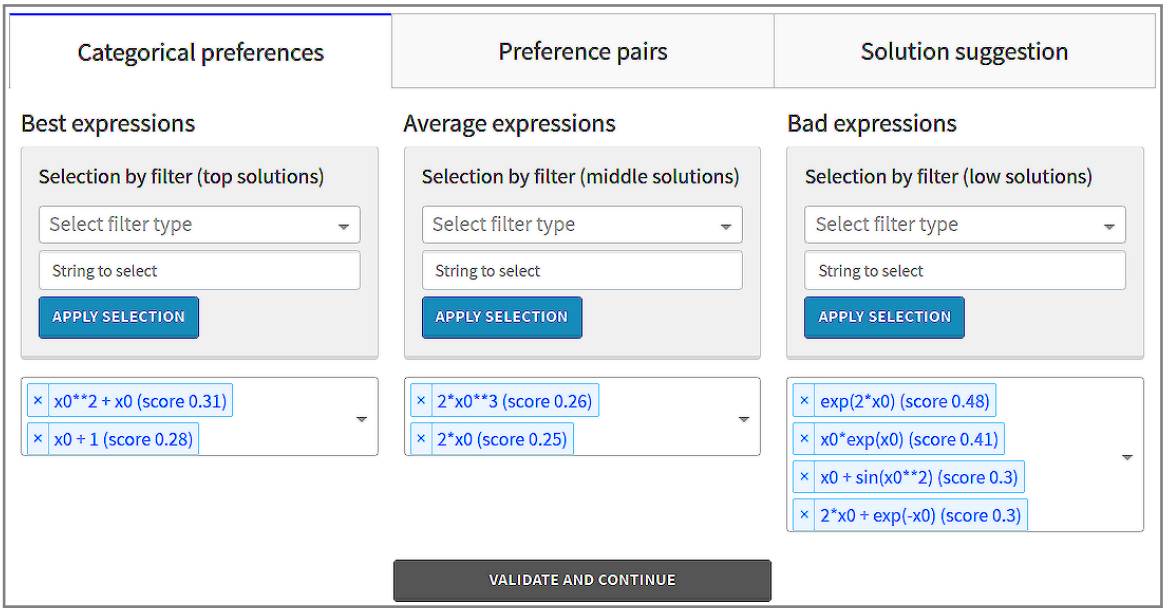


FIGURE 6.6 – Onglet d'interaction par des préférences catégorielles.

6.4.2 Interface d'interaction

À une fréquence d'interaction choisie dans l'étape P1, les meilleures expressions du batch sont présentées à l'utilisateur, ordonnées de manière décroissante selon leur récompense cumulée sur la trajectoire. Les préférences sur les expressions sont collectées pendant la procédure P3 (voir figure 6.5) selon trois mécanismes différents :

Préférences par catégories Comme le montre la figure 6.6, on demande à l'utilisateur de classer les expressions dans trois catégories ordonnées : "Best" ("Meilleure"), "Average" ("Moyenne"), et "Bad" ("Mauvaise"). Nous générons ensuite des paires de catégories, en considérant que chaque expression de la catégorie "Best" est meilleure que toutes les expressions des deux catégories inférieures, (de même "Average" > "Bad"). Cette stratégie, dérivée des travaux de (Kuhlman *et al.*, 2018), permet de générer plus de paires en moins d'interactions que les paires de préférences directes. Nous proposons un filtrage par Expressions Régulières sur les expressions restantes pour réduire encore le nombre de clics de l'utilisateurs.

Préférences sur des paires Le deuxième onglet présenté dans la figure 6.7 présente initialement à l'utilisateur une liste de paires d'expressions supérieures sélectionnées au hasard. Il est inspiré de la stratégie de préférence sur les segments (Christiano *et al.*, 2017). L'utilisateur peut préférer l'une, l'autre, les deux, ou aucune des deux expressions de chaque paire. De nouvelles paires peuvent être ajoutées manuellement à la liste. Cet onglet, présenté dans la figure 6.7, est complémentaire à la première interaction et vise à affiner la préférence sur les catégories. Par exemple, l'utilisateur peut sélectionner la préférence sur les expressions classées dans la même catégorie dans le premier onglet. Cependant, comme cette interaction est plus longue (Kuhlman *et al.*, 2019), elle n'est pas considérée comme le principal mécanisme de collecte des préférences.

Suggestion directe de solutions Le troisième onglet de la figure 6.8 se concentre sur la suggestion directe de solutions susceptibles d'améliorer une expression échantillonnée par l'al-

FIGURE 6.7 – Onglet d'interaction sur des paires de préférences.

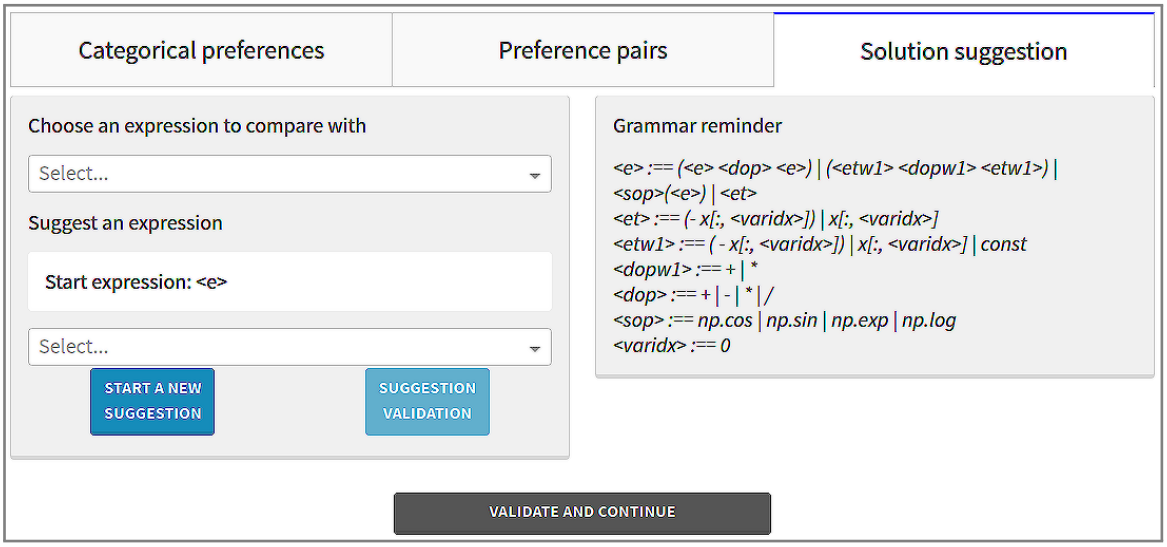


FIGURE 6.8 – Onglet de suggestion de solutions.

gorithme. Ici, l'utilisateur sélectionne d'abord une expression dans le batch vis-à-vis de laquelle comparer celle suggérée directement. Ensuite, il crée une nouvelle expression à partir de zéro en sélectionnant itérativement des actions-règles dans la grammaire au format BNF. L'expression proposée directement par l'utilisateur sera considérée comme meilleure que celle sélectionnée pour comparaison, ceci afin de créer une paire. Cet onglet est destiné aux utilisateurs familiers avec le fonctionnent des grammaires non-contextuelles. Il peut permettre d'éviter les optima locaux lorsque l'algorithme manque de diversité.

6.5 Cas d'utilisation

Nous avons implémenté une plateforme basée sur le web en utilisant la bibliothèque Dash Plotly en Python (Hossain *et al.*, 2019). L'architecture du réseau neuronal utilise l'implémentation PyTorch pour le CPU (Paszke *et al.*, 2019). Deux cas d'utilisation ont été associés à la plateforme : un premier volontairement simple pour calibrer l'interface, puis un second permettant de représenter un exemple simplifié du réseau électrique avec lequel faire interagir le plus grand nombre possible d'utilisateurs.

6.5.1 Exploration d'une fonction issue d'un benchmark

Tout d'abord, nous proposons de tester notre plateforme sur une fonction du benchmark standard de SR. L'expression symbolique cible réelle est connue de l'équipe de recherche mais ne l'est pas par l'utilisateur. L'objectif est ici pour l'utilisateur de récupérer la formule exacte d'une expression sélectionnée dans le benchmark SR de Nguyen (Uy *et al.*, 2011). La grammaire décrite dans la figure 5.4 du chapitre 5 est fournie. Elle comprend des fonctions telles que : +, -, ×, / exp, log, sin, cos. Deux cas peuvent être étudiés : 1) L'expression exacte n'est pas connue a priori. L'utilisateur doit explorer les expressions proposées à travers des comparaisons visuelles et basées sur le score pour se forger itérativement une opinion sur la solution correcte. 2) L'utilisateur peut avoir à l'esprit (ou recevoir) des concepts liés au domaine qui sont nécessaires pour représenter l'expression cible. Avec ces concepts en tête, l'utilisateur peut alors essayer d'agir comme un enseignant (Mosqueira-Rey *et al.*, 2021) en trouvant des exemples pour représenter chaque concept et les faire enseigner à l'algorithme en itérant sur les étapes d'explication/révision.

6.5.2 Exemple tiré des réseaux électriques

Ce deuxième exemple se concentre sur un cas d'utilisation industrielle : représenter un ensemble de mesures historiques de capteurs sous la forme d'une expression symbolique qui respecte les propriétés physiques liées au domaine. Plus précisément, nous proposons d'effectuer une tâche de RS sur des données simulées de réseau électrique afin de découvrir une relation physique simplifiée inconnue sur une ligne électrique donnée. Les données simulées sont obtenues à l'aide de la plateforme Grid2Op (Donnot, 2020a), où des experts en systèmes électriques ont sélectionné la ligne électrique à étudier comme étant représentative de leur objectif industriel. Une grammaire prédéfinie (Crochepierre *et al.*, 2020) est proposée pour insérer les relations physiques telles que les lois d'Ohm et de Kirchoff.

6.6 Expériences : Calibrage de l'interaction par simulation de comportements utilisateurs

6.6.1 Objectifs

Dans cette expérience nous cherchons à construire une procédure automatisée permettant de calibrer l'interaction avec l'utilisateur selon différents paramètres tels que la fréquence d'interaction et le type de préférences fourni par l'utilisateur. En simulant la capacité d'apprentissage de l'algorithme selon différentes catégories de préférences utilisateurs, et pour différentes fréquences d'interactions, nous proposons une expérience permettant de répondre aux questions suivantes :

Question de recherche 1 : La version interactive de l'algorithme permet-elle d'améliorer l'apprentissage ?

Question de recherche 2 : Quelle est la fréquence d'interaction maximale jusqu'à laquelle l'algorithme est capable d'apprendre ?

Question de recherche 3 : Un apprentissage basé uniquement sur les préférences et sur la réutilisation des préférences entre les interactions est-elle utile ?

Ainsi pour calibrer au mieux la fréquence d'interaction et répondre aux questions ci-dessus, nous proposons dans un premier temps un test préliminaire de l'interface décrit en section 6.4 sur un panel d'utilisateurs restreints afin de mettre en évidence les concepts et les hypothèses formulés puis testés par les utilisateurs. Puis, à partir de l'analyse des tests préliminaires, certaines préférences fréquentes des utilisateurs ont été encodées sous forme de règles pouvant ensuite simuler le comportement des utilisateurs, comme décrit respectivement dans les sous-sections 6.6.3 et 6.6.4.

6.6.2 Interviews préliminaires

Afin de calibrer au mieux l'interaction entre algorithme iRBG2-SR et l'utilisateur, des tests de l'interface ont été réalisés par trois utilisateurs, sur le cas de la fonction Nguyen4 notée f_4 avec pour rappel $f_4(x) = x^6 + x^5 + x^4 + x^3 + x^2 + x$. Cette fonction a été choisie a partir des résultats du chapitre 5 montrant qu'au bout des 2000 itérations d'apprentissage f_4 est trouvé de façon exacte seulement une faible proportion du temps avec l'algorithme RBG2-SR. Un deuxième argument pour le choix de cette fonction est la simplicité de sa formulation, qui nous permettra par la suite de la faire tester par un plus grand panel d'utilisateurs, n'ayant pas forcément une connaissance mathématique avancée.

Déroulement des tests préliminaires

Pour cette expérience, chaque utilisateur disposent d'un niveau d'information différent sur les données étudiées :

— Le premier utilisateur (U1) ne dispose d'aucune information spécifique sur la fonction symbolique à trouver. La tâche de cet utilisateur, la plus complexe d'entre les 3 utilisateurs, étant de se forger une intuition sur la forme de la solution en explorant les solutions fournies par l'algorithme.

— Le deuxième utilisateur (U2) possède comme information le fait qu'il s'agisse d'une fonction polynomiale. La tâche de ce deuxième utilisateur étant alors de d'analyser les expressions symboliques proposées et de guider l'apprentissage de manière à respecter la connaissance a priori de la forme de la solution.

— Le dernier (U3) connaît la solution exacte f_4. Sa tâche consistait alors à retrouver la solution exacte.

Les trois utilisateurs sont ingénieurs de formation (dont un docteur), avec au moins 7 ans d'ancienneté dans le domaine de l'énergie. Après s'être fait expliqué le fonctionnement de l'interface selon la description donnée en section 6.4, les utilisateurs réalisent des tests libres de l'interface. L'interaction entre l'algorithme et l'utilisateur a lieu toutes les 5 étapes d'apprentissage de l'agorithme RBG2-SR. Tous les utilisateurs apprennent à partir de la même grammaire définie en figure 5.4 du chapitre 5. Ainsi, 4 étapes sur 5 de l'algorithme fonctionnent en apprenant avec la fonction objective du chapitre 5 basée sur l'équation 5.4 et, pour une étape sur 5, l'apprentissage est réalisé à partir des préférences des utilisateurs d'après l'équation 6.1. Une fois les tests réalisés, les utilisateurs sont ensuite interrogés sous forme d'interviews libres afin de recueillir leurs impressions sur l'apprentissage ainsi que les *concepts* qu'ils ont cherché à faire apprendre au fil des interactions.

Synthèse des interviews

Le premier utilisateur U1 a réalisé deux tests successifs de l'interface de respectivement 3 et 15 interactions. Le premier apprentissage a semble-t-il servi à se familiariser avec l'utilisation de l'interface. Le deuxième utilisateur a réalisé un apprentissage sur 21 interactions. Enfin, U3 a réalisé deux apprentissages de respectivement 16 et 26 interactions au cours desquelles il est parvenu à retrouver l'expression exacte. Les apprentissages des utilisateurs U1 et U2 ont été arrêtés lorsque les utilisateurs n'avaient plus de préférences à fournir à l'utilisateur. Bien que l'interface permette de sauter des étapes d'interactions, il aurait été intéressant pour eux de pouvoir automatiquement faire poursuivre l'apprentissage sans aucune interaction pour voir si les préférences données par les utilisateurs au cours des premières interactions étaient suffisantes pour guider l'apprentissage.

Concernant les concepts construits par les utilisateurs, qu'ils ont ensuite cherché à faire apprendre à l'algorithme, certains groupes de raisonnements ont été identifiés. Tout d'abord à partir de l'analyse des nuages de points, comme représenté à droite dans la figure 6.7, notamment l'analyse des points d'inflexion, l'utilisateur U1 a d'abord pu appréhender que la fonction à trouver ne pouvait être une sinusoïde, ni un logarithme. Il s'agit du premier concept principalement utilisé par cet utilisateur pour créer les paires préférences. L'utilisateur U2 a lui également utilisé le même concept que celui de U1, mais s'est aussi servi de sa connaissance a priori de la forme de la fonction pour générer les préférences mettant dans la catégorie "Bad" toutes les expressions comportant une fonction exponentielle. Enfin l'utilisateur U3 a utilisé les deux concepts décrits précédemment, qu'il a complété avec une préférence sur les solutions ayant le score le plus élevé. Ce dernier concept était majoritaire dans les derniers interactions de l'apprentissage, une fois que les solutions générées n'étaient ni des sinusoïdes, ni des logarithmes.

A partir des informations fournies par les utilisateurs, des concepts ont été dégagés. Ceux-ci serviront ensuite pour modéliser le comportement des utilisateurs sur des paramètres pouvant influencer l'apprentissage interactif.

6.6.3 Modélisation du comportement utilisateur

Les différents concepts des utilisateurs U1-3 recueillis lors des interviews, ont permis de définir 3 modèles de comportements :

— un premier modèle de comportement, m_{best}, dans lequel un utilisateur simulé choisis toujours pour chaque paire la solution ayant le score de récompense la plus élevée ;

— un second modèle de comportement, $m_{random\_prob\_a}$ où $a \in \{0.2, 0.5, 0.8\}$, dans lequel un utilisateur simulé choisis aléatoirement sa préférence $a\%$ du temps et la solution de récompense la plus élevée le reste du temps ;

— un dernier modèle de comportement, $m_{regex\_string\_rule}$, où $string\_rule$ représente une Expression Régulière (RegEx). Ce dernier modèle permet par exemple de modéliser la préférences de fonctions autre que sinusoïdales, logarithmes.

Ces modèles de comportement ont été implémentés en Python afin de les simuler selon différentes fréquences d'interaction prise entre 1, 2, 5, 10, 15 et 20. Une fréquence d'interaction de 1 correspondant à un apprentissage basé uniquement sur les préférences à toutes les itérations. Afin de ne pas donner un trop grand avantage à la machine, capable de comparer rapidement un grand nombre d'expressions symboliques, nous avons choisi de restreindre l'algorithme à ne comparer que 5 paires par itération de l'algorithme. Sachant que les modèles construits ne sont pas réalistes individuellement, mais composent un partie du raisonnement d'un utilisateur (qui pourra en réalité alterner entre plusieurs de ces modèles au cours de l'apprentissage), l'objectif des simulations sera notamment de trouver une fréquence d'interaction adaptée pour à peu près tous les modèles. Nous choisirons donc de manière privilégiée la fréquence d'interaction la plus élevée possible qui nous permettra d'obtenir l'erreur la plus faible, en moyenne sur tous les modèles. Les résultats des simulations de ces modèles de comportement sont synthétisés dans la sous-section 6.6.4.

6.6.4 Résultats des simulations d'utilisateurs

Pour chaque fréquence d'interaction et chaque modèle de comportement, 10 apprentissages ont été réalisés. Deux types d'utilisations de préférences ont également été testés. Un premier mode d'apprentissage, sans réutilisation des préférences, utilise l'algorithme RBG2-SR décrit dans le chapitre 5 entre les étapes d'interactions sur les préférences. Le deuxième mode d'apprentissage testé, réutilise les préférences fournies par les utilisateurs entre les étapes d'interaction.

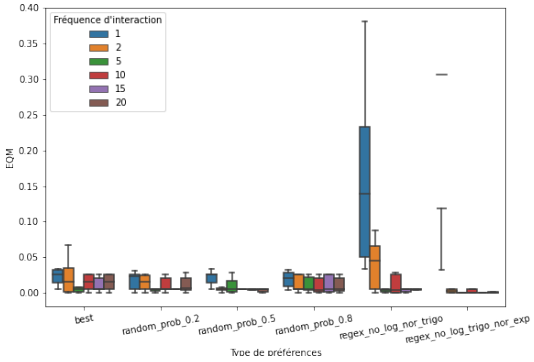
Les résultats obtenus et les scores en terme d'EQM et de coefficient de détermination R2 sont synthétisés dans la figure 6.9, la figure 6.10 et le tableau 6.1.

Tout d'abord dans la figure 6.9, nous comparons les scores d'EQM et R2 selon le modèle de préférence utilisé et la fréquence d'interaction avec l'utilisateur simulé. Dans les figures 6.9b et 6.9a sont représentés les scores d'EQM respectivement avec et sans réutilisation des préférences. De même, les figures 6.9d et 6.9c montre les scores de R2 respectivement avec et sans réutilisation des préférences.

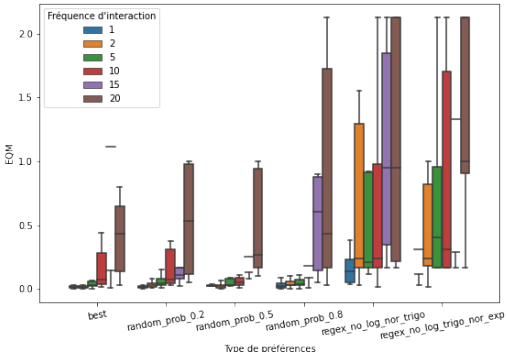
Utilité de la réutilisation des préférences

En comparant dans un premier temps les apprentissages avec et sans réutilisation de préférences, nous remarquons que l'erreur EQM et R2 est plus élevée lorsque les préférences sont réutilisées entre les interactions. Nous pouvons également remarquer que l'erreur est également plus élevée pour les modèles de comportements utilisant des RegEx (Expression Régulière) fixes. Une explication possible de ce résultat est le manque de diversité engendré par l'utilisation de RegEx. Par ailleurs, ce comportement perd en réalisme au bout de quelques itérations d'interactions, notamment lorsque toutes les expressions générée par la politique π_θ respectent la RegEx. Le modèle ne fournit alors plus aucune préférence. En mettant en évidence que la réutilisation des préférences n'est pas utile, ce premier résultat répond à la **question de recherche 3**.

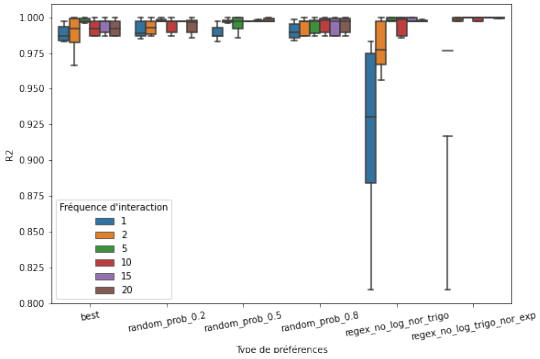
Concentrons nous maintenant sur les figures 6.9a et 6.9c. En regardant dans un premier temps les modèles basés sur les RegEx, nous pouvons voir que l'erreur est significativement plus élevée



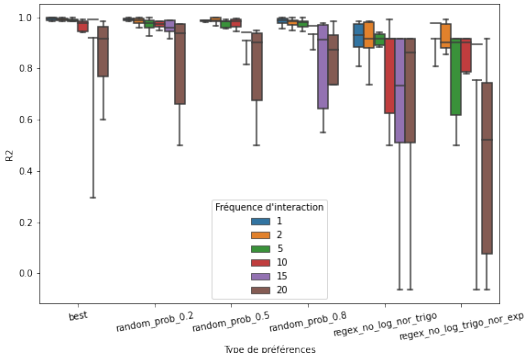
(a) EQM sans réutilisation des préférences entre deux interactions



(b) EQM avec réutilisation des préférences

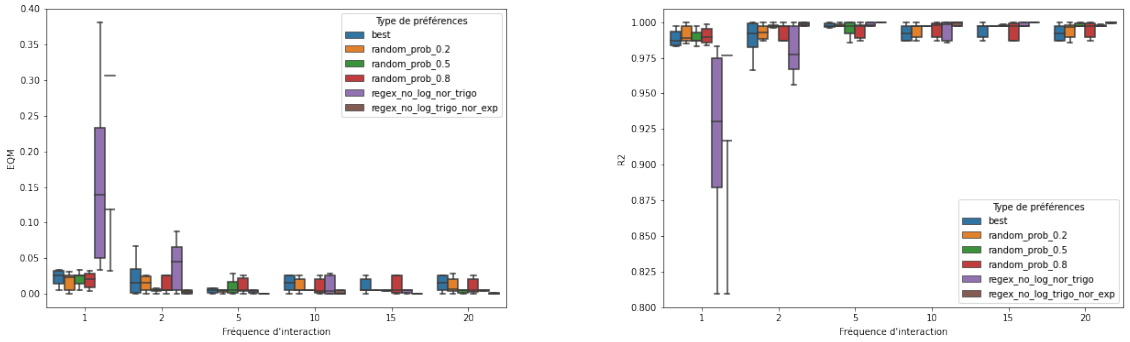


(c) R2 sans réutilisation des préférences.



(d) R2 avec réutilisation des préférences

FIGURE 6.9 – Boîtes à moustaches représentant les scores d'EQM et coefficients de détermination R2 en fonction du type de préférences simulées et de la fréquence d'interaction, pour la simulation de certains comportements utilisateurs sur le jeu de données Nguyen-4 sur 10 apprentissages, avec ou sans réutilisation des préférences entre les interactions.



(a) Scores d'EQM.

(b) Scores R2.

FIGURE 6.10 – Boîtes à moustaches des scores EQM (a) et R2 (b) en fonction de la fréquence d'interaction r sur 10 apprentissages, et séparé selon le modèle de préférence d'utilisateur.

pour une fréquence d'interaction de 1 (supérieure à 0.1 pour les deux RegEx) ou 2 (supérieure à 0.05 pour la Regex no_log_nor_trigo), vraisemblablement due à un manque de diversité des préférences sur les expressions comparées. L'erreur devient plus faible à partir d'une fréquence d'interaction de 5. En comparant les modèles best (préférant les expressions de meilleures récompenses) et random_prob (choisissant aléatoirement les préférences un certain pourcentage du temps), nous obtenons des résultat assez similaires pour ces 4 modèles. En moyenne pour tous les modèles, pour les fréquences d'interaction strictement supérieures à 2, l'erreur d'EQM est inférieure à 0.019 et le score de R2 est supérieur à 0.99. Ainsi, choisir systématiquement la solution de meilleur score ne constitue pas le meilleur apprentissage, mais plutôt choisir une fois sur deux une préférence basée sur un autre critère (ici modélisée par un choix aléatoire semble pertinent).

Il semblerait également que pour les fréquences d'interaction supérieures à 2, les modèles random_prob_0.5 et regex_no_log_trigo_nor_exp obtiennent des scores similaires et de bons scores. Le modèle choisissant systématiquement la récompense la plus élevée ne donne donc pas les meilleurs scores d'EQM et R2. Les préférences basées sur des critères autres que la récompense semblent donc plus informatives. Ces éléments fournissent donc un début de réponse à la **question de recherche 2**, en faveur de l'apprentissage interactif.

Regardons maintenant la figure 6.10. Celle-ci représente les mêmes données que celles des figures 6.9a et 6.9c, mais cette fois-ci avec la fréquence d'interaction sur l'axe des abscisses et séparé selon le type de préférences. Ces figures récapitulent l'information sur la fréquence d'interaction mentionnée plus haut : pour obtenir les meilleurs scores d'EQM et R2, une fréquence d'interaction supérieure ou égale à 5 semble nécessaire.

Analyse des scores moyennés sur tous les modèles de préférences

Le tableau 6.1 présente les scores d'EQM et R2, présentés par fréquence d'interaction et aggrégés en moyennant sur les 6 types de préférences. Pour rappel, nous cherchons à identifier une fréquence d'interaction permettant de minimiser le nombre d'interactions de l'utilisateur tout en obtenant l'erreur la plus faible possible. Le score obtenu dans le chapitre 5 est rappelé au début du tableau et servira de référence. Tout d'abord, en comparant les résultats obtenus de l'apprentissage avec réutilisation de préférences, nous rappelons comme précédemment que quelque

Ré-utilisation des préférences	Fréquence d'interaction	EQM moyenne (± écart-type)	R2 moyenne (± écart-type)
Référence (chap. 5)	-	$1.62 \times 10^{-2} (\pm 1.6 \times 10^{-2})$	9.92×10^{-1} $(\pm 8.2 \times 10^{-3})$
Non	1	$6.57 \times 10^{-2} \pm (8.91 \times 10^{-2})$	$9.67 \times 10^{-1} \pm (4.45 \times 10^{-2})$
	2	$6.14 \times 10^{-2} \pm (2.09 \times 10^{-1})$	$9.69 \times 10^{-1} \pm (1.04 \times 10^{-1})$
	5	$\mathbf{7.21 \times 10^{-3} \pm (8.93 \times 10^{-3})}$	$\mathbf{9.96 \times 10^{-1} \pm (4.46 \times 10^{-3})}$
	10	$1.14 \times 10^{-2} \pm (1.79 \times 10^{-2})$	$9.94 \times 10^{-1} \pm (8.92 \times 10^{-3})$
	15	$8.56 \times 10^{-3} \pm (1.03 \times 10^{-2})$	$\mathbf{9.96 \times 10^{-1} \pm (5.12 \times 10^{-3})}$
	20	$1.13 \times 10^{-2} \pm (1.71 \times 10^{-2})$	$9.94 \times 10^{-1} \pm (8.57 \times 10^{-3})$
Oui	1	$6.14 \times 10^{-2} \pm (8.88 \times 10^{-2})$	$9.69 \times 10^{-1} \pm (4.43 \times 10^{-2})$
	2	$1.17 \times 10^{-1} \pm (2.51 \times 10^{-1})$	$9.42 \times 10^{-1} \pm (1.25 \times 10^{-1})$
	5	$1.99 \times 10^{-1} \pm (3.57 \times 10^{-1})$	$9.01 \times 10^{-1} \pm (1.78 \times 10^{-1})$
	10	$2.77 \times 10^{-1} \pm (5.11 \times 10^{-1})$	$8.62 \times 10^{-1} \pm (2.55 \times 10^{-1})$
	15	$4.66 \times 10^{-1} \pm (6.03 \times 10^{-1})$	$7.67 \times 10^{-1} \pm (3.01 \times 10^{-1})$
	20	$7.17 \times 10^{-1} \pm (6.98 \times 10^{-1})$	$6.42 \times 10^{-1} \pm (3.49 \times 10^{-1})$

TABLE 6.1 – Moyennes et écart-types des scores d'EQM et coefficient de détermination R2. Moyennes réalisées sur 10 exécutions de l'algorithme et sur toutes les types de préférences donnés. Les tests sont réalisés pour plusieurs fréquences d'interactions, avec et sans réutilisation des préférences. La référence utilisée est celle de l'algorithme non interactif RBG2-SR, présenté dans le chapitre 5. Les meilleurs résultats sont en gras sont significativement supérieurs à la référence.

soit la fréquence d'interaction, l'EQM est plus grande que sans réutilisation des préférences. La comparaison avec la référence du chapitre 5 met également en évidence que la ré-utilisation des préférences est moins performante qu'aucune interaction, notamment liée au manque d'exploration de l'algorithme d'AR utilisant cet apprentissage.

Regardons maintenant le cas où nous ne réutilisons pas les préférences entre les interactions. Comme attendu après l'analyse des figures 6.9 et 6.10, lorsqu'une interaction est réalisée toutes les iterations et toutes les 2 iterations, l'EQM est supérieure à celle obtenue lors de l'apprentissage RBG2-SR non-interactif. Pour des interactions toutes les 5 à 20 pas de temps, l'erreur est plus faible et l'EQM est même divisée par deux pour une interaction toutes les 5 itérations.

Enfin, en comparant le score R2 sans réutilisation de préférences avec la référence, nous pouvons constater que le score R2 est meilleur que la référence pour une interaction calibrée toutes les 5 à 20 itérations. De plus, le score est très similaire pour les fréquences de 5 et 15, ce qui laisse présager que l'on puisse augmenter la fréquence d'interaction sans pour autant perdre trop en terme de score.

Ainsi, en réponse à la **question de recherche 1**, il semble donc que la version interactive de l'algorithme permette d'améliorer l'apprentissage et de diminuer l'erreur sur l'expression trouvée. Les préférences fournies par les utilisateurs semblent ici pousser la politique π_θ à *exploiter* certaines zones pertinentes de l'espace, sans pour autant trop contraindre l'*exploration*.

Par ailleurs, pour répondre à la **question de recherche 2**, la meilleure fréquence d'interaction identifiée ici semble être une interaction tous les 5 itérations en moyenne, bien que les valeurs moyennes semblent correctes jusqu'à une interaction tous les 20 itérations.

6.7 Interface unique pour la régression symbolique

6.7.1 Objectifs

Dans le chapitre 5, nous avons identifié que l'algorithme RBG2-SR fonctionnait mieux que l'EG (Evolution Grammaticale) sur la plupart des cas du benchmark étudié. Néanmoins, sur les

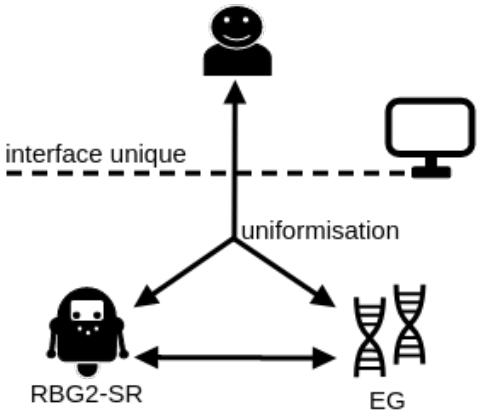


FIGURE 6.11 – Vue générale du paradigme multi-algorithmes avec une interface unique. Les algorithmes EG et RBG2-SR s'échangent tous des expressions symboliques (i.e. des individus et trajectoires) susceptibles d'améliorer l'apprentissage. Des expressions symboliques sont présentées à l'utilisateur. L'utilisateur fournit des indications sur la pertinences des expressions pouvant être utilisées par l'EG et/ou la RBG2-SR.

cas où RBG2-SR obtenait de moins bons scores, l'EG avait cette fois-ci de meilleurs résultats. Il semble donc que les algorithmes évolutionnaires et par renforcement puissent être complémentaires sur des tâches de RS. Dans la littérature, cette complémentarité a été illustrée en RS sans grammaire, par l'utilisation du renforcement pour l'initialisation de populations génétiques, aussi appelé Neural-Guided Genetic Programming Population Seeding, noté NG2P2S (Mundhenk *et al.*, 2021). L'AR (Apprentissage par Renforcement) pour obtenir de meilleures initialisations d'AG (Algorithme Génétique) semble donc permettre de tirer parti au mieux des capacités des deux méthodes.

Par ailleurs, comme nous l'avons précisé plus haut en section 6.3, l'interactivité en AG et en AR peut se faire selon différentes modalités et combiner les deux algorithmes pourrait alors fournir à l'utilisateur une plus grande diversité dans ses interactions avec l'algorithme et plus précisément, potentiellement plusieurs manières de prendre en compte des informations fournies par les utilisateurs.

6.7.2 Description du paradigme multi-algorithmes dans une interface unique

A partir de ces éléments, nous proposons de définir comment apprendre conjointement avec l'algorithme d'EG et de RBG2-SR et de manière à ce que l'utilisateur utilise une seule interface pour interagir. Nous illustrons ceci dans la figure 6.11. Comme nous l'avons présenté en introduction du chapitre 5, une tâche de RS résolue par des algorithmes évolutionnaires ou par renforcement présente des similarités notamment vis-à-vis de la structure des éléments générés. Des individus génétiques et des trajectoires de renforcement sont traduites sous forme d'expressions symboliques présentées ensuite à l'utilisateur. L'utilisateur peut ainsi fournir des informations sur la pertinence des expressions symboliques quelque soit les algorithmes les ayant générés.

Afin de pouvoir tester ce paradigme et l'illustrer sur un algorithme déjà existant pour des tâches de RS combinant AG (Algorithme Génétique) et AR (Apprentissage par Renforcement), nous avons choisi d'inclure l'interactivité dans l'algorithme NG2P2S mentionné plus haut (Mundhenk *et al.*, 2021). A partir du choix d'un algorithme de renforcement (ici RBG2-SR, présenté dans le chapitre 5) et un algorithme génétique (ici EG, Evolution Grammaticale), l'apprentis-

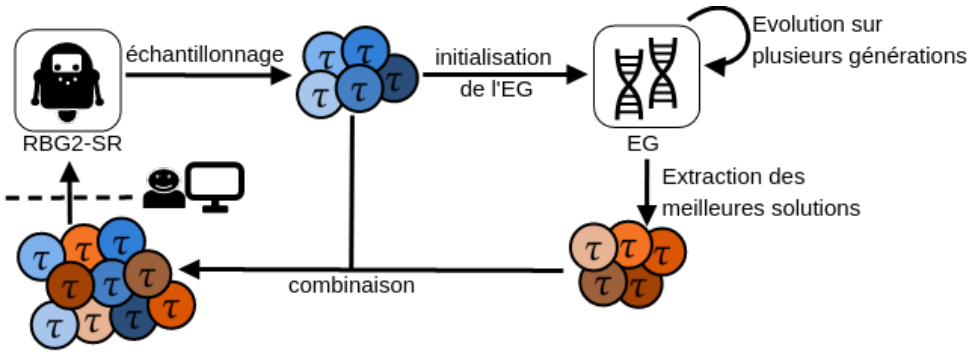


FIGURE 6.12 – Apprentissage joint RBG2-SR et EG utilisant un seul interface d'interaction, illustré sur l'algorithme NG2P2S. Le schéma est inspiré de la Figure 1 du papier (Mundhenk *et al.*, 2021).

sage fonctionne de la manière suivante. Un ensemble de trajectoires est d'abord échantillonné par renforcement (AR). Ces trajectoires sont utilisées pour initialiser une population génétique évoluées sur plusieurs générations (AG). A la fin de l'évolution, les meilleurs individus de l'AR et l'AG sont sélectionnées et combinées pour apprendre une politique optimale π_θ de renforcement selon l'équation 5.4 du chapitre 5. Dans notre cas d'utilisation nous choisissons l'EG comme AG, et RBG2-SR comme AR. Ces itérations d'échantillonnage/évolution/combinaisons sont répétées jusqu'à convergence de l'algorithme ou sur un nombre fixe d'itérations. Selon la présentation donnée en 6.11, nous proposons d'inclure une étape d'interactivité de manière à être agnostique aux deux algorithmes, c'est à dire une fois les meilleures solutions de l'AR et AG sont combinées et avant la mise à jour de la politique π_θ.

6.7.3 Premiers résultats

Une expérience a été menée afin de mettre en évidence de manière quantitative l'intérêt d'une telle approche, combinant interactivité, AR et AG pour la RS sur la fonction f_4. A partir de l'algorithme illustré dans la figure 6.11, et les paramètres utilisés dans les expériences de la 5.4 du chapitre 5 ainsi que les plages de fréquences d'interactions identifiées comme pertinentes dans les simulations de la sous-section 6.6.4. Plus précisément, à partir de l'algorithme RBG2-SR et EG tel que définit dans le chapitre 5, nous réalisons 10 apprentissages sans interactivité. Pour chaque type de préférence tel que simulés dans la partie précédente, 10 apprentissages AR+AG sont également réalisés avec une interaction tous les 5, 15 ou 25 pas de temps. Nous nous comparons aux résultats du chapitre 5 et de la sous-section 6.6.4. Les résultats moyennés sur tous les modèles d'interaction sont résumés dans la tableau 6.2.

En haut du tableau les différents scores sont rappelés : obtenus par RBG2-SR et EG (Evolution Grammaticale) dans le chapitre 5, ou par iRBG2-SR dans la sous-section 6.6.4). Dans un premier temps en comparant les scores obtenus par combinaison des deux méthodes RBG2-SR et EG (notée RBG2-SR + EG), nous pouvons constater que cette méthode obtient de meilleurs scores que les méthodes RBG2-SR et EG utilisées indépendemment. Cependant, en comparant RBG2-SR + EG sans interactivité, avec la version interactive de l'algorithme RBG2-SR (noté iRBG2-SR), nous n'obtenons pas de meilleurs résultats. Ainsi, l'ajout de l'interactivité apporte ici une plus grande amélioration de résultats que la combinaison de RBG2-SR avec une approche génétique d'EG.

Algorithme	Fréquence d'interaction	EQM moyenne (± écart-type)	R2 moyenne (± écart-type)
RBG2-SR (chap. 5)	-	$1.62 \times 10^{-2}(\pm 1.6 \times 10^{-2})$	9.92×10^{-1} $(\pm 8.2 \times 10^{-3})$
EG (chap. 5)	-	$1.48 \times 10^{-2}(\pm 1.6 \times 10^{-2})$	9.93×10^{-1} $(\pm 6.5 \times 10^{-3})$
iRBG2-SR (sec.6.6.4)	5	$\mathbf{7.21 \times 10^{-3}(\pm 8.93 \times 10^{-3})}$	$\mathbf{9.96 \times 10^{-1}(\pm 4.46 \times 10^{-3})}$
	15	$8.56 \times 10^{-3}(\pm 1.03 \times 10^{-2})$	$\mathbf{9.96 \times 10^{-1}(\pm 5.12 \times 10^{-3})}$
RBG2-SR + EG	-	$1.24 \times 10^{-2}(\pm 1.56 \times 10^{-2})$	$9.94 \times 10^{-1}(\pm 7.79 \times 10^{-3})$
i(RBG2-SR + EG)	5	$1.98 \times 10^{-2}(\pm 4.75 \times 10^{-2})$	$9.90 \times 10^{-1}(\pm 2.37 \times 10^{-2})$
	15	$\mathit{1.11 \times 10^{-2}(\pm 1.47 \times 10^{-2})}$	$\mathit{9.94 \times 10^{-1}(\pm 6.98 \times 10^{-3})}$
	25	$1.12 \times 10^{-2}(\pm 1.78 \times 10^{-2})$	$9.94 \times 10^{-1}(\pm 8.90 \times 10^{-3})$

TABLE 6.2 – Comparaison des scores d'EQM et coefficients de déterminations R2 en moyenne (± écart-type) pour différentes méthodes avec et sans interactivité sur une tâche de RS appliquée à la fonction f_4.

Regardons maintenant les performances d'une approche combinant RBG2-SR EG et interactivité, notée i(RBG2-SR + EG), pour plusieurs fréquences d'interaction. Pour cela nous comparons les résultats obtenus avec i(RBG2-SR + EG) et les résultats de l'iRBG2-SR. Tout d'abord en regardant l'approche i(RBG2-SR + EG) avec une interaction tous les 5 pas de temps, nous constatons une erreur supérieure à celle de toutes les autres approches. Néanmoins en regardant une interaction tous les 15 et 25 pas de temps, nous obtenons une légère diminution de l'erreur par rapport à l'approche non-interactive RBG2-SR + EG. Le meilleur score de l'approche i(RBG2-SR+EG) en terme d'EQM et de R2 semble ici obtenu pour une fréquence d'interaction de 15. Il semble donc qu'ici l'algorithme i(RBG2-SR+EG) s'appuie plus sur son co-apprentissage avec l'algorithme génétique que sur l'interactivité.

En conclusion, l'expérience réalisée semble mettre en évidence l'intérêt de l'interactivité pour l'amélioration des approches RBG2-SR et RBG2-SR + EG. En l'état, ces expériences présentent l'algorithme iRBG2-SR comme étant celui obtenant l'erreur la plus faible et le coefficient de détermination le plus élevé. De plus amples investigations semblent néanmoins nécessaires pour valider ou invalider quantitativement l'approche interactive i(RBG2-SR + EG) et également pour parvenir à calibrer au mieux l'interactivité avec l'algorithme RBG2-SR+EG. Il pourra par exemple être exploré d'autres valeurs de fréquences d'interaction, d'autres fonctions d'apprentissage sur des préférences, ou une meilleure estimation du nombre d'évolutions nécessaire à l'algorithme d'EG par itération de l'algorithme RBG2-SR. Par ailleurs, comme précédemment dans la section 6.6, les expériences ont pour l'instant été réalisées sur une seule fonction f_4 pour faciliter l'évaluation de l'interface par des personnes ayant un niveau de connaissance suffisant pour la résolution du problème. L'application des ces différentes méthodes à d'autres données réelles du réseau électrique ou issues de benchmarks permettraient une validation complémentaire des résultats obtenus dans les sections 6.6 et 6.7.

6.8 Conclusion du chapitre

Après avoir présenté des travaux préliminaires utilisant des algorithmes génétiques interactifs, nous avons décrit une interface interactive permettant d'effectuer une régression symbolique guidée par la grammaire à partir des préférences humaines. La plateforme utilise trois mécanismes de collecte de préférences et propose des cas d'utilisation à complexité incrémentale. Cette approche d'apprentissage basée sur les préférences est particulièrement intéressante pour étudier

symboliquement des ensembles de données de manière exploratoire, lorsque l'utilisateur peut avoir des idées sur les contraintes de la solution à considérer mais ne connaît pas l'expression symbolique exacte au préalable.

Dans un second temps, nous avons fait tester l'interface à un panel restreint d'utilisateurs afin d'identifier des concepts utiles pour la résolution du problème étudié. Ces concepts nous ont ensuite permis d'établir des modèles de préférences simulables. Bien qu'imparfait et peu réalistes individuellement, l'aggrégation de ces modèles nous a permis d'évaluer la qualité de la prise en compte par l'algorithme des préférences simulées, ceci afin notamment de réaliser l'hyperparamétrisation et le calibrage du modèle sans avoir à faire interagir un grand nombre d'utilisateurs. Cette expérience nous a permis également de mettre en évidence la pertinence de l'interaction pour l'apprentissage de fonctions symbolique sur un benchmark. Par ailleurs, dans l'objectif de minimiser le nombre d'interactions avec l'utilisateur, les expériences réalisées nous ont permis d'identifier une plage de fréquence d'interactions acceptable (une interaction toutes les 5 à 20 itérations).

Enfin dans un dernier temps, nous avons présenté une piste permettant de réaliser un apprentissage interactif combinant RBG2-SR avec l'EG de manière à ce que l'algorithme à l'origine de la génération des expressions symboliques soit agnostique pour l'utilisateur.

Les travaux réalisés dans ce chapitre sont une première étape pour l'apprentissage interactif d'expressions symboliques et ouvre sur de nombreuses perspectives. Ainsi, grâce au calibrage réalisé dans la section 6.6, l'interface présenté permettra de tester et de comparer les modalités d'interaction (préférences sur des paires, par catégorie, et utilisation de suggestions) lorsqu'elles sont utilisées séparément et conjointement. Notre objectif est maintenant de réaliser une ou plusieurs expériences sur un panel d'utilisateurs plus large afin d'obtenir des résultats objectifs sur la pertinence de l'approche, ainsi que des résultats subjectifs sur le ressenti des utilisateurs. Ces informations pourraient par exemple être recueillies au moyen de questionnaires tels que le NASA Task Load Index (NASA-TLX) (Hart, 2006), éventuellement modifié pour l'usage dans le cadre d'algorithmes interactifs (Krening et Feigh, 2019). A ces fins, ces tests pourraient être réalisés à la fois sur le benchmark f_4 utilisé pour la calibration de l'algorithme mais également sur le 2e benchmark implémenté avec la plateforme sur des données du réseau électrique IEEE 14 noeuds simulées avec la librairie Grid2Op (Donnot, 2020a).

Par ailleurs, nous avons utilisé ici une première formulation de l'apprentissage sur des préférences où nous apprenons une politique qui respecte au maximum les préférences de l'utilisateur. Cependant d'autres formulations de l'apprentissage sur des préférences existent (Wirth *et al.*, 2017) où les préférences sont par exemples apprises via une fonction d'utilité (qui estime les évaluations de l'utilisateur via une fonction retournant des valeurs numériques), ou un modèle des préférences (qui approxime les préférences de l'expert).

Chapitre 7

Conclusion et perspectives

7.1 Conclusion

Dans cette thèse nous nous intéressons au problème de création d'indicateurs pour l'estimation des transits électriques. Pour répondre à cette problématique, nous avons présenté des approches par régression symbolique pour créer des expressions symboliques pouvant être ensuite utilisés comme indicateurs. Les algorithmes développés proposent d'incorporer de la connaissance experte à la fois explicite et implicite, via l'utilisation d'une grammaire non-contextuelle spécifique, et de l'interactivité pendant l'apprentissage.

Dans cette optique, une première approche a été développée dans le chapitre 4 utilisant un algorithme interactif basé sur de l'Evolution Grammaticale, noté EGI. Cet algorithme utilise les commentaires des utilisateurs, obtenus lorsqu'ils analysent les meilleurs individus génétiques découverts, pour corriger incrémentalement les règles de la grammaire et au fil des itérations améliorer la qualité des expressions symboliques construites. Cette approche a été évaluée sur des données réelles issue de l'historique du réseau électrique sur une période de 2014 à 2017 et a été analysée sur le plan de l'interprétabilité à la fois fonctionnelle et d'un point de vue humain, des solutions quelle fournit. L'utilisation de données réelles a également permis de mettre en avant les difficultés intrinsèques au problème industriel étudié tel que la difficulté de trouver une métrique adaptée au problème et l'effet de la complexité des individus génétiques sur la qualité des solutions.

A partir de ces éléments, nous définissons ensuite un processus de décision markovien partiellement observable pour apprendre par renforcement une politique d'action stochastique dans l'environnement grammatical défini, à partir d'algorithme de renforcement REINFORCE. Cette approche, présentée dans le chapitre 5, est validée sur un ensemble de jeux de données issus de la littérature de la régression symbolique ainsi que sur un jeu de données comportant des contraintes physiques sur les unités des variables. Les résultats mettent en évidence une réduction de l'erreur par rapport aux méthodes évolutionnaires et incrémentales testées utilisant une grammaire. A taille d'expression maximale fixée, les algorithmes par renforcement obtiennent de bons résultats pour parcourir l'espace grammatical d'une tâche de régression symbolique. Du fait de la difficulté de travailler avec des données réelles bruitées, parfois incomplètes, cette approche n'a pour l'instant pas été testée sur des données issues des réseaux électriques mais les résultats obtenus poussent dans cette direction. Enfin, dans le chapitre 6, nous étudions l'interaction directement pendant l'apprentissage, avec les algorithmes présentés dans les chapitres 4 et 5. Dans un premier temps l'interaction avec un algorithme d'EG est étudiée sur des données du réseau électrique. Une interface est proposée et a permis de dégager des cas d'utilisations pertinents pour l'étude

de données du réseau. Dans un deuxième temps, une interface d'interaction est proposée pour l'algorithme d'apprentissage par renforcement. L'objectif de ce travail est d'apprendre à partir de préférences de l'utilisateur sur des paires d'exemples, avec une fonction de coût adaptée. L'interface est testée par un panel restreint d'utilisateurs afin d'identifier des comportements utiles pour la résolutions du problème. Certains de ces comportements sont codés afin de pouvoir les simuler et d'évaluer l'algorithme interactif à moindre coût selon différents paramètres d'interaction. Enfin, une approche plus générale d'interaction est proposée avec plusieurs algorithmes, dans laquelle l'interaction se fait à la fois entre deux algorithmes et avec l'utilisateur, de manière agnostique pour celui-ci. Ce paradigme est testé dans le cadre de l'interaction avec un algorithme d'Evolution Grammaticale et un algorithme d'Apprentissage par Renforcement et les premiers résultats obtenu nous encouragent à développer cette forme d'interaction.

Ensemble, les travaux présentés proposent différents moyens de représenter symboliquement les données du réseau électrique, à partir d'algorithmes automatiques ainsi qu'en incluant un expert dans la boucle d'apprentissage. Nous présentons par la suite plusieurs intéressantes pour de futurs travaux dans ce domaine.

7.2 Perspectives et travaux futurs

Ces travaux ouvrent sur trois catégories de perspectives à la fois sur le plan algorithmique, interactif et applicatif.

7.2.1 Pistes algorithmiques

Dans les chapitres 4 et 5, nous nous appuyons respectivement sur un algorithme d'Evolution Grammaticale (EG), puis d'Apprentissage par Renforcement (AR) avec une stratégie favorable au risque. Cependant, d'autres approches du domaine de la Régression Symbolique (RS) pourraient être bénéfiquement combinées à celles déjà présentées. Dans cette direction, nous envisageons dans le chapitre 6 la combinaisons de l'EG et l'AR dans un cadre interactif. D'autres approches, pour l'instant étudiée dans un cadre non-grammatical, telle que *divide and conquer* (Udrescu et Tegmark, 2020) ou Monte-Carlo Tree Search (Lu *et al.*, 2021), seront incluses dans de prochains tests. Afin de comparer ces différentes méthodes, des benchmarks de la régression symbolique comportant des équations physiques connues pourront être également utilisés (La Cava *et al.*, 2021).

Le chapitre 6, propose par ailleurs de combiner algorithmes génétiques, apprentissage par renforcement et interactivité avec l'utilisateur pour la régression symbolique. Ces travaux pourront être poursuivis par la réalisation de nouvelles expériences interactives. L'algorithme de renforcement interactif, pour l'instant testé sur une fonction simple du domaine de la régression symbolique, pourra être étendu à la réalisation de tests sur les données du réseau électrique. A ces fins, la plateforme interactive décrite dans la section 6.5 inclue d'ores et déjà des données simulées du réseau électrique obtenues grâce à la librairie Grid2Op (Donnot, 2020a). Là où les données issues de l'historique du réseau n'étaient analysables que par un nombre restreint d'experts, ces données simulées permettent d'envisager un plus large panel d'utilisateurs ayant une compréhension du réseau simulé étudié, incluant notamment des utilisateurs avec différents niveaux de connaissances.

Enfin, bien que la grammaire soit considérée dans ce travail comme une entrée du problème, nous envisageons à plus long terme que celle-ci puisse également être *apprise*. Cette approche d'inférence grammaticale, appelée *grammatical induction* a pour objectif de construire une grammaire à partir d'individus (AG) (De La Higuera, 2005) ou de trajectoires (AR) (Lange et Faisal,

2019). L'intérêt est pluriel. Tout d'abord, d'un point de vue de l'apprentissage, nous avons vu dans le chapitre 4 la sensibilité de l'apprentissage à la grammaire choisie. La construction "automatique" d'une grammaire permettrait donc de choisir une grammaire la plus à même de réaliser un apprentissage efficace. Un second intérêt est lié à la *généralisation*. Par exemple, à partir d'une grammaire construite par inférence grammaticale et/ou avec un expert, un "étudiant" (i.e. un expert apprenant) pourrait alors construire une séquence (individus/trajectoires) générée à partir d'un langage optimal, composée d'un nombre fini d'actions de contrôle de bas niveau, et portant les contraintes pertinentes dès le départ.

7.2.2 Pistes d'interactivité

Dans le chapitre 6, nous avons proposé une interface d'interaction utilisant les préférences des utilisateurs sur des paires d'expressions symboliques. L'apprentissage sur des préférences a été évalué sur un ensemble de comportements simulés afin d'évaluer l'interface développée dans la section 6.4. Cependant, d'autres fonctions de coûts pourraient également être envisagées dans le domaine de l'apprentissage par renforcement sur des préférence, tel que l'apprentissage d'une récompense apprise à partir des préférences humaines (Christiano *et al.*, 2017).

Comme mentionné ci-dessus, les opérateurs gagnent de l'expérience au fil du temps. Un opérateur expérimenté et un jeune opérateur, auront donc une compréhension différente du réseau. Cet élément pourrait également être sujet d'analyse à moyen terme grâce aux cas d'utilisation détaillés dans la section 6.5 du chapitre 6. En effet, dans ce chapitre une première expérience comprenant des utilisateurs avec différents niveaux de connaissance est proposée. Celle-ci a permis de mettre en évidence les stratégies communes et spécifique à chaque opérateur sur un jeu de données simple correspondant à une fonction polynomiale. L'expérience pourrait donc être poursuivie, sur un réseau électrique simple, avec des utilisateurs ayant différents niveaux de connaissances du réseau électrique et du problème étudié. Cette expérience permettrait, par exemple, de mettre en évidence les spécificités et points communs sur le cas de l'étude du réseau électrique. La confiance dans les résultats proposés par l'algorithme pourrait également être surveillée au fil des interactions et selon le type de profil (novice/expert) de l'utilisateur. Ces perspectives pourraient permettre, in fine, de développer des interfaces et algorithmes capables de répondre adéquatement selon le niveau d'expertise et de connaissance de son utilisateur.

Par ailleurs, les travaux de cette thèse se sont focalisés sur l'interactivité homme-machine avec un seul expert. Cependant, à plus long terme, plusieurs experts pourraient être impliqués. Dans un cadre applicatif tel que de l'exploitation du réseau électrique, plusieurs experts pourraient avoir des réponses différentes à un même problème. En effet, au fil de leurs années de d'exploitation du réseau, les opérateurs gagnent une expérience et une connaissance appliquée de son fonctionnement, acquises grâce à la résolution d'événements survenus au cours de leur période de travail. Travailler avec différents experts offre donc accès à différentes perspectives, ce qui est bénéfique pour la résolution de problèmes et permet d'approfondir la compréhension, la confiance et la validation des résultats (Boukhelifa *et al.*, 2019).

Enfin, le choix de visualisations adaptées est important pour permettre à un utilisateur de comprendre la situation du réseau électrique de manière rapide et efficace (Overbye *et al.*, 2001). Ainsi, pour développer l'interactivité de façon adéquate, des visualisations adaptées doivent être choisies, à la fois dans le cadre de l'interaction homme/machine mais également pour la restitution des résultats. Par exemple, dans les travaux présentés, l'analyse des expressions symboliques se font sous forme de texte, d'arbre, ou de série temporelle selon le cas. Concernant la restitution des résultats, une fois les expressions symboliques créées, celles-ci pourraient dans un premier temps être intégrées dans les outils existants au sein de RTE. Cependant, d'autres représentations des

indicateurs pourraient être envisagées, telles que des tableaux de bords incorporant du contexte géospatial (Jing *et al.*, 2019), et la construction de nouvelles visualisations pourrait être l'objet de futures recherches.

7.2.3 Pistes d'applications aux réseaux électriques

Sur le plan applicatif, nous avons principalement centré nos évaluations du chapitre 4 sur le cas de l'étude du cas en "N". Dans la suite de ces travaux de thèse, les algorithmes d'AR des chapitres 5 et 6 pourront également être très prochainement évalués sur les mêmes données du réseau électrique. En outre, les algorithmes développés dans cette thèse sur l'étude en "N" pourront aussi être utilisés, par la suite, sur le second cas applicatif en "N-1" présenté dans le chapitre 2.

Nous avons également mis en évidence dans les expériences du chapitre 4 que les expressions symboliques construites sur les données réelles incluent moins fréquemment les variables de topologie que les autres types de variables. Or, au sein de RTE, d'autres travaux de recherche s'intéressent spécifiquement aux variables de topologie et plus spécifiquement au clustering de topologies sur le réseau (Marot *et al.*, 2018b). A moyen terme, l'utilisation des clusters construits grâce à ces algorithmes, pourrait être bénéfiques à la construction de variables. En effet, ces clusters pourraient être considérés comme des variables d'entrées du problème de régression symbolique et inclus dans la constructions de variables sous la forme de conditions sur la topologie d'un cluster donné.

Lors de futures évolutions des outils logiciels, les travaux menés et les algorithmes développés pourront plus largement s'intégrer dans le développement d'assistants pour les opérateurs du réseau présenté en introduction. Dans un premier temps, les indicateurs pourraient être introduits dans l'outil COMPARE (voir figure 1.7), en exécutant à intervalles réguliers les méthodes des chapitres 4 à 6 en cas de perte de qualité des indicateurs existants, en cas de disponibilité de nouvelles données du réseau, ou encore lors d'évolutions majeures du réseau. A plus long terme, ces méthodes pourraient constituer un module de l'assistant, permettant d'hybrider l'IA et la connaissance des opérateurs, comme présenté dans la figure 1.8 du chapitre 1.

Enfin, dans cette thèse, la représentation de la physique et des connaissances liées aux réseaux électriques ont été représentées dans un format BNF. Les grammaires développées dans le chapitre 4 pourront donc être ré-utilisées pour d'autres applications présentant des règles physiques similaires. Dans le cas de problèmes incluant des dépendances temporelles, les grammaires pourront par ailleurs être étendues avec des opérateurs de logique temporelle (Pigozzi *et al.*, 2021; Lucena-Sánchez *et al.*, 2021).

7.3 Liste des publications

1. **Laure Crochepierre**, Lydia Boudjeloud-Assala, Vincent Barbesant "Interactive reinforcement learning for symbolic regression from multi-format human-preference feedbacks.", In Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence (IJCAI 2022), pages 5900–5903. International Joint Conferences on Artificial Intelligence Organization. Demo Track. (Crochepierre *et al.*, 2022c)
2. Antoine Marot, Alexandre Rozier, Matthieu Dussartre, **Laure Crochepierre**, Benjamin Donnot "Towards an AI assistant for power grid operators", In The First International Conference on Hybrid Human Artificial Intelligence (HHAI 2022), pages 1–19. IOS Press. (Marot *et al.*, 2022b).

3. **Laure Crochepierre**, Lydia Boudjeloud-Assala, Vincent Barbesant "Interactive feature extraction using implicit knowledge elicitation : Application to power system expertise." In Hawaii International Conference on System Sciences (HICSS 2022), pages 1–10. ScholarSpace. (Crochepierre *et al.*, 2022b)
4. **Laure Crochepierre**, Lydia Boudjeloud-Assala, Vincent Barbesant "Approche interactive d'extraction de variables interprétables et explicatives pour la gestion des contraintes du réseau électrique français." Revue des Nouvelles Technologies de l'Information, volume Extraction et Gestion des Connaissances (EGC 2021), volume RNTI-E-37, pages 373–380. (Crochepierre *et al.*, 2021)
5. **Laure Crochepierre**, Lydia Boudjeloud-Assala, Vincent Barbesant "Interpretable dimensionally-consistent feature extraction from electrical network sensors." In Machine Learning and Knowledge Discovery in Databases : Applied Data Science Track - European Conference, ECML PKDD 2020, Proceedings, Part IV, volume 12460 de Lecture Notes in Computer Science, pages 444–460. (Crochepierre *et al.*, 2020)
6. Antoine Marot, Antoine Rosin, **Laure Crochepierre**, Benjamin Donnot, Pierre Pinson, Lydia Boudjeloud-Assala "Interpreting Atypical Conditions in Systems with Deep Conditional Autoencoders : The Case of Electrical Consumption". n Machine Learning and Knowledge Discovery in Databases : European Conference, (ECML PKDD 2019), Proceedings, Part III, volume 11908 de Lecture Notes in Computer Science, pages 638–654. (Marot *et al.*, 2019)
7. **Laure Crochepierre**, Antoine Marot, Vincent Barbesant, Benjamin Donnot, Lydia Boudjeloud - Assala "Utilisation de réseau de neurones siamois en clustering : application aux événements du réseau électrique français" Société Francophone de Classification (SFC 2019) Actes des 26èmes Rencontres, pages 111-116 (Crochepierre *et al.*, 2019)

Publications en cours de préparation ou soumises dans un journal international

1. **Laure Crochepierre**, Lydia Boudjeloud-Assala, Vincent Barbesant "Learning from Human - Preferences in Symbolic Regression." en préparation pour soumission dans un journal international en anglais.
2. **Laure Crochepierre**, Lydia Boudjeloud-Assala, Vincent Barbesant "A Reinforcement Learning Approach to Domain-Knowledge Inclusion Using Grammar Guided Symbolic Regression." (Crochepierre *et al.*, 2022a), soumis pour publication en anglais dans le journal international Machine Learning.

Annexe A

Grammaires initiales et finale

Décrivons les deux grammaires initiales et finales utilisées dans le chapitre 4.

La grammaire de la figure A.1 correspond à une grammaire qui n'as pas encore subit plusieurs itérations d'apprentissages évolutionnaires et de révisions. Celle-ci comprend les unités p, q, v, et i. Ces 4 unités sont autorisées comme unité de l'expression en sortie de l'algorithme. Cet élément est signifié par le symbole grammatical $< expr >$.

Le lien entre les variable du jeu de données et leur unité est réalisé par les symboles $< pvar >$, $< qvar >$, $< vvar >$ et $< ivar >$. Chaque unité peut être additionnée avec, soustraite à une variable de même unité, et il est également possible d'y appliquer les fonctions valeur absolue `abs`, partie négative `neg`, ou partie positive `pos`. Outre les opérations d'additions, il est également possible de prendre la valeur minimale `minimum` ou maximale `maximum` d'une liste de variables, tel que décrit par le symbole $< list\_op >$.

Notons enfin que dans cette grammaire, les variables topologiques sont également décrites. La variable de sortie combinant des variables topologiques (booléen sans unité) et des variables numériques (avec unité) sont combinées dans le symbole $< expr\_and\_topo >$. Les symboles de la grammaire correspondant aux variables topologiques sont $< topo\_var >$ et $< b\_var >$. Ces variables peuvent être combinées par des opérateurs logiques tel que *et* `logical_and` et *ou* `logical_or` dans les règles du symbole $< b >$ pour former des règles de type *if_ then_ else*. Cependant, la construction d'expressions utilisant ces variables topologiques n'est pas systématiquement réalisé et plutôt que d'obtenir une expression définie par le symbole de départ $< expr\_and\_topo >$, l'expression peut ne comporter que des variables numériques correspondant au symbole $< expr >$.

En regardant maintenant la grammaire décrite dans la figure A.2, nous pouvons remarquer que de nouvelles variables et de nouvelles opérations ont été introduites. Cette nouvelle grammaire comprend les unités p, q, v, i, ainsi que p^2, q^2, v^2, s. Parmi ces unités, seules p, p/v, s et i sont autorisées comme unité de l'expression en sortie de l'algorithme. Cet élément est signifié par le symbole grammatical $< expr >$.

Des opérations de multiplications par des grandeurs sans unités est introduit dans toutes les règles des symboles correspondant aux variables avec des unités physiques. Ces grandeurs sans unités peuvent être définies par des ratios entre variables de mêmes unités et des multiplications d'expressions sans dimensions.

Des relations physiques ont également été introduites pour passer d'une dimension à l'autre, tel que la règle $< s > \leftarrow sqrt\_add(< p2 >, < q2 >$ (définition du triangle des puissances), ou encore $< i > \leftarrow < s > / < v >$ (intensité).

```
<expr_and_topo> ::= <expr> | if_then_else( <b> , <expr_and_topo> , <expr> )
<expr>          ::= <p> | <q> | <v> | <i>

<p>             ::=  ( - <p> ) | ( <p> - <p> ) | <list_op> ([ <p> , <p_list> ]) | <sop>( <p> )
                | x[ <pvar> ]
<p_list>        ::= <p> | <p> , <p_list>
<q>             ::=  ( - <q> ) | ( <q> - <q> ) | <list_op> ([ <q> , <q_list> ]) | <sop>( <q> )
                | x[ <qvar> ]
<q_list>        ::= <q> | <q> , <q_list>
<v>             ::=  ( - <v> ) | ( <v> - <v> ) | <list_op> ([ <v> , <v_list> ]) | <sop>( <v> )
                | x[ <vvar> ]
<v_list>        ::= <v> | <v> , <v_list>

<i>             ::=  ( - <i> ) | <list_op> ([ <i> , <i_list> ]) | <sop> ( <i> ) | x[ <ivar> ]
<i_list>        ::= <i> | <i> , <i_list>

<sop>           ::=abs|neg|pos
<list_op>       ::=sum|minimum|maximum
<integer>       ::= GE_RANGE:10 | probs [1 for x in range(100)]

<b>             ::= logical_and( <b> , <b> ) | logical_or( <b> , <b> ) | x[ <bvar> ]
                | (<topo> == <integer> ) | (<topo> in [ <integer> , <integer_list> ] )
                | logical_not( <b> )
<topo>          ::= x[ <topo_var> ]
<integer_list>  ::= <integer> , <integer_list> | <integer>

<pvar>          ::="p_bus_or_"+<line> |"p_bus_ex_"+<line> |"p_line_or_"+<frontier_line>
                |"p_line_ex_"+<frontier_line>
<qvar>          ::="q_bus_or_"+<line> |"q_bus_ex_"+<line> |"q_line_or_"+<frontier_line>
                |"q_line_ex_"+<frontier_line>
<vvar>          ::="v_or_"+<line> |"v_ex_"+<line>
<topo_var>      ::="topo_id_"+<sub>
<bvar>          ::="connected_or_"+<line> |"connected_ex_"+<line>
<ivar>          ::="i_or_"+<frontier_line>

<frontier_line> ::= frontiere_lines[ <id_fl> %len(frontiere_lines)]
<line>          ::= lines_list[ <id_l> %len(lines_list)]
<sub>           ::= subs_list[ <id_s> %len(subs_list)]

<id_fl>         ::= GE_RANGE:nb_fl
<id_l>          ::= GE_RANGE:nb_l
<id_s>          ::= GE_RANGE:nb_s
```

FIGURE A.1 – Grammaire initiale avant entraînement.

```
<expr_and_topo> ::= <expr> | if_then_else(<b>,<expr_and_topo>,<expr>)
<expr>          ::= <p> | pdiv(<p>,<v>) | <s> | <i>

<p>             ::= ( - <p>) | <list_op>([<p>,<p_list>]) | <sop>(<p>) | multiply(<p>,<f>)
                | multiply(<f>,<p>) | x[<pvar>]
<p_list>        ::= <p> | <p>,<p_list>
<q>             ::= ( - <q>) | <list_op>([<q>,<q_list>]) | <sop>(<q>) | multiply(<q>,<f>)
                | multiply(<f>,<q>) | x[<qvar>]
<q_list>        ::= <q> | <q>,<q_list>
<v>             ::= ( - <v>) | <list_op>([<v>,<v_list>]) | <sop>(<v>) | multiply(<v>,<f>)
                | multiply(<f>,<v>) | x[<vvar>]
<v_list>        ::= <v> | <v>,<v_list>

<p2>            ::= multiply(<p>,<p>) | square(<p>) | multiply(<p2>,<f>) | multiply(<f>,<p2>)
<q2>            ::= multiply(<q>,<q>) | square(<q>) | multiply(<q2>,<f>) | multiply(<f>,<q2>)
<v2>            ::= multiply(<v>,<v>) | square(<v>) | multiply(<v2>,<f>) | multiply(<f>,<v2>)

<s>             ::= ( - <p>) | <list_op>([<s>,<s_list>]) | sqrt_add(<p2>,<q2>) | multiply(<s>,<f>)
<s_list>        ::= <s> | <s>,<s_list>
<i>             ::= ( - <i>) | pdiv(<s>,<v>) | <list_op>([<i>,<i_list>]) | <sop>(<i>)
                | multiply(<i>,<f>) | multiply(<f>,<i>) | x[<ivar>]
<i_list>        ::= <i> | <i>,<i_list>

<f>             ::= multiply(<f>,<f>) | pdiv(<p>,<p>) | pdiv(<q>,<q>) | pdiv(<v>,<v>)
<integer>       ::= GE_RANGE:100

<b>             ::= logical_and(<b>,<b>) | logical_or(<b>,<b>) | x[<bvar>] | (<topo> == <integer>)

| (<topo> in [<integer>,<integer_list>]) | logical_not(<b>)
<topo>          ::= x[<topo_var>] | probs[1]
<integer_list>  ::= <integer>,<integer_list> | <integer>

<sop>           ::= abs | neg | pos
<list_op>       ::= sum | minimum | maximum
<pvar>          ::= "p_bus_or_"+<line> | "p_bus_ex_"+<line> | "p_line_or_"+<frontier_line>
                | "p_line_ex_"+<frontier_line>
<qvar>          ::= "q_bus_or_"+<line> | "q_bus_ex_"+<line> | "q_line_or_"+<frontier_line>
                | "q_line_ex_"+<frontier_line>
<vvar>          ::= "v_or_"+<line> | "v_ex_"+<line>
<topo_var>      ::= "topo_id_"+<sub> | probs[1]
<bvar>          ::= "connected_or_"+<line> | "connected_ex_"+<line>
<ivar>          ::= "i_or_"+<frontier_line>

<frontier_line> ::= frontiere_lines[<id_fl>%len(frontiere_lines)]
<line>          ::= lines_list[<id_l>%len(lines_list)]
<sub>           ::= subs_list[<id_s>%len(subs_list)]

<id_fl>         ::= GE_RANGE:nb_fl
<id_l>          ::= GE_RANGE:nb_l
<id_s>          ::= GE_RANGE:nb_s
```

FIGURE A.2 – Grammaire finale complète. Des spécifications sur les unités autorisées en sortie sont été rajoutées. De nouvelles unités sont maintenant présentes telles que `<p2_var>`, `<q2_var>`, `<s_var>`.

Bibliographie

Abdellaoui, I. A. et Mehrkanoon, S. (2021). Symbolic regression for scientific discovery : an application to wind speed forecasting. *In IEEE Symposium Series on Computational Intelligence (SSCI)*, pages 01–08. IEEE.

Altman, N. S. (1992). An introduction to kernel and nearest-neighbor nonparametric regression. *The American Statistician*, 46(3):175–185.

Amershi, S., Cakmak, M., Knox, W. B. et Kulesza, T. (2014). Power to the People : The Role of Humans in Interactive Machine Learning. *Ai Magazine*, 35(4):105–120.

Anjum, A., Sun, F., Wang, L. et Orchard, J. (2019). A Novel Neural Network-Based Symbolic Regression Method : Neuro-Encoded Expression Programming. *In Artificial Neural Networks and Machine Learning – ICANN 2019 : Deep Learning*, pages 373–386. Springer International Publishing.

Araki, B., Choi, J., Chin, L., Li, X. et Rus, D. (2022). Learning Policies by Learning Rules. *IEEE Robotics and Automation Letters*, 7(2):1284–1291.

Assemblée Nationale (2020). Proposition de résolution visant à inscrire le caractère de « bien commun » de l'énergie et l'organisation nationale de sa production. `https://www.assemblee-nationale.fr/dyn/15/textes/l15b3545_proposition-resolution`. - Dernier accès 27 septembre 2022.

Backus, J. W., Bauer, F. L., Green, J., Katz, C., McCarthy, J., Perlis, A. J., Rutishauser, H., Samelson, K., Vauquois, B., Wegstein, J. H., van Wijngaarden, A., Woodger, M. et Naur, P. (1963). Revised Report on the Algorithm Language ALGOL 60. *Commununication of the ACM*, 6(1):1–17.

Baker, J. E. (1985). Adaptive Selection Methods for Genetic Algorithms. *In Proceedings of the 1st International Conference on Genetic Algorithms*, page 101–111. L. Erlbaum Associates Inc.

Banzhaf, W., Nordin, P., Keller, R. E. et Francone, F. D. (1998). *Genetic programming : an introduction : on the automatic evolution of computer programs and its applications.* Morgan Kaufmann Publishers Inc.

Barjot, D. (2013). Reconstruire la france après la seconde guerre mondiale : les débuts d'électricité de france (1946-1953). *Entreprises et histoire*, 70(1):54–75.

Beltran, A. (2004). Quelle approche «culturelle» de l'histoire de l'électricité ? *Annales historiques de lelectricite*, 2(1):139–145.

Bengio, Y., Louradour, J., Collobert, R. et Weston, J. (2009). Curriculum Learning. *In Proceedings of the 26th Annual International Conference on Machine Learning*, ICML '09, page 41–48. Association for Computing Machinery.

Bergman, M. K. (2018). *A Knowledge Representation Practionary - Guidelines Based on Charles Sanders Peirce*. Springer.

Berti-Équille, L. (2020). Active Reinforcement Learning for Data Preparation : Learn2Clean with Human-In-The-Loop. *In 10th Conference on Innovative Data Systems Research, CIDR*. www.cidrdb.org.

Biggio, L., Bendinelli, T., Lucchi, A. et Parascandolo, G. (2020). A seq2seq approach to symbolic regression. *In Proceedings of the International Workshop on Learning Meets Combinatorial Algorithms (LMCA) co-located with Advances in Neural Information Processing Systems (Neurips)*.

Biggio, L., Bendinelli, T., Neitz, A., Lucchi, A. et Parascandolo, G. (2021). Neural Symbolic Regression that scales. *In Proceedings of the 38th International Conference on Machine Learning*, volume 139 de *Proceedings of Machine Learning Research*, pages 936–945. PMLR.

Bojarczuk, C. C., Lopes, H. S., Freitas, A. A. et Michalkiewicz, E. L. (2004). A constrained-syntax genetic programming system for discovering classification rules : application to medical data sets. *Artificial Intelligence in Medicine*, 30(1):27–48.

Boudjeloud-Assala, L. (2005). *Visualisation et algorithmes génétiques pour la fouille de grands ensembles de données*. Thèse de doctorat, École polytechnique universitaire de Nantes Université.

Boukhelifa, N., Bezerianos, A. et Lutton, E. (2018). Evaluation of Interactive Machine Learning Systems. *In Human and Machine Learning : Visible, Explainable, Trustworthy and Transparent*, Human-Computer Interaction Series, pages 341–360. Springer International Publishing.

Boukhelifa, N., Bezerianos, A., Trelea, I. C., Perrot, N. M. et Lutton, E. (2019). An Exploratory Study on Visual Exploration of Model Simulations by Multiple Types of Experts. *In Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems, CHI*, CHI '19, page 1–14. Association for Computing Machinery.

Bouneau, C. (2004). Les réseaux de transport d'électricité en europe occidentale depuis la fin du xixe siècle : de la diversité des modèles nationaux à la recherche de la convergence européenne. *Annales historiques de l'electricite*, 2(1):23–37.

Brabazon, A. et O'Neill, M. (2006). *Biologically inspired algorithms for financial modelling*. Springer Science & Business Media.

Breiman, L., Friedman, J. H., Olshen, R. A. et Stone, C. J. (1984). *Classification and Regression Trees*. Chapman and Hall/CRC.

Brence, J., Todorovski, L. et Džeroski, S. (2021). Probabilistic grammars for equation discovery. *Knowledge-Based Systems*, 224:107077.

Brooks, T. F., Pope, D. S. et Marcolini, M. A. (1989). Airfoil self-noise and prediction. Rapport technique, National Aeronautics and Space Administration, Office of Management, Scientific and Technical Information Division.

Burlacu, B., Kronberger, G., Kommenda, M. et Affenzeller, M. (2019). Parsimony measures in multi-objective genetic programming for symbolic regression. *In Proceedings of the Genetic and Evolutionary Computation Conference Companion*, pages 338–339. Association for Computing Machinery.

Carvalho, D. V., Pereira, E. M. et Cardoso, J. S. (2019). Machine Learning Interpretability : A Survey on Methods and Metrics. *Electronics*, 8(8):832.

Castle, T. et Johnson, C. G. (2010). Positional Effect of Crossover and Mutation in Grammatical Evolution. *In Genetic Programming, 13th European Conference, EuroGP 2010. Proceedings*, volume 6021 de *Lecture Notes in Computer Science*, pages 26–37. Springer.

Chabin, T., Barnabé, M., Boukhelifa, N., Fonseca, F., Tonda, A. P., Velly, H., Lemaitre, B., Perrot, N. et Lutton, E. (2018). LIDeOGraM : An Interactive Evolutionary Modelling Tool. *In Artificial Evolution - 13th International Conference, Évolution Artificielle, EA 2017*, volume 10764 de *Lecture Notes in Computer Science*, pages 189–201. Springer.

Chen, G. (2020). Learning Symbolic Expressions via Gumbel-Max Equation Learner Networks. *arXiv preprint arXiv :2012.06921*.

Cherrier, N., Poli, J., Defurne, M. et Sabatié, F. (2019). Consistent Feature Construction with Constrained Genetic Programming for Experimental Physics. *In IEEE Congress on Evolutionary Computation (CEC)*, pages 1650–1658. IEEE.

Chomsky, N. (1959). On certain formal properties of grammars. *Information and control*, 2(2):137–167.

Christiano, P. F., Leike, J., Brown, T. B., Martic, M., Legg, S. et Amodei, D. (2017). Deep Reinforcement Learning from Human Preferences. *In Advances in Neural Information Processing Systems*, volume 30, pages 4299–4307. Curran Associates, Inc.

Chung, W., Thomas, V., Machado, M. C. et Roux, N. L. (2021). Beyond Variance Reduction : Understanding the True Impact of Baselines on Policy Optimization. *In Proceedings of the 38th International Conference on Machine Learning, (ICML 2021)*, volume 139 de *Proceedings of Machine Learning Research*, pages 1999–2009. PMLR.

Commission de Régulation de l'Energie (CRE) (2018). Les interconnexions électriques et gazières en france - juillet 2018. `https://www.cre.fr/Documents/Publications/Rapports-thematiques/Rapport-interconnexions-2018`. - Dernier accès 27 septembre 2022.

Contreras, I., Bertachi, A., Biagi, L., Vehí, J. et Oviedo, S. (2018). Using Grammatical Evolution to Generate Short-term Blood Glucose Prediction Models. *In Proceedings of the 3rd Workshop on Knowledge Discovery in Healthcare Data co-located with the 27th International Joint Conference on Artificial Intelligence and the 23rd European Conference on Artificial Intelligence (IJCAI-ECAI 2018)*, volume 2148 de *CEUR Workshop Proceedings*, pages 91–96. CEUR-WS.org.

Crochepierre, L., Boudjeloud-Assala, L. et Barbesant, V. (2020). Interpretable Dimensionally-Consistent Feature Extraction from Electrical Network Sensors. *In Machine Learning and Knowledge Discovery in Databases : Applied Data Science Track - European Conference, ECML PKDD 2020, Proceedings, Part IV*, volume 12460 de *Lecture Notes in Computer Science*, pages 444–460. Springer.

Crochepierre, L., Boudjeloud-Assala, L. et Barbesant, V. (2021). Approche interactive d'extraction de variables interprétables et explicatives pour la gestion des contraintes du réseau électrique français. *Revue des Nouvelles Technologies de l'Information*, Extraction et Gestion des Connaissances, RNTI-E-37:373–380.

Crochepierre, L., Boudjeloud-Assala, L. et Barbesant, V. (2022a). A Reinforcement Learning Approach to Domain-Knowledge Inclusion Using Grammar Guided Symbolic Regression. *arXiv preprint arXiv :2202.04367*.

Crochepierre, L., Boudjeloud-Assala, L. et Barbesant, V. (2022b). Interactive Feature Extraction using Implicit Knowledge Elicitation : Application to Power System Expertise. *In Hawaii International Conference on System Sciences (HICSS 2022)*, pages 1–10. ScholarSpace.

Crochepierre, L., Boudjeloud-Assala, L. et Barbesant, V. (2022c). Interactive reinforcement learning for symbolic regression from multi-format human-preference feedbacks. *In Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence (IJCAI 2022)*, pages 5900–5903. International Joint Conferences on Artificial Intelligence Organization. Demo Track.

Crochepierre, L., Marot, A., Barbesant, V., Donnot, B. et Boudjeloud-Assala, L. (2019). Utilisation de réseau de neurones siamois en clustering : application aux événements du réseau électrique français. *Société Francophone de Classification (SFC) Actes des 26èmes Rencontres*, pages 111–116.

Cruz, C. A. et Igarashi, T. (2020). A Survey on Interactive Reinforcement Learning : Design Principles and Open Challenges. *In Designing Interactive Systems Conference (DIS '20)*, pages 1195–1209. Association for Computing Machinery.

Cunha, J. M. T. d. (2021). *Probabilistic Grammatical Evolution*. Thèse de doctorat, Universidade de Coimbra.

Dabek, F. et Caban, J. J. (2017). A Grammar-based Approach for Modeling User Interactions and Generating Suggestions During the Data Exploration Process. *IEEE Transactions on Visualization and Computer Graphics*, 23(1):41–50.

Dalkir, K. (2013). *Knowledge management in theory and practice*. Routledge.

Darwin, C. (1859). *On the Origin of Species by Means of Natural Selection, or the Preservation of Favoured Races in the Struggle for Life*, volume 2. Books, Incorporated, Pub.

d'Ascoli, S., Kamienny, P.-A., Lample, G. et Charton, F. (2022). Deep Symbolic Regression for Recurrent Sequences. *arXiv preprint arXiv :2201.04600*.

De La Higuera, C. (2005). A bibliographical study of grammatical inference. *Pattern recognition*, 38(9):1332–1348.

Dellermann, D., Calma, A., Lipusch, N., Weber, T., Weigel, S. et Ebel, P. (2019a). The Future of Human-AI Collaboration : A Taxonomy of Design Knowledge for Hybrid Intelligence Systems. *In Proceedings of the 52nd Hawaii International Conference on System Sciences (HICSS - 2019)*. ScholarSpace.

Dellermann, D., Ebel, P., Söllner, M. et Leimeister, J. M. (2019b). Hybrid Intelligence. *Business & Information Systems Engineering*, 61(5):637–643.

Doan, A. (2018). Human-in-the-Loop Data Analysis : A Personal Perspective. *In Proceedings of the Workshop on Human-In-the-Loop Data Analytics (HILDA)*. Association for Computing Machinery.

Donnot, B. (2019). *Deep learning methods for predicting flows in power grids : novel architectures and algorithms*. Thèse de doctorat, Université Paris Saclay (COmUE).

Donnot, B. (2020a). Grid2op- A testbed platform to model sequential decision making in power systems. . `https://GitHub.com/rte-france/grid2op`.

Donnot, B. (2020b). Lightsim2grid - A c++ backend targeting the Grid2Op platform. . `https://GitHub.com/bdonnot/lightsim2grid`.

Donnot, B., Guyon, I., Marot, A., Schoenauer, M. et Panciatici, P. (2018). Optimization of computational budget for power system risk assessment. *In 2018 IEEE PES Innovative Smart Grid Technologies Conference Europe (ISGT-Europe)*, pages 1–6. IEEE.

Donnot, B., Guyon, I., Schoenauer, M., Panciatici, P. et Marot, A. (2017). Introducing machine learning for power system operation support. *In Proceedings of the International Institute for Research and Education in Power Systems Symposium on Bulk Power System Dynamics and Control X (IREP 2017)*.

Donon, B., Clément, R., Donnot, B., Marot, A., Guyon, I. et Schoenauer, M. (2020). Neural networks for power flow : Graph neural solver. *Electric Power Systems Research*, 189:106547.

Donon, B., Donnot, B., Guyon, I. et Marot, A. (2019). Graph Neural Solver for Power Systems. *In International Joint Conference on Neural Networks, (IJCNN)*, pages 1–8. IEEE.

Doshi-Velez, F. et Kim, B. (2017). Towards a rigorous science of interpretable machine learning. *arXiv preprint arXiv :1702.08608*.

Downing, K. L. (2001). Adaptive genetic programs via reinforcement learning. *In Proceedings of the 3rd Annual Conference on Genetic and Evolutionary Computation*, pages 19–26.

Drori, I., Krishnamurthy, Y., Lourenço, R., Rampin, R., Cho, K., Silva, C. et Freire, J. (2019). Automatic Machine Learning by Pipeline Synthesis using Model-Based Reinforcement Learning and a Grammar. *arXiv preprint arXiv :1905.10345*.

Du, M., Liu, N. et Hu, X. (2019). Techniques for interpretable machine learning. *Communications of the ACM*, 63(1):68–77.

Dua, D. et Graff, C. (2017). UCI Machine Learning Repository. `http://archive.ics.uci.edu/ml`.

Duchesne, L., Karangelos, E. et Wehenkel, L. (2020). Recent developments in machine learning for energy systems reliability management. *Proceedings of the IEEE*, 108(9):1656–1676.

Duchi, J. C., Mackey, L. W. et Jordan, M. I. (2010). On the Consistency of Ranking Algorithms. *In Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, pages 327–334.

Dudley, J. J. et Kristensson, P. O. (2018). A Review of User Interface Design for Interactive Machine Learning. *ACM Transactions on Interactive Intelligent Systems (TiiS)*, 8(2):1–37.

Dufek, A. S., Augusto, D. A., Dias, P. L. et Barbosa, H. J. (2013). Evaluating the feasibility of grammar-based gp in combining meteorological forecast models. *In 2013 IEEE Congress on Evolutionary Computation*, pages 32–39.

Dyer, C., Kuncoro, A., Ballesteros, M. et Smith, N. A. (2016). Recurrent Neural Network Grammars. *In NAACL HLT 2016, Proceedings of The 2016 Conference of the North American Chapter of the Association for Computational Linguistics : Human Language Technologies*, pages 199–209. The Association for Computational Linguistics.

EDF (2022). La consommation d'électricité en chiffres. `https://www.edf.fr/groupe-edf/espaces-dedies/l-energie-de-a-a-z/tout-sur-l-energie/l-electricite-au-quotidien/la-consommation-d-electricite-en-chiffres`. - Dernier accès 27 septembre 2022.

Emani, C. K., Silva, C. F. D., Fiés, B. et Ghodous, P. (2019). NALDO : From natural language definitions to OWL expressions. *Data & Knowledge Engineering*, 122:130–141.

Evans, B. P., Xue, B. et Zhang, M. (2019). What's inside the black-box ? : a genetic programming method for interpreting complex machine learning models. *In Proceedings of the Genetic and Evolutionary Computation Conference*, pages 1012–1020.

Evans, J. D. (1996). *Straightforward statistics for the behavioral sciences.* Thomson Brooks/Cole Publishing Co.

Fagan, D., Fenton, M. et O'Neill, M. (2016). Exploring position independent initialisation in grammatical evolution. *In 2016 IEEE Congress on Evolutionary Computation (CEC)*, pages 5060–5067. IEEE.

Fenton, M., McDermott, J., Fagan, D., Forstenlechner, S., Hemberg, E. et O'Neill, M. (2017). PonyGE2 : Grammatical evolution in Python. *In Proceedings of the Genetic and Evolutionary Computation Conference Companion*, pages 1194–1201. Association for Computing Machinery.

Feuerriegel, S., Dolata, M. et Schwabe, G. (2020). Fair AI. *Business & information systems engineering*, 62(4):379–384.

Feurer, M., Klein, A., Eggensperger, K., Springenberg, J. T., Blum, M. et Hutter, F. (2015). Efficient and Robust Automated Machine Learning. *In Advances in Neural Information Processing Systems*, volume 28, pages 2962–2970. Curran Associates, Inc.

for Electrical System Security within Large Areas (ITESLA), I. T. (2012). Project final report. `https://bit.ly/2BLpWME`. - Dernier accès 27 septembre 2022.

Fortin, F.-A., De Rainville, F.-M., Gardner, M.-A., Parizeau, M. et Gagné, C. (2012). DEAP : Evolutionary Algorithms Made Easy. *Journal of Machine Learning Research*, 13: 2171–2175.

Françoise, J., Caramiaux, B. et Sanchez, T. (2021). Marcelle : Composing Interactive Machine Learning Workflows and Interfaces. *In UIST '21 : The 34th Annual ACM Symposium on User Interface Software and Technology*, pages 39–53. Association for Computing Machinery.

Friedman, J., Hastie, T. et Tibshirani, R. (2010). Regularization paths for generalized linear models via coordinate descent. *Journal of statistical software*, 33(1):1–22.

Friedman, J. H. (1991). Multivariate Adaptive Regression Splines. *The annals of statistics*, 19(1):1–67.

Friedman, J. H. et Popescu, B. E. (2008). Predictive learning via rule ensembles. *The annals of applied statistics*, 2(3):916–954.

Friedman, J. H. et Stuetzle, W. (1981). Projection Pursuit Regression. *Journal of the American statistical Association*, 76(376):817–823.

Fürnkranz, J. et Hüllermeier, E. (2010). Preference learning and ranking by pairwise comparison. *In Preference learning*, pages 65–82. Springer.

Gehring, J., Auli, M., Grangier, D., Yarats, D. et Dauphin, Y. N. (2017). Convolutional sequence to sequence learning. *In International Conference on Machine Learning*, pages 1243–1252. PMLR.

Glatt, R., da Silva, F. L., VAN HAI, B., Huang, C., Xue, L., Wang, M., Chang, F., Murphey, Y. et Su, W. (2021). Deep Symbolic Optimization for Electric Component Sizing in Fixed Topology Power Converters. *In Workshop on AI for Design and Manufacturing (ADAM) as part of the AAAI Conference on Artificial Intelligence.* AAAI Press.

Good, P. I. et Hardin, J. W. (2012). *Common errors in statistics (and how to avoid them).* John Wiley & Sons, 4ème édition.

Goodfellow, I., Bengio, Y. et Courville, A. (2016). *Deep learning.* MIT press.

Graves, A., Mohamed, A.-r. et Hinton, G. (2013). Speech recognition with deep recurrent neural networks. *In 2013 IEEE international conference on acoustics, speech and signal processing*, pages 6645–6649. IEEE.

Gutzwiller, R. S. et Reeder, J. (2017). Human interactive machine learning for trust in teams of autonomous robots. *In 2017 IEEE Conference on Cognitive and Computational Aspects of Situation Management (CogSIMA)*, pages 1–3. IEEE.

Haes Alhelou, H., Hamedani-Golshan, M. E., Njenda, T. C. et Siano, P. (2019). A Survey on Power System Blackout and Cascading Events : Research Motivations and Challenges. *Energies*, 12(4):682.

Hahn, W. (1931). Load studies on the DC calculating table. *General Electric Review*, 34:444.

Hancock, P. J. (1994). An empirical comparison of selection methods in evolutionary algorithms. *In AISB workshop on evolutionary computing*, pages 80–94. Springer.

Harper, R. (2010). GE, explosive grammars and the lasting legacy of bad initialisation. *In IEEE Congress on Evolutionary Computation*, pages 1–8.

Hart, S. G. (2006). Nasa-Task Load Index (NASA-TLX) ; 20 Years Later. *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, 50(9):904–908.

Hastie, T. et Tibshirani, R. (1986). Generalized Additive Models. *Statistical Science*, 1(3):297–310.

Hastie, T., Tibshirani, R., Friedman, J. H. et Friedman, J. H. (2009). *The elements of statistical learning : data mining, inference, and prediction*, volume 2 de *Springer series in statistics*. Springer, 2nd édition.

Hein, D., Udluft, S. et Runkler, T. A. (2018). Interpretable policies for reinforcement learning by genetic programming. *Engineering Applications of Artificial Intelligence*, 76:158–169.

Hepp, M., De Leenheer, P., De Moor, A. et Sure, Y. (2007). *Ontology management : semantic web, semantic web services, and business applications.* Springer Science & Business Media.

Hochreiter, S. et Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, 9(8):1735–1780.

Hoerl, A. E. et Kennard, R. W. (1970). Ridge Regression : Biased Estimation for Nonorthogonal Problems. *Technometrics*, 12(1):55–67.

Holzinger, A., Plass, M., Kickmeier-Rust, M. D., Holzinger, K., Crisan, G. C., Pintea, C. et Palade, V. (2019). Interactive machine learning : experimental evidence for the human in the algorithmic loop - A case study on Ant Colony Optimization. *Applied Intelligence*, 49(7):2401–2414.

Honeycutt, D. R., Nourani, M. et Ragan, E. D. (2020). Soliciting Human-in-the-Loop User Feedback for Interactive Machine Learning Reduces User Trust and Impressions of Model Accuracy. *In Proceedings of the AAAI Conference on Human Computation and Crowdsourcing (HCOMP)*, volume 8, pages 63–72. AAAI Press.

Hornik, K., Stinchcombe, M. et White, H. (1989). Multilayer feedforward networks are universal approximators. *Neural networks*, 2(5):359–366.

Hossain, S., Calloway, C., Lippa, D., Niederhut, D. et Shupe, D. (2019). Visualization of Bioinformatics Data with Dash Bio. *In Proceedings of the 18th Python in Science Conference (SciPy)*, pages 126–133.

Huang, J., Angelov, P. P. et Yin, C. (2020). Interpretable policies for reinforcement learning by empirical fuzzy sets. *Engineering Applications of Artificial Intelligence*, 91:103559.

Huang, J. et Cakmak, M. (2015). Supporting mental model accuracy in trigger-action programming. *In Proceedings of the 2015 ACM International Joint Conference on Pervasive and Ubiquitous Computing, (UbiComp 2015)*, pages 215–225. Association for Computing Machinery.

Huang, S. et Ontañón, S. (2020). A closer look at invalid action masking in policy gradient algorithms. *arXiv preprint arXiv :2006.14171*.

Javed, N., Gobet, F. et Lane, P. (2022). Simplification of genetic programs : a literature survey. *Data Mining and Knowledge Discovery*, pages 1–22.

Jin, M., Ma, Z., Jin, K., Zhuo, H. H., Chen, C. et Yu, C. (2022). Creativity of AI : Automatic Symbolic Option Discovery for Facilitating Deep Reinforcement Learning. *In Thirty-Sixth AAAI Conference on Artificial Intelligence, AAAI 2022, Thirty-Fourth Conference on Innovative Applications of Artificial Intelligence, IAAI 2022, The Twelveth Symposium on Educational Advances in Artificial Intelligence, EAAI 2022 Virtual Event, February 22 - March 1, 2022*, pages 7042–7050. AAAI Press.

Jing, C., Du, M., Li, S. et Liu, S. (2019). Geospatial Dashboards for Monitoring Smart City Performance. *Sustainability*, 11(20):5648.

Kaelbling, L. P., Littman, M. L. et Cassandra, A. R. (1998). Planning and acting in partially observable stochastic domains. *Artificial Intelligence*, 101(1):99–134.

Kaliakatsos-Papakostas, M. A., Epitropakis, M. G., Floros, A. et Vrahatis, M. N. (2012). Interactive Evolution of 8-Bit Melodies with Genetic Programming towards Finding Aesthetic Measures for Sound. *In Evolutionary and Biologically Inspired Music, Sound, Art and Design - First International Conference, EvoMUSART 2012. Proceedings*, volume 7247 de *Lecture Notes in Computer Science*, pages 141–152. Springer.

Kamalian, Ying Zhang, Takagi et Agogino (2005). Reduced human fatigue interactive evolutionary computation for micromachine design. *In 2005 International Conference on Machine Learning and Cybernetics*, volume 9, pages 5666–5671.

Katz, M., Ram, P., Sohrabi, S. et Udrea, O. (2020). Exploring Context-Free Languages via Planning : The Case for Automating Machine Learning. *In Proceedings of the International Conference on Automated Planning and Scheduling (ICAPS)*, volume 30, pages 403–411.

Keijzer, M. (2003). Improving symbolic regression with interval arithmetic and linear scaling. *In European Conference on Genetic Programming*, pages 70–82. Springer.

Keijzer, M. et Babovic, V. (1999). Dimensionally Aware Genetic Programming. *In Proceedings of the 1st Annual Conference on Genetic and Evolutionary Computation*, volume 2, pages 1069–1076.

Kelly, A., O'Sullivan, A., de Mars, P. et Marot, A. (2020). Reinforcement Learning for Electricity Network Operation. *arXiv preprint arXiv :2003.07339*.

Kerrigan, D., Hullman, J. et Bertini, E. (2021). A Survey of Domain Knowledge Elicitation in Applied Machine Learning. *Multimodal Technologies and Interaction*, 5(12):73.

Kezunovic, M., Pinson, P., Obradovic, Z., Grijalva, S., Hong, T. et Bessa, R. (2020). Big data analytics for future electricity grids. *Electric Power Systems Research*, 189:106788.

Kim, H.-T. et Ahn, C. W. (2015). A New Grammatical Evolution Based on Probabilistic Context-free Grammar. *In Proceedings of the 18th Asia Pacific Symposium on Intelligent and Evolutionary Systems*, volume 2, pages 1–12. Springer International Publishing.

Kim, J. T., Kim, S. et Petersen, B. K. (2020). An Interactive Visualization Platform for Deep Symbolic Regression. *In Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence*, pages 5261–5263. International Joint Conferences on Artificial Intelligence Organization.

Kim, S., Lu, P. Y., Mukherjee, S., Gilbert, M., Jing, L., Ceperic, V. et Soljacic, M. (2021). Integration of Neural Network-Based Symbolic Regression in Deep Learning for Scientific Discovery. *IEEE Transactions on Neural Networks and Learning Systems*, 32(9): 4166–4177.

Kingma, D. P., Mohamed, S., Jimenez Rezende, D. et Welling, M. (2014). Semi-supervised learning with deep generative models. *In Advances in neural information processing systems*, volume 27. Curran Associates, Inc.

Kingma, D. P. et Welling, M. (2014). Auto-Encoding Variational Bayes. *In 2nd International Conference on Learning Representations, (ICLR 2014), Conference Track Proceedings*.

Knuth, D. E. (1964). backus normal form vs. Backus Naur form. *Communications of the ACM*, 7(12):735–736.

Kosorukoff, A. (2001). Human based genetic algorithm. *In 2001 IEEE International Conference on Systems, Man and Cybernetics. e-Systems and e-Man for Cybernetics in Cyberspace*, volume 5, pages 3464–3469. IEEE.

Kotthoff, L., Thornton, C., Hoos, H. H., Hutter, F. et Leyton-Brown, K. (2019). Auto-WEKA : Automatic model selection and hyperparameter optimization in WEKA. *In Automated Machine Learning*, pages 81–95. Springer.

Koza, J. R. (1990). Concept Formation and Decision Tree Induction Using the Genetic Programming Paradigm. *In Parallel Problem Solving from Nature, 1st Workshop, PPSN I, Proceedings*, volume 496, pages 124–128.

Koza, J. R. (1992). Hierarchical Automatic Function Definition in Genetic Programming. *In Proceedings of the Second Workshop on Foundations of Genetic Algorithms.*, pages 297–318.

Krening, S. et Feigh, K. M. (2019). Effect of Interaction Design on the Human Experience with Interactive Reinforcement Learning. *In Proceedings of the 2019 on Designing Interactive Systems Conference*, pages 1089–1100.

Krening, S., Harrison, B., Feigh, K. M., Isbell, C. L., Riedl, M. et Thomaz, A. (2017). Learning From Explanations Using Sentiment and Advice in RL. *IEEE Transactions on Cognitive and Developmental Systems*, 9(1):44–55.

Kreps, D. M. (1988). *Notes on the Theory of Choice*. Routledge.

Krizhevsky, A., Sutskever, I. et Hinton, G. E. (2012). ImageNet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25.

Kubalík, J., Derner, E., Zegklitz, J. et Babuska, R. (2021). Symbolic Regression Methods for Reinforcement Learning. *IEEE Access*, 9:139697–139711.

Kuhlman, C., Doherty, D., Nurbekova, M., Deva, G., Phyo, Z., Schoenhagen, P., Valkenburg, M. V., Rundensteiner, E. A. et Harrison, L. (2019). Evaluating Preference Collection Methods for Interactive Ranking Analytics. *In Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems, (CHI 2019)*, CHI '19, page 1–11. Association for Computing Machinery.

Kuhlman, C., Valkenburg, M. V., Doherty, D., Nurbekova, M., Deva, G., Phyo, Z., Rundensteiner, E. A. et Harrison, L. (2018). Preference-driven Interactive Ranking System for Personalized Decision Support. *In Proceedings of the 27th ACM International Conference on Information and Knowledge Management, (CIKM 2018)*, pages 1931–1934. Association for Computing Machinery.

Kulesza, T., Burnett, M., Wong, W.-K. et Stumpf, S. (2015). Principles of Explanatory Debugging to Personalize Interactive Machine Learning. *In Proceedings of the 20th International Conference on Intelligent User Interfaces*. Association for Computing Machinery.

Kundur, P. (1994). *Power system stability and control*. CRC press New York, NY, USA.

Kusner, M. J., Paige, B. et Hernández-Lobato, J. M. (2017). Grammar variational autoencoder. *In International Conference on Machine Learning*, pages 1945–1954. PMLR.

La Cava, W., Orzechowski, P., Burlacu, B., de Franca, F., Virgolin, M., Jin, Y., Kommenda, M. et Moore, J. (2021). Contemporary symbolic regression methods and their relative performance. *In Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks*, volume 1.

Lample, G. et Charton, F. (2020). Deep Learning For Symbolic Mathematics. *In 8th International Conference on Learning Representations, (ICLR 2020)*. OpenReview.net.

Landajuela, M., Petersen, B. K., Kim, S., Santiago, C. P., Glatt, R., Mundhenk, N., Pettit, J. F. et Faissol, D. (2021). Discovering symbolic policies with deep reinforcement learning. *In International Conference on Machine Learning*, pages 5979–5989. PMLR.

Lange, R. T. et Faisal, A. (2019). Semantic rl with action grammars : Data-efficient learning of hierarchical task abstractions. *arXiv preprint arXiv :1907.12477*.

Lau, K., López, R. et Oñate, E. (2009). A neural networks approach to aerofoil noise prediction. Rapport technique PI 335, Universidad Politécnica de Cataluña.

Laugel, T., Lesot, M., Marsala, C., Renard, X. et Detyniecki, M. (2019). The Dangers of Post-hoc Interpretability : Unjustified Counterfactual Explanations. *In Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, (IJCAI 2019)*, pages 2801–2807. ijcai.org.

Le, N., Xuan, H. N., Brabazon, A. et Thi, T. P. (2016). Complexity measures in Genetic Programming learning : A brief review. *In IEEE Congress on Evolutionary Computation (CEC 2016)*, pages 2409–2416. IEEE.

Le Monde (2021). Tribune de marianne laigneau - 05/11/2021. https://www.lemonde.fr/smart-cities/article/2021/11/05/marianne-laigneau-l-electricite-est-l-energie-d-avenir-pour-repondre-aux-nouveaux-usages-et-aux-enjeux-environnementaux_6101016_4811534.html. - Dernier accès 27 septembre 2022.

Lee, J., Lee, Y., Kim, J., Kosiorek, A., Choi, S. et Teh, Y. W. (2019). Set transformer : A framework for attention-based permutation-invariant neural networks. *In International Conference on Machine Learning*, pages 3744–3753. PMLR.

Lee, K., Smith, L. M., Dragan, A. D. et Abbeel, P. (2021). B-pref : Benchmarking preference-based reinforcement learning. *In Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks 1, NeurIPS Datasets and Benchmarks 2021*.

Lehman, J., Clune, J., Misevic, D., Adami, C., Altenberg, L., Beaulieu, J., Bentley, P. J., Bernard, S., Beslon, G., Bryson, D. M., Cheney, N., Chrabaszcz, P., Cully, A., Doncieux, S., Dyer, F. C., Ellefsen, K. O., Feldt, R., Fischer, S., Forrest,

S., Fŕenoy, A., Gagńe, C., Le Goff, L., Grabowski, L. M., Hodjat, B., Hutter, F., Keller, L., Knibbe, C., Krcah, P., Lenski, R. E., Lipson, H., MacCurdy, R., Maestre, C., Miikkulainen, R., Mitri, S., Moriarty, D. E., Mouret, J.-B., Nguyen, A., Ofria, C., Parizeau, M., Parsons, D., Pennock, R. T., Punch, W. F., Ray, T. S., Schoenauer, M., Schulte, E., Sims, K., Stanley, K. O., Taddei, F., Tarapore, D., Thibault, S., Watson, R., Weimer, W. et Yosinski, J. (2020). The Surprising Creativity of Digital Evolution : A Collection of Anecdotes from the Evolutionary Computation and Artificial Life Research Communities. *Artificial Life*, 26(2):274–306.

Linardatos, P., Papastefanopoulos, V. et Kotsiantis, S. (2021). Explainable AI : A Review of Machine Learning Interpretability Methods. *Entropy*, 23(1):18.

Linz, P. (2011). *An Introduction to Formal Languages and Automata*. Jones & Bartlett Publishers.

Lipton, Z. C. (2018). The mythos of model interpretability. *Commununication of the ACM*, 61(10):36–43.

Liu, W., Dai, B., Humayun, A., Tay, C., Yu, C., Smith, L. B., Rehg, J. M. et Song, L. (2017). Iterative machine teaching. *In International Conference on Machine Learning*, pages 2149–2158. PMLR.

Liu, Z., Madhavan, V. et Tegmark, M. (2022). AI Poincaré 2.0 : Machine Learning Conservation Laws from Differential Equations. *arXiv preprint arXiv :2203.12610*.

Loughran, R., McDermott, J. et O'Neill, M. (2015). Tonality driven piano compositions with grammatical evolution. *In 2015 IEEE Congress on Evolutionary Computation (CEC)*, pages 2168–2175. IEEE.

Lu, Q., Tao, F., Zhou, S. et Wang, Z. (2021). Incorporating Actor-Critic in Monte Carlo tree search for symbolic regression. *Neural Computing and Applications*, 33(14):8495–8511.

Lucena-Sánchez, E., Sciavicco, G. et Stan, I. E. (2021). Feature and Language Selection in Temporal Symbolic Regression for Interpretable Air Quality Modelling. *Algorithms*, 14(3): 76.

Luke, S. et Panait, L. (2006). A Comparison of Bloat Control Methods for Genetic Programming. *Evolutionary Computation*, 14(3):309–344.

Lundberg, S. M. et Lee, S.-I. (2017). A Unified Approach to Interpreting Model Predictions. *Advances in neural information processing systems*, 30:4765–4774.

Lv, J., Zhu, M., Pan, W. et Liu, X. (2019). Interactive Genetic Algorithm Oriented toward the Novel Design of Traditional Patterns. *Information*, 10(2):36.

Lynch, D., McDermott, J. et O'Neill, M. (2020). Program Synthesis in a Continuous Space Using Grammars and Variational Autoencoders. *In Parallel Problem Solving from Nature – PPSN XVI*, volume 12270 de *Lecture Notes in Computer Science*, pages 33–47. Springer International Publishing.

Ma, Z., Zhuang, Y., Weng, P., Zhuo, H. H., Li, D., Liu, W. et Hao, J. (2021). Learning Symbolic Rules for Interpretable Deep Reinforcement Learning. *arXiv preprint arXiv :2103.08228*.

Maclin, R., Shavlik, J. W., Torrey, L., Walker, T. et Wild, E. W. (2005). Giving Advice about Preferred Actions to Reinforcement Learners Via Knowledge-Based Kernel Regression. *In Proceedings, The Twentieth National Conference on Artificial Intelligence and the Seventeenth Innovative Applications of Artificial Intelligence Conference*, pages 819–824. AAAI Press / The MIT Press.

Mandow, L., de-la Cruz, J.-L. P., Rodríguez-Gavilán, A. B. et Ruiz-Montiel, M. (2020). Architectural planning with shape grammars and reinforcement learning : Habitability and energy efficiency. *Engineering Applications of Artificial Intelligence*, 96:103909.

Mann, H. B. et Whitney, D. R. (1947). On a Test of Whether one of Two Random Variables is Stochastically Larger than the Other. *Annals of Mathematical Statistics*, 18(1):50–60.

Maricau, E., De Jonghe, D. et Gielen, G. (2012). Hierarchical analog circuit reliability analysis using multivariate nonlinear regression and active learning sample selection. *In 2012 Design, Automation Test in Europe Conference Exhibition (DATE)*, pages 745–750.

Marot, A., Donnot, B., Romero, C., Donon, B., Lerousseau, M., Veyrin-Forrer, L. et Guyon, I. (2020). Learning to run a power network challenge for training topology controllers. *Electric Power Systems Research*, 189:106635.

Marot, A., Donnot, B., Tazi, S. et Panciatici, P. (2018a). Expert System for topological remedial action discovery in smart grids. *In Mediterranean Conference on Power Generation, Transmission, Distribution and Energy Conversion (MEDPOWER 2018)*. IET Digital Library.

Marot, A., Kelly, A., Naglic, M., Barbesant, V., Cremer, J., Stefanov, A. et Viebahn, J. (2022a). Perspectives on future power system control centers for energy transition. *Journal of Modern Power Systems and Clean Energy*, 10(2):328–344.

Marot, A., Rosin, A., Crochepierre, L., Donnot, B., Pinson, P. et Boudjeloud-Assala, L. (2019). Interpreting Atypical Conditions in Systems with Deep Conditional Autoencoders : The Case of Electrical Consumption. *In Machine Learning and Knowledge Discovery in Databases : European Conference, (ECML PKDD 2019), Proceedings, Part III*, volume 11908 de *Lecture Notes in Computer Science*, pages 638–654.

Marot, A., Rozier, A., Dussartre, M., Crochepierre, L. et Donnot, B. (2022b). Towards an AI assistant for human grid operators. *In The First International Conference on Hybrid Human Artificial Intelligence (HHAI 2022)*, pages 1–19. IOS Press.

Marot, A., Tazi, S., Donnot, B. et Panciatici, P. (2018b). Guided Machine Learning for Power Grid Segmentation. *In 2018 IEEE PES Innovative Smart Grid Technologies Conference Europe (ISGT-Europe)*, pages 1–6. IEEE.

Martin, P. et Poli, R. (2002). Crossover operators for a hardware implementation of gp using fpgas and handel-c. *In Proceedings of the 4th Annual Conference on Genetic and Evolutionary Computation*, GECCO'02, pages 845–852. Morgan Kaufmann Publishers Inc.

Martius, G. et Lampert, C. H. (2017). Extrapolation and learning equations. *In 5th International Conference on Learning Representations (ICLR 2017), Workshop Track Proceedings*. OpenReview.net.

Mathewson, K. W. et Pilarski, P. M. (2022). A Brief Guide to Designing and Evaluating Human-Centered Interactive Machine Learning. *arXiv preprint arXiv :2204.09622*.

McConaghy, T. (2011). FFX : Fast, Scalable, Deterministic Symbolic Regression Technology. *In Genetic Programming Theory and Practice IX*, pages 235–260. Springer New York.

McDermott, J., White, D. R., Luke, S., Manzoni, L., Castelli, M., Vanneschi, L., Jaskowski, W., Krawiec, K., Harper, R., De Jong, K. et O'Reilly, U.-M. (2012). Genetic Programming Needs Better Benchmarks. *In Proceedings of the 14th Annual Conference on Genetic and Evolutionary Computation*, page 791–798. ACM Press.

Miller, B. L., Goldberg, D. E. *et al.* (1995). Genetic algorithms, tournament selection, and the effects of noise. *Complex systems*, 9(3):193–212.

Ministère de la transition écologique (2021). Évolution de la production primaire d'énergie renouvelable par filière en france. `https://www.statistiques.developpement-durable.gouv.fr/sites/default/files/2021-07/chiffres_cles_enr_edition2021.xlsx`. - Dernier accès 27 septembre 2022.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., Petersen, S., Beattie, C., Sadik, A., Antonoglou, I., King, H., Kumaran, D., Wierstra, D., Legg, S. et Hassabis, D. (2015). Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533.

Molnar, C. (2020). *Interpretable machine learning : a guide for making Black Box Models interpretable*. Lulu.com.

Monarch, R. M. (2021). *Human-in-the-Loop Machine Learning : Active Learning and Annotation for Human-Centered AI*. Simon and Schuster.

Montana, D. J. (1995). Strongly Typed Genetic Programming. *Evolutionary computation*, 3(2):199–230.

Mosqueira-Rey, E., Alonso-Ríos, D. et Baamonde-Lozano, A. (2021). Integrating Iterative Machine Teaching and Active Learning into the Machine Learning Loop. *Procedia Computer Science*, 192:553–562. Knowledge-Based and Intelligent Information & Engineering Systems : Proceedings of the 25th International Conference KES2021.

Mundhenk, T., Landajuela, M., Glatt, R., Santiago, C. P., Petersen, B. K. *et al.* (2021). Symbolic Regression via Deep Reinforcement Learning Enhanced Genetic Programming Seeding. *Advances in Neural Information Processing Systems*, 34:24912–24923.

Narvekar, S., Peng, B., Leonetti, M., Sinapov, J., Taylor, M. E. et Stone, P. (2020). Curriculum Learning for Reinforcement Learning Domains : A Framework and Survey. *Journal of Machine Learning Research*, 21:1–50.

Nelder, J. A. et Wedderburn, R. W. (1972). Generalized Linear Models. *Journal of the Royal Statistical Society : Series A (General)*, 135(3):370–384.

Nicolau, M. (2017). Understanding grammatical evolution : initialisation. *Genetic Programming and Evolvable Machines*, 18(4):467–507.

Nicolau, M. et Agapitos, A. (2018). *Understanding grammatical evolution : Grammar design*, pages 23–53. Springer International Publishing.

O'Neill, M., McDermott, J., Swafford, J. M., Byrne, J., Hemberg, E., Brabazon, A., Shotton, E., McNally, C. et Hemberg, M. (2010). Evolutionary design using grammatical evolution and shape grammars : Designing a shelter. *International Journal of Design Engineering*, 3(1):4–24.

O'Neill, M. et Ryan, C. (2001). Grammatical evolution. *IEEE Transactions on Evolutionary Computation*, 5(4):349–358.

Overbye, T. J., Wiegmann, D. A., Rich, A. M. et Sun, Y. (2001). Human Factors Analysis of Power System Visualizations. *In Proceedings of the 34th Annual Hawaii International Conference on System Sciences (HICSS-34)*, pages 647–652. IEEE Computer Society.

O'Neill, M. et Brabazon, A. (2005). mGGA : The meta-grammar genetic algorithm. *In European Conference on Genetic Programming*, pages 311–320. Springer.

Pagie, L. et Hogeweg, P. (1997). Evolutionary Consequences of Coevolving Targets. *Evolutionary computation*, 5:401–18.

Pandey, H. M. (2021). *State of the Art on Grammatical Inference Using Evolutionary Method.* Academic Press.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Köpf, A., Yang, E. Z., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J. et Chintala, S. (2019). PyTorch : An Imperative Style, High-Performance Deep Learning Library. *In Advances in Neural Information Processing Systems*, volume 32, pages 8024–8035. Curran Associates, Inc.

Personnaz, A., Amer-Yahia, S., Berti-Équille, L., Fabricius, M. et Subramanian, S. (2021). DORA THE EXPLORER : Exploring very large data with interactive deep reinforcement learning. *In CIKM '21 : The 30th ACM International Conference on Information & Knowledge Management*, pages 4769–4773. ACM.

Petersen, B. K., Larma, M. L., Mundhenk, T. N., Santiago, C. P., Kim, S. et Kim, J. T. (2021). Deep symbolic regression : Recovering mathematical expressions from data via risk-seeking policy gradients. *In 9th International Conference on Learning Representations, (ICLR 2021).*

Pigozzi, F., Medvet, E. et Nenzi, L. (2021). Mining Road Traffic Rules with Signal Temporal Logic and Grammar-Based Genetic Programming. *Applied Sciences*, 11(22):10573.

Poli, R. et Langdon, W. B. (1998). Genetic programming with one-point crossover. *In Soft Computing in Engineering Design and Manufacturing*, pages 180–189. Springer.

Pommeranz, A., Broekens, J., Wiggers, P., Brinkman, W.-P. et Jonker, C. M. (2012). Designing interfaces for explicit preference elicitation : a user-centered investigation of preference representation and elicitation process. *User Modeling and User-Adapted Interaction*, 22(4):357–397.

Prieschl, S., Girardi, D. et Kronberger, G. (2019). Using Ontologies to Express Prior Knowledge for Genetic Programming. *In International Cross-Domain Conference for Machine Learning and Knowledge Extraction*, pages 362–376.

Projet GARPUR (2015). Projet GARPUR, Funtional Analysis of System Operation processes Workpackage 6.1. `https://www.sintef.no/globalassets/project/garpur/deliverables/garpur-d6.1-functional-analysis-system-operation.pdf`. - Dernier accès 27 septembre 2022.

Prostejovsky, A. M., Brosinsky, C., Heussen, K., Westermann, D., Kreusel, J. et Marinelli, M. (2019). The future role of human operators in highly automated electric power systems. *Electric Power Systems Research*, 175:105883.

Purohit, A., Bhardwaj, A., Tiwari, A. et Chaudhari, N. S. (2011). Handling the Problem of Code Bloating to Enhance the Performance of Classifier Designed Using Genetic Programming. *In Proceedings of the 5th Indian International Conference on Artificial Intelligence, IICAI*, pages 333–342. IICAI.

Puusa, A. et Eerikäinen, M. (2010). Is tacit knowledge really tacit ? *Electronic Journal of Knowledge Management*, 8(3):307–318.

Radford, A., Narasimhan, K., Salimans, T. et Sutskever, I. (2018). Improving language understanding by generative pre-training. Rapport technique, OpenAI.

Raghu, M., Poole, B., Kleinberg, J., Ganguli, S. et Sohl-Dickstein, J. (2017). On the expressive power of deep neural networks. *In Proceedings of the 34th International*

Conference on Machine Learning, volume 70 de *Proceedings of Machine Learning Research*, pages 2847–2854. PMLR.

Ramos, G. A., Meek, C., Simard, P. Y., Suh, J. et Ghorashi, S. (2020). Interactive machine teaching : a human-centered approach to building machine-learned models. *Human–Computer Interaction*, 35(5-6):413–451.

Ratle, A. et Sebag, M. (2000). Genetic Programming and Domain Knowledge : Beyond the Limitations of Grammar-Guided Machine Discovery. *In Parallel Problem Solving from Nature PPSN VI*, volume 1917, pages 211–220. Springer.

Ratle, A. et Sebag, M. (2001). Avoiding the bloat with stochastic grammar-based genetic programming. *In Selected Papers from the 5th European Conference on Artificial Evolution*, page 255–266. Springer-Verlag.

Ribeiro, R. (2013). Tacit knowledge management. *Phenomenology and the cognitive sciences*, 12(2):337–366.

Robert, S., Büttner, S., Röcker, C. et Holzinger, A. (2016). Reasoning under uncertainty : Towards collaborative interactive machine learning. *In Machine Learning for Health Informatics : State-of-the-Art and Future Challenges*, pages 357–376. Springer International Publishing.

Rosenblatt, F. (1958). The perceptron : A probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6):386.

Rte (2021). Futurs énergétiques 2050 - octobre 2021. `https://assets.rte-france.com/prod/public/2021-10/Futurs-Energetiques-2050-principaux-resultats_0.pdf`. - Dernier accès 27 septembre 2022.

Rudin, C. (2019). Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. *Nature Machine Intelligence*, 1(5):206–215.

Ruiz-Montiel, M., Boned, J., Gavilanes, J., Jiménez, E., Mandow, L. et de-la Cruz, J.-L. P. (2013). Design with shape grammars and reinforcement learning. *Advanced Engineering Informatics*, 27(2):230–245.

Rummery, G. A. et Niranjan, M. (1994). On-line Q-learning using connectionist systems. Rapport technique TR 166, Cambridge University Engineering Department.

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A. C. et Fei-Fei, L. (2015). ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252.

Ryan, C., Collins, J. J. et O'Neill, M. (1998). Grammatical Evolution : Evolving Programs for an Arbitrary Language. *In Genetic Programming, First European Workshop, EuroGP'98, Proceedings*, volume 1391, pages 83–96. Springer.

Sahoo, S., Lampert, C. et Martius, G. (2018). Learning Equations for Extrapolation and Control. *In Proceedings of the 35th International Conference on Machine Learning*, volume 80 de *Proceedings of Machine Learning Research*, pages 4442–4450. PMLR.

Sakakibara, Y. (2017). Probabilistic Context-Free Grammars. *In Encyclopedia of Machine Learning and Data Mining*, pages 1013–1017. Springer.

Sambasivan, N. et Veeraraghavan, R. (2022). The Deskilling of Domain Expertise in AI Development. *In CHI Conference on Human Factors in Computing Systems*, CHI '22, pages 587 :1–587 :14. Association for Computing Machinery.

Sammut, C. et Webb, G. I. (2011). *Encyclopedia of machine learning*. Springer Science & Business Media.

Savic, D. A., Walters, G. A. et Davidson, J. W. (1999). A genetic programming approach to rainfall-runoff modelling. *Water resources management*, 13(3):219–231.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A. et Klimov, O. (2017). Proximal Policy Optimization Algorithms. *arXiv preprint arXiv :1707.06347*.

Sereeter, B., Vuik, C. et Witteveen, C. (2019). On a comparison of Newton–Raphson solvers for power flow problems. *Journal of Computational and Applied Mathematics*, 360:157–169.

Settles, B. (2009). *Active learning literature survey*. University of Wisconsin-Madison Department of Computer Sciences.

Shadrick, S. B., Lussier, J. W. et Hinkle, R. (2005). Concept development for future domains : A new method of knowledge elicitation. Rapport technique, U.S. Army Research Institute for the Behavioral and Social Sciences.

Silva, S. G. O. d. (2008). *Controlling bloat : individual and population based approaches in genetic programming*. Thèse de doctorat, Universidade de Coimbra.

Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., Van Den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M. *et al.* (2016). Mastering the game of Go with deep neural networks and tree search. *nature*, 529(7587):484–489.

Sotto, L. F. D. P. et de Melo, V. V. (2017). A probabilistic linear genetic programming with stochastic context-free grammar for solving symbolic regression problems. *In Proceedings of the Genetic and Evolutionary Computation Conference*, pages 1017–1024. ACM.

Specht, D. F. (1991). A general regression neural network. *IEEE Transactions on Neural Networks*, 2(6):568–576.

Stott, B., Jardim, J. et Alsac, O. (2009). DC Power Flow Revisited. *IEEE Transactions on Power Systems*, 24(3):1290–1300.

Studer, R., Benjamins, V. R. et Fensel, D. (1998). Knowledge engineering : Principles and methods. *Data & knowledge engineering*, 25(1-2):161–197.

Sutton, R. S. et Barto, A. G. (2018). *Reinforcement learning : An introduction*. MIT press.

Takagi, H. (2001). Interactive evolutionary computation : fusion of the capabilities of ec optimization and human evaluation. *Proceedings of the IEEE*, 89(9):1275–1296.

Taye, M. M. (2010). Understanding semantic web and ontologies : Theory and applications. *arXiv preprint arXiv :1006.4567*.

Tegen, A., Davidsson, P. et Persson, J. A. (2020). A taxonomy of interactive online machine learning strategies. *In Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 137–153. Springer.

Tolmeijer, S., Gadiraju, U., Ghantasala, R., Gupta, A. et Bernstein, A. (2021). Second Chance for a First Impression ? Trust Development in Intelligent System Interaction. *In Proceedings of the 29th ACM Conference on User Modeling, Adaptation and Personalization*, page 77–87. Association for Computing Machinery.

Udrescu, S., Tan, A. K., Feng, J., Neto, O., Wu, T. et Tegmark, M. (2020). AI Feynman 2.0 : Pareto-optimal symbolic regression exploiting graph modularity. *In Advances in Neural Information Processing Systems*, volume 33. Curran Associates, Inc.

Udrescu, S.-M. et Tegmark, M. (2020). AI Feynman : A physics-inspired method for symbolic regression. *Science Advances*, 6(16):eaay2631.

Uy, N. Q., Hoai, N. X., O'Neill, M., McKay, R. I. et Galván-López, E. (2011). Semantically-based crossover in genetic programming : application to real-valued symbolic regression. *Genetic Programming and Evolvable Machines*, 12(2):91–119.

Vaddireddy, H. et San, O. (2019). Equation Discovery Using Fast Function Extraction : a Deterministic Symbolic Regression Approach. *Fluids*, 4(2):111.

Valipour, M., You, B., Panju, M. et Ghodsi, A. (2021). SymbolicGPT : A Generative Transformer Model for Symbolic Regression. *arXiv preprint arXiv :2106.14131*.

Vapnik, V. (1999). *The nature of statistical learning theory*. Springer science & business media.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L. u. et Polosukhin, I. (2017). Attention is All you Need. *In Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

Ventocilla, E., Helldin, T., Riveiro, M., Bae, J., Boeva, V., Falkman, G. et Lavesson, N. (2018). Towards a taxonomy for interpretable and interactive machine learning. *In 2nd Workshop on Explainable AI (XAI-18), as part of the 27th International Joint Conferences on Artificial Intelligence (IJCAI)*, pages 151–157.

Verma, A., Murali, V., Singh, R., Kohli, P. et Chaudhuri, S. (2018). Programmatically Interpretable Reinforcement Learning. *In Proceedings of the 35th International Conference on Machine Learning*, volume 80 de *Proceedings of Machine Learning Research*, pages 5045–5054.

Videau, M., Leite, A., Teytaud, O. et Schoenauer, M. (2022). Multi-objective Genetic Programming for Explainable Reinforcement Learning. *In European Conference on Genetic Programming (Part of EvoStar)*, pages 278–293. Springer.

Vladislavleva, E. J., Smits, G. F. et den Hertog, D. (2009). Order of Nonlinearity as a Complexity Measure for Models Generated by Symbolic Regression via Pareto Genetic Programming. *IEEE Transactions on Evolutionary Computation*, 13(2):333–349.

von Rueden, L., Mayer, S., Beckh, K., Georgiev, B., Giesselbach, S., Heese, R., Kirsch, B., Walczak, M., Pfrommer, J., Pick, A., Ramamurthy, R., Garcke, J., Bauckhage, C. et Schuecker, J. (2021). Informed Machine Learning - A Taxonomy and Survey of Integrating Prior Knowledge into Learning Systems. *IEEE Transactions on Knowledge and Data Engineering*, pages 1–1.

Vovk, V. (2013). *Kernel Ridge Regression*, pages 105–116. Springer.

Wallace, B. C., Small, K., Brodley, C. E., Lau, J. et Trikalinos, T. A. (2012). Deploying an interactive machine learning system in an evidence-based practice center : abstrackr. *In ACM International Health Informatics Symposium, (IHI '12)*, pages 819–824. Association for Computing Machinery.

Wang, S. et Takagi, H. (2005). Improving the performance of predicting users' subjective evaluation characteristics to reduce their fatigue in IEC. *Journal of physiological anthropology and applied human science*, 24(1):81–85.

Wang, Z. J., Kale, A., Nori, H., Stella, P., Nunnally, M., Chau, D. H., Vorvoreanu, M., Vaughan, J. W. et Caruana, R. (2021). GAM Changer : Editing Generalized Additive Models with Interactive Visualization. *arXiv preprint arXiv :2112.03245*.

Watkins, C. J. C. H. (1989). *Learning from delayed rewards*. Thèse de doctorat, King's College, Cambridge United Kingdom.

Whigham, P. A. *et al.* (1995). Grammatically-based genetic programming. *Proceedings of the workshop on genetic programming : from theory to real-world applications*, 16(3):33–41.

White, D. R., McDermott, J., Castelli, M., Manzoni, L., Goldman, B. W., Kronberger, G., Jaśkowski, W., O'Reilly, U.-M. et Luke, S. (2013). Better GP benchmarks : community survey results and proposals. *Genetic Programming and Evolvable Machines*, 14(1):3–29.

Wierstra, D., Foerster, A., Peters, J. et Schmidhuber, J. (2007). Solving Deep Memory POMDPs with Recurrent Policy Gradients. *In Artificial Neural Networks - ICANN 2007, 17th International Conference, Proceedings, Part I*, volume 4668 de *Lecture Notes in Computer Science*, pages 697–706. Springer.

Williams, R. J. (1992). Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3-4):229–256.

Wirth, C., Akrour, R., Neumann, G., Fürnkranz, J. *et al.* (2017). A survey of preference-based reinforcement learning methods. *Journal of Machine Learning Research*, 18(136):1–46.

Xu, Y., Zhang, H., Singh, A., Dubrawski, A. et Miller, K. (2017). Noise-Tolerant Interactive Learning Using Pairwise Comparisons. *In Advances in Neural Information Processing Systems*, volume 30, pages 2431–2440. Curran Associates, Inc.

Yang, Y. et Tian, X. (2019). Combining Users' Cognition Noise with Interactive Genetic Algorithms and Trapezoidal Fuzzy Numbers for Product Color Design. *Computational Intelligence and Neuroscience*, 2019:1–11.

Zhang, H. et Zhou, A. (2021). RL-GEP : Symbolic Regression via Gene Expression Programming and Reinforcement Learning. *In 2021 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8.

Zhang, W. et Weiss, J. C. (2021). Fair Decision-making Under Uncertainty. *In 2021 IEEE International Conference on Data Mining (ICDM)*, pages 886–895. IEEE.

Zhang, Y., Tiňo, P., Leonardis, A. et Tang, K. (2021). A Survey on Neural Network Interpretability. *IEEE Transactions on Emerging Topics in Computational Intelligence*, 5(5):726–742.

Zhu, X. (2015). Machine Teaching : An Inverse Problem to Machine Learning and an Approach Toward Optimal Education. *In Proceedings of the AAAI Conference on Artificial Intelligence*, volume 29. AAAI Press.

Zou, H. et Hastie, T. (2005). Regularization and variable selection via the elastic net. *Journal of the royal statistical society : series B (statistical methodology)*, 67(2):301–320.

Zou, H., Hastie, T. et Tibshirani, R. (2007). On the "degrees of freedom" of the lasso. *The Annals of Statistics*, 35(5):2173–2192.

Zytek, A., Arnaldo, I., Liu, D., Berti-Équille, L. et Veeramachaneni, K. (2022). The need for interpretable features : Motivation and taxonomy. *SIGKDD Explorations Newsletter*, 24(1):1–13.

Résumé

Dans le contexte de la transition énergétique et de l'augmentation des interconnexions entre les réseaux de transport d'électricité en Europe, les opérateurs du réseau français doivent désormais faire face à davantage de fluctuations et des dynamiques nouvelles sur le réseau. Pour garantir la sûreté de ce réseau, les opérateurs s'appuient sur des logiciels informatiques permettant de réaliser des simulations, ou de suivre l'évolution d'indicateurs créés manuellement par des experts grâce à leur connaissance du fonctionnement du réseau. Le gestionnaire de réseau de transport d'électricité français RTE (Réseau de Transport d'Electricité) s'intéresse notamment aux développements d'outils permettant d'assister les opérateurs dans leur tâche de surveillance des transits sur les lignes électriques. Les transits sont en effet des grandeurs particulièrement importantes pour maintenir le réseau dans un état de sécurité, garantissant la sûreté du matériel et des personnes. Cependant, les indicateurs utilisés ne sont pas faciles à mettre à jour du fait de l'expertise nécessaire pour les construire et les analyser.

Pour répondre à la problématique énoncée, cette thèse a pour objet la construction d'indicateurs, sous la forme d'expressions symboliques, permettant d'estimer les transits sur les lignes électriques. Le problème est étudié sous l'angle de la Régression Symbolique et investigué à la fois par des approches génétiques d'Evolution Grammaticale et d'Apprentissage par Renforcement dans lesquelles la connaissance experte, explicite et implicite, est prise en compte. Les connaissances explicites sur la physique et l'expertise du domaine électrique sont représentées sous la forme d'une grammaire non-contextuelle délimitant l'espace fonctionnel à partir duquel l'expression est créée. Une première approche d'Evolution Grammaticale Interactive propose d'améliorer incrémentalement les expressions trouvées par la mise à jour d'une grammaire entre les apprentissages évolutionnaires. Les expressions obtenues sur des données réelles issues de l'historique du réseau sont validées par une évaluation de métriques d'apprentissages, complétée par une évaluation de leur interprétabilité. Dans un second temps, nous proposons une approche par renforcement pour chercher dans un espace délimité par une grammaire non-contextuelle afin de construire une expression symbolique pertinente pour des applications comportant des contraintes physiques. Cette méthode est validée sur des données de l'état de l'art de la régression symbolique, ainsi qu'un jeu de données comportant des contraintes physiques pour en évaluer l'interprétabilité.

De plus, afin de tirer parti des complémentarités entre les capacités des algorithmes d'apprentissage automatique et de l'expertise des opérateurs du réseau, des algorithmes interactifs de Régression Symbolique sont proposés et intégrés dans des plateformes interactives. L'interactivité est employée à la fois pour mettre à jour la connaissance représentée sous forme grammaticale, analyser, interagir avec et commenter les solutions proposées par les différentes approches. Ces algorithmes et interfaces interactifs ont également pour but de prendre en compte de la connaissance implicite, plus difficile à formaliser, grâce à l'utilisation de mécanismes d'interactions basés sur des suggestions et des préférences de l'utilisateur.

Mots-clés: Apprentissage Automatique, Régression Symbolique, Evolution Grammaticale, Apprentissage par Renforcement, Interactivité, Connaissance experte, Réseau de transport d'électricité.

Abstract

In the energy transition context and the increase in interconnections between the electricity transmission networks in Europe, the French network operators must now deal with more fluctuations and new network dynamics. To guarantee the safety of the network, operators rely on computer software that allows them to carry out simulations or to monitor the evolution of indicators created manually by experts, thanks to their knowledge of the operation of the network. The French electricity transmission network operator RTE (Réseau de Transport d'Electricité) is particularly interested in developing tools to assist operators in monitoring flows on power lines. Flows are notably important to maintain the network in a safe state, guaranteeing the safety of equipment and people. However, the indicators used are not easy to update because of the expertise required to construct and analyze them.

In order to address the stated problem, this thesis aims at constructing indicators, in the form of symbolic expressions, to estimate flows on power lines. The problem is studied from the Symbolic Regression perspective and investigated using both Grammatical Evolution and Reinforcement Learning approaches in which explicit and implicit expert knowledge is taken into account. Explicit knowledge about the physics and expertise of the electrical domain is represented in the form of a Context-Free Grammar to limit the functional space from which an expression is created. A first approach of Interactive Grammatical Evolution proposes to incrementally improve found expressions by updating a grammar between evolutionary learnings. Expressions are obtained on real-world data from the network history, validated by an analysis of learning metrics and an interpretability evaluation. Secondly, we propose a reinforcement approach to search in a space delimited by a Context-Free Grammar in order to build a relevant symbolic expression to applications involving physical constraints. This method is validated on state-of-the-art Symbolic Regression benchmarks and also on a dataset with physical constraints to assess its interpretability.

Furthermore, in order to take advantage of the complementarities between the capacities of machine learning algorithms and the expertise of network operators, interactive Symbolic Regression algorithms are proposed and integrated into interactive platforms. Interactivity allows updating the knowledge represented in grammatical form and analyzing, interacting with, and commenting on the solutions found by the different approaches. These algorithms and interactive interfaces also aim to take into account implicit knowledge, which is more difficult to formalize, through interaction mechanisms based on suggestions and user preferences.

Keywords: Machine Learning, Symbolic Regression, Grammatical Evolution, Reinforcement Learning, Interactivity, Expert Knowledge, Electricity transmission networks.