CHRISTIAN GAGNÉ

ALGORITHMES ÉVOLUTIONNAIRES APPLIQUÉS À LA RECONNAISSANCE DES FORMES ET À LA CONCEPTION OPTIQUE

Thèse présentée
à la Faculté des études supérieures de l'Université Laval
dans le cadre du programme de doctorat en génie électrique
pour l'obtention du grade de philosophiæ doctor (Ph.D.)

FACULTÉ DES SCIENCES ET DE GÉNIE
UNIVERSITÉ LAVAL
QUÉBEC

MAI 2005

Résumé

Les algorithmes évolutionnaires (AE) constituent une famille d'algorithmes inspirés de l'évolution naturelle. Ces algorithmes sont particulièrement utiles pour la résolution de problèmes où les algorithmes classiques d'optimisation, d'apprentissage ou de conception automatique sont incapables de produire des résultats satisfaisants. On propose dans cette thèse une approche méthodologique pour le développement de systèmes intelligents basés sur les AE. Cette approche méthodologique repose sur cinq principes : 1) utiliser des algorithmes et des représentations adaptés au problème ; 2) développer des hybrides entre des AE et des heuristiques du domaine d'application ; 3) tirer profit de l'optimisation évolutionnaire à plusieurs objectifs ; 4) faire de la co-évolution pour résoudre simultanément plusieurs sous-problèmes d'une application ou favoriser la robustesse ; et 5) utiliser un outil logiciel générique pour le développement rapide d'AE non conventionnels. Cette approche méthodologique est illustrée par quatre applications des AE à des problèmes difficiles. De plus, le cinquième principe est appuyé par l'étude sur la généricité dans les outils logiciels d'AE.

Le développement d'applications complexes avec les AE exige l'utilisation d'un outil logiciel générique. Six critères sont proposés ici pour évaluer la généricité des outils d'AE. De nombreux outils logiciels d'AE sont disponibles dans la communauté, mais peu d'entre eux peuvent être véritablement qualifiés de génériques. En effet, une évaluation de quelques outils relativement populaires nous indique que seulement trois satisfont pleinement à tous ces critères, dont la framework d'AE Open BEAGLE, développée durant le doctorat. Open BEAGLE est organisé en trois couches logicielles principales, avec à la base les fondations orientées objet, sur lesquelles s'ajoute une framework générique comprenant les mécanismes généraux de l'outil, ainsi que plusieurs frameworks spécialisées qui implantent différentes saveurs d'AE. L'outil comporte également deux extensions servant à distribuer des calculs sur plusieurs ordinateurs et à visualiser des résultats.

Ensuite, trois applications illustrent différentes approches d'utilisation des AE dans un contexte de reconnaissance des formes. Premièrement, on optimise des classifieurs

basés sur la règle du plus proche voisin avec la sélection de prototypes par un algorithme génétique, simultanément à la construction de mesures de voisinage par programmation génétique (PG). À cette co-évolution coopérative à deux espèces, on ajoute la co-évolution compétitive d'une troisième espèce pour la sélection de données de test, afin d'améliorer la capacité de généralisation des solutions. La deuxième application consiste en l'ingénierie de représentations par PG pour la reconnaissance de caractères manuscrits. Cette ingénierie évolutionnaire s'effectue par un positionnement automatique de régions dans la fenêtre d'attention jumelé à la sélection d'ensembles flous pour l'extraction de caractéristiques. Cette application permet d'automatiser la recherche de représentations de caractères, opération généralement effectuée par des experts humains suite à un processus d'essais et erreurs. Pour la troisième application en reconnaissance des formes, on propose un système extensible pour la combinaison hiérarchique de classifieurs dans un arbre de décision flou. Dans ce système, la topologie des arbres est évoluée par PG alors que les paramètres numériques des unités de classement sont déterminés par des techniques d'apprentissage spécialisées. Le système est testé avec trois types simples d'unités de classement. Pour toutes ces applications en reconnaissance des formes, on utilise une mesure d'adéquation à deux objectifs afin de minimiser les erreurs de classement et la complexité des solutions.

Une dernière application démontre l'efficacité des AE pour la conception de systèmes de lentilles. On utilise des stratégies d'évolution auto-adaptatives hybridées avec une technique d'optimisation locale spécialisée pour la résolution de deux problèmes complexes de conception optique. Dans les deux cas, on démontre que les AE hybrides sont capables de générer des résultats comparables ou supérieurs à ceux produits par des experts humains. Ces résultats sont prometteurs dans la perspective d'une automatisation plus poussée de la conception optique. On présente également une expérience supplémentaire avec une mesure à deux objectifs servant à maximiser la qualité de l'image et à minimiser le coût du système de lentilles.

Abstract

Evolutionary Algorithms (EA) encompass a family of robust search algorithms loosely inspired by natural evolution. These algorithms are particularly useful to solve problems for which classical algorithms of optimization, learning, or automatic design cannot produce good results. In this thesis, we propose a common methodological approach for the development of EA-based intelligent systems. This methodological approach is based on five principles : 1) to use algorithms and representations that are problem specific ; 2) to develop hybrids between EA and heuristics from the application field ; 3) to take advantage of multi-objective evolutionary optimization ; 4) to do co-evolution for the simultaneous resolution of several sub-problems of a common application and for promoting robustness ; and 5) to use generic software tools for rapid development of unconventional EA. This methodological approach is illustrated on four applications of EA to hard problems. Moreover, the fifth principle is explained in the study on genericity of EA software tools.

The application of EA to complex problems requires the use of generic software tool, for which we propose six genericity criteria. Many EA software tools are available in the community, but only a few are really generic. Indeed, an evaluation of some popular tools tells us that only three respect all these criteria, of which the framework Open BEAGLE, developed during the Ph.D. Open BEAGLE is organized into three main software layers. The basic layer is made of the object oriented foundations, over which there is the generic framework layer, consisting of the general mechanisms of the tool, and then the final layer, containing several specialized frameworks implementing different EA flavors. The tool also includes two extensions, respectively to distribute the computations over many computers and to visualize results.

Three applications illustrate different approaches for using EA in the context of pattern recognition. First, nearest neighbor classifiers are optimized, with the prototype selection using a genetic algorithm simultaneously to the Genetic Programming (GP) of neighborhood metrics. We add to this cooperative two species co-evolution a third co-evolving competitive species for selecting test data in order to improve the generalization

capability of solutions. A second application consists in designing representations with GP for handwritten character recognition. This evolutionary engineering is conducted with an automatic positioning of regions in a window of attention, combined with the selection of fuzzy sets for feature extraction. This application is used to automate character representation search, which is usually conducted by human experts with a trial and error process. For the third application in pattern recognition, we propose an extensible system for the hierarchical combination of classifiers into a fuzzy decision tree. In this system, the tree topology is evolved with GP while the numerical parameters of classification units are determined by specialized learning techniques. The system is tested with three simple types of classification units. All of these applications in pattern recognition have been implemented using a two-objective fitness measure in order to minimize classification errors and solutions complexity.

The last application demonstrate the efficiency of EA for lens system design. Self-adaptative evolution strategies, hybridized with a specialized local optimisation technique, are used to solve two complex optical design problems. In both cases, the experiments demonstrate that hybridized EA are able to produce results that are comparable or better than those obtained by human experts. These results are encouraging from the standpoint of a fully automated optical design process. An additional experiment is also conducted with a two-objectives fitness measure that tries to maximize image quality while minimizing lens system cost.

Avant-propos

Cette thèse porte sur des travaux de recherche dont l'origine est antérieure à mon passage au doctorat en septembre 2001. En effet, j'ai débuté ma carrière en recherche lors d'un stage au Laboratoire de vision et systèmes numériques (LVSN) à l'été 1998. C'est à ce moment que le professeur Marc Parizeau, qui fut mon directeur durant toutes mes études graduées, m'a orienté vers le monde passionnant des algorithmes évolutionnaires. Il faut souligner l'appui remarquable qu'il m'a consacré durant ces sept années, la porte de son bureau étant toujours grande ouverte pour me recevoir. Il faut également mentionner le temps et l'énergie qu'il a consacré pour la révision des articles scientifiques qui sont à la base du présent document. J'estime que Marc Parizeau est l'une des quelques personnes qui ont eu une influence déterminante dans ma vie.

Durant toutes ces années, j'ai grandement apprécié la collaboration et les nombreux échanges avec plusieurs étudiants du LVSN : Marc Dubreuil, Alexandre Lemieux, Nicolas Dubé, Patrick-Emmanuel Boulanger-Nadeau, Jean-François Dupuis et Jean-François Hébert. Également, il faut souligner la collaboration spéciale de Simon Thibault de l'Institut national d'optique, qui nous a fait gracieusement bénéficier de son expertise en conception optique pour valider les résultats du chapitre 6 et nous fournir un problème pour une compétition homme-machine. Je tiens aussi à souligner la collaboration spéciale du professeur Denis Laurendeau, qui a accepté d'effectuer une pré-lecture accélérée de ma thèse.

Cette thèse serait toute autre sans l'aide précieuse apportée par ma conjointe Julie Beaulieu. En effet, en plus d'être à l'origine de l'application en conception optique (chapitre 6), elle m'a fait bénéficier gracieusement de ses services de traduction scientifique, par la traduction de plusieurs chapitres ainsi que la révision méticuleuse de la thèse. Une bonne partie du mérite lui revient quant à la qualité linguistique du document. Il faut de plus souligner son soutien moral et sa patience durant toutes mes études universitaires. Pour toutes ces raisons et plus encore, je lui dédie affectueusement cette thèse.

J'ai bénéficié de différentes bourses durant mes études graduées. Je remercie le Conseil de recherche en sciences naturelles et en génie du Canada (CRSNG) ainsi que le Fonds québécois de la recherche sur la nature et les technologies (FQRNT) pour leur soutien financier.

À Julie, mon amour, qui ensoleille ma vie...

Table des matières

Résumé **ii**

Abstract **iv**

Avant-propos **vi**

Table des matières **ix**

Liste des tableaux **xiii**

Table des figures **xv**

Glossaire **xviii**

1 Introduction **1**

1.1 Systèmes intelligents 2

1.1.1 Intelligence artificielle 2

1.1.2 Intelligence computationnelle 3

1.1.3 Apprentissage automatique 3

1.1.4 Reconnaissance des formes 3

1.2 Algorithmes évolutionnaires 5

1.2.1 Algorithmes génétiques 5

1.2.2 Programmation génétique 7

1.2.3 Stratégies d'évolution 7

1.2.4 Programmation évolutionnaire 8

1.2.5 Optimisation multi-objectif 8

1.2.6 Algorithmes mémétiques 9

1.2.7 Co-évolution 10

1.3 Objectifs de recherche 11

1.4 Approche méthodologique 13

1.4.1 Algorithmes et représentations adaptés au problème 13

1.4.2 Hybrides entre des AE et des heuristiques du domaine d'application 14

1.4.3 Optimisation évolutionnaire à plusieurs objectifs 15

1.4.4 Co-évolution de composants et de cas de test 15
1.4.5 Développer à l'aide d'outils logiciels génériques d'AE 16
1.5 Plan de la thèse 17

2 Outils logiciels d'algorithmes évolutionnaires 20
2.1 Outils logiciels génériques d'algorithmes évolutionnaires 21
2.1.1 Revue de littérature 22
2.1.2 Framework d'algorithmes évolutionnaires 23
2.1.3 Progression d'une framework d'algorithmes évolutionnaires 24
2.1.4 Critères de généricité 25
2.1.5 Analyse de la généricité d'outils logiciels 28
2.2 Open BEAGLE : framework générique d'algorithmes évolutionnaires . . . 31
2.2.1 Architecture d'Open BEAGLE 32
2.2.2 Fondations orientées objet 32
2.2.3 Framework générique d'algorithmes évolutionnaires 33
2.2.4 Frameworks spécialisées 38
2.2.5 Exemple de code : OneMax 41
2.3 Distributed BEAGLE : distribution des calculs évolutionnaires 42
2.4 BEAGLE Visualizer : interface Web d'analyse de résultats 46
2.5 Conclusion 49

3 Co-évolution de classifieurs basés sur la règle du plus proche voisin 50
3.1 Classement par la règle du plus proche voisin 51
3.2 Ensembles de données synthétiques 53
3.3 Sélection de prototypes avec un algorithme génétique multi-objectif . . . 56
3.4 Évolution des mesures de voisinage avec la programmation génétique . . 62
3.5 Co-évolution de classifieurs 66
3.6 Résultats sur des données réelles 70
3.7 Discussion 74
3.8 Conclusion 74

4 Évolution de représentations de caractères cursifs 76
4.1 Représentations floues régionales 77
4.1.1 Extraction de caractéristiques 77
4.1.2 Topologies de grille 79
4.1.3 Résultats expérimentaux 80
4.2 Représentations guidées par les données 82
4.2.1 Résultats expérimentaux 83
4.3 Ingénierie de représentations par programmation génétique 84
4.3.1 Évolution des représentations 85
4.3.2 Résultats expérimentaux 88

4.4 Conclusion 94

5 Arbres de décision flous, extensibles et évolutionnaires 97
5.1 Principes et motivations 98
5.2 Revue de littérature 100
5.3 Système EFFECT 101
5.3.1 Combinaison floue et entraînement des unités 102
5.3.2 Unité ADALINE 105
5.3.3 Unité plus proche voisin 110
5.3.4 Unité à K-moyennes floues 110
5.3.5 Autres unités 113
5.3.6 Évolution de topologies d'arbres 116
5.4 Résultats expérimentaux 118
5.4.1 Résultats avec le système EFFECT 121
5.5 Gel des paramètres appris 127
5.5.1 Résultats avec gel des paramètres 128
5.6 Conclusion 129

6 Conception automatisée de systèmes de lentilles 132
6.1 Stratégies d'évolution 133
6.2 Conception de systèmes de lentilles 134
6.3 Optimisation de systèmes de lentilles 137
6.4 Problème du *monochromatic quartet* 139
6.5 Conception de systèmes de lentilles par algorithmes évolutionnaires . . . 142
6.6 Problème du système d'imagerie 146
6.7 Optimisation multi-objectif 152
6.8 Conclusion 156

7 Conclusion 160
7.1 Contributions au développement logiciel 161
7.2 Contributions applicatives 162
7.2.1 Application : co-évolution de classifieurs 162
7.2.2 Application : ingénierie évolutionnaire de représentations de l'écriture manuscrite 163
7.2.3 Application : évolution d'arbres de décision flous 165
7.2.4 Application : conception automatisée de systèmes de lentilles . . . 166
7.2.5 Technique : minimisation de la complexité des modèles évolués par programmation génétique 166
7.2.6 Technique : utilisation d'un ensemble de validation distinct pour la sélection de la meilleure solution d'une évolution 167

7.2.7 Technique : primitives de programmation génétique avec valeurs générées aléatoirement 168
7.2.8 Technique : stratégies d'évolutions multi-objectif avec NSGA-II . 168
7.3 Contributions méthodologiques 169
7.4 Perspectives 171

A Liste des publications **173**

Bibliographie **175**

Index **188**

Liste des tableaux

2.1 Évaluation de la généricité d'outils logiciels d'AE. 28

3.1 Performances obtenues pour les ensembles de données synthétiques avec des classifieurs 1-PPV classiques. 55
3.2 Résultats de l'édition de Wilson et de la condensation de Hart sur les ensembles de données synthétiques. 57
3.3 Résultats de la sélection de prototypes avec un AG multi-objectif sur les ensembles de données synthétiques. 59
3.4 Ensemble des primitives utilisées pour les évolutions des mesures de voisinage avec la PG. 64
3.5 Résultats de l'évolution de mesures de voisinage obtenues avec la PG sur les ensembles de données synthétiques. 66
3.6 Résultats de la co-évolution de la sélection de prototypes et de la mesure de voisinage sur les ensembles de données synthétiques. 68
3.7 Résultats de la co-évolution de trois espèces sur les ensembles de données synthétiques. 69
3.8 Description des ensembles de données choisis du *UCI Machine Learning Repository*. 71
3.9 Performances obtenues pour les ensembles de données UCI des classifieurs 1-PPV classiques. 71
3.10 Résultats de l'édition de Wilson et de la condensation de Hart sur les ensembles de données UCI. 72
3.11 Résultats de la co-évolution de la sélection de prototypes et de la mesure de voisinage sur les ensembles de données UCI. 72
3.12 Résultats de la co-évolution de trois espèces sur les ensembles de données UCI. 73

4.1 Taux de reconnaissance pour différentes topologies de grille de représentations floues régionales. 81
4.2 Taux de reconnaissance pour différentes topologies de grille de représentations guidées par les données. 83
4.3 Ensemble des primitives utilisées pour les évolution de PG. 86

4.4 Taux de reconnaissance obtenus pour les meilleures représentations des quatre évolutions distinctes de PG. 89

5.1 Ensemble des primitives de PG du système EFFECT. 117
5.2 Description des ensembles de données utilisés pour évaluer et comparer la performance du système EFFECT. 119
5.3 Performances de différents classifieurs standards. 120
5.4 Performances du système EFFECT. 123
5.5 Performances du système EFFECT avec gel des paramètres appris. . . . 128

6.1 Nombre de paramètres et transformations de valeurs pour le *monochromatic quartet*. 143
6.2 Contraintes physiques du *monochromatic quartet*. 143
6.3 Spécifications pour le problème du système d'imagerie. 146
6.4 Paramètres à optimiser et transformation des valeurs pour le problème du système d'imagerie. 148
6.5 Contraintes physiques pour le problème du système d'imagerie. 149
6.6 Liste des coûts relatifs, en unités par gramme, pour des verres courants utilisés dans la fabrication de lentilles. 153

Table des figures

1.1 Système de reconnaissance des formes. 4
1.2 Processus évolutionnaire classique des algorithmes génétiques. 6

2.1 Architecture de la framework Open BEAGLE. 32
2.2 Framework générique d'AE. 34
2.3 Configuration en XML d'un évoluateur générationnel d'AG à chaîne de bits avec Open BEAGLE. 36
2.4 Configuration en XML d'un évoluateur *steady-state* d'AG à chaîne de bits avec Open BEAGLE. 39
2.5 Relation entre l'ensemble de primitives et les arbres de PG. 40
2.6 Opérateur d'évaluation de l'adéquation pour le problème *OneMax*. 42
2.7 Routine principale pour le problème *OneMax*. 43
2.8 Architecture du deuxième prototype de Distributed BEAGLE 45
2.9 Capture d'écran d'un rapport BEAGLE Visualizer pour le problème *OneMax*. 47
2.10 Capture d'écran d'un rapport BEAGLE Visualizer pour le problème de la régression symbolique. 48

3.1 Ensembles de données synthétiques. 54
3.2 Algorithme d'édition de Wilson pour la sélection de prototypes. 56
3.3 Algorithme de condensation de Hart pour la sélection de prototypes. . . . 56
3.4 Fronts de Pareto du meilleur des 10 essais. 61
3.5 Utilisation de mesures de voisinage adaptées au problème à résoudre. . . 62
3.6 Espace des caractéristiques d'entrée partitionné par les meilleures mesures de voisinage obtenues avec la PG. 67

4.1 Trois caractères cursifs d'écriture en ligne. 77
4.2 Ensembles flous de la représentation floue régionale originale. 78
4.3 Différentes topologies de grille. 79
4.4 Topologie de grille guidée par les données en arbre quin à 2 niveaux (DDR-quin2) pour différents chiffres manuscrits 83
4.5 Gain moyen du taux de reconnaissance par rapport à la RFR-3x2 de base pour les RFR et DDR testées. 84

4.6 Forme de l'ensemble flou utilisé pour extraire les caractéristiques d'orientation et de courbure. 87
4.7 Fronts de Pareto pour les quatre évolutions de représentations de caractères cursifs. 91
4.8 S-expression de l'arbre Evol1. 92
4.9 Structure d'arbre de la représentation Evol1. 93
4.10 Caractères de DevTest-R02/V02 mal classés par au moins trois des quatre représentations évoluées. 96

5.1 Variables générales utilisées dans la description des algorithmes du système. 101
5.2 Fonction récursive de l'algorithme de classement général. 103
5.3 Illustration du classement dans les arbres de décision flous. 104
5.4 Fonction récursive de l'algorithme d'entraînement général. 106
5.5 Algorithme de ré-étiquetage en deux classes. 107
5.6 Algorithme de classement des unités ADALINE. 108
5.7 Algorithme d'entraînement des unités ADALINE. 109
5.8 Algorithme de classement des unités PPV. 111
5.9 Algorithme d'entraînement des unités PPV. 112
5.10 Algorithme de classement des unités à K-moyennes floues. 113
5.11 Algorithme d'entraînement des unités à K-moyennes floues. 114
5.12 Tracé de la fonction d'amplification des degrés d'appartenance flous. . . . 115
5.13 Tracé de la fonction de clarification de degrés d'appartenance flous. . . . 117
5.14 Fronts de Pareto de quatre évolutions d'arbres de décision flous du système EFFECT. 125
5.15 Performance du système EFFECT sur les données tae avec différentes tailles de l'ensemble de validation. 126
5.16 Comparaison des taux de reconnaissance pour des algorithmes testés. . . 129

6.1 Paramètres d'un système de lentilles. 135
6.2 La première loi de la réfraction de Snell-Descartes. 136
6.3 Deux exemples d'aberration de Seidel. 137
6.4 Mesure de la distorsion avec le centroïde du *blur spot*. 140
6.5 Le meilleur système de lentilles de chacune des deux classes de solutions, présentés par des experts humains à la ILDC 1990. 141
6.6 Meilleur système de lentilles obtenu avec la SA-ES anisotrope $(\mu + \lambda)$ pour le problème du *monochromatic quartet*. 145
6.7 Meilleur système de lentilles obtenu avec la CMA-ES pour le problème du *monochromatic quartet*. 145
6.8 Solution présentée par les experts de l'INO pour le problème du système d'imagerie. 147

6.9 Algorithme de mutation et de croisement utilisé pour générer la nouvelle population dans les expérimentations avec la SA-ES anisotrope $(\mu + \lambda)$ pour le problème du système d'imagerie. 150
6.10 Meilleur système de lentilles obtenu avec la SA-ES anisotrope $(\mu + \lambda)$ pour le problème du système d'imagerie. 150
6.11 Meilleur système de lentilles obtenu avec la CMA-ES pour le problème du système d'imagerie. 152
6.12 Front de Pareto de l'évolution ayant produit la meilleure solution. 155
6.13 Meilleur système de lentilles obtenu en utilisant la SA-ES anisotrope avec NSGA-II pour le problème du système d'imagerie. 155

Glossaire

Le présent document porte sur des sujets pointus de différents domaines de l'informatique et de l'ingénierie des systèmes intelligents. Voici donc un glossaire qui devrait alléger le texte et le rendre accessible aux lecteurs non-initiés.

Adéquation Aptitude d'un individu à résoudre le problème posé.

Algorithmes évolutionnaires (AE) (*evolutionary computation* ou *evolutionary algorithms*) Discipline de l'intelligence computationnelle comprenant l'ensemble des méthodes simulant l'évolution naturelle, habituellement sur des ordinateurs. Ces méthodes utilisent généralement une population à laquelle est appliquée des opérations de modifications aléatoires et de sélection naturelle. Les algorithmes évolutionnaires existent en quatre saveurs : les algorithmes génétiques, la programmation génétique, la stratégie d'évolution et la programmation évolutionnaire. (Adapté de [3].)

Algorithmes génétiques (AG) Algorithmes évolutionnaires développés par John H. Holland et ses étudiants dans les années 1960. La formulation originale des AG représente les solution par des chaînes de bits. De nouvelles représentations telles que les AG de nombres réels ont été développées depuis. Les algorithmes génétiques consistent à faire évoluer une population d'individus en se basant essentiellement sur des opérateurs de recombinaison (croisement) et de sélection naturelle, et plus secondairement des opérateurs de mutation. (Adapté de [3].)

Algorithmes mémétiques (*memetic algorithms*) Combinaison d'algorithmes évolutionnaires avec des heuristiques de recherche locale adaptées au problème à résoudre.

Apprentissage automatique (*machine learning*) Sous-domaine des systèmes intelligents visant à développer des programmes informatiques qui apprennent automatiquement avec l'expérience. (Adapté de [98].)

Croisement Opération génétique consistant en un échange de matériel génétique entre des individus.

Co-évolution Un système avec co-évolution comprend différentes populations qui interagissent de sorte que la fonction d'évaluation de l'adéquation d'une population

peut dépendre de l'état du processus évolutionnaire d'autres populations. (Adapté de [3].)

Dème Sous-population indépendante dans les processus évolutifs avec migration des algorithmes évolutionnaires parallèles. (Adapté de [3].)

Design pattern Formalisme désignant, justifiant et expliquant systématiquement un patron général pour résoudre un problème de conception récurrent en programmation orientée objet. Un design pattern décrit le problème, la solution, le contexte d'application de la solution et ses conséquences. Il donne également des trucs et des exemples sur son implantation. La solution est un arrangement général d'objets, de classes résolvant le problème. La solution est personnalisée et implémentée dans le contexte particulier du problème à résoudre. (Adapté de [52].)

Framework Ensemble de composants formant un design réutilisable pour un type particulier de logiciel. Une framework donne des spécifications architecturales en partitionnant le design en classes abstraites et en définissant les responsabilités et collaborations entre celles-ci. Un développeur spécialise une framework pour ses besoins particuliers en utilisant les classes de la framework comme base à ses propres instances. (Adapté de [52].)

Génotype Représentation de base des individus nécessitant souvent une transformation pour exprimer une solution valide (le phénotype).

Individu Membre unique de la population représentant une solution possible au problème à résoudre (un point dans l'espace de recherche). Un individu contient généralement d'autres informations telles que l'adéquation de la solution au problème à résoudre. (Adapté de [3].)

Intelligence artificielle Sous-domaine des systèmes intelligents qui vise à simuler les facultés de compréhension, de raisonnement et de déduction de l'homme.

Intelligence computationnelle (*computational intelligence*) Domaine des systèmes intelligents qui englobe un ensemble de techniques qui, à l'opposé des techniques classiques d'intelligence artificielle, traitent de l'information de bas niveau vers des niveaux d'abstraction supérieurs. Le terme intelligence computationnelle englobe essentiellement trois techniques importantes des systèmes intelligents : les réseaux de neurones, les algorithmes évolutionnaires et les systèmes flous. (Adapté de [32].)

Mixture d'experts (*mixture of experts*) Ensemble de classifieurs où les classifieurs composants sont hautement spécialisés sur différentes aspects du problème, souvent une région de l'espace des caractéristiques. (Adapté de [41].)

Mutation Opération génétique consistant en une modification aléatoire d'un individu.

Opération génétique Opération qui modifie la structure des individus dans les algorithmes évolutionnaires. Deux types d'opérations génétiques sont couramment employés dans les algorithmes évolutionnaires : le croisement et la mutation.

Phénotype Dans certains algorithmes évolutionnaires, résultat de la transformation de la représentation de base d'un individu (le génotype) en une solution valide au problème posé. Le phénotype exprime une solution valide qui est généralement utilisée pour déterminer l'adéquation de l'individu associé.

Polymorphisme Habileté de substituer des objets logiciels par d'autres ayant une interface d'utilisation compatible. (Adapté de [52].)

Population Dans les algorithmes évolutionnaires, groupe d'individus qui interagissent entre eux, par exemple, par recombinaison (croisements) et reproduction. (Adapté de [3].)

Programmation évolutionnaire (PE) (*evolutionary programming*) Algorithme évolutionnaire initialement développé dans le but de faire évoluer des machines à états finis et par la suite étendu aux problèmes d'optimisation de paramètres. La programmation évolutionnaire est basée sur des opérateurs de variation (mutation) adaptés au problème. Le paradigme utilise également des opérations de sélection naturelle par tournois où μ individus survivent parmi les μ parents et μ descendants générés par mutation. (Adapté de [3].)

Programmation génétique (PG) Paradigme dérivé des algorithmes génétiques consistant à utiliser des algorithmes évolutionnaires pour programmer des ordinateurs automatiquement. À cette fin, chaque individus de la population représente un programme informatique complet dans un langage donné. La représentation la plus courante utilise des arbres dans le langage LISP. D'autres représentations (incluant des chaînes de bits) et d'autres langages de programmation (incluant le code machine) ont été fructueusement utilisés pour faire de la programmation génétique. (Adapté de [3].)

Reconnaissance des formes (*pattern recognition*) Discipline consistant à reproduire dans des systèmes intelligents les capacités perceptuelles de l'homme. La reproduction de ces capacités perceptuelles s'effectue généralement par le classement automatique d'objets observés selon des patrons reconnus dans les données brutes provenant de capteurs.

Réseau de neurones Domaine de l'intelligence computationnelle utilisant le cerveau comme modèle pour résoudre des problèmes. (Adapté de [32].)

Sélection Opérateur d'algorithmes évolutionnaires basé sur les principes de la sélection naturelle. Cet opérateur est utilisé pour diriger le processus évolutif dans les régions de l'espace de recherche les plus intéressantes en favorisant la survivance des individus ayant une adéquation élevée. (Adapté de [3].)

Stratégie d'évolution (SE) (*evolution strategy*) Algorithme évolutionnaire développé par I. Rechenberg and H.-P. Schwefel dans les années 1960. La stratégie d'évolution utilise typiquement des paramètres de nombres réels, bien que la stratégie d'évolution a déjà été appliquée à des problèmes discrets. La stratégie d'évolution

se caractérise par la distinction entre une population de parents (de dimension μ) et une population de descendants (de taille $\lambda \geq \mu$), l'emphase sur des mutations suivant une distribution normale, l'utilisation de différentes formes de croisement et l'incorporation du principe d'auto-adaptation des paramètres de la stratégie. Ces paramètres spécifient la fonction de densité de probabilité de mutation, et évoluent en-ligne selon les mêmes principes utilisés pour faire évoluer les paramètres des solutions potentielles. (Adapté de [3].)

Système flou Domaine de l'intelligence computationnelle qui, au lieu d'utiliser la logique Booléenne, utilise des notions de presque vrai et presque faux pour résoudre des problèmes. (Adapté de [32].)

Système intelligent Discipline visant le développement de systèmes exhibant un comportement dit intelligent. (Adapté de [32].)

Chapitre 1

Introduction

> *Instead of trying to produce a programme to simulate the adult mind, why not rather try to produce one which simulates the child's ? [...] We cannot expect to find a good child-machine at the first attempt. One must experiment with teaching one such machine and see how well it learns. One can then try another and see if it is better or worse. There is an obvious connection between this process and evolution, by the identifications*
>
> *Structure of the child machine = Hereditary material*
> *Changes of the child machine = Mutations*
> *Natural selection = Judgment of the experimenter*
>
> Alan M. Turing, *Computing Machinery and Intelligence* [145], 1950.

Les formes d'intelligence les plus développées sur la Terre se retrouvent chez les être vivants. En effet, malgré tous les espoirs qui ont été placés dans l'informatique depuis les débuts, dans les années 1940, le développement d'une machine dotée d'une intelligence s'approchant du niveau de celle des animaux vertébrés relève encore de la science fiction. Dans ces circonstances, ne serait-il pas possible d'utiliser le mécanisme qui est à la base du développement de l'intelligence sur Terre, soit l'évolution naturelle, afin de doter les machines d'une plus grande intelligence ? L'idée n'est pas nouvelle et a déjà été abordée par plusieurs, dont le pionnier de l'informatique Alan M. Turing [68, 79, 145]. Dans les années 1960 et 1970, la simulation de l'évolution naturelle sur des ordinateurs a été indépendamment développée avec les algorithmes génétiques [66], la programmation évolutionnaire [46] et les stratégies d'évolution [125]. Ces différentes techniques ont grandement évolué depuis leurs débuts et ont été réunies sous une même appellation, les *algorithmes évolutionnaires* (AE), qui rassemblent tous les algorithmes

inspirés de l'évolution naturelle [3, 5].

La présente thèse porte sur l'utilisation des AE pour l'ingénierie de systèmes intelligents. Plus précisément, elle est divisée selon trois applications en reconnaissance des formes et une application en conception automatisée de systèmes de lentilles. Ces quatre applications reposent toutes sur une approche similaire basée sur des techniques modernes d'AE. De plus, le développement d'applications des AE requiert l'utilisation d'outils logiciels flexibles et extensibles. À cet effet, nous proposons des principes de développement d'outils logiciels d'AE génériques et nous présentons l'outil Open BEAGLE développé durant le projet de recherche.

À la section 1.1, nous présentons plus amplement les systèmes intelligents avec une emphase particulière sur les domaines abordés dans la thèse. À la section 1.2, nous introduisons les quatre saveurs principales d'AE ainsi que trois techniques modernes d'AE particulièrement intéressantes dans un contexte applicatif. Nous proposons ensuite, à la section 1.4, une approche méthodologique qui est à la base des applications de la thèse. Finalement, nous présentons l'organisation générale du document à la section 1.5.

1.1 Systèmes intelligents

Le domaine des systèmes intelligents englobe les disciplines visant le développement de systèmes exhibant un comportement dit intelligent. Nous présentons brièvement dans les sections qui suivent quelques domaines et sous-domaines composant les systèmes intelligents ainsi que les relations entre eux, tout en insistant sur l'un des domaines principaux d'intérêt du projet, la reconnaissance des formes.

1.1.1 Intelligence artificielle

Dans l'esprit populaire, le terme intelligence artificielle est souvent synonyme de systèmes intelligents. Cependant, le sens de cette expression a dérivé depuis les années 1960 pour maintenant désigner un sous-domaine des systèmes intelligents simulant les facultés de compréhension, de raisonnement et de déduction de l'homme à un niveau d'abstraction relativement élevé. Cette simulation des capacités cognitives humaines s'effectue généralement à l'aide de techniques telles que la logique propositionnelle.

1.1.2 Intelligence computationnelle

L'intelligence computationnelle est un domaine des systèmes intelligents englobant un ensemble de techniques qui, à l'opposé des techniques classiques d'intelligence artificielle, traitent de l'information de bas niveau vers des niveaux d'abstraction supérieurs [32]. Le terme intelligence computationnelle comprend essentiellement trois techniques importantes des systèmes intelligents : les réseaux de neurones, les algorithmes évolutionnaires et les systèmes flous. L'intelligence computationnelle est également connue sous l'expression *soft computing*, par opposition à la recherche opérationnelle, parfois désignée sous l'appellation *hard computing*. Ces deux disciplines partagent les mêmes domaines applicatifs mais diffèrent du fait que les techniques utilisées en recherche opérationnelle comportent des conditions strictes sur le domaine d'application, mais assurent la découverte d'une solution satisfaisante au problème (souvent une solution optimale). À l'opposé, les méthodes d'intelligence computationnelle ne posent pas de condition d'applicabilité, mais ne comportent également aucune garantie sur la qualité des solutions trouvées. Cette déficience est compensée par la grande robustesse des méthodes employées pour la recherche de solutions.

1.1.3 Apprentissage automatique

L'apprentissage automatique est un sous-domaine des systèmes intelligents visant à développer des programmes informatiques qui apprennent automatiquement avec l'expérience [98]. L'apprentissage automatique désigne concrètement un vaste ensemble de techniques telles que les arbres de décision et la programmation logique inductive. Les techniques d'intelligence computationnelle sont également utilisées pour effectuer de l'apprentissage automatique. Cette discipline s'avère fructueuse pour des applications dans des domaines tels que le forage de données (*data mining*) et l'aide à la décision.

1.1.4 Reconnaissance des formes

La reconnaissance des formes est une discipline consistant à reproduire les capacités perceptuelles de l'homme, en catégorisant un ensemble de données brutes selon le patron observé. La reconnaissance des formes ne se limite pas seulement à la perception visuelle, mais inclut également tous les types de perception artificielle exigeant de reconnaître des patrons de données[1]. La reconnaissance des formes partage de nombreuses tech-

[1] En ce sens, le terme anglais *pattern recognition* décrit plus précisément le domaine.

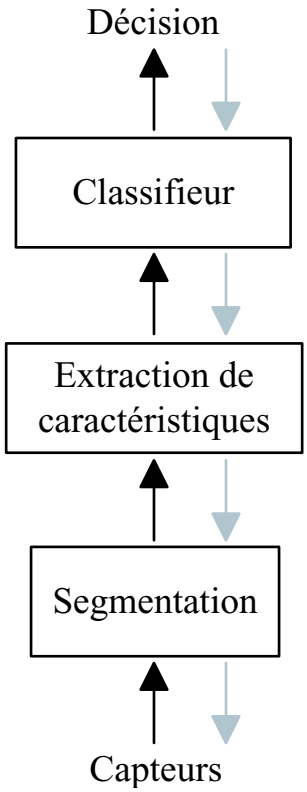


FIG. 1.1 – Système de reconnaissance des formes.

niques avec l'apprentissage automatique. Les différences entre ces deux disciplines sont essentiellement historiques et culturelles. Le domaine de la reconnaissance des formes est apparu à la fin des années 1960 et consiste à développer des systèmes intelligents dotés de capacités perceptuelles, traitant souvent des données numériques. D'autre part, l'apprentissage automatique est un domaine plus récent (années 1980) dont l'objectif est de développer des systèmes apprenant de façon autonome et traitant des données souvent discrètes. On peut donc affirmer que la reconnaissance des formes effectue généralement un traitement de plus bas niveau que l'apprentissage automatique, quoiqu'une distinction nette entre les deux domaines reste difficile à faire.

Typiquement, la plupart des systèmes de reconnaissance des formes peuvent être séparés en trois modules distincts, tel que présenté à la figure 1.1 (inspiré de [41]). Dans ce système, le module de segmentation transforme le signal des capteurs perceptuels, souvent des images et/ou du son, en regroupant les éléments du signal relatifs aux objets d'intérêt et en les isolant du signal d'arrière-plan. Le module d'extraction de caractéristiques (*feature extraction*) vise à mesurer des propriétés utiles pour la reconnaissance, qui proviennent des objets fournis par la segmentation. Finalement, le classifieur permet de catégoriser les objets extraits en les associant à une classe donnée. Le flot principal

des données du schéma va de bas en haut, mais il peut également aller de haut en bas, ce qui symbolise la rétroaction souvent présente entre les modules de tels systèmes.

1.2 Algorithmes évolutionnaires

Les AE constituent une discipline impliquant la simulation du processus de l'évolution naturelle sur un ordinateur [3]. Ceci peut être vu comme un processus d'optimisation où des individus évoluent dans le temps, afin de devenir de plus en plus adéquats à un environnement donné, le problème à résoudre. Les AE ont été appliqués avec succès à de nombreux problèmes où les algorithmes classiques d'optimisation, d'apprentissage et de conception automatisée sont incapables de produire des résultats satisfaisants. En général, les AE sont divisés en quatre saveurs principales, présentées dans cette section : les algorithmes génétiques, la programmation génétique, les stratégies d'évolution et la programmation évolutionnaire. De plus, nous présentons dans cette section trois techniques avancées d'AE : l'optimisation multi-objectif, les algorithmes mémétiques et la co-évolution.

1.2.1 Algorithmes génétiques

Les algorithmes génétiques (AG) [66, 97] impliquent l'évolution d'une population de vecteurs de dimension fixe composés de caractères, généralement des bits. L'idée des AG est vaguement inspirée des chaînes d'ADN, qui composent tout organisme vivant. Les AG sont utilisés dans le but de découvrir une solution, généralement numérique, résolvant un problème donné, et ce sans avoir de connaissance *a priori* sur l'espace de recherche. Seul un critère de qualité est nécessaire pour discriminer différentes solutions. Dans la plupart des cas, ce critère, l'adéquation, est une mesure objective qui permet de quantifier la capacité de l'individu à résoudre le problème donné. Le processus de recherche s'effectue en appliquant itérativement à une population de solutions potentielles des opérations de variation génétique (généralement le croisement et la mutation), et des opérations de sélection naturelle biaisées vers les individus les plus forts. En utilisant ce processus Darwinien, la population de solutions potentielles évolue dans le temps jusqu'à ce qu'un certain critère d'arrêt soit atteint. La figure 1.2 présente plus en détails le processus évolutionnaire des AG. Dans un contexte d'optimisation de fonctions à paramètres réels, une variante importante des AG consiste à représenter les individus directement par des vecteurs de nombres réels.

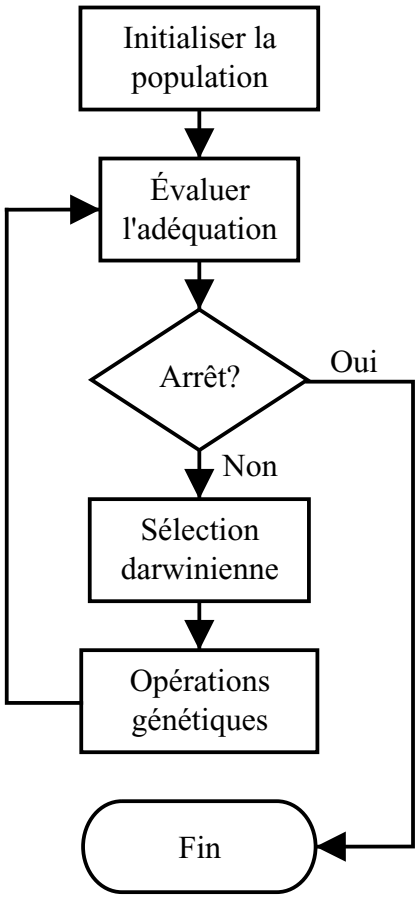


FIG. 1.2 – Processus évolutionnaire classique des algorithmes génétiques.

1.2.2 Programmation génétique

La programmation génétique (PG) [9, 76] est un paradigme permettant la programmation automatique d'ordinateurs par des heuristiques basées sur les mêmes principes d'évolution que les AG : des opérations de variation génétique, par des croisements et des mutations, et des opérations de sélection naturelle. La différence entre la PG et les AG réside essentiellement dans la représentation des individus. En effet, la PG consiste à faire évoluer des individus dont la structure est similaire à des programmes informatiques. La PG a été exprimée formellement par Koza au début des années 1990 [76, 77]. La PG utilisée par Koza représente les individus sous forme d'arbres, c'est-à-dire des graphes orientés et sans cycle, dans lesquels chaque nœud est associé à une opération élémentaire relative au domaine du problème. Plusieurs autres représentations comme des programmes linéaires [9] et des graphes cycliques [139], ont été utilisées depuis. La PG est particulièrement adaptée à l'évolution de structures complexes de dimensions variables.

1.2.3 Stratégies d'évolution

Les stratégies d'évolution (SE) constituent une technique d'AE développée par I. Rechenberg et H.-P. Schwefel à Berlin dans les années 1960 [3, 125]. Les SE représentent les individus comme un ensemble de caractéristiques de la solution potentielle. En général, cet ensemble prend la forme d'un vecteur de nombres réels de dimension fixe. Les SE s'appliquent à une population de parents (de dimension μ) à partir de laquelle des individus sont sélectionnés aléatoirement afin de générer une population de descendants (de dimension $\lambda \geq \mu$). Ceux-ci sont ensuite modifiés par des mutations qui consistent à ajouter une valeur générée aléatoirement selon une fonction de densité de probabilité paramétrée. Les paramètres de la fonction de densité de probabilité, nommés les *paramètres de la stratégie*, évoluent eux aussi dans le temps selon les mêmes principes que les paramètres caractérisant les individus. Pour former la nouvelle population, μ individus sont choisis (approche (μ, λ)) parmi les μ meilleurs individus des λ descendants, ou (approche $(\mu + \lambda)$) parmi les μ meilleurs individus des μ parents et λ descendants. Les SE modernes peuvent aussi intégrer une approche multi-échelle où des stratégies d'évolution sont imbriquées.

1.2.4 Programmation évolutionnaire

La programmation évolutionnaire (PE) a été développée par L.J. Fogel dans les années 1960 et reprise par D.B. Fogel et d'autres chercheurs dans les années 1990 [3, 46]. La PE a été initialement conçue dans le but de faire évoluer des machines à états finis et a été par la suite étendue aux problèmes d'optimisation de paramètres. Cette approche met l'emphase sur la relation entre les parents et leurs descendants plutôt que de simuler des opérateurs génétiques d'inspiration naturelle. Contrairement aux trois autres AE classiques, la PE n'utilise pas une représentation spécifique mais plutôt un modèle évolutionnaire de haut niveau, jumelé à une représentation et à un opérateur de mutation directement appropriés au problème à résoudre.

Pour effectuer de la PE, une population de μ solutions potentielles au problème est d'abord générée aléatoirement. Chaque individu i de la population produit λ descendants résultant de mutations. Une opération de sélection naturelle est alors appliquée afin de former une nouvelle population de μ individus. Le processus de mutation - sélection est répété itérativement jusqu'à ce qu'une solution acceptable soit trouvée.

1.2.5 Optimisation multi-objectif

De nombreux problèmes d'ingénierie peuvent être formulés comme des problèmes d'optimisation à plusieurs objectifs. Dans un contexte d'optimisation classique à un objectif, une solution est jugée meilleure qu'une autre en fonction de l'objectif unique. Pour le cas à plusieurs objectifs, cette comparaison valeur à valeur n'est pas possible. En effet, les différents objectifs d'un problème donné sont très souvent contradictoires ; l'amélioration selon l'un de ces objectifs est généralement faite au détriment des autres objectifs.

Une approche souvent employée consiste à transformer par une somme pondérée un problème à plusieurs objectifs en un problème à un seul objectif. Ceci permet une utilisation directe des algorithmes d'optimisation classiques basés sur une mesure unique de qualité. Cependant, la relation entre les différents objectifs à optimiser étant souvent non-linéaire, la somme pondérée biaise le processus d'optimisation vers certaines régions de l'espace de recherche. Une autre approche consiste à utiliser une notion de dominance appelée optimalité de Pareto. Selon ce principe, une solution domine une deuxième solution si elle est équivalente ou plus performante, pour chacun des objectifs, et si elle est plus performante pour au moins un objectif. L'optimisation multi-objectif basée sur l'optimalité de Pareto est particulièrement adaptée aux techniques de recherche

simultanée avec plusieurs solutions. C'est le cas des AE, où une population est formée de plusieurs solutions évoluant en parallèle.

Il existe de nombreux algorithmes de sélection naturelle pour AE basés sur le critère d'optimalité de Pareto [26]. À l'heure actuelle, l'algorithme le plus reconnu est certainement le *Non-Dominated Sorting Genetic Algorithm 2* (NSGA-II) de Deb et al. [37]. Cet algorithme consiste à générer une population d'enfants à partir de la population de parents, à fusionner ces deux populations pour appliquer une procédure de tri basée sur l'optimalité de Pareto. Le tri du NSGA-II classe les solutions de la population fusionnée selon les rangs de Pareto. Le premier rang de Pareto, aussi appelé front de Pareto, est composé de l'ensemble des solutions non-dominées de la population. Le deuxième rang comprend l'ensemble des solutions qui sont non-dominées lorsqu'on exclut les solutions du premier rang de la population, et ainsi de suite pour les autres rangs. La nouvelle population est générée en sélectionnant la moitié supérieure des solutions de la population de parents et d'enfants selon leur rang de Pareto. Pour le rang de Pareto à cheval sur la moitié supérieure et la moitié inférieure de la population fusionnée, une procédure de tri supplémentaire basée sur la notion de niche est utilisée. Ce deuxième tri permet de sélectionner le bon nombre de solutions pour former la nouvelle population.

Le résultat d'une évolution d'un AE dans un contexte multi-objectif est généralement donné par le front de Pareto. Ce front contient les solutions non-dominées de la dernière génération. Le choix de la meilleure solution de l'évolution dans le front de Pareto se fait selon des critères spécifiques au problème à résoudre. Le compromis entre les différents objectifs est donc reporté à la fin du processus d'optimisation, ce qui permet de faire un choix en toute connaissance de cause. Cette approche s'oppose à une optimisation à un objectif par sommes pondérées, où des poids arbitraires biaisant la recherche doivent être choisis au préalable.

1.2.6 Algorithmes mémétiques

Avec des techniques d'optimisation telles que les AE, il importe de considérer le dilemme *exploration versus exploitation*. Il s'agit de configurer un algorithme d'optimisation de manière à faire un bon compromis entre l'exploration de régions de l'espace de recherche qui n'ont pas été visitées, pour permettre la découverte de meilleures solutions, et l'exploitation des meilleures solutions connues, qui sont possiblement situées dans des régions riches en bonnes solutions.

À la lumière de ces considérations, il semble pertinent de développer des approches hybrides entre les AE et des heuristiques adaptées au problème. Les techniques basées

sur de tels hybrides sont souvent désignées sous l'appellation *algorithmes mémétiques* [101]. Du point de vue du dilemme exploration versus exploitation, les algorithmes mémétiques permettent de faire un compromis judicieux entre la capacité des AE à explorer l'espace de recherche et l'habileté des heuristiques adaptées au problème à raffiner les solutions de la population. En pratique, il s'avère que de telles approches performent plutôt bien, généralement mieux que l'utilisation seule de l'AE ou de l'heuristique adaptée au problème. Les algorithmes mémétiques ont été particulièrement développés dans le contexte de problèmes difficiles d'optimisation combinatoire comme le problème du voyageur de commerce.

Dans un cadre d'apprentissage automatique, les algorithmes mémétiques consistent souvent à hybrider les AE avec un algorithme d'apprentissage plus classique. De tels hybrides forment des systèmes comprenant d'une part une recherche évolutionnaire qui correspond à une recherche globale menant à l'obtention de caractères innés, et d'autre part un apprentissage qui correspond à une recherche locale menant à l'émergence de caractères acquis. Dans ce type de systèmes, une interaction positive peut être observée entre les modules d'évolution et d'apprentissage, où l'évolution génère des solutions comportant des caractères particulièrement favorables à l'apprentissage. Cette interaction peut aussi produire un *effet de Baldwin* [6, 63], où des caractéristiques initialement acquises par apprentissage se transforment en des caractéristiques innées au cours de l'évolution. En utilisant la terminologie de l'apprentissage automatique, ce phénomène consiste en un passage de biais faibles, où les classifieurs sont dotés d'une grande adaptabilité par apprentissage, vers des biais forts, où la configuration des classifieurs est plus rigide par l'inclusion d'une connaissance sur la structure des données [146]. Une autre avenue consiste à intégrer explicitement des caractères acquis par apprentissage dans les solutions évoluées par une opération génétique. Cette approche, biologiquement peu plausible, correspond aux théories de l'évolution de Lamarck.

1.2.7 Co-évolution

La co-évolution [4] consiste en plusieurs populations, souvent appelées espèces, qui évoluent simultanément avec une interaction entre elles de sorte que l'opération d'évaluation de l'adéquation d'une espèce dépend de l'état de l'évolution des autres espèces. Il existe deux types de co-évolution : 1) la co-évolution compétitive [62], où les espèces co-évoluées sont guidées par des objectifs opposés, la mesure d'adéquation des individus d'une espèce étant inversement proportionnelle à la performance des individus des autres espèces, et 2) la co-évolution coopérative [116], où les espèces évoluent ensemble afin de résoudre différents éléments d'un même problème.

La co-évolution compétitive peut être utile pour une évolution simultanée d'espèces dont l'adéquation des individus est inversement proportionnelle à celle des membres des autres espèces. On note deux formes principales de co-évolution compétitive : 1) une co-évolution compétitive où les individus de toutes les espèces sont des solutions à part entière qui s'affrontent, par exemple des solutions qui sont des adversaires d'un jeux donné ; 2) une co-évolution compétitive où une espèce consiste en des solutions au problème à résoudre et où une deuxième espèce consiste en des cas de test utilisés pour évaluer la qualité des solutions de la première espèce. La deuxième approche permet d'augmenter la robustesse des solutions tout en réduisant le nombre de cas de test nécessaires, et donc l'effort computationnel. Cependant, la co-évolution compétitive soulève d'autres problématiques telles que le désengagement, la sur-spécialisation et l'intransitivité [35]. Le désengagement (aussi appelé perte du gradient) survient lorsque les cas de tests deviennent trop faciles ou trop difficiles pour permettre une discrimination adéquate des solutions. La sur-spécialisation consiste en une amélioration des performances des solutions sur une minorité d'aspects du problème, sans progrès notable pour la majorité des autres aspects. Finalement, l'intransitivité est observée lorsque la relation de transitivité dans l'évaluation des solutions n'est pas respectée. Par exemple, si l'on suppose que la solution B est supérieure à A et que la solution C est supérieure à B, la transitivité est respectée lorsque la solution C est supérieure à A. Le processus de recherche darwinien peut mener à une impasse par un bouclage sur un ensemble de solutions où la transitivité n'est pas respectée.

La co-évolution coopérative est intéressante pour résoudre des problèmes formés de sous-problèmes distincts. Elle permet ainsi une décomposition du problème principal en plusieurs sous-problèmes afin de rendre la tâche de résolution plus facile. Cependant, la co-évolution coopérative soulève ses propres problématiques, essentiellement liées au choix de la stratégie d'appariement à utiliser entre individus d'espèces différentes pour l'évaluation de l'adéquation. Il n'existe pas de stratégie d'appariement universellement valide, le choix étant étroitement lié à la nature du problème. Ce choix comporte trois aspects principaux [155] : la méthode utilisée pour assigner l'adéquation aux individus (problème de l'assignation du crédit), la pression sélective à appliquer lors de la sélection des collaborateurs et le nombre d'individus par espèce à employer pour évaluer une adéquation.

1.3 Objectifs de recherche

La présentation sur les systèmes intelligents et les algorithmes évolutionnaires des sections précédentes ne se veut pas exhaustive. En effet, ces domaines sont très vastes

et trop complexes pour pouvoir être synthétisés adéquatement en quelques pages. De plus, il existe déjà des documents proposant des états de l'art généraux détaillés sur la reconnaissance des formes [41, 71], les AE [5, 49] et plus spécifiquement la PG [9, 87], les AG [96, 97] et les SE [2, 12]. Étant donné la diversité des sujets couverts dans la présente thèse, il a été opté de présenter au fur et à mesure des états de l'art plus spécifiques, selon la nature des travaux des différents chapitres.

Les thèse porte sur le développement d'infrastructures et d'expertises pour la résolution de problèmes difficiles par AE. Plus particulièrement, les objectifs de recherche du projet sont les suivants.

- **Développer des outils logiciels génériques pour le déploiement d'applications basées sur les AE**
  Le développement d'une framework logicielle générique et versatile est essentiel à la réalisation technique du projet. En effet, le projet implique l'expérimentation avec différents AE classiques pour la résolution de problèmes difficiles, ainsi que le développement de nouveaux algorithmes adaptés au problème, inspirés d'AE classiques.

- **Appliquer les AE, en particulier la PG, à la résolution de problèmes difficiles de reconnaissance des formes**
  Les AE ont un potentiel remarquable pour résoudre une panoplie de problèmes difficiles dans des domaines tels que l'optimisation, l'apprentissage automatique et la conception automatisée. En effet, les AE constituent un judicieux compromis entre la recherche guidée par une fonction objectif, le hasard et la force brute. La PG possède en plus la spécificité de pouvoir aisément créer et optimiser non seulement des valeurs numériques, mais également des structures telles que des programmes informatiques. En fait, la PG est dotée de capacités de création darwinienne exceptionnelles [78] idéales pour la résolution de problèmes difficiles dans un cadre d'ingénierie. De plus, plusieurs projets de recherche au Laboratoire de vision et systèmes numériques portent sur des aspects de la reconnaissance des formes. Ces projets de recherche comportent des problèmes difficiles d'apprentissage et d'optimisation, qui pourraient grandement bénéficier des caractéristiques exceptionnelles des AE. En ce sens, le présent doctorat s'inscrit dans la lignée des projets en reconnaissance des formes en général, avec un accent particulier sur la reconnaissance de l'écriture cursive.

- **Développer une méthodologie pour la réalisation de systèmes intelligents basés sur les AE**
  Le projet de recherche comporte un aspect applicatif important avec le dévelop-

pement d'applications basées sur les AE. Il s'avère important de proposer une méthodologie générale qui fait une synthèse de l'expertise acquise durant le doctorat. Malheureusement, les contributions de cette nature sont souvent négligées alors qu'elles s'avèrent d'une grande importance pour l'ingénierie de systèmes intelligents dans un contexte réaliste.

La thèse est structurée selon quatre grandes applications des AE à des problèmes difficiles, en plus de comporter une étude sur les outils logiciels génériques. En plus des principaux objectifs de recherche, la thèse comporte de nombreux objectifs plus spécifiques à chacune des applications. En effet, le projet de doctorat comporte différentes applications qui ont chacune une certaine portée dans leur contexte particulier. Cependant, les aspects spécifiques aux différentes applications sont introduits graduellement dans les sections pertinentes du document, afin de ne pas alourdir inutilement le présent chapitre.

1.4 Approche méthodologique

Un des objectifs principaux de la thèse est de proposer une approche méthodologique pour réaliser des systèmes intelligents basés sur les AE. L'approche méthodologique proposée repose sur cinq grands principes : 1) utiliser des algorithmes et des représentations adaptés au problème ; 2) développer des hybrides entre des AE et des heuristiques du domaine d'application ; 3) optimiser avec plusieurs objectifs ; 4) co-évoluer des composants et des cas de test ; et 5) utiliser des outils logiciels génériques d'AE.

1.4.1 Algorithmes et représentations adaptés au problème

Les AE constituent une famille de techniques d'optimisation inspirée de la nature. Le principe général est le même pour toutes les saveurs : une application itérative d'opérations de variation et de sélection naturelle à une population de solutions. Chaque représentation et chaque algorithme est adapté à certaines classes de problèmes. Par exemple, les SE sont particulièrement adaptées à l'optimisation de fonctions à paramètres réels, alors que les AG à chaînes de bits sont plus appropriés à des problèmes de sélection d'éléments. Dans le cadre de la thèse, la PG est utilisée pour l'évolution de structures de longueur variable qui sont utilisées comme modèles par les systèmes de reconnaissance des formes. Dans l'application des AE, il faut utiliser l'algorithme le plus

approprié à la nature du problème ou du sous-problème à résoudre afin de permettre l'obtention des meilleurs résultats possibles.

1.4.2 Hybrides entre des AE et des heuristiques du domaine d'application

L'utilisation d'hybrides entre des AE et des heuristiques adaptées au problème permet d'obtenir des résultats souvent supérieurs aux résultats obtenus avec des AE ou des heuristiques seuls. Les AE sont des méthodes d'optimisation globale d'usage général qui peuvent être appliquées à de nombreux problèmes en autant que l'on puisse représenter une solution par une structure de données et que l'on ait une mesure de qualité permettant de guider la recherche. Cependant, cette généricité a un coût, soit une certaine inefficacité computationnelle et une non-assurance de convergence vers des solutions optimales. En ce sens, il est souvent plus avantageux d'utiliser des méthodes spécialisées adaptées au problème afin de réduire les temps de calcul et d'avoir une assurance, parfois forte, de convergence. Par contre, il est fréquent que ce type de méthode ne puisse être appliqué qu'à une partie du problème ou que l'on puisse effectuer une optimisation efficace seulement dans les cas simples. Il peut alors s'avérer intéressant de pallier à ces difficultés en hybridant des AE à des heuristiques, afin d'effectuer une recherche évolutionnaire de solutions brutes qui sont ensuite raffinées par l'application des heuristiques. Cette approche permet souvent d'obtenir un compromis intéressant entre l'exploration et l'exploitation.

Prenons le cas particulier des systèmes de reconnaissance des formes. Tel que présenté à la figure 1.1, ces systèmes se comparent à une chaîne de traitement incluant la détection de données brutes, la segmentation, l'extraction de caractéristiques, le classement et la prise de décision [41]. Toutes ces étapes sont essentiellement spécifiques au problème à résoudre, à l'exception du classement pour lequel il existe plusieurs méthodes génériques bien connues. La conception d'un système de reconnaissance nécessite donc une compréhension approfondie du processus de reconnaissance et implique le plus souvent une part d'essais et erreurs afin d'obtenir une performance suffisante. Les étapes de segmentation et d'extraction de caractéristiques sont particulièrement critiques, car même un classifieur très puissant ne peut jamais compenser complètement le fait que les données puissent posséder des caractéristiques bruitées ou non discriminantes. Dans ces circonstances, il peut être intéressant, par exemple, d'utiliser les AE pour optimiser le module d'extraction de caractéristiques, par une sélection ou même une construction des caractéristiques fournies au module de classement. De cette façon, il serait possible de construire un système hybride où le module d'extraction de données est conçu par un

AE afin d'optimiser les performances du classifieur, qui constitue l'heuristique adaptée au problème.

1.4.3 Optimisation évolutionnaire à plusieurs objectifs

Il est reconnu que les méthodes modernes de sélection naturelle basées sur l'optimalité de Pareto permettent d'obtenir en général de bonnes performances pour des optimisations multi-objectif. De plus, les problèmes réels comportent souvent plusieurs objectifs à optimiser. Il ne faut donc pas hésiter à recourir à une optimisation multi-objectif évolutionnaire lorsque la situation s'y prête. Plus encore, certains types de problèmes, qui ne comportent pas *a priori* plusieurs objectifs, auraient potentiellement intérêt à être transformées en problèmes d'optimisation à plusieurs objectifs. On peut penser à des situations où le coût, la robustesse ou d'autres caractéristiques sont des critères inhérents à la conception d'un système. Dans ce cas, il serait judicieux de formuler ces objectifs explicitement, afin de guider la recherche évolutionnaire vers des solutions intéressantes selon tous les aspects pratiques du problème. Par exemple, l'optimisation d'un système d'apprentissage automatique comporte généralement un objectif principal qui consiste à minimiser le risque (ou à maximiser les marges). Cependant, de tels systèmes comportent d'autres objectifs à ne pas négliger, comme minimiser la complexité ou minimiser la variance entre les résultats de plusieurs entraînements afin d'assurer une certaine stabilité dans le processus d'apprentissage (compromis biais-variance [41]).

1.4.4 Co-évolution de composants et de cas de test

La co-évolution peut être particulièrement efficace pour deux types de tâches, soit l'évolution simultanée coopérative des différents composants d'un système, soit l'évolution simultanée compétitive de solutions et de cas de tests utilisés pour l'évaluation de l'adéquation afin d'augmenter la robustesse générale des solutions.

La co-évolution compétitive de cas de tests comporte deux avantages. D'une part, une compétition entre les solutions et les cas de test peut amener une robustesse accrue des solutions. D'autre part, une réduction du nombre de cas de tests utilisés pour chaque évaluation de l'adéquation permet une réduction de la charge de calcul. En effet, la co-adaptation dynamique des cas de test consiste en un ré-échantillonage continu de l'espace des données de test. En ce sens, l'utilisation d'un ensemble fixe de cas de test risque de donner une moins bonne couverture de l'espace des données de test, et ce même si le nombre de cas de test utilisés à chaque évaluation de l'adéquation est plus

grand.

Une méthode conventionnelle pour résoudre un problème complexe consiste à effectuer une décomposition en plusieurs sous-problèmes plus simples. Parfois, cette décomposition est naturelle, la tâche se présentant directement comme un ensemble de problèmes à résoudre. Une approche alors souvent utilisée consiste à résoudre le problème de manière séquentielle pour chacun des sous-problèmes. Avec cette approche, il peut être difficile d'avoir une vue d'ensemble de la tâche et de concevoir chaque composant du système de façon optimale par rapport aux autres composants. On peut donc comparer l'optimisation séquentielle des sous-problèmes à une recherche vorace où chaque résultat intermédiaire correspond au choix potentiellement le plus intéressant étant donné l'information disponible au moment de sa résolution, mais où le résultat final est potentiellement sous-optimal étant donné que les décisions prises hâtivement durant le processus n'ont pas nécessairement mené à la meilleure solution.

La co-évolution coopérative permet de résoudre simultanément plusieurs sous-problèmes, en évoluant une espèce par sous-problème et en évaluant la performance de solutions potentielles de chaque espèce en conjonction avec celle des solutions des autres espèces. De cette façon, la mesure de qualité guidant la recherche évolutionnaire est évaluée globalement, donc sans prise de décision hâtive et en permettant une adaptation dynamique des différents composants du système selon les changements de configuration des autres composants.

Par exemple, supposons que l'on veuille utiliser des classifieurs évolutionnaires à deux classes pour résoudre un problème de classement à plusieurs classes. On peut alors utiliser la co-évolution coopérative pour faire évoluer autant d'espèces qu'il y a de classes, chaque espèce étant formée de classifieurs chargés de discriminer les données d'une classe des autres classes. La co-évolution coopérative devrait permettre l'émergence d'un système de classifieurs à une classe co-adaptés pour le classement à plusieurs classes.

1.4.5 Développer à l'aide d'outils logiciels génériques d'AE

Le développement logiciel pour l'implantation d'applications d'AE basées sur les quatre principes précédents peut se faire de différentes façons. Premièrement, l'ensemble du code relatif aux AE peut être programmé spécifiquement pour l'application. Ceci a l'avantage d'offrir une très grande flexibilité, mais demande un investissement important en temps et en énergie. Cette approche est envisageable pour l'utilisation d'AE simples, par exemple un AG à chaînes de bits standard, mais devient rapidement trop coûteuse

pour des AE plus complexes.

Une approche de rechange consiste à récupérer une implantation existante d'un AE précis. Cela pourrait permettre de réduire significativement les coûts de développement comparativement à la première approche. Cependant, la librairie utilisée risque d'être trop limitative et donc inadéquate pour être employée telle quelle. En effet, en respectant les quatre principes précédents, il est probable que l'algorithme et la représentation implantés dans la librairie monolithique ne soient pas convenables pour le problème à résoudre, qu'il soit difficile d'intégrer une heuristique du domaine d'application avec l'AE, ou qu'il ne soit pas possible de faire une optimisation évolutionnaire multi-objectif ou de la co-évolution. L'utilisateur devra alors modifier substantiellement la librairie monolithique pour pouvoir implanter son problème, ce qui se rapproche de la première approche de développement.

Une troisième approche de développement consiste à utiliser des outils logiciels d'AE génériques. Ces outils offrent toute la flexibilité requise, tout en favorisant une réutilisation du code, ce qui minimise les coûts de développement. Selon cette approche, les représentations, les algorithmes et les autres composants ne sont pas limités à quelques modèles ou quelques valeurs pré-établis. Un bon outil générique inclut des librairies de composants qui implantent des AE standards. S'ils sont satisfaisants pour le problème à résoudre, le développeur peut exploiter ces composants tels quels. Sinon, l'outil doit offrir la possibilité de définir de nouveaux composants répondant aux spécificités de l'application. Cette approche favorise également le prototypage rapide d'applications. Son grand désavantage reste cependant l'effort nécessaire à un néophyte pour maîtriser les différents mécanismes inhérent à la généricité de l'outil.

1.5 Plan de la thèse

Le prochain chapitre porte sur l'outillage logiciel pour l'application des AE à des problèmes concrets. Par la suite, la thèse est structurée selon quatre applications des AE. Aux chapitres 3, 4 et 5, nous décrivons des applications particulières en reconnaissance des formes, tandis qu'au chapitre 6, nous présentons une application à la conception optique.

Le chapitre 2 porte sur l'utilisation d'outils logiciels pour effectuer des AE. En premier lieu, les caractéristiques et avantages de l'utilisation d'outils logiciels génériques d'AE sont décrits. Ensuite, six critères de généricité d'outils logiciels d'AE sont proposés et sont utilisés pour analyser la généricité de six outils logiciels d'AE disponibles dans

la communauté. Ensuite, un apport important du projet de doctorat, soit la framework générique d'AE Open BEAGLE, est présentée. L'organisation de cette framework est détaillée avec des explications sur les mécanismes déployés pour permettre la généricité selon les critères proposés. Mentionnons dès maintenant qu'Open BEAGLE est utilisée pour l'implantation de toutes les applications présentées dans la thèse.

Ensuite, on propose au chapitre 3 une méthode d'optimisation des classifieurs basés sur la règle du plus proche voisin (PPV). Cette méthode consiste en la co-évolution coopérative de la sélection de prototypes par un AG à chaîne de bits et d'une mesure de voisinage spécifique aux données à apprendre par la PG. Ces deux espèces utilisent chacune une mesure d'adéquation à deux objectifs afin de minimiser simultanément l'erreur de classement et la complexité des solutions. À la co-évolution à deux espèces s'ajoute la co-évolution compétitive d'une troisième espèce de cas de test. Cette co-évolution de classifieurs PPV illustre bien les différents aspects de l'approche méthodologique proposée, avec des représentations de solutions adaptées aux différents sous-problèmes, la règle du PPV comme heuristique pour le classement automatique, l'optimisation à plusieurs objectifs et une co-évolution coopérative des différentes parties du problème à laquelle se greffe une co-évolution compétitive des cas de test.

Au chapitre 4, la PG est utilisée pour l'évolution de représentations de caractères manuscrits. Le système de reconnaissance exploité dans cette application repose sur une représentation régionale floue de l'écriture cursive [60]. Les programmes génétiques consistent en un découpage de la fenêtre d'attention jumelé à une génération de caractéristiques pour la reconnaissance. Des réseaux de neurones de type perceptron multicouche sont utilisés pour classer les données selon l'information fournie par les représentations évoluées. La recherche de représentations est guidée par une mesure à deux objectifs visant à minimiser à la fois le taux d'erreur de classement et le nombre de caractéristiques. Cette application illustre l'utilisation d'un AE pour la construction de caractéristiques d'un système de reconnaissance des formes.

On présente au chapitre 5 une troisième application à la reconnaissance des formes avec un système d'arbres de décision flous, extensibles et évolutionnaires. Ce système est une méthode générique pour la combinaison hiérarchique de classifieurs de différentes natures. L'information se propage entre les noeuds de l'arbre par un traitement flou du classement. La topologie de ces arbres flous évolue par une PG à plusieurs objectifs, alors que les paramètres des différentes unités de classement sont déterminés par l'entraînement de chaque arbre. On effectue donc, d'une part, une acquisition de caractères innés par l'évolution des topologies et, d'autre part, une optimisation locale par apprentissage. Cette approche consiste donc en une optimisation du module de décision de systèmes de reconnaissance (voir figure 1.1) par une combinaison floue et

évolutionnaire de classifieurs de différentes natures.

Finalement, le chapitre 6 porte sur la conception automatisée de systèmes de lentilles avec des SE. La conception de systèmes de lentilles peut être formulée comme un problème d'optimisation de fonctions avec paramètres à valeur réelle. Une résolution analytique du problème est trop complexe pour être possible, excepté pour quelques cas triviaux. La conception optique moderne repose sur le savoir-faire des experts humains. Malgré quelques essais, il ne semble pas pour l'instant y avoir de méthode de conception optique entièrement automatisée qui soit suffisamment performante pour être utilisée dans la pratique. On propose dans ce chapitre une approche hybride pour la conception automatique de systèmes de lentilles par l'utilisation des SE jumelées aux algorithmes d'optimisation locale d'outils commerciaux de CAO d'optique. Pour deux problèmes relativement difficiles de conception optique, la méthode proposée est capable de générer des résultats comparables ou meilleurs que ceux produits par des experts humains. De plus, des résultats supplémentaires démontrent l'intérêt des AE pour une optimisation à plusieurs objectifs de systèmes optiques.

La thèse s'articule donc autour d'exemples concrets illustrant l'approche méthodologique proposée pour l'application des AE à des problèmes difficiles. Le chapitre 2 est une réponse directe au principe de la section 1.4.5 sur l'utilisation d'outils logiciels génériques d'AE. Quant aux applications faisant l'objet des quatre chapitres suivants, elles illustrent clairement les principes méthodologiques proposés en démontrant la justesse de l'approche.

Chapitre 2

Outils logiciels d'algorithmes évolutionnaires

Software Engineering is that part of Computer Science which is too difficult for the Computer Scientist.
F. L. Bauer

L'utilisation d'un outil logiciel générique et flexible est essentielle à la réalisation d'un projet de recherche d'envergure dans le domaine des algorithmes évolutionnaires (AE). À cet effet, il existe de nombreux outils logiciels disponibles sur le Web[1]. Cependant, une minorité de ceux-ci ont été créés avec l'objectif de donner à l'utilisateur toute la flexibilité nécessaire pour expérimenter aisément avec les AE sans devoir tout reprogrammer.

Dans ce chapitre nous proposons une liste de caractéristiques que devrait posséder une framework générique d'AE et nous présentons des détails sur l'implantation d'*Open BEAGLE*, une framework C++ d'AE qui a été développée durant le projet de doctorat. Finalement, nous exposons brièvement deux extensions d'Open BEAGLE, *Distributed BEAGLE* et *BEAGLE Visualizer*.

[1]Le site Web d'EvoNet liste plus d'une centaine de logiciels d'AE (`http://evonet.lri.fr/evoweb/resources/software/index.php`).

2.1 Outils logiciels génériques d'algorithmes évolutionnaires

Les AE comportent traditionnellement trois branches principales [5] : les algorithmes génétiques (AG) [66], initialement développés aux États-Unis dans les années 1960 et 1970 par J.H. Holland et ses étudiants[2], les stratégies d'évolution (SE) [3, 125], développées à peu près en même temps à Berlin par I. Rechenberg et H.-P. Schwefel et finalement la programmation évolutionnaire (PE) [3, 46], fondée par L.J. Fogel aux États-Unis dans les années 1960. Cette taxonomie tient essentiellement à des facteurs historiques et géographiques, les différentes communautés scientifiques ayant évolué de façon relativement isolée jusqu'au début des années 1990 [5]. L'idée de réunir les trois paradigmes sous une même appellation, les *algorithmes évolutionnaires*, a alors fait son chemin étant donné les similarités évidentes entre les approches. Plusieurs conférences et journaux scientifiques englobant l'ensemble des AE ont vu le jour durant les années 1990, permettant une atténuation graduelle des distinctions historiques et une unification des idées. Un exemple de cette tendance est le développement de techniques agnostiques vis-à-vis les différentes saveurs d'AE comme l'optimisation multi-objectif [26] et la co-évolution [4, 62]. Un autre exemple est l'idée à l'effet que dans une dizaine d'années d'ici, le GECCO[3] ne sera possiblement plus compartimenté selon les différentes saveurs d'AE (AG, PG, etc.), mais bien selon les différents aspects du domaine (représentations, algorithmes, applications, etc.)[4].

Malgré la tendance lourde visant l'unification des AE, le développement de la discipline est toujours empreint d'artefacts liés au cloisonnement historique. La plupart des experts dans le domaine sont associés à une saveur particulière d'AE et conservent souvent une approche scientifique et technique teintée par cette appartenance. Un exemple flagrant est l'utilisation toujours répandue de logiciels spécialisés implantant une saveur d'AE précise. On peut s'attendre qu'avec le temps, le décloisonnement et la généralisation des idées dans le domaine devraient mener à une utilisation répandue d'outils logiciels qui ne sont pas spécifiques à une saveur d'AE précise, d'où l'intérêt pour le développement et l'utilisation d'outils logiciels d'AE génériques. Sans oublier qu'avec de tels outils, le développement de nouvelles approches s'avère possible et beaucoup moins coûteux. À notre avis, l'utilisation de tels outils logiciels génériques pour implanter des AE est incontournable et devrait se répandre dans la communauté au cours des

[2] Malgré ses caractéristiques distinctives et son importance, la programmation génétique (PG) [76, 9] est généralement considérée comme une composante des AG selon la taxonomie historique des AE.

[3] *Genetic and Evolutionary Computation COnference*, une des plus importantes conférences scientifiques sur les AE.

[4] Opinion avancée par quelques chercheurs au cours du GECCO 2003.

prochaines années.

Le développement d'un outil logiciel d'AE véritablement générique s'avère complexe sur le plan du génie logiciel. Dans la présente section, nous tentons d'éclaircir la question afin de bien analyser et de bien comprendre en quoi consiste le développement d'un outil logiciel d'AE générique. Premièrement, une revue de littérature est faite sur l'état de l'art dans le domaine. Nous exposons ensuite les avantages d'un outil logiciel générique d'AE sous la forme d'une framework programmée en langage orienté objet (OO). Puis, nous présentons les différents stades de développement d'une framework d'AE générique, ainsi que les six critères de qualité proposés pour évaluer la généricité d'un outil logiciel d'AE. Finalement, nous analysons de la généricité de quelques outils logiciels d'AE existants.

2.1.1 Revue de littérature

Au milieu des années 1990, Filho et al. [45] ont présenté différents outils logiciels d'AE existant à l'époque, avec une emphase particulière sur les outils d'AG. Cette présentation, quoique très axée sur les caractéristiques techniques des outils, classe les outils logiciels d'AE en trois grandes catégories : 1) les outils orientés application, utilisés essentiellement dans un contexte applicatif précis, 2) les outils orientés algorithme, typiquement des librairies implantant un AE particulier et 3) les *toolkits*, des ensembles logiciels relativement génériques. Dans la même année, deux magazines ont publié des articles [29, 135] présentant des implantations particulières de systèmes de PG en C++. On note également Keith et Martin [73], qui ont fait un analyse judicieuse des différentes façons d'implanter une représentation de programmes génétiques.

Plus récemment, plusieurs publications portant sur des outils logiciels d'AE ont été recensées. Certaines [138, 92] portent sur l'architecture et l'interface avec l'utilisateur d'outils d'AE spécialisés. Wilson et al. [157] ont entre autres fait une analyse superficielle de trois outils logiciels de PG, soit lil-gp, ECJ et Grammatical Evolution, avec la perspective d'un utilisateur novice. D'autres [50, 72, 81, 93, 129, 143, 148] présentent des architectures d'outils logiciels génériques d'AE, sans toutefois aborder de front les concepts généraux liés au développement de tels outils. Lenaerts et Manderick [88] se démarquent en abordant la question du développement d'un outil générique dans une perspective plus globale. Ils présentent les avantages d'utiliser et de développer une framework d'AE avec une méthodologie OO. Leur article expose les problèmes rencontrés à l'utilisation de la plupart des librairies disponibles et des avantages qu'auraient les chercheurs dans le domaine des AE, en particulier de la PG, à utiliser des frameworks génériques et extensibles. Les auteurs font également une présentation très intéressante

de différents *design patterns* [52] qui peuvent être appliqués lors de la conception d'une framework de PG.

2.1.2 Framework d'algorithmes évolutionnaires

On peut voir les AE comme une classe abstraite d'algorithmes et ses différentes saveurs telles que la PG, les AG et les SE, comme une instanciation concrète de ceux-ci. L'instanciation d'un AE se fait en spécifiant deux éléments caractéristiques principaux : la représentation de la population (structure des données) et les opérations génétiques et de sélection naturelle (algorithmes) utilisées. Dans le cadre du développement d'un outil logiciel d'AE, cette modélisation demande cependant des efforts particuliers de la part du concepteur afin de découpler le plus possible les différents aspects d'une instanciation d'un AE, tout en permettant une certaine ré-utilisation du code. En effet, certains types d'opérations, par exemple les opérations de sélection naturelle, peuvent s'appliquer à toutes les représentations, alors que d'autres types d'opérations, par exemple les opérations de croisement et de mutation, sont spécifiques à la représentation utilisée. De plus, le domaine des AE est en constant développement et il est difficile de prévoir les nouvelles approches ou variations qui seront intéressantes à implanter dans l'outil logiciel. Il faut donc intégrer des mécanismes flexibles pour éviter de devoir effectuer des modifications importantes dans les structures fondamentales de l'outil. Pour toutes ces raisons, le développement d'un outil logiciel générique d'AE comporte un ensemble de problématiques qui, sur le plan du génie logiciel, en fait un candidat idéal pour une modélisation sous forme de framework.

On peut définir le terme framework dans un cadre de programmation OO de la façon suivante (adapté de [52]) :

> Ensemble de composants formant un design réutilisable pour un type particulier de logiciel. Une framework donne des spécifications architecturales en partitionnant le design en classes abstraites et en définissant les responsabilités et collaborations entre celles-ci. Un développeur spécialise une framework pour ses besoins particuliers en utilisant les classes de la framework comme base à ses propres instances.

Une caractéristique des AE est la nécessité de définir une façon d'évaluer l'adéquation d'une solution pour chaque problème à résoudre. En ce sens, développer un outil logiciel d'AE exige de laisser une partie de l'architecture incomplète, afin de permettre à l'utilisateur d'implanter, au minimum, l'opération d'évaluation de l'adéquation. D'autre part, selon le problème à résoudre, l'utilisateur peut prendre quatre approches pour implanter son AE : 1) employer une configuration standard d'un AE ; 2) définir son propre

AE en utilisant différents composants d'algorithmes standards ; 3) définir ses propres opérations génétiques et de sélection naturelle ; ou 4) définir une nouvelle représentation d'individus. En ce sens, un bon outil logiciel d'AE doit offrir une très grande versatilité, tout en implantant des composants effectuant par défaut des opérations standards et en spécifiant les relations entre les différentes entités du système.

2.1.3 Progression d'une framework d'algorithmes évolutionnaires

Roberts et Johnson [127] présentent différents patrons capturant l'essence de frameworks à différents stades de développement, partant d'une framework de type *white box*, où l'utilisateur doit définir un ensemble de composants qui héritent (en termes OO) de composants de base, jusqu'à une framework de type *black box*, où la majorité des composants spécialisés existent déjà et où il n'est donc plus nécessaire d'en définir de nouveaux mais seulement de former un arrangement de composants pour résoudre un problème donné. Une framework débute généralement comme une *white box* pour progresser vers un modèle *black box*. Un scénario typique d'une telle progression débute par le développement d'un ensemble de composants relativement volumineux qui forment une librairie capturant le domaine d'application. Ces composants sont définis en dérivant de nouvelles classes des composants de base. Cependant, la progression de la framework tend à ce que les éléments de l'architecture qui changent régulièrement soient modélisés par des composants de plus en plus simples, faiblement couplés et pouvant être composés ensemble sans devoir définir de nouveaux objets. De plus, une framework mûrit généralement par un enrichissement continu de sa librairie de composants. La framework atteint alors le stade de *black box* lorsqu'il est possible d'implanter des applications seulement en connectant des composants existants de la framework. Lorsqu'une framework a atteint ce stade, il est courant que des langages de script et/ou des interfaces graphiques soient développés, afin de permettre un environnement de développement n'exigeant pas de l'utilisateur une connaissance approfondie du fonctionnement interne de la framework pour des applications simples.

Les stades de développement d'une framework d'AE cadrent relativement bien dans le modèle de la progression d'une framework de type *white box* vers une framework de type *black box*. Cette progression se traduit souvent par l'encapsulation des opérations génétiques et de sélection sous la forme de composants qui peuvent être connectés pour former les AE utilisés. Cependant, l'évolution d'une framework d'AE vers un modèle *black box* ne permet pas d'éliminer totalement la nécessité de programmer de nouveaux composants, car l'utilisateur doit toujours fournir sous une forme ou une autre

la fonction d'évaluation de l'adéquation pour son problème particulier. Ceci peut se faire soit par la définition d'un nouveau composant, soit par un script dans un langage de haut niveau. Il est aussi possible de fournir une banque de composants d'évaluation de l'adéquation pour un certain ensemble de problèmes classiques. Mais en général, cette approche n'est pas applicable car les fonctions d'évaluation de l'adéquation sont souvent très spécifiques au problème à résoudre et impossibles à déterminer à l'avance.

L'évolution d'une framework d'AE vers un modèle *black box* jumelée à une interface graphique permet à l'utilisateur d'employer l'outil sans devoir maîtriser la mécanique interne. De plus, l'utilisateur expert peut définir de nouveaux composants en tirant parti de la flexibilité de l'architecture pour ses développements. Cependant, la conception d'une telle framework d'AE demande des efforts considérables de la part des développeurs, la preuve étant qu'à notre connaissance, il existe seulement trois frameworks d'AE véritablement génériques, flexibles et faciles d'utilisation ayant atteint le stade de *black box*[5].

2.1.4 Critères de généricité

Pour qualifier une framework d'AE de générique, six critères doivent selon nous être pris en compte : 1) la représentation, 2) l'adéquation, 3) les opérations, 4) le modèle évolutionnaire, 5) le mécanisme de gestion des paramètres et 6) une sortie configurable.

Représentation générique De nouvelles représentations d'individus pour un AE personnalisé doivent pouvoir être définies par l'utilisateur sans limitation quant à la structure de données utilisée. Ces nouvelles représentations peuvent être définies en s'appuyant sur des représentations existantes, par exemple en se basant sur une représentation standard d'AG pour définir une représentation d'un vecteur d'indices dans un graphe afin de résoudre un problème d'optimisation combinatoire [96]. L'utilisateur peut aussi définir des représentations inusitées, par exemple la programmation génétique à base de graphe [140], qui est significativement différente de la représentation classique de la PG à base d'arbre. Selon la singularité des représentations, il doit être possible de récupérer des opérations génétiques et de sélection existantes pour les appliquer aux nouvelles représentations.

Adéquation générique La mesure de l'adéquation des individus doit être indépendante autant que possible des représentations et des opérations de sélection. Il doit être possible de définir et d'utiliser des mesures d'adéquation propres à une application donnée. Par exemple, l'utilisateur doit pouvoir changer de mesure d'adéquation en passant d'une mesure où les valeurs élevées caractérisent les bons

[5]Soit ECJ, EO et Open BEAGLE, qui seront présentées plus loin.

individus (maximisation de l'adéquation) à une mesure d'adéquation où les valeurs faibles sont désirées (minimisation de l'adéquation), sans devoir recoder les représentations ou même les opérations de sélection. Un autre exemple est la nécessité de supporter de façon transparente des mesures d'adéquation multi-objectif.

Opérations génériques Un minimum de limitation doit être imposé sur la nature des opérations génétiques et de sélection qui peuvent être implantées dans l'outil logiciel. Les opérations peuvent être relativement classiques, par exemple une opération de croisement à deux parents, ou totalement inusitées. De plus, les opérations doivent être relativement indépendantes et avoir un minimum d'effet de bord, afin qu'il soit possible de les utiliser en conjonction avec d'autres opérations. Lorsque c'est possible, les opérations doivent être indépendantes des représentations. Cette approche permet le développement de nouvelles opérations sans altérer celles qui existent déjà, ce qui favorise la création d'une librairie de composants et facilite le développement de nouvelles saveurs d'AE. Par exemple, on peut imaginer une librairie de composants comprenant plusieurs opérateurs de variation pour une représentation à chaîne de bits (mutation par inversion de bits, croisement en un point, en deux points, uniforme, etc.). Pour une application basée sur une telle représentation, l'utilisateur a le choix d'utiliser un ou plusieurs des ces opérateurs, avec d'autres opérateurs de son cru s'il le désire, sans être contraint quant à leur arrangement ou leur compatibilité.

Modèle évolutionnaire générique Les opérations génétiques et de sélection doivent être appliquées à la population selon des algorithmes flexibles et configurables. Le modèle évolutionnaire doit donc être le plus malléable possible, sans structure rigide. Le modèle devrait idéalement pouvoir se définir en connectant des opérateurs ensemble, selon un certain ordre. Par exemple, dans le cas d'un AG générationnel, l'AE peut être vu comme une application successive d'opérations de sélection naturelle, de croisement et de mutation sur tous les individus d'une population. D'autre part, un AG de type *steady-state* est caractérisé par des opérations de sélection naturelle, de croisement et de mutation appliquées aléatoirement aux individus, chaque itération correspondant à la création d'un nouvel individu remplaçant un individu actuel de la population. Et finalement les modèles (μ, λ) et $(\mu + \lambda)$ des SE représentent un modèle encore plus complexe d'opérations de sélection, de croisement et de mutation sur les individus d'une population. Des opérateurs inusités doivent pouvoir être introduits dans un modèle existant sans devoir le réécrire.

Gestion des paramètres Une framework d'AE comporte très souvent un mécanisme permettant de modifier les paramètres (la taille de la population, la probabilité de mutation, etc.) dynamiquement, par un fichier de configuration ou toute autre interface avec l'utilisateur. Il est alors souhaitable que le mécanisme de gestion des paramètres inclue uniquement les paramètres pertinents à l'AE utilisé et que ce

mécanisme permette d'ajouter, avec peu d'effort, de nouveaux paramètres lorsque l'AE est personnalisé par l'utilisateur. Finalement, il peut être très intéressant de configurer l'algorithme directement dans un fichier, sans devoir recompiler une nouvelle application.

Sortie configurable La sortie de l'information dans un fichier, ou via une autre interface avec l'utilisateur, doit pouvoir être configurable. Pour les sorties sur l'état de l'évolution, chaque représentation, mesure d'adéquation ou opération comporte son lot d'informations spécifiques qui peuvent être intéressantes en sortie. Par exemple, les sorties relatives aux statistiques sur la progression de l'évolution sont différentes selon que l'on utilise une mesure d'adéquation à un ou plusieurs objectifs. Un autre exemple est l'intérêt en PG d'avoir en sortie des statistiques sur les tailles et les profondeurs des arbres de PG. Dans un outil générique, l'ajout de nouvelles sorties telles que ces statistiques doit être possible et celles-ci devraient s'intégrer harmonieusement aux sorties actuelles, que ce soit les sorties d'information à l'utilisateur ou les sorties de résultats dans des fichiers.

Face à une framework générique d'AE, l'utilisateur doit initialement fournir un certain effort afin d'assimiler les mécanismes essentiels à la flexibilité de l'outil. Ceci a tendance à repousser certains utilisateurs vers des outils logiciels d'AE de type librairies monolithiques spécialisées pour un AE donné, car ceux-ci demandent moins d'effort pour être utilisés à court terme. Mais cette simplicité apparente s'avère assez coûteuse à moyen terme, au moment où les utilisateurs découvrent les limitations posées par le cadre initial de la librairie. Ils doivent alors modifier le code de la librairie afin de supporter de nouvelles fonctionnalités ou modifier des fonctionnalités existantes. Les avantages initiaux sont donc perdus face au besoin d'assimiler l'ensemble du code de la librairie pour pouvoir le modifier. De plus, l'apport de nouveaux composants, tels que de nouvelles opérations génétiques ou de nouvelles représentations, est pratiquement impossible car il implique généralement que chaque modification est permanente et irréversible. Il devient alors très difficile de conjuguer différentes modifications ou d'ajouter de nouvelles fonctionnalités qui concurrencent celles qui existent déjà. Par exemple, l'extension d'une librairie d'AG monolithique afin de supporter des mesures d'adéquation d'évolutions multi-objectif pourrait possiblement rendre incompatible la librairie avec les anciennes applications qui utilisent des mesures d'adéquation d'évolution à un objectif. En ce sens, l'apprentissage d'une framework d'AE générique est un investissement qui sera grandement rentable à l'utilisateur à moyen terme, s'il compte expérimenter avec différentes variations de l'AE de base.

TAB. 2.1 – Évaluation de la généricité d'outils logiciels d'AE.

Critère de généricité	ECJ 12	EO 0.9.3a	GAlib 2.4.5	lil-gp 1.1	GPLAB 2	Open BEAGLE 2.2.0
Représentation générique	2	2	2	0	0	2
Adéquation générique	2	2	0	0	0	2
Opération générique	2	2	1	2	2	2
Modèle évolutionnaire générique	2	2	1	1	1	2
Gestion des paramètres	2	2	2	1	2	2
Sortie configurable	2	1	0	1	0	2

(2 = complet, 1 = partiel, 0 = absent)

2.1.5 Analyse de la généricité d'outils logiciels

Malgré la panoplie d'outils logiciels d'AE existants, peu d'entre eux sont couramment utilisés dans la communauté, et encore moins peuvent être qualifiés de génériques. Afin d'illustrer les idées avancées sur la généricité dans les outils logiciels d'AE, six outils logiciels sont revus en détails du point de vue de la généricité. Il s'agit de ECJ [91], EO [95, 72], GAlib [154], lil-gp [118], GPLAB [134] et Open BEAGLE. Le choix des ces logiciels s'est fait sur les bases suivantes : 1) ils permettent d'effectuer de la PG, qui est un paradigme important des AE particulièrement complexe à implanter, 2) ils font preuve d'une certaine flexibilité sous une forme ou une autre, 3) ils connaissent une relative popularité dans la communauté et 4) ils comportent des particularités qui les rendent intéressants pour la présente étude, que ce soit sur le plan du langage de programmation ou de l'architecture. Le tableau 2.1 présente une évaluation de la généricité de ces différents outils selon les six critères présentés à la section 2.1.4.

ECJ[6] est une framework générique d'AE programmée en langage Java. Il s'agit probablement du système d'AE en Java le plus populaire disponible publiquement. La framework respecte l'ensemble des critères de généricité d'une framework d'AE. Elle possède un exécutable Java qui nécessite seulement un fichier de configuration et une composante Java avec la fonction d'évaluation de l'adéquation. Ce fichier de configuration spécifie les éléments de ECJ à utiliser pour composer l'AE ainsi que les

[6] `http://cs.gmu.edu/~eclab/projects/ecj`

paramètres mêmes de l'AE. Comparativement aux outils programmés dans un langage tel que le C++, le fait que ECJ soit programmé en langage Java, un langage orienté objet de haut niveau, facilite la programmation de nouveaux modules tout en produisant des programmes très gourmands en mémoire et avec de plus longs temps d'exécution [117]. Les opérations d'ECJ peuvent être composées ensemble selon un modèle évolutionnaire générique, sans devoir programmer quelque classe que ce soit. En ce sens, ECJ est une framework d'AE de type *black box* à part entière.

EO[7] [72] est une framework générique C++ d'AE. EO, qui signifie *Evolving Object*, a été conçue dans l'optique d'appliquer un processus évolutionnaire à n'importe quel type de représentation, en autant qu'une mesure objective de qualité puisse être établie. EO comporte différents opérateurs pour manipuler les représentations d'individus et effectuer des traitements évolutionnaires, ainsi que des opérateurs d'intégration. EO possède également plusieurs classes utilitaires pour gérer les paramètres et, dans une certaine mesure, configurer la sortie. Si l'utilisateur souhaite un modèle évolutionnaire plus complexe, il doit alors concevoir des opérateurs d'intégration pour bâtir son modèle évolutionnaire. La mise au point de ces différents opérateurs spécialisés demande une bonne compréhension de la framework et de ses composants. Dans son ensemble, EO est une framework générique de type *black box* assez difficile à exploiter car elle demande la maîtrise d'une mécanique complexe..

En relation avec EO, EASEA[8] [27, 28] est un outil permettant de simplifier grandement l'utilisation d'un outil logiciel d'AE donné. En effet, EASEA permet d'intégrer les spécifications d'un AE dans un langage de programmation de haut niveau et de transformer ces spécifications en code C++ compilable avec EO. De plus, un programme doté d'une interface graphique nommé GUIDE est associé à EASEA pour permettre la génération des spécifications EASEA via une interface graphique. La suite EASEA/GUIDE forme un ensemble logiciel cohérent intéressant pour développer une application simple sans devoir maîtriser la mécanique interne d'EO. En ce sens, EASEA/GUIDE permet de pallier à une certaine complexité liée à l'utilisation d'EO. Il est cependant clair que d'un point de vue de généricité, la suite EASEA/GUIDE impose certaines limitations sur les algorithmes utilisables qui rendent son utilisation beaucoup moins intéressante pour des experts dans le domaine.

GAlib[9] est une librairie C++ d'AE basée sur une représentation générique. Cependant, hors de l'aspect générique des représentations, la librairie est relativement limitative du point de vue de sa flexibilité. Premièrement, la mesure d'adéquation est

[7] Evolving Objects, `http://eodev.sourceforge.net`

[8] EAsy Specification of Evolutionary Algorithms, `http://fractales.inria.fr/evo-lab/EVO-easea-engl.html`

[9] `http://lancet.mit.edu/ga`

fixée comme étant une valeur scalaire. De plus, il existe seulement six types d'opérateurs : initialisation de la population, sélection d'individus, croisement à deux parents, mutation, évaluation de l'adéquation et terminaison de l'évolution. Il n'est pas possible de définir des opérateurs sortant de ce cadre avec les modèles évolutionnaires existants. Pour une évolution donnée, un (et un seul) opérateur de chaque type doit être fourni à un modèle évolutionnaire particulier. Les modèles évolutionnaires de GAlib ne sont pas génériques. En effet, ceux-ci sont codés directement dans une classe. Des paramètres peuvent être ajoutés dynamiquement au système mais la sortie n'est pas modifiable. Cependant, la simplification du modèle évolutionnaire à quelques algorithmes et à des opérations bien définies rend GAlib relativement facile à utiliser et à maîtriser pour un novice.

Les deux outils logiciels suivants sont conçus spécifiquement pour effectuer de la PG. Nous les évaluons selon les critères de généricité intéressants pour la présente discussion. Premièrement, lil-gp[10] est une ré-implantation en langage C du système de PG *little-lisp* [76]. Le système est très utilisé dans la communauté et a la réputation d'être l'un des systèmes de PG les plus rapides. Évidemment, la représentation est fixée à des arbres de PG, mais la mesure d'adéquation est également restreinte à la mesure d'adéquation de PG de Koza [76]. Le modèle évolutionnaire consiste en une application successive d'opérateurs sur la population, une application de tous les opérateurs résultant en une génération. Le système permet donc une certaine généricité en n'imposant aucune restriction sur les opérateurs utilisés, mais limite l'utilisateur à des algorithmes générationnels. Des paramètres peuvent être ajoutés, mais l'utilisateur doit fournir une routine pour analyser le fichier de configuration et extraire les données de ces paramètres. lil-gp est l'exemple même de la librairie monolithique spécifique facile à utiliser lorsqu'on reste dans le cadre initial, mais qui est difficile à modifier ou à étendre.

D'autre part, GPLAB[11] est une *toolbox* MATLAB pour effectuer de la PG. Tout comme lil-gp, GPLAB supporte une seule représentation et la mesure d'adéquation est limitée à un nombre réel. Le modèle évolutionnaire est fixé *a priori*, mais les opérateurs composant ce modèle peuvent être de tous types. Bien que remplissant moins que la moitié des critères de généricité, GPLAB comporte l'avantage certain d'être bâti dans un environnement pouvant être qualifié lui-même de framework générique pour la programmation scientifique. En effet, MATLAB est un environnement de développement de très haut niveau offrant un ensemble de fonctionnalités mathématiques et graphiques incomparable, à la condition que l'utilisateur soit prêt à payer le prix en temps d'exécution significativement plus longs. En ce sens, la présente évaluation de la généricité de GPLAB ne fait pas complètement honneur aux qualités de l'outil et au fait qu'il soit

[10] `http://garage.cps.msu.edu/software/lil-gp/index.html`

[11] `http://gplab.sourceforge.net`

programmé dans MATLAB.

Finalement, Open BEAGLE est une framework *black box* d'AE programmée en C++. En faisant abstraction des différences liées aux langages de programmation, Open BEAGLE est comparable à ECJ sur le plan de la flexibilité et de la facilité de programmation. Il est sans aucun doute plus générique que GAlib, lil-gp ou GPLAB, et plus facile à utiliser que EO. Un peu comme ECJ, le modèle évolutionnaire d'une application d'Open BEAGLE peut être modifié dynamiquement sans recompilation, via le fichier de configuration. Il est envisagé, dans une version future, que ce fichier de configuration puisse être généré par une application comportant une interface graphique, un peu comme avec EASEA/GUIDE.

2.2 Open BEAGLE : framework générique d'algorithmes évolutionnaires

Open BEAGLE est une framework C++ d'AE de type *black box*. L'acronyme récursif BEAGLE signifie *Beagle est un Environnement d'Apprentissage Génétique Logiciel Évolué*[12]. Beagle est également le nom du vaisseau anglais sur lequel Charles Darwin s'est engagé comme naturaliste pour effectuer son célèbre tour du monde. Le nom Beagle a été utilisé dans les années 1980 pour désigner un logiciel de reconnaissance des formes basé sur des principes évolutionnaires, développé par Forsyth [48]. L'adjectif *Open* a été ajouté au nom de la framework pour la distinguer du logiciel de Forsyth, et également pour insister sur l'aspect *open source* du projet. Le projet a débuté en 1998 par le développement d'une librairie C++ pour effectuer de la PG. Ce premier prototype a été complètement réécrit en 1999 pour pallier à certains problèmes fondamentaux dans l'architecture. Au cours des années 2001 et 2002, une nouvelle écriture du logiciel à partir de presque rien a été effectuée, cette fois pour transformer l'outil en une framework générique d'AE. En 2002, Open BEAGLE a été lancée publiquement sur le Web[13]. À la fin de l'année 2003, le développement s'est déplacé sur *SourceForge.net*[14], un portail offrant un ensemble de services pour le développement collaboratif. Les développements de la framework suivent une méthodologie *open source* [124], les contributions futures provenant d'utilisateurs externes seront évaluées et intégrées à la framework si jugées intéressantes.

Depuis que l'outil est disponible publiquement, la popularité d'Open BEAGLE est

[12] En anglais, l'acronyme signifie *the Beagle Engine is an Advanced Genetic Learning Environment.*
[13] `http://beagle.gel.ulaval.ca`.
[14] `http://sourceforge.net/projects/beagle`

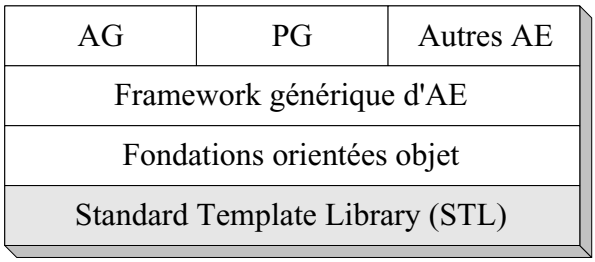


FIG. 2.1 – Architecture de la framework Open BEAGLE.

en croissance continue. À la fin février 2005, le compteur de la page Web indique qu'un peu plus de 22 500 personnes ont visité la page Web depuis ses débuts. Il y a eu près de 2140 téléchargements du code source entre octobre 2003 et février 2005. Tous ces chiffres illustrent le succès relatif de la framework dans la communauté.

2.2.1 Architecture d'Open BEAGLE

L'architecture de la framework suit les principes directeurs de la programmation OO, où des abstractions sont représentées par des objets faiblement couplés et où il est aisé et courant de réutiliser le code. Open BEAGLE a une architecture structurée en trois niveaux distincts, tel que présenté à la figure 2.1. Les fondations OO sont à la base de l'architecture comme une extension OO du C++ et de la *Standard Template Library* (STL). La framework générique est construite sur ces fondations. Elle est composée des éléments caractéristiques de tous les AE. Finalement, différents modules spécialisent la framework générique en implantant un AE spécifique.

2.2.2 Fondations orientées objet

Les fondations OO forment la base de l'architecture d'Open BEAGLE. Elles sont inspirées de *design patterns* [52] et d'autres environnements tels que la STL [102], la librairie Java [19] et CORBA [61].

Les classes C++ d'Open BEAGLE sont toutes dérivées de la même classe abstraite, `Object`. Cette classe comporte un ensemble de fonctionnalités, dont un compteur de référence qui permet, en conjonction avec des pointeurs intelligents, une gestion automatisée de la désallocation de la mémoire similaire à ce que l'on retrouve dans des langages OO de plus haut niveau. Open BEAGLE fait une utilisation importante de

polymorphisme par héritage. Cela signifie que les objets doivent être instanciés dynamiquement. De plus, il est difficile de copier ou de cloner un objet donné lorsque son type exact est inconnu. À cette fin, des *allocateurs* ont été intégrés à la framework. Ces allocateurs agissent comme des *object factories* qui peuvent allouer, cloner et copier un certain type d'objet. Un *conteneur* générique d'objets est aussi intégré aux fondations OO d'Open BEAGLE. Ce conteneur est un tableau dynamique de pointeurs d'objets Open BEAGLE compatible avec l'interface des conteneurs génériques de la STL, tout en utilisant un allocateur pour allouer les objets contenus.

La lecture et l'écriture de fichiers contenant les données des AE, telles que la représentation de la population, les paramètres et les résultats des évolutions, est un aspect important d'une framework d'AE. Le XML (*eXtensible Markup Language*) [16] est le langage idéal pour la modélisation de données car il est flexible, standard, lisible par des humains et facilement éditable. De plus, tout format de fichier XML peut être transformé en un autre format de fichier XML en utilisant des XSLT (*eXtensible Stylesheet Language Transformations*) [25], tant que les informations nécessaires sont présentes et correctement identifiées. Cela est un point important car il permet sans difficulté une compatibilité arrière pour le changement de format de fichiers, une interaction avec d'autres systèmes[15] utilisant des fichiers XML[16] et la transformation de fichier XML en fichier XHTML pour la visualisation des données dans un navigateur Web. La framework Open BEAGLE inclut des classes pour lire et écrire du XML interagissant avec les flots standards de lecture et d'écriture du C++. De plus, les classes d'Open BEAGLE comportent des fonctions pour être lues et être écrites en XML, ce qui permet une intégration complète du langage XML au système.

2.2.3 Framework générique d'algorithmes évolutionnaires

La framework générique d'AE est une extension des fondations OO. Elle offre une base solide pour l'implantation d'AE. Elle est composée d'une structure générique de populations, d'un système d'évolution et d'un ensemble d'opérateurs compris dans un évoluateur. Tous les composants de la framework générique d'AE sont intégrés ensemble comme des modules qui peuvent être remplacés ou spécialisés indépendamment. Ce design modulaire donne beaucoup de flexibilité à la framework et simplifie l'implantation de n'importe quel AE.

Dans Open BEAGLE, les populations sont structurées selon quatre niveaux hié-

[15] On peut imaginer qu'il serait simple de développer un outil pour convertir des fichiers de format EAML [152] en fichiers de configuration d'Open BEAGLE et vice-versa.

[16] Et même certains types de format texte de fichier.

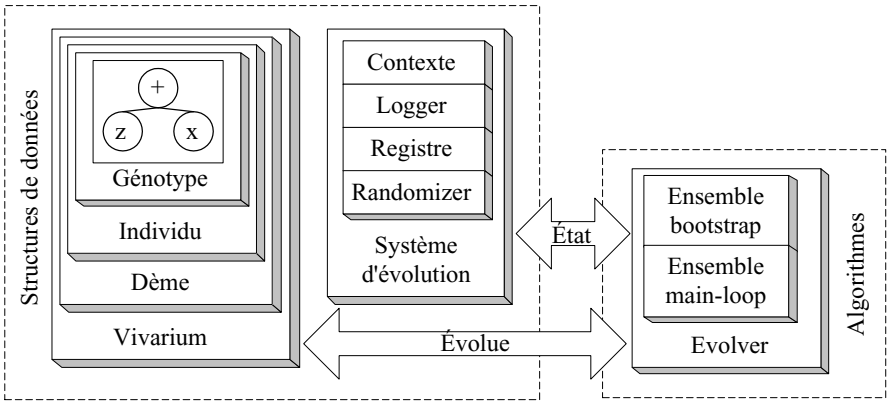


FIG. 2.2 – Framework générique d'AE.

rarchiques : le vivarium, les dèmes, les individus et les génotypes (voir la figure 2.2). Le vivarium comprend les statistiques de la dernière génération, un *hall-of-fame* des meilleurs individus de l'évolution et tous les individus vivant dans le système évolutionnaire. Ces individus sont regroupés en dèmes [87] qui représentent des groupes d'individus évoluant dans des environnements clos. Un dème comprend également les statistiques de la dernière génération et un *hall-of-fame* des meilleurs individus. En général, à chaque génération, des individus peuvent migrer entre les différents dèmes d'un vivarium.

Les individus représentent des solutions potentielles à un problème. Un individu est défini par deux types de données : la mesure de son adéquation (dans un environnement donné) et un ou plusieurs génotypes. Le génotype comprend le bagage génétique d'un individu. Le génotype dans la framework générique d'AE est une interface qui doit être spécialisée. Par exemple, dans le cas de la PG, le génotype est défini comme un arbre. L'organisation des individus, des génotypes et de la mesure de l'adéquation permet de répondre aux critères de généricité d'une représentation générique et d'une adéquation générique, tels que présentés à la section 2.1.4.

La framework générique comporte également un système d'évolution contenant la configuration de l'engin d'évolution. Il est formé de quatre composants : l'allocateur de contexte, le registre, le *logger* et le *randomizer* (voir la figure 2.2). Le contexte de la framework générique représente l'état actuel du processus évolutif. Il contient quelques informations essentielles telles que le dème, l'individu et le génotype actuellement traités, ainsi que le numéro de la génération actuelle. Pour certains AE, un contexte plus spécialisé peut être défini. Par exemple, une pile associée à l'arbre de PG traité (un génotype) est ajoutée au contexte de base dans la framework spécialisée de PG. Le concept du contexte est similaire à celui de l'exécution dans les ordinateurs, qui comprend entre

autres les valeurs des registres, compteurs et pointeurs de l'unité de traitement.

Étant donné que les paramètres d'Open BEAGLE sont distribués sur plusieurs éléments disparates, un agent nommé le registre est utilisé pour centraliser l'information. Les références aux objets Open BEAGLE représentant des paramètres y sont associées à des identifiants. De plus, les éléments tels que les opérateurs peuvent accéder dynamiquement à ces paramètres. Le registre est également responsable de lire les fichiers de configuration XML. Le registre implante donc des mécanismes répondant au critère de généricité de la gestion des paramètres.

Le *logger* agit comme interface avec l'utilisateur en traitant tous les messages générés par la framework. Ces messages sont associés à un type (entité architecturale), une classe (le type C++ de l'objet à l'origine du message) et un niveau de détail. De plus, le *logger* peut être configuré par l'utilisateur pour signaler uniquement les messages d'un niveau de détail inférieur ou égal à une certaine valeur. Le *logger* peut être également spécialisé pour transmettre les messages de sortie à différentes entités, par exemple une fenêtre graphique. Le *logger* utilisé par défaut transmet des messages de sortie en format XML à un fichier et/ou à la console. Le *logger* est également très pratique pour déboguer une application, en utilisant un niveau de détail très élevé. Le *logger* remplit les spécifications du critère de généricité d'une sortie configurable de la section 2.1.4.

Le *randomizer* est le générateur de nombres aléatoires du système. Il peut générer des entiers ou des nombres à virgule flottante suivant des distributions uniformes ou gaussiennes. La semence du générateur peut être assignée à une valeur arbitraire. Cette valeur est inscrite dans le registre de sorte que l'on peut reproduire les évolutions.

Les opérateurs et évoluateurs sont des concepts centraux de la framework générique d'AE. Dans Open BEAGLE, le processus évolutif consiste en une séquence d'opérations qui sont itérativement appliquées aux dèmes du vivarium. Chaque opération génétique est définie comme un opérateur. L'évoluateur possède deux ensembles d'opérateurs : l'ensemble d'opérateurs du *bootstrap* et celui de la *main-loop*. L'ensemble d'opérateurs du *bootstrap* consiste en une liste d'opérations appliquées à chaque dème pour construire la nouvelle population. L'ensemble d'opérateurs de la *main-loop* est une liste d'opérations appliquées itérativement à chaque dème, à chaque génération partant de la génération 1. Le modèle des opérateurs et évoluateurs est basé sur le *design pattern Strategy*, appliqué au cas particulier des AE.

La figure 2.3 présente la configuration en XML d'un évoluateur générationnel d'AG tel qu'utilisé dans Open BEAGLE. Dans l'ensemble des opérateurs du *bootstrap*, l'opérateur `GA-InitBitStrOp` génère la population initiale, l'opérateur `MyEvalOp` fait l'éva-

```
1  <?xml version="1.0"?>
2  <Beagle>
3    <Evolver>
4      <BootStrapSet>
5        <GA-InitBitStrOp/>
6        <MyEvalOp/>
7        <StatsCalcFitnessSimpleOp/>
8      </BootStrapSet>
9      <MainLoopSet>
10       <SelectTournamentOp/>
11       <GA-CrossoverOnePointBitStrOp/>
12       <GA-MutationFlipBitStrOp/>
13       <MyEvalOp/>
14       <StatsCalcFitnessSimpleOp/>
15       <TermMaxGenOp/>
16       <MilestoneWriteOp/>
17     </MainLoopSet>
18   </Evolver>
19 </Beagle>
```

FIG. 2.3 – Configuration en XML d'un évoluateur générationnel d'AG à chaîne de bits avec Open BEAGLE.

luation de l'adéquation, alors que l'opérateur `StatsCalcFitnessSimpleOp` permet un calcul des statistiques. Pour ce qui est de l'ensemble des opérateurs de la *main-loop*, une opération de sélection par tournois est appliquée par l'opérateur `SelectTournamentOp`, puis les opérations de croisement à un point (`GA-CrossoverOnePointBitStrOp`) et de mutation par inversion de bit (`GA-MutationFlipBitStrOp`) sont appliquées. Ensuite, on retrouve de nouveau les opérations d'évaluation de l'adéquation (`MyEvalOp`) et de calcul des statistiques (`StatsCalcFitnessSimpleOp`). La génération est complétée par le test d'un critère d'arrêt (`TermMaxGenOp`), dans le cas présent un nombre maximum de générations. Finalement, l'opérateur `MilestoneWriteOp` permet l'écriture d'un fichier XML comprenant l'état actuel de l'évolution (paramètres, évoluateur, statistiques, population), ce qui est pratique pour analyser les résultats d'une évolution et même redémarrer le processus évolutionnaire.

Ce modèle de l'évoluateur et des opérateurs fonctionne bien dans le cas d'AE générationnels, car seul un mécanisme de traitement évolutionnaire au niveau de la population est nécessaire. Cependant, pour d'autres types d'AE tels que des AG *steady-state* ou les SE, un mécanisme de traitement au niveau des individus est indispensable. À cette fin, le modèle du *breeder* a été développé. Il consiste en une extension du modèle de l'évoluateur et des opérateurs. Il comporte deux éléments architecturaux principaux : les *stratégies de remplacement*, qui sont des opérateurs standards présents dans les ensembles *bootstrap* ou *main-loop*, et les opérateurs-breeders qui peuvent se connecter entre eux ainsi qu'aux stratégies de remplacement pour effectuer un traitement au niveau des individus.

Une chaîne de traitement de type *breeder* est une structure en arbre avec à la racine une stratégie de remplacement et comme autres nœuds des opérateurs-breeders. Une stratégie de remplacement appelle les sous-arbres d'opérateurs-breeders afin de générer de nouveaux individus selon un algorithme qui lui est propre. Ces appels sont généralement paramétrés selon les probabilités de *breeding* de chaque sous-arbre. Les nouveaux individus sont insérés dans la population selon l'algorithme spécifique de la stratégie de remplacement, d'où le nom. Chaque appel à un sous-arbre d'opérateurs-breeders a pour but de générer un nouvel individu. Un nœud non terminal effectue une opération sur les individus reçus des ses fils, pour en retourner le résultat à son parent. Un nœud terminal consiste en une opération de sélection afin de choisir un individu dans la population actuelle et le retourner à son parent.

Prenons par exemple une stratégie de remplacement de type *steady-state*, qui produit une nouvelle génération d'individus par un nombre d'appels à ses sous-arbres égal à la taille de la population (un appel = un individu généré). La probabilité d'appeler chaque sous-arbre de la stratégie de remplacement est paramétrée par la probabilité

de *breeding* de chacun de ses sous-arbres. Supposons qu'il y a trois sous-arbres, chacun représentant respectivement un croisement, une mutation et une reproduction d'un individu, respectivement. La probabilité de *breeding* de chacun de ces sous-arbres est la probabilité de croisement, de mutation et de reproduction. La figure 2.4 présente la configuration en XML d'un tel évoluateur *steady-state* d'AG. L'opérateur d'évaluation est à la racine des sous-arbres de croisement et de mutation car les nouveaux individus générés dans un algorithme *steady-state* doivent avoir une adéquation valide avant d'être insérés dans la population par la stratégie de remplacement. Cela n'est pas nécessaire pour le sous-arbre de reproduction, composé seulement de l'opérateur de sélection de la ligne 22 de la figure 2.4. En effet, les individus générés sont des copies d'individus existant déjà dans la population, ayant donc une adéquation valide.

Les concepts de l'évoluateur, des opérateurs et du *breeder* permettent de remplir les critères d'opérateurs génériques et de modèle évolutionnaire générique, tels que présentés à la section 2.1.4. Ce sont aussi des mécanismes coordonnant des composants qui permettent en grande partie de qualifier Open BEAGLE de framework *black box*.

2.2.4 Frameworks spécialisées

Les frameworks spécialisées sont bâties sur la framework générique. Actuellement, trois frameworks spécifiques ont été implantées, soit une framework d'AE à représentation linéaire avec un support pour des représentations en chaînes de bits, en vecteurs de nombres réels et en vecteurs de paires (x_i, σ_i) pour les SE, une framework de PG à base d'arbres, et finalement une framework de co-évolution, permettant une évolution simultanée de plusieurs espèces de représentation différente ou non. Un utilisateur peut implanter sa propre saveur d'AE à partir d'une framework spécialisée existante, ou directement à partir de la framework générique.

La framework d'AE à représentation linéaire est relativement simple. Pour chacune des trois représentations supportées, elle définit un génotype spécifique et comporte des opérateurs d'initialisation, trois opérateurs génériques de croisement (croisement en un point, en deux points et uniforme) ainsi que des opérateurs de mutation spécifiques (inversion de bits, ajout de valeur selon une distribution normale et mutation adaptative des SE). La framework spécialisée comporte également quelques fonctionnalités permettant entre autres la conversion d'une chaîne de bits en un vecteur de nombres réels défini selon un intervalle et un encodage donnés.

La framework spécialisée de PG est plus complexe. De nouveaux mécanismes spécifiques au paradigme utilisé ont été définis. Pour programmer génétiquement un ordina-

```
1  <?xml version="1.0"?>
2  <Beagle>
3    <Evolver>
4      <BootStrapSet>
5        <GA-InitBitStrOp/>
6        <MyEvalOp/>
7        <StatsCalcFitnessSimpleOp/>
8      </BootStrapSet>
9      <MainLoopSet>
10       <SteadyStateOp>
11         <MyEvalOp>
12           <GA-CrossoverOnePointBitStrOp matingpb="ga.cx1p.prob">
13             <SelectTournamentOp/>
14             <SelectTournamentOp/>
15           </GA-CrossoverOnePointBitStrOp>
16         </MyEvalOp>
17         <MyEvalOp>
18           <GA-MutationFlipBitStrOp mutationpb="ga.mutflip.indpb">
19             <SelectTournamentOp/>
20           </GA-MutationFlipBitStrOp>
21         </MyEvalOp>
22         <SelectTournamentOp repropb="ec.repro.prob"/>
23       </SteadyStateOp>
24       <StatsCalcFitnessSimpleOp/>
25       <TermMaxGenOp/>
26       <MilestoneWriteOp/>
27     </MainLoopSet>
28   </Evolver>
29 </Beagle>
```

FIG. 2.4 – Configuration en XML d'un évoluateur *steady-state* d'AG à chaîne de bits avec Open BEAGLE.

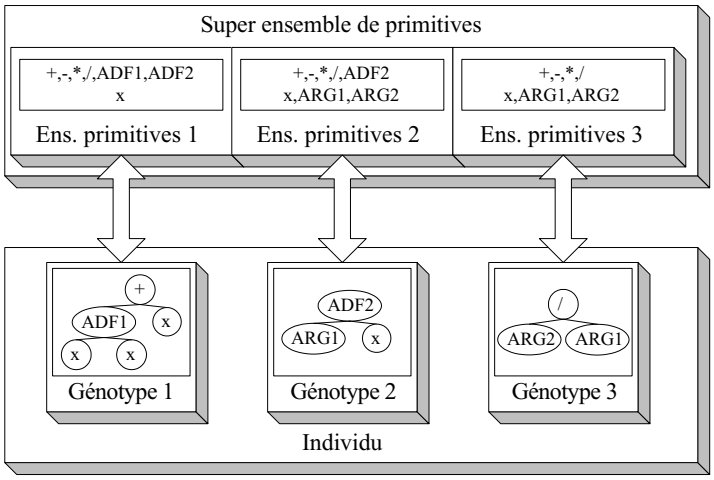


FIG. 2.5 – Relation entre l'ensemble de primitives et les arbres de PG.

teur, deux points spécifiques au domaine du problème à résoudre doivent être établis. Premièrement, l'utilisateur doit définir le type du *datum*, c'est-à-dire le type des données (variables) qui seront manipulées par les programmes génétiques. Une fois le datum défini, les *primitives* utilisées pour construire les individus de PG doivent également être spécifiées. Une primitive est une opération spécifique à une application qui est associée aux nœuds de l'arbre de PG. Elle correspond aux terminaux et aux fonctions utilisés dans une application tel que décrit dans [76]. Les primitives traitent et retournent des variables du type du datum utilisé.

Le type du datum dans Open BEAGLE doit être dérivé de la classe `Object`. Ceci peut être fait généralement en utilisant un type prédéfini d'Open BEAGLE ou en adaptant un type étranger à l'interface de la classe. Pour créer une primitive qui peut être utilisée dans les arbres de PG, l'utilisateur doit définir une classe concrète dérivée de la classe abstraite de primitives, où une fonction virtuelle pure doit être surdéfinie afin d'implanter l'opération caractéristique de la primitive. L'interface de la classe de primitives permet également d'expérimenter avec des fonctionnalités avancées de la PG telles que la PG fortement typée (*strongly-typed*) [99] et les constantes éphémères générées aléatoirement [76].

Les primitives utilisées pour une application donnée doivent être insérées dans l'ensemble des primitives utilisables. Les arbres de PG sont générés uniquement à partir de cet ensemble. Le super-ensemble des primitives est une extension du système d'évolution contenant plusieurs ensembles de primitives. Le nombre d'ensembles dans le super-ensemble détermine le nombre d'arbres (de génotypes) des individus de PG, tel qu'illustré à la figure 2.5. Ce mécanisme permet, entre autre, l'implantation de fonctions définies automatiquement (*automatically defined functions*) [76, 77].

Finalement, la framework de co-évolution [4, 62] est différente des deux précédentes par le fait qu'elle implante non pas une nouvelle représentation, mais des mécanismes pour une évolution simultanée de plusieurs espèces d'individus. Cette framework est basée sur la programmation concurrente, où chaque fil d'exécution (*thread*) fait l'évolution d'une espèce. La framework de co-évolution définit un opérateur d'évaluation de l'adéquation permettant l'appariement d'individus d'espèces (de fils d'exécution) différentes. Autrement, les individus évoluent en utilisant les mécanismes standards définis dans les autres frameworks spécialisées, ou définis par l'utilisateur.

2.2.5 Exemple de code : OneMax

Malgré toute la complexité inhérente à une framework générique d'AE, l'utilisation d'Open BEAGLE peut être relativement simple du point de vue d'un novice. En effet, l'ensemble des composants ont des valeurs et des politiques par défaut satisfaisantes pour la plupart des applications simples. L'utilisateur n'a qu'à définir un opérateur d'évaluation de l'adéquation et un programme initialisant les différentes composantes pour effectuer une évolution. Un exemple classique illustrant les AG à chaînes de bit est le problème *OneMax*, qui consiste tout simplement à rechercher des chaînes de bits avec un maximum de uns. La figure 2.6 présente l'implantation d'un opérateur d'évaluation pour cet exemple classique, tandis que la figure 2.7 présente la routine principale de l'exemple.

La ligne 5 de la figure 2.6 permet de construire un opérateur d'évaluation avec comme nom `OneMaxEvalOp`. Les lignes 6 à 15 de la même figure appellent la fonction pour évaluer l'adéquation d'un individu pour le problème *OneMax*. Les lignes 9 et 10 effectuent un retypage de l'individu générique à évaluer en un individu à chaîne de bits. Les lignes 11 à 13 comptent le nombre de 1 dans la chaîne de bits et la ligne 14 retourne l'adéquation comme une mesure de maximisation à une valeur réelle.

Pour ce qui est de la routine principale de l'application présentée à la figure 2.7, les lignes 8 et 9 permettent de bâtir une population de chaînes de bits. La ligne 10 instancie l'opérateur d'évaluation de l'adéquation défini précédemment, alors que les lignes 11 à 13 définissent un évoluateur d'AG à chaîne de bits où les individus sont initialisés comme une chaîne de 20 bits chacun. La ligne 14 crée le système d'évolution tel que défini à la figure 2.2. À la ligne 15, l'évoluateur ainsi que le système d'évolution sont initialisés, la ligne de commande est analysée et les fichiers de configuration sont lus. Par défaut, un AG générationnel similaire à celui de la figure 2.3 est utilisé. Mais si l'utilisateur préfère employer un AG de type *steady-state* par exemple, il doit alors

```
1  #include "beagle/GA.hpp"
2  using namespace Beagle;
3  class OneMaxEvalOp : public EvaluationOp {
4  public:
5    OneMaxEvalOp() : EvaluationOp("OneMaxEvalOp") { }
6    virtual Fitness::Handle evaluate(Individual& inIndividual,
7                                     Context& ioContext)
8    {
9      GA::BitString::Handle lBitString =
10       castHandleT<GA::BitString>(inIndividual[0]);
11     unsigned int lNumberOnes = 0;
12     for(unsigned int i=0; i<lBitString->size(); ++i)
13       if((*lBitString)[i]) ++lNumberOnes;
14     return new FitnessSimple(float(lNumberOnes));
15   }
16 };
```

FIG. 2.6 – Opérateur d'évaluation de l'adéquation pour le problème *OneMax*.

définir un fichier de configuration XML similaire[17] à celui de la figure 2.4 et exécuter le programme avec une option sur la ligne de commande référant au fichier de configuration approprié. Finalement, la ligne 16 lance l'évolution. L'ensemble de cette routine est dans un bloc *try-catch* pour intercepter les exceptions lancées par Open BEAGLE lorsqu'une situation problématique est détectée à l'exécution. Cet exemple, ainsi que de nombreux autres, accompagnent la distribution d'Open BEAGLE disponible sur le Web.

2.3 Distributed BEAGLE : distribution des calculs évolutionnaires

Le projet intègre également une extension, Distributed BEAGLE, permettant la distribution du processus évolutif sur plusieurs processeurs. Cette extension permet d'exploiter les importantes ressources informatiques du Laboratoire de vision et systèmes numériques (LVSN) et d'appliquer les AE à des problèmes difficiles demandant des calculs trop longs pour être envisagés sur un seul processeur.

Nous avons développé un prototype de Distributed BEAGLE au cours de l'été 2000. Ce prototype est une implantation simple basée sur le modèle maître-esclave de dis-

[17] En prenant soin évidemment de remplacer les balises `MyEvalOp` par des balises `OneMaxEvalOp`.

```
1  #include <cstdlib>
2  #include <iostream>
3  #include "beagle/GA.hpp"
4  #include "OneMaxEvalOp.hpp"
5  using namespace Beagle;
6  int main(int argc, char** argv) {
7    try {
8      GA::BitString::Alloc::Handle lBSAlloc = new GA::BitString::Alloc;
9      Vivarium::Handle lVivarium = new Vivarium(lBSAlloc);
10     OneMaxEvalOp::Handle lEvalOp = new OneMaxEvalOp;
11     const unsigned int lNumberOfBitsPerGenotype = 20;
12     IntegerVector lEncoding(1, lNumberOfBitsPerGenotype);
13     GA::EvolverBitString lEvolver(lEvalOp, lEncoding);
14     System::Handle lSystem = new System;
15     lEvolver.initialize(lSystem, argc, argv);
16     lEvolver.evolve(lVivarium);
17   }
18   catch(Exception& inException) {
19     inException.terminate(std::cerr);
20   }
21   return 0;
22 }
```

FIG. 2.7 – Routine principale pour le problème *OneMax*.

tribution d'AE. Le standard CORBA [61] a été utilisé pour gérer les communications entre les processus. Ce système est pleinement fonctionnel et a été utilisé pour des applications d'AE demandant de longs temps de calculs. Cependant, des lacunes ont été observées à l'utilisation : latences trop élevées dues à la structure même de l'algorithme de distribution, complexité du modèle utilisé, lourdeur du standard CORBA, etc.

Une équipe d'étudiants[18] au baccalauréat en génie informatique de l'Université Laval a complètement ré-implanté Distributed BEAGLE dans le cadre du cours *Design IV*. Un étudiant de cette équipe, Marc Dubreuil, a poursuivi le développement dans le cadre d'un projet de maîtrise au LVSN. Cette nouvelle version de Distributed BEAGLE utilise une base de données de type SQL, assurant la persistance des données, des *sockets* TCP-IP pour la communication entre les processeurs et le langage XML pour encoder les messages. Le système est bien intégré à Open BEAGLE et son utilisation est relativement transparente à l'utilisateur : seulement quelques lignes de code doivent être modifiées afin de transformer une application monoprocesseur en une application multiprocesseur. Le système est disponible publiquement sur le Web[19] depuis juin 2004.

Les évolutions sont distribuées par cette nouvelle version de Distributed BEAGLE selon un modèle maître-esclave inspiré du *design pattern* des algorithmes évolutionnaires distribués et persistants [14]. Le système fait de l'équilibrage de la charge (*load balancing*) permettant d'ajuster dynamiquement, selon les temps de réponse observés pour les différents esclaves, le nombre d'individus envoyés aux nœuds de calculs à chaque requête. Finalement, les charges de calcul en retard par rapport au temps estimé initialement peuvent être redistribuées sur d'autres esclaves afin d'accélérer le traitement et d'augmenter la robustesse du système.

La figure 2.8 présente différents composants de Distributed BEAGLE. La base de données contient deux types de données : les dèmes qui doivent évoluer d'une génération et les individus dont l'adéquation doit être évaluée. Le serveur sert d'interface entre les requêtes provenant des clients d'évaluation et d'évolution et la base de données. Ce serveur est chargé de marquer l'état des dèmes et des individus de la base de données et d'effectuer l'équilibrage de la charge. Les clients d'évolution font des requêtes au serveur du système afin de recevoir des dèmes à traiter. Les dèmes sont évolués d'une génération en leur appliquant des opérations génétiques et de sélection naturelle. Les clients d'évaluation font quant à eux des requêtes de calculs au serveur centralisé afin de recevoir des ensembles d'individus dont l'adéquation doit être évaluée.

Contrairement aux autres systèmes connus d'AE parallèles et distribués [1, 109], les

[18] L'équipe était formé de Marc Dubreuil, Frédéric Jean, Jacques Labrie et Hélène Torresan.

[19] `http://beagle.gel.ulaval.ca/distributed`

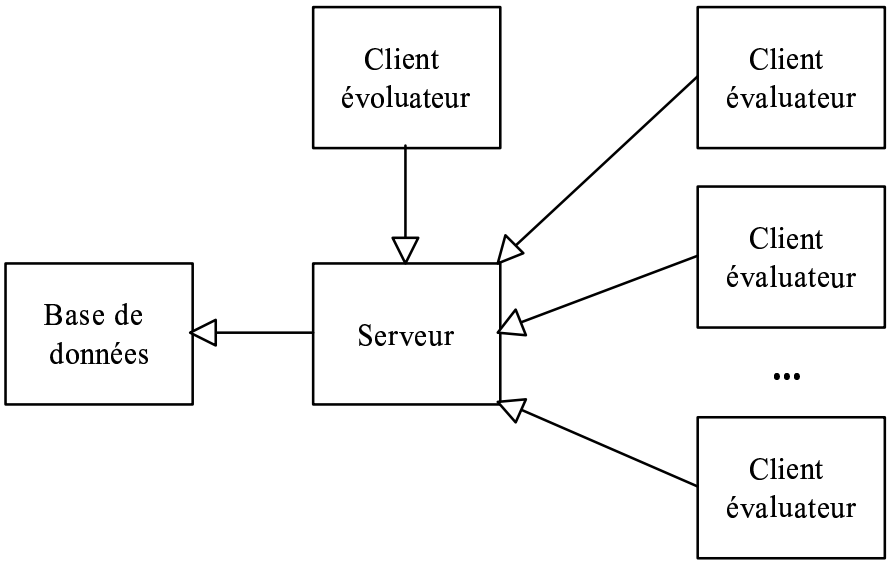


FIG. 2.8 – Architecture du deuxième prototype de Distributed BEAGLE

systèmes Open BEAGLE et Distributed BEAGLE effectuent un découplage de la gestion du parallélisme, c'est-à-dire une séparation de l'évolution simultanée de plusieurs sous-populations et de la distribution des calculs sur plusieurs processeurs. En effet, les populations dans Open BEAGLE sont structurées en plusieurs dèmes qui évoluent en vase clos, simulant ainsi des sous-populations évoluant en parallèle. Il est donc possible avec Open BEAGLE de simuler des AE à plusieurs populations sans devoir distribuer le processus évolutif. Distributed BEAGLE répartit les tâches de calculs sur plusieurs processeurs tout en équilibrant dynamiquement la charge. Cette abstraction entre les aspects parallèles et distribués des AE sur plusieurs processeurs est bénéfique à de multiples égards. En effet, Distributed BEAGLE peut exploiter des réseaux d'ordinateurs de puissance et de disponibilité variable sans devoir se soucier d'ajuster en conséquence les paramètres d'évolution tels que la taille des dèmes et la fréquence des migrations. De plus, lorsque des nœuds de calculs tombent en panne ou deviennent indisponibles sur de longues périodes, le processus évolutif continue de façon robuste, sans aucune perte d'information. Cependant, le désavantage de cette approche par rapport aux systèmes basés sur le modèle en îles multiprocesseur [21] reste les communications beaucoup plus fréquentes et importantes entre les différents composants du système. Néanmoins, nous avons démontré [40, 51] qu'un modèle maître-esclave permet l'exploitation de plusieurs centaines de machines avant que la charge des communications sur un réseau contemporain ne devienne significative.

2.4 BEAGLE Visualizer : interface Web d'analyse de résultats

Les résultats générés par des AE consistent en une quantité importante d'information qui peut être difficile à analyser en un simple coup d'oeil. À cette fin, un outil générant un ensemble de représentations visuelles peut faciliter grandement la compréhension et l'analyse des résultats. Une nouvelle extension d'Open BEAGLE a donc été développée sous la forme d'un interface Web d'analyse de résultats. Le développement de cette interface a été confié à une équipe de deux étudiants[20] au baccalauréat en génie informatique de l'Université Laval, dans le cadre d'un projet pour le cours *Design IV*. L'interface Web nommée *BEAGLE Visualizer* est accessible publiquement[21].

Cet outil graphique permet l'analyse de résultats en utilisant les fichiers XML générés par Open BEAGLE. En effet, le langage XML est un composant central des technologies Web tout à fait approprié à ce type d'interface, tout en permettant une neutralité vis-à-vis du système d'exploitation et en n'exigeant pas l'installation de programmes, autre qu'un navigateur Web moderne. Les données extraites des fichiers et représentées dans l'interface Web sont la configuration et les paramètres de l'évolution, les différentes statistiques sur la progression de l'évolution et une présentation graphique des individus évolués. De plus, l'interface Web permet de synthétiser des populations entières en des figures simples, afin de faire ressortir les structures communes à plusieurs individus. Finalement, l'interface Web est disponible sous deux formes, soit comme un service Web accessible publiquement et comme un outil ligne de commande générant, dans le répertoire local, l'ensemble des fichiers de résultats d'analyse.

Les figures 2.9 et 2.10 présentent des captures d'écran de rapports Web générés par BEAGLE Visualizer. La figure 2.9 présente un extrait de l'item *statistics* d'un rapport Web de BEAGLE Visualizer, avec un graphique de l'adéquation pour le problème *OneMax*, où l'on recherche des chaînes de bits composées d'un maximum de 1. Sur le graphique, la courbe en trait continu représente la moyenne de l'adéquation alors que les courbes en traits discontinus représentent l'adéquation maximale et minimale. La zone grise représente un écart-type des mesures d'adéquation de la population par rapport à la moyenne de l'adéquation. La figure 2.10 est un extrait de l'item *vivarium* d'un rapport Web de BEAGLE Visualizer sur une évolution de programmes génétiques pour la résolution du problème de la régression symbolique [76]. La figure présente les statistiques numériques sur la mesure d'adéquation ainsi qu'une représentation circulaire de la population de programmes génétiques, tel que proposé par Daida *et al.* [34].

[20] L'équipe était formée de Patrick-Emmanuel Boulanger-Nadeau et Vincent-Olivier Gravel.

[21] `http://beagle.gel.ulaval.ca/visualizer`

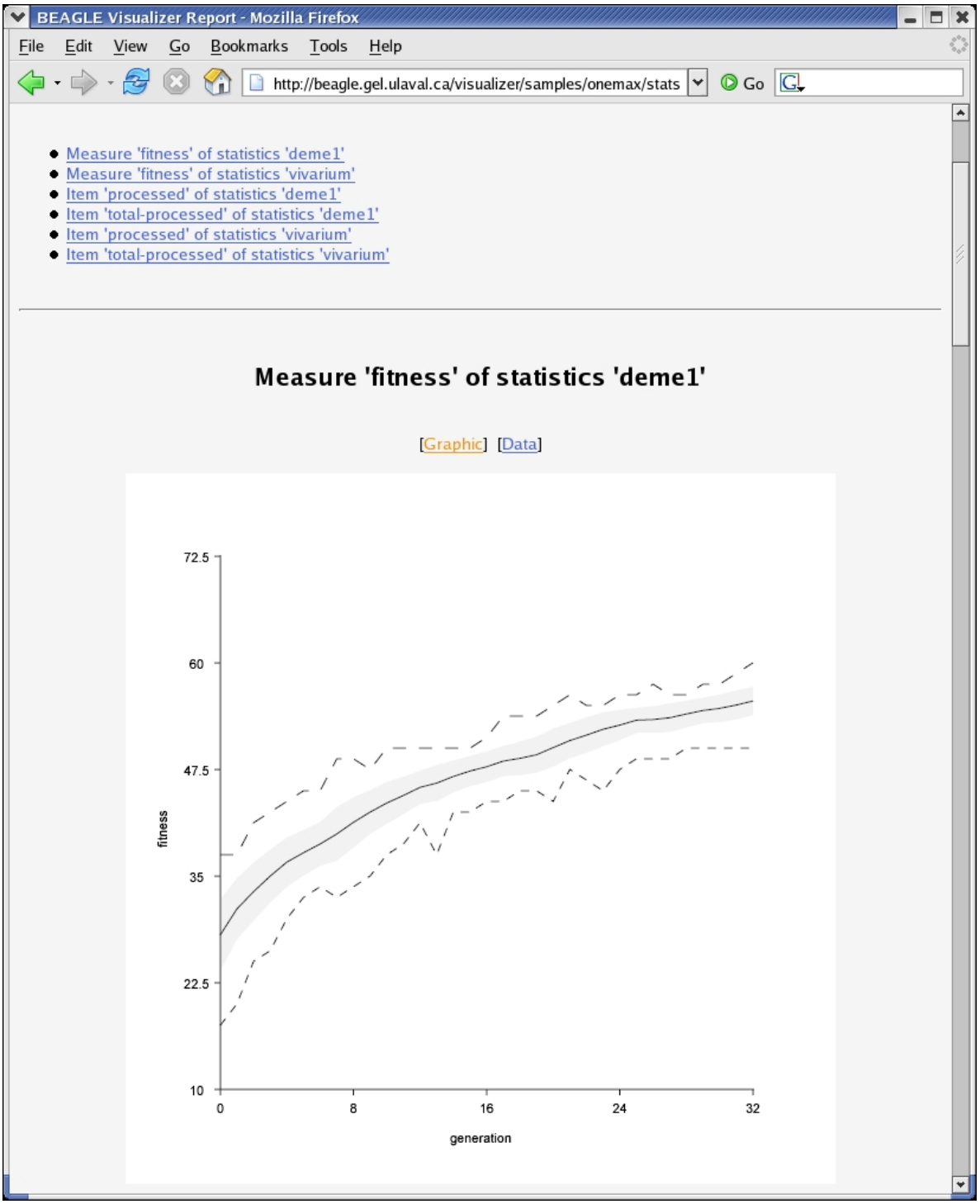


FIG. 2.9 – Capture d'écran d'un rapport BEAGLE Visualizer pour le problème *OneMax*.

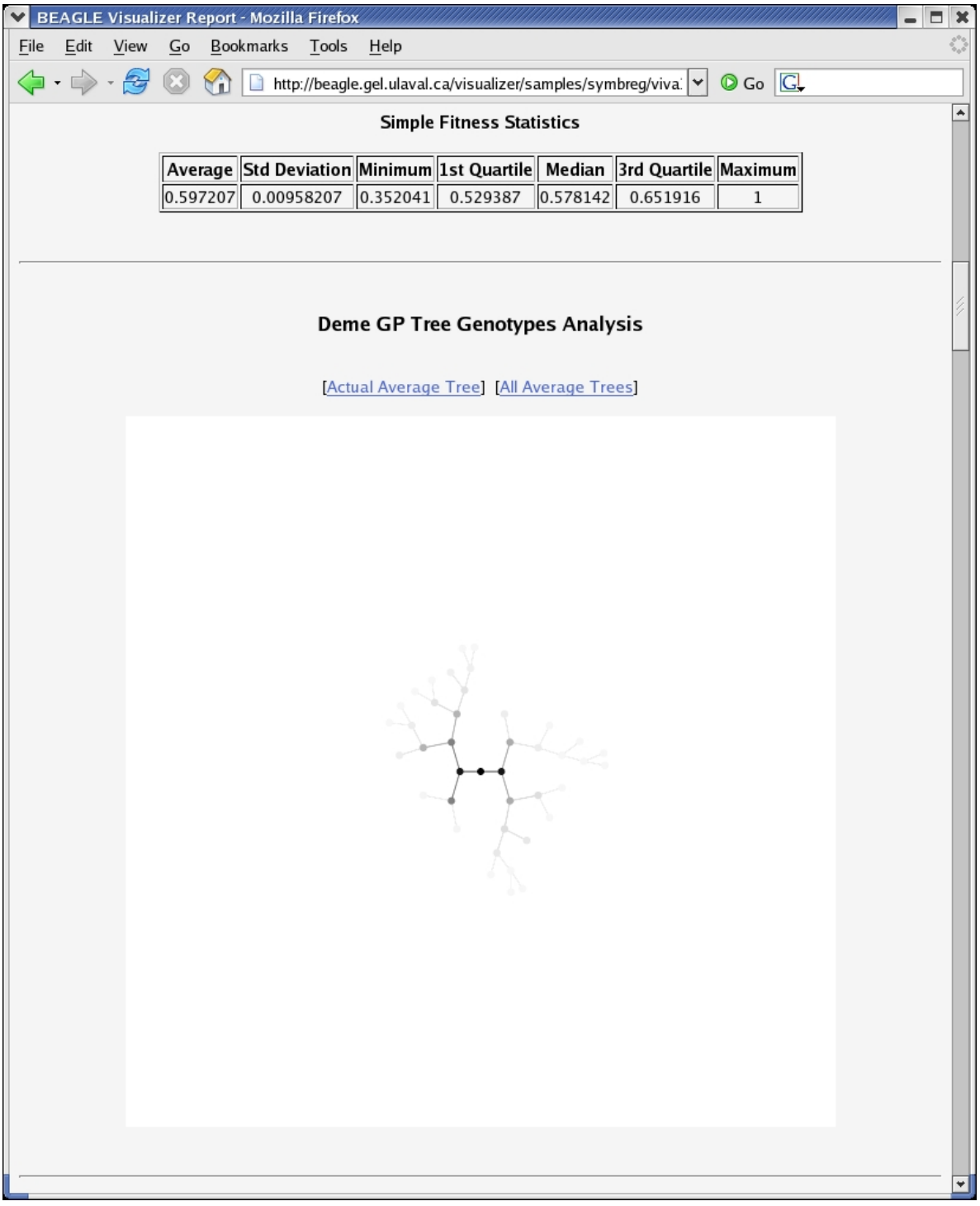


FIG. 2.10 – Capture d'écran d'un rapport BEAGLE Visualizer pour le problème de la régression symbolique.

2.5 Conclusion

La framework d'AE Open BEAGLE est une partie importante du projet de doctorat. Le développement d'une telle framework générique d'AE est une tâche relativement complexe sur le plan du génie logiciel. En effet, il existe de nombreux environnements logiciels pour effectuer des AE dans le domaine public, mais très peu d'entre eux peuvent être considérés comme étant à la fois génériques, versatiles et faciles d'utilisation. Open BEAGLE fait partie du groupe restreint des frameworks génériques d'AE ayant atteint un stade de développement *black box*. L'architecture d'Open BEAGLE se décompose en trois couches principales. À la base, on retrouve les fondations OO qui tendent à élever le niveau du langage C++. Bâtie sur ces fondations, la framework générique d'AE implante l'essentiel des structures qui sont communes aux différents AE. Finalement, il y a les frameworks spécialisées, qui implantent différentes saveurs d'AE en suivant les lignes directrices données par la framework générique d'AE. Open BEAGLE comporte également deux extensions développées par d'autres étudiants dans le cadre de projets de maîtrise et de fin de baccalauréat : Distributed BEAGLE, qui permet de distribuer des évolutions sur des réseaux d'ordinateurs, et BEAGLE Visualizer, qui permet d'analyser visuellement les résultats d'évolution.

Chapitre 3

Co-évolution de classifieurs basés sur la règle du plus proche voisin

Un mauvais voisin est une calamité, un bon voisin un vrai trésor.
Hésiode (poète grec)

Le classement par le plus proche voisin (PPV) [17, 31, 41], appelé parfois classement à base d'instances, est une technique très utilisée en reconnaissance des formes et en apprentissage automatique. Il s'agit d'une technique simple mais efficace pour des problèmes d'apprentissage supervisé avec des caractéristiques (ou attributs) continus. Après plus de 35 ans, les classifieurs PPV sont encore largement étudiés et utilisés pour résoudre des problèmes de reconnaissance des formes. On utilise aussi souvent ce classifieur comme système de base par rapport auquel on compare les nouveaux systèmes de classement.

Cependant, cette méthode de classement comporte des sous-problèmes difficiles tels que la sélection de prototypes ou le choix de la mesure de voisinage, qui s'avèrent des bons candidats à l'utilisation d'algorithmes évolutionnaires (AE). Dans ce chapitre, nous voyons différentes techniques d'AE pour concevoir des classifieurs PPV efficaces et robustes.. Quatre approches sont étudiées : 1) la sélection de prototypes avec un algorithme génétique (AG) multi-objectif ; 2) la conception de mesures de voisinage (distance) avec la programmation génétique (PG) ; 3) la co-évolution coopérative de la sélection de prototypes et de la conception de mesures de voisinage et 4) la co-évolution compétitive de la sélection d'ensembles d'évaluation de l'adéquation, d'une part, et de la sélection de prototypes et de la conception de mesures de voisinage en coopération, d'autre part. Pour chaque approche, les résultats seront présentés et ana-

lysés en utilisant des ensembles de données synthétiques et des ensembles de données tirées d'expériences réelles. Ces approches se veulent une illustration assez éloquente de l'approche méthodologique proposée à la section 1.4 de la thèse.

À la section 3.1, nous résumons les caractéristiques d'un classifieur PPV bien connu. Nous insistons particulièrement sur les trois caractéristiques principales de ce classifieur, dont deux seront optimisées (la sélection de prototypes et la mesure de voisinage). À la section 3.2, nous décrivons les quatre ensembles de données synthétiques utilisés dans ce chapitre. Ces ensembles de données ont été conçus pour souligner différents aspects du problème de classement. Ensuite, les quatre approches considérées sont présentées aux sections 3.3, 3.4 et 3.5. À la section 3.6, nous présentons des résultats obtenus pour cinq problèmes tirés du *UCI machine learning repository* [13]. Finalement, nous présentons à la section 3.7 une discussion sur les approches utilisées et les résultats obtenus, puis sur la contribution possible des AE à la conception de systèmes efficaces de reconnaissance des formes.

3.1 Classement par la règle du plus proche voisin

La conception de classifieurs PPV requiert trois spécifications distinctes [41] : un ensemble de prototypes, une règle de classement et une mesure de voisinage. Les prototypes sont des données représentatives utilisées par le classifieur pour attribuer des étiquettes de classes aux nouvelles données. La sélection de prototypes est une étape critique puisqu'elle implique d'utiliser un ensemble de données d'entraînement pour partitionner l'espace vectoriel d'entrée selon la mesure de voisinage. Si les prototypes sont bruités ou s'ils ne sont pas représentatifs des données du problème, alors la partition sera inadéquate et contribuera à diminuer le taux de reconnaissance. Aussi, il faut tenter de minimiser le nombre de prototypes sélectionnés car celui-ci a un effet direct sur le temps de classement.

La règle de classement par le PPV consiste essentiellement à choisir l'étiquette associée au plus proche voisin d'un vecteur d'entrée $\mathbf{x}$ inconnu. Plus formellement, supposons que $P = \{\mathbf{p}_1, ..., \mathbf{p}_m\}$ représente l'ensemble des m prototypes étiquetés et que $\mathbf{p} \in P$ est le prototype le plus près de $\mathbf{x}$. Selon la règle de base du PPV, $\mathbf{x}$ est assigné à la même étiquette que $\mathbf{p}$. Il s'agit d'une règle directe très efficace pour le classement de données exemptes de bruit. Toutefois, pour des problèmes réels, où le bruit est omniprésent dans les données d'entraînement, on utilise une règle légèrement modifiée, appelée règle des k-PPV, qui considère les k plus proches voisins pour assigner à $\mathbf{x}$ l'étiquette de classe le plus souvent rencontré, en brisant arbitrairement les égalités. Lorsque la valeur de k

augmente, l'effet du bruit est atténué mais les limites des classes sont adoucies.

On peut utiliser différentes mesures de voisinage pour le classement par les k-PPV. La norme L_a est une mesure de voisinage commune et générale qui peut être utilisée pour mesurer une distance entre deux vecteurs, $\mathbf{x} = [x_1 \cdots x_n]^T$ et $\mathbf{y} = [y_1 \cdots y_n]^T$.

$$L_a(\mathbf{x}, \mathbf{y}) = \left(\sum_{i=1}^{n} |x_i - y_i|^a \right)^{\frac{1}{a}} \tag{3.1}$$

L'instanciation la plus répandue de cette norme est la distance euclidienne, qui correspond au cas $a = 2$:

$$L_2(\mathbf{x}, \mathbf{y}) = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2} \tag{3.2}$$

Deux autres cas connus sont la distance de Manhattan ($a = 1$) :

$$L_1(\mathbf{x}, \mathbf{y}) = \sum_{i=1}^{n} |x_i - y_i|, \tag{3.3}$$

et la norme L_∞ :

$$L_\infty(\mathbf{x}, \mathbf{y}) = \max_{i=1}^{n}(|x_i - y_i|) \tag{3.4}$$

L'inconvénient avec la norme L_a est que les différentes dimensions de l'espace des caractéristiques doivent être normalisées à des échelles comparables. Le choix de a est arbitraire et permet d'accorder une importance différente aux composantes les plus grandes de $\mathbf{x} - \mathbf{y}$ relativement aux composantes les plus petites. Par exemple, dans L_1, l'importance est égale pour toutes les composantes. Les caractéristiques d'entrée doivent donc être à la même échelle (mêmes unités) afin que les distances soient cohérentes. Plus la valeur de a est grande, plus les grandes composantes ont de l'importance. À la limite, lorsque $a \to \infty$, la distance mesurée dépend seulement de la composante la plus grande. Il est important de mentionner que la famille des normes L_a représente seulement une possibilité parmi une infinité de mesures de voisinage. L'un des objectifs de ce chapitre est de développer une approche générale afin de découvrir automatiquement la meilleure mesure pour un problème de classement donné.

Il est possible de normaliser l'espace des caractéristiques plutôt que de changer cette mesure. Par exemple, il peut s'avérer utile dans certains cas d'effectuer une simple transformation d'échelle de chaque caractéristique dans l'intervalle $[0, 1]$. Toutefois, cette transformation peut déformer la distribution des données et compliquer la discrimination. Une solution plus sophistiquée consiste à appliquer une transformation blanchissante [41], qui élimine d'abord la dépendance linéaire entre les caractéristiques par une translation et une rotation (transformation rigide) avant d'appliquer la transformation

d'échelle. Selon une hypothèse des distributions multinormales, cette transformation blanchissante est pertinente, quoiqu'il n'est pas certain qu'elle augmente le pouvoir de discrimination de la mesure choisie. Cela dépend des distributions véritables des données.

Enfin, il faut noter que le classifieur par les k-PPV possède une complexité de calcul de $O(m \log k)$ pour traiter un vecteur $\mathbf{x}$ inconnu en entrée. Il y a donc un intérêt certain à réduire à la fois le nombre de voisins (k) et la taille de l'ensemble des prototypes (m) afin d'obtenir des classifieurs plus efficaces.

3.2 Ensembles de données synthétiques

Les ensembles de données synthétiques conçus pour la première série d'expériences sont représentés à la figure 3.1. Les données possèdent un espace 2D des caractéristiques et deux classes, elles peuvent donc être visualisées facilement sur papier. Les ensembles de données sont :

1. *Chevauchées* — deux distributions multinormales de 250 points allongées horizontalement, avec un chevauchement important entre les classes selon le plus petit de leurs axes principaux (le taux de reconnaissance bayésien optimal est de 76.8 %) ;
2. *Inclinées* — des distributions semblables à celles décrites précédemment mais inclinées de 30° (le taux de reconnaissance bayésien optimal est de 90.4 %) ;
3. *Dentelées* — deux distributions uniformes non chevauchées de 971 et 1029 points respectivement, imbriquées à la frontière des deux classes de données (séparables à 100 %) ;
4. *Spirales* — les spirales classiques entrelacées de 193 points chacune.

Chaque ensemble de données a été partitionné aléatoirement en quatre sous-ensembles de même taille : un ensemble de données d'*entraînement*, un ensemble de données d'*évaluation de l'adéquation*, un ensemble de données de *validation* et un ensemble de données de *test*. Ces mêmes sous-ensembles sont utilisés dans toutes les expérimentations impliquant des données synthétiques. L'AG sélectionne les prototypes de classes à partir de l'ensemble d'entraînement. L'ensemble d'évaluation de l'adéquation est utilisé par l'AG et la PG pour évaluer l'adéquation de chaque individu. L'ensemble de validation est utilisé pour sélectionner le meilleur individu de chaque génération, c'est-à-dire celui qui a la meilleure capacité de généralisation. Finalement, l'ensemble de test est utilisé seulement à la fin du processus d'évolution pour évaluer la performance des meilleures solutions obtenues. Les évolutions sont répétées 10 fois pour permettre une bonne évaluation statistique des résultats.

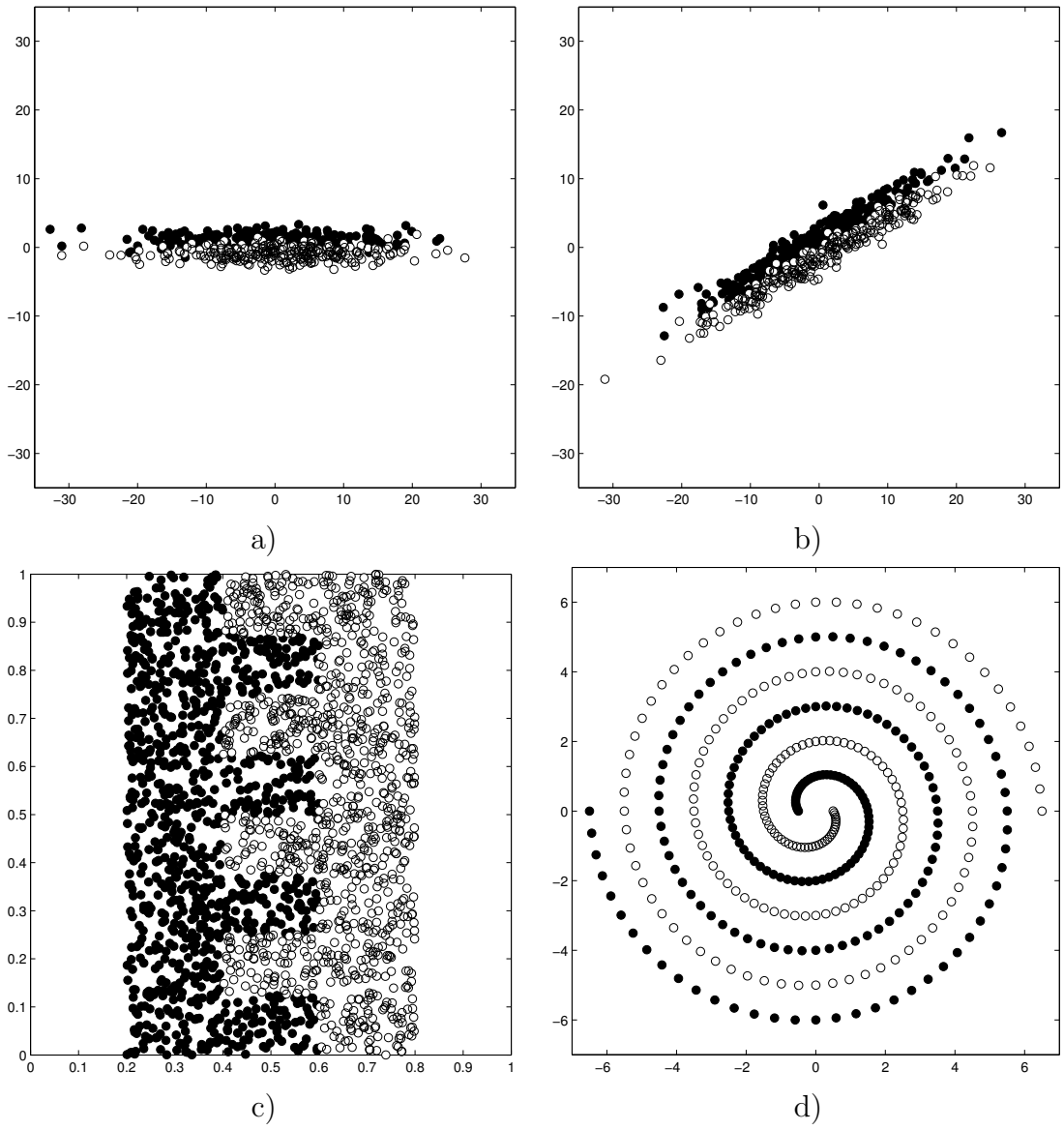


FIG. 3.1 – Ensembles de données synthétiques : a) données chevauchées ; b) données inclinées ; c) données dentelées ; d) spirales.

TAB. 3.1 – Performances obtenues pour les ensembles de données synthétiques des classifieurs 1-PPV classiques, en faisant varier la norme L_a ($a = 1, 2, \infty$) et la normalisation (aucune, d'échelle et blanchissante) de l'espace des caractéristiques d'entrée. Les résultats donnés sont sur les ensembles de test.

Ensemble de données	Performance de base (L_2)	Meilleure configuration		
		Distance	Normalisation	Gain en reconnaissance
Chevauchées	69.6 %	L_∞	d'échelle	2.4 %
Inclinées	80.0 %	L_2	blanchissante	8.8 %
Dentelées	94.2 %	L_2	aucune	0.0 %
Spirales	81.4 %	L_1	d'échelle	−2.0 %

Le tableau 3.1 présente un résumé des performances de base obtenues pour ces ensembles de données synthétiques avec un classifieur 1-PPV, la norme L_2 et aucune normalisation de l'espace des caractéristiques d'entrée. Le tableau donne également la meilleure performance qui peut être obtenue en variant les paramètres de base du 1-PPV : utilisation de la norme L_1, L_2 ou L_∞ ; soit sans normalisation, soit avec une transformation d'échelle ou blanchissante. Dans toutes les expériences présentées dans ce chapitre, on considère un seul voisin.

La colonne *Gain en reconnaissance* du tableau 3.1 donne, pour chaque ensemble de données, l'augmentation de la performance observée pour la meilleure combinaison de paramètres. Par exemple, pour l'ensemble des données chevauchées, le meilleur individu obtenu par un classifieur avec la norme L_∞ et une transformation d'échelle affiche une augmentation de 2.4 % par rapport à la performance de base. Dans ce cas, une simple transformation d'échelle semble optimale puisque les données sont allongées et les deux caractéristiques sont linéairement indépendantes. Pour l'ensemble des données inclinées, où les deux caractéristiques sont linéairement dépendantes, une transformation blanchissante semble donner le meilleur résultat car elle permet une décorrélation des caractéristiques. Pour les ensembles des données dentelés et des spirales, il semble qu'il ne soit pas possible d'améliorer les performances de base. La valeur négative du gain en reconnaissance pour l'ensemble des spirales s'explique par le fait que le meilleur individu sélectionné en utilisant les ensembles d'évaluation et de validation (qui surpassait légèrement la performance de base) était en fait un peu en deçà de la performance de base lorsque évalué sur l'ensemble de test.

Aux trois prochaines sections, nous étudions quatre approches pour améliorer la performance du classifieur PPV. Nous voyons également qu'il est possible à la fois de

1. Initialiser l'ensemble des prototypes P avec l'ensemble complet d'entraînement ;
2. Pour chaque $\mathbf{p}_i \in P$ (aléatoirement) :
   (a) Trouver dans $P - \{\mathbf{p}_i\}$ les k plus proches voisins de $\mathbf{p}_i$;
   (b) Trouver l'étiquette de classe majoritaire θ parmi ces voisins, avec bris arbitraire d'égalité ;
   (c) Si θ correspond à la véritable étiquette de $\mathbf{p}_i$, retirer $\mathbf{p}_i$ de P.

FIG. 3.2 – Algorithme d'édition de Wilson pour la sélection de prototypes.

1. Soit $X = \{\mathbf{x}_1 \cdots \mathbf{x}_m\}$, l'ensemble complet d'entraînement permuté aléatoirement ;
2. Initialiser $P = \{\mathbf{x}_1\}$, l'ensemble initial des prototypes, et $X = X - \{\mathbf{x}_1\}$;
3. Pour chaque élément $\mathbf{x}_i \in \mathbf{X}$:
   (a) Soit θ, l'étiquette de classe du plus proche voisin de $\mathbf{x}_i$ dans P ;
   (b) Si θ est différent de l'étiquette de $\mathbf{x}_i$, alors $P = P + \{\mathbf{x}_i\}$ et $X = X - \{\mathbf{x}_i\}$.
4. Si $|X| > 0$ et si la condition 3(b) est vraie au moins une fois, alors retourner à l'étape 3 ;
5. Retourner P comme ensemble condensé des prototypes.

FIG. 3.3 – Algorithme de condensation de Hart pour la sélection de prototypes.

réduire le nombre de prototypes et d'améliorer le taux de reconnaissance.

3.3 Sélection de prototypes avec un algorithme génétique multi-objectif

La sélection de prototypes pour des classifieurs PPV est un thème important qui a été souvent abordé dans les domaines de la reconnaissance des formes et de l'apprentissage automatique. Les travaux classiques de Wilson [156] tournent autour de l'algorithme simple décrit à la figure 3.2. L'édition de Wilson tend à éliminer les instances bruitées de l'ensemble d'entraînement. La règle de condensation de Hart [58], présentée à la figure 3.3, est une autre approche classique qui consiste à éliminer les prototypes qui ne contribuent pas au classement. Il est courant d'appliquer l'édition de Wilson suivie de l'algorithme de condensation de Hart pour une réduction maximale de la taille de

TAB. 3.2 – Résultats de l'édition de Wilson et de la condensation de Hart sur les ensembles de données synthétiques ($k = 1$ avec la norme L_2 et aucune normalisation). La colonne *Perf. base* correspond au taux de reconnaissance obtenu avec l'ensemble complet d'entraînement comme ensemble de prototypes, la colonne *Gain recon.* est la différence entre le taux de reconnaissance après la sélection de prototypes et la performance de base et la colonne *Taux sélect.* donne le rapport du nombre de prototypes sélectionnés à la taille de l'ensemble d'entraînement.

		Édition de Wilson		Cond. de Hart		Wilson + Hart	
Ensemble de données	Perf. base	Gain recon.	Taux sélect.	Gain recon.	Taux sélect.	Gain recon.	Taux sélect.
Chevauchées	69.6 %	0.8 %	68.8 %	−8.0 %	51.2 %	0.0 %	16.8 %
Inclinées	80.0 %	0.0 %	87.2 %	4.0 %	28.0 %	3.2 %	15.2 %
Dentelées	94.2 %	−0.4 %	95.2 %	−1.8 %	14.8 %	−0.4 %	9.0 %
Spirales	81.4 %	−17.5 %	79.2 %	−5.2 %	53.1 %	−23.7 %	28.1 %

l'ensemble des prototypes. Le tableau 3.2 présente les taux de reconnaissance et de sélection obtenus sur les ensembles de test après l'édition de Wilson, la condensation de Hart et après l'édition de Wilson suivie de la condensation de Hart, respectivement. Les résultats démontrent que l'édition de Wilson améliore légèrement le taux de reconnaissance pour l'ensemble des données chevauchées, qu'elle n'a aucun effet sur l'ensemble des données inclinées et qu'elle diminue le taux de reconnaissance pour les ensembles des données dentelées et des spirales. En général, la réduction du nombre de prototypes est faible. La condensation de Hart est beaucoup plus efficace dans la réduction du nombre de prototypes mais elle tend aussi à diminuer le taux de reconnaissance, sauf pour l'ensemble des données inclinées. La combinaison de l'édition de Wilson et de la condensation de Hart réduit encore davantage le nombre de prototypes et conserve le taux de reconnaissance à un niveau comparable à celui de l'application de l'édition de Wilson seule. En conclusion, ces méthodes classiques de sélection de prototypes sont très efficaces pour réduire le nombre de prototypes mais ne peuvent pas améliorer en même temps le taux de reconnaissance. La pire performance concerne l'ensemble des spirales qui sont essentiellement sans bruit et quelque peu sous-échantillonnées dans l'ensemble d'entraînement (1/4 des spirales).

Les AE ont été fréquemment utilisés pour la sélection de prototypes. Cano et al. [20] ont présenté une analyse détaillée sur l'utilisation de différents AE pour la sélection de prototypes et Ho et al. [64] et Kuncheva et Jain [83] ont démontré l'efficacité des AG pour la sélection simultanée des prototypes et des caractéristiques. Dans les trois cas, on utilise une somme pondérée comme mesure d'adéquation. Dans ce chapitre, nous avons

délibérément choisi d'effectuer seulement la sélection de prototypes, comme il a été fait dans [20], étant donné que la sélection des caractéristiques est faite indirectement dans les sections suivantes lorsque nous évoluons des mesures de voisinage spécifiques aux problèmes à résoudre. Yang et Honavar [159] et Oliveira et al. [104] ont également abordé la sélection de caractéristiques avec des AG, approche qui peut être directement adaptée au problème de la sélection de prototypes. Dans les deux cas, un AG multi-objectif est utilisé.

Comme représentation naturelle d'un AG pour la sélection de prototypes, on utilise des chaînes de bits où chaque bit est associé à un échantillon distinct de l'ensemble d'entraînement. Une valeur de 1 signifie que l'échantillon est sélectionné comme un prototype et une valeur de 0 signifie que l'échantillon est rejeté. Il y a deux objectifs d'optimisation : minimiser le nombre de prototypes et maximiser le taux de reconnaissance. Ces objectifs sont contradictoires puisque tel qu'observé au tableau 3.2, la sélection d'un plus petit nombre de prototypes cause une diminution du taux de reconnaissance. La mesure d'adéquation à un objectif utilisée dans [64, 83] est une somme pondérée du taux de reconnaissance et du nombre de prototypes (et de caractéristiques) sélectionnés. L'avantage d'utiliser une mesure d'adéquation multi-objectif et un algorithme de sélection naturelle basé sur l'optimalité de Pareto est d'éliminer tout biais induit par une somme pondérée, ce qui permet de reporter le choix du meilleur rapport du taux de reconnaissance au nombre de prototypes à la fin du processus d'optimisation. De plus, cela uniformise la pression sur les deux objectifs durant le processus d'optimisation et élimine tout biais prématuré vers une région spécifique du front de Pareto. À la fin de l'évolution, on peut donc sélectionner la solution finale selon le choix approprié du rapport du taux de reconnaissance au nombre de prototypes.

Dans un contexte d'optimisation multi-objectif, le choix du meilleur individu de l'évolution peut être fait de différentes façons. Pour la présente application, on pose que le meilleur individu d'une génération est tout simplement celui qui obtient le meilleur taux de reconnaissance sur l'ensemble d'évaluation de l'adéquation. Cette méthode permet une sélection de l'individu à une extrémité du front de Pareto. On pose également qu'à la fin du processus d'évolution, le meilleur individu est celui qui obtient le meilleur taux de reconnaissance sur l'ensemble de données de validation, parmi les meilleurs individus de chaque génération sur l'ensemble d'évaluation de l'adéquation. Ceci permet de sélectionner l'individu ayant la meilleure performance globale, sans occasionner de surapprentissage des ensembles de données. Cette procédure est utilisée dans toutes les expériences suivantes du chapitre. À notre connaissance, seuls Yu *et al.* [160] ont appliqué une procédure similaire dans un contexte de prédiction de valeur de titres financiers. On estime qu'une telle procédure appliquée à chaque génération peut être extrêmement bénéfique dans un contexte d'apprentissage évolutionnaire où le surapprentissage est

TAB. 3.3 – Résultats de la sélection de prototypes avec un AG multi-objectif sur les ensembles de données synthétiques pour un classifieur 1-PPV utilisant la norme L_2 et aucune normalisation de l'espace des caractéristiques d'entrée. La colonne *Meilleur des* 10 *essais* correspond au meilleur résultat parmi les meilleurs individus des 10 évolutions indépendantes et la colonne *Moyenne des* 10 *essais* correspond à la moyenne de ces 10 meilleurs individus.

Ensemble de données	Performance de base	Meilleur des 10 essais		Moyenne des 10 essais	
		Gain en reconnaissance	Taux de sélection	Gain en reconnaissance	Taux de sélection
Chevauchées	69.6 %	2.4 %	60.8 %	6.4 %	60.0 %
Inclinées	80.0 %	−1.6 %	46.4 %	−0.8 %	67.6 %
Dentelées	94.2 %	0.6 %	66.0 %	0.7 %	67.1 %
Spirales	81.4 %	−4.1 %	65.6 %	−5.1 %	75.4 %

parfois très présent en fin d'évolution. Il est cependant certain qu'avec une telle procédure, une quantité non négligeable de données n'est pas utilisée pour entraîner les individus.

Le tableau 3.3 présente un résumé des résultats obtenus sur les quatre ensembles de données synthétiques, en utilisant un AG multi-objectif pour la sélection de prototypes, une mesure L_2 et aucune normalisation de espace des caractéristiques d'entrée. Il est à noter que le meilleur individu est choisi en fonction du taux de reconnaissance et non en fonction du taux de sélection. Un autre point à souligner concerne les gains en reconnaissance pour le meilleur des 10 essais, qui sont inférieurs aux gains de la moyenne des 10 essais pour les ensembles de données chevauchées et dentelées. Cette anormalité est explicable du fait que le meilleur des 10 essais est sélectionné selon le taux de reconnaissance sur l'ensemble de données de validation, alors que les résultats du tableau sont calculés sur l'ensemble de test.

L'algorithme utilisé pour la sélection multi-objectif est le *Niched Pareto Genetic Algorithm 2* (NPGA2) [44]. Les paramètres de l'AG sont : une population de taille 1000 ; 100 générations ; la sélection par tournoi NPGA2 avec 2 individus et un rayon de niche (σ_{share}) de 1.0 ; une probabilité *a priori* d'initialisation des bits à 1 de 0.75 ; une probabilité de croisement uniforme de 0.3 et une probabilité de mutation par bit de 0.01. Les valeurs utilisées pour la taille de la population et le nombre de générations de l'évolution ont été estimées suite à quelques expérimentations préliminaires, afin de générer des résultats intéressants avec des temps de calculs raisonnables. Il faut garder à l'esprit que l'efficacité computationnelle n'est pas un objectif de l'application et donc

que l'on a possiblement sur-estimé la taille de la population et le nombre de générations nécessaires à l'obtention de résultats satisfaisants. Les autres paramètres de l'algorithme sont des valeurs par défaut de l'outil logiciel, excepté pour la probabilité *a priori* d'initialisation des bits qui a fait l'objet d'un réglage spécial suite à des observations faites durant les expérimentations préliminaires.

Puisque le premier ensemble des données est composé de nuages de points chevauchés, la sélection de prototypes tend à éliminer les échantillons qui introduisent de mauvais classements. Étrangement, le taux de reconnaissance est légèrement plus faible que la performance de base pour l'ensemble des données inclinées. Les améliorations du taux de reconnaissance pour l'ensemble des données dentelées sont très faibles, ce qui n'est pas surprenant puisque les classes ne se chevauchent pas dans cet ensemble de données. De plus, le taux de sélection n'est pas aussi bon qu'avec la condensation de Hart. Pour les spirales, la reconnaissance est un peu plus faible mais le taux de sélection est très bon. La figure 3.4 illustre les fronts de Pareto associés avec le *meilleur des 10 essais* du tableau 3.3. Ces fronts de Pareto montrent que toutes les solutions non dominées obtenues au cours d'une évolution couvrent une étendue importante des rapports du taux de reconnaissance au taux de sélection.

En général, les résultats présentés démontrent que la sélection de prototypes permet des améliorations du taux de reconnaissance, tout en atteignant des taux de sélection acceptables. En comparaison aux résultats obtenus avec les algorithmes d'édition de Wilson et de condensation de Hart, les taux de reconnaissance sont égaux et parfois meilleurs, principalement à cause de la sélection naturelle basée sur l'optimalité de Pareto et de la stratégie de sélection du meilleur individu de chaque évolution. D'autre part, les taux de sélection de prototypes ne sont pas comparables à ceux obtenus avec l'algorithme de condensation de Hart en utilisant la même stratégie de sélection du meilleur individu de chaque évolution. Il est possible de tracer le front de Pareto des solutions et de choisir la solution non dominée la plus près du rapport désiré du taux de reconnaissance au taux de sélection. Les résultats présentés à la figure 3.4 illustrent cette méthode en montrant que l'on peut obtenir des solutions non dominées différentes de celles obtenues avec l'édition de Wilson et la condensation de Hart. De plus, il apparaît évident que le processus de recherche darwinienne des AG permet une exploration plus complète des différentes sélections de prototypes. Comme à l'habitude, la nature stochastique des AE fait en sorte qu'ils ne garantissent pas la convergence ni la répétabilité des résultats. De plus, le processus d'évaluation de l'adéquation requiert une lourde tâche de calcul. Sur un PC ordinaire (AMD Athlon 1.2 GHz), il faut environ une demi-heure pour effectuer une telle évolution de sélection de prototypes en utilisant un AG, comparativement à quelques secondes pour des heuristiques déterministes tels que l'édition de Wilson et la condensation de Hart.

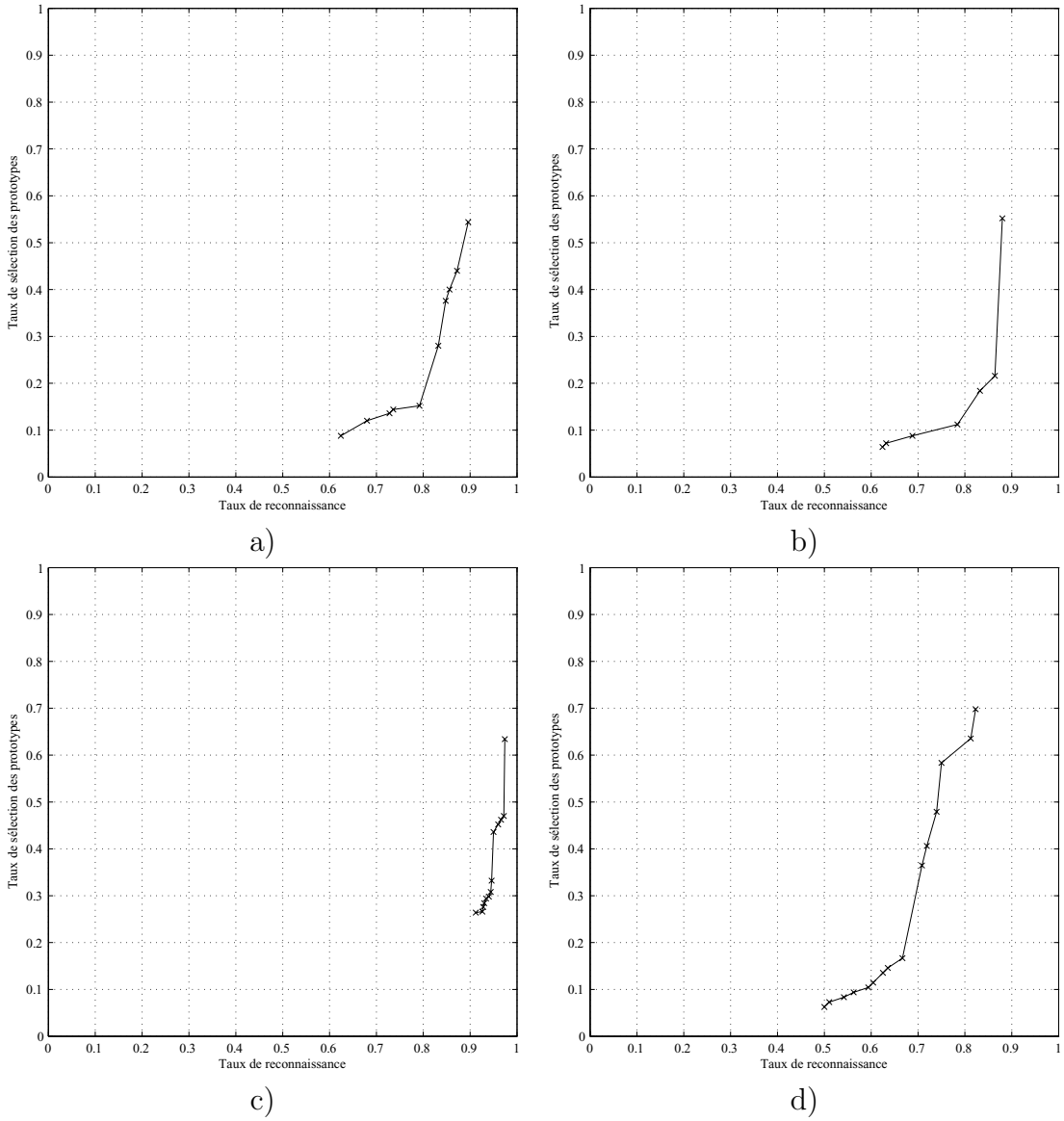


FIG. 3.4 – Fronts de Pareto du meilleur des 10 essais : a) données chevauchées ; b) données inclinées ; c) données dentelées et d) spirales.

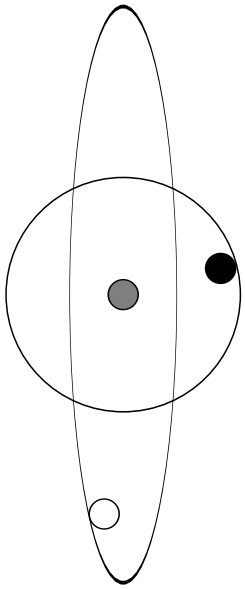


FIG. 3.5 – Utilisation de mesures de voisinage adaptées au problème à résoudre.

3.4 Évolution des mesures de voisinage avec la programmation génétique

Le choix de la mesure de voisinage peut être déterminant sur les performances d'un classifieur PPV. La figure 3.5 illustre l'idée d'utiliser une mesure de distance adaptée au problème à résoudre. Supposons que les trois points de couleurs différentes (noir, gris et blanc) de la figure représentent trois données dans un espace à deux caractéristiques. Supposons aussi qu'une distance euclidienne classique est utilisée, avec une distance unitaire au point gris représentée par le grand cercle qui l'entoure. Selon cette distance, on voit clairement que le point noir est plus près du point gris que le point blanc. Supposons maintenant qu'une autre mesure de distance est utilisée, avec cette fois une distance unitaire représentée par l'ellipse entourant le point gris. Dans ce cas, la donnée représentée par le point blanc est plus près du point gris. S'il s'avère que la véritable distribution statistique des données à classer est vaguement elliptique, comme c'est le cas pour les ensembles de données chevauchées et inclinées de la section 3.2, la mesure de voisinage avec une distance unitaire représentée à la figure 3.5 par une ellipse pourrait alors être plus adaptée aux données qu'une mesure de distance euclidienne. L'utilisation d'une mesure de voisinage adaptée aux données semble donc être une excellent chose. Cependant, le choix de cette mesure peut être très difficile à effectuer par des humains dans le cas général. Il pourrait alors être très intéressant de développer des méthodes automatiques pour générer de telles mesures de voisinage adaptées à la distribution des

données du problème à résoudre.

La PG permet l'évolution de programmes qui utilisent un ensemble d'instructions spécifiques au problème à résoudre. Dans le contexte de la reconnaissance des formes, la PG peut être utilisée pour concevoir automatiquement un modèle de données spécifique au problème. Pour le classifieur PPV, cette modélisation peut se faire en évoluant les mesures de voisinage afin de maximiser le taux de reconnaissance. Cela permet une très grande flexibilité pour l'exploration au-delà de la modélisation linéaire paramétrique habituelle des données.

Demiröz et Güvenir [38] ont utilisé un AG spécialisé pour évoluer les poids de chaque caractéristique dans l'espace des distances euclidiennes. D'autre part, Raymer et al. [123] ont fait évoluer une matrice de transformation linéaire de dimensions $n \times m$, où n est la taille de l'espace d'entrée initiale, m est la taille de l'espace d'entrée transformée et $m < n$ afin de réduire la dimensionnalité de l'espace des caractéristiques utilisé dans le classement par les k-PPV. Il ne s'agit pas d'une évolution explicite de la mesure de voisinage, mais cette approche est pertinente dans le présent contexte étant donné qu'elle cible le développement d'un modèle de données spécifique au problème à résoudre, en appliquant une transformation linéaire de l'espace d'entrée. Pour ce qui est de l'application de la PG à l'ingénierie de caractéristiques dans le contexte de la reconnaissance des formes, les travaux de Sherrah et al. [132, 133] et de Bot [15] utilisent la PG pour la construction de caractéristiques de haut niveau à partir de caractéristiques brutes, dans le but d'améliorer le taux de classement sur un certain ensemble de données. Les ensembles de fonctions et de terminaux utilisés dans [15, 132] permettent une construction linéaire des caractéristiques de haut niveau à partir de caractéristiques brutes, tandis que dans [133], des fonctions non linéaires sont utilisées, ce qui peut s'avérer intéressant.

Nous proposons ici d'évoluer des programmes où la structure de base des données traitées par les arbres est un vecteur de n valeurs, où n correspond à la taille de l'espace des caractéristiques. Un cas particulier de ce vecteur correspond à $n = 1$, ce qui représente un scalaire. Pour assurer la propriété de fermeture des programmes évolués, toutes les primitives de PG sont conçues pour traiter les combinaisons de scalaires et de vecteurs. Par exemple, la primitive ADD peut traiter deux vecteurs, deux scalaires ou un vecteur et un scalaire. Dans ce dernier cas, le scalaire est additionné à chaque composante du vecteur pour produire un autre vecteur de dimension n. Le tableau 3.4 présente l'ensemble complet des primitives utilisées pour toutes les évolutions de PG. L'ensemble comprend des primitives arithmétiques telles que $+$, $-$, $\times$ et $\div$; des fonctions qui retournent le maximum, le minimum et la moyenne de deux arguments ; des mesures de distance telles que la distance de Manhattan, euclidienne et L_∞ ; des fonctions qui changent la dimension des données traitées de n vers 1 en extrayant la

TAB. 3.4 – Ensemble des primitives utilisées pour les évolutions des mesures de voisinage avec la PG.

Nom	Args	Description
ADD	2	Addition de deux vecteurs.
SUB	2	Soustraction de deux vecteurs.
MUL	2	Multiplication de deux vecteurs.
DIV	2	Division protégée [76] de deux vecteurs ; division où un dénominateur dans $[-0.001, 0.001]$ retourne 1.0.
MAX	2	Maximum de deux vecteurs.
MIN	2	Minimum de deux vecteurs.
MEA	2	Moyenne de deux vecteurs.
MAN	1	Norme L_1 d'un vecteur.
EUC	1	Norme L_2 d'un vecteur.
LIF	1	Norme L_∞ d'un vecteur.
SUM	1	Somme des composantes d'un vecteur.
MXV	1	Composante maximale d'un vecteur.
MIV	1	Composante minimale d'un vecteur.
ROT	1	Rotation aléatoire du vecteur en argument. Si l'argument est un scalaire, celui-ci est retourné sans modification. Sinon, la primitive retourne le résultat d'une rotation du vecteur en argument, en utilisant une matrice de rotation formé à partir de $n-1$ angles générés aléatoirement dans l'intervalle $[0, \pi]$. La matrice de rotation est générée soit pendant l'initialisation, soit après une mutation, de façon similaire aux constantes éphémères générées aléatoirement [76].
EBV	0	Constantes éphémères générées aléatoirement [76] formées de vecteurs binaires de dimension n, où n est le nombre d'attributs des données. Les vecteurs sont dits binaires car leurs composantes ont comme valeur 0.0 ou 1.0.
ESC	0	Constantes éphémères générées aléatoirement [76] formées de vecteurs scalaires générés dans l'intervalle $[-1.0, 1.0]$.
PMX	0	Vecteur maximal de l'ensemble des prototypes (coin supérieur droit du rectangle englobant les données en 2D).
PMI	0	Vecteur minimum de l'ensemble des prototypes (coin inférieur gauche du rectangle englobant les données en 2D).
PME	0	Vecteur moyen de l'ensemble des prototypes.
P	0	Une instance de l'ensemble des prototypes.
I	0	Vecteur d'entrée inconnu (la donnée à classer).

somme, le maximum ou le minimum des composantes d'un vecteur et une fonction de rotation éphémère générée aléatoirement. Les feuilles utilisées sont des vecteurs binaires éphémères générés aléatoirement ; des scalaires éphémères générés aléatoirement dans le domaine $[-1, 1]$; les vecteurs maximal, minimal et moyen des données de l'ensemble des prototypes ; le vecteur d'entrée du prototype spécifique et le vecteur d'entrée inconnu. Lorsque le résultat reçu de la racine d'un programme est un vecteur de n valeurs, ce vecteur est transformé en un scalaire à l'aide de la norme L_1. L'évaluation de l'adéquation a été implantée en calculant les distances entre un vecteur de test donné et tous les prototypes par une seule interprétation de l'arbre de PG. Cette approche a été utilisée dans [87] afin de réduire significativement le temps de calcul comparativement à une interprétation pour chaque prototype, avec une vitesse observée de 25 % à 50 % supérieure selon l'ensemble de données.

Une mesure d'adéquation multi-objectif est utilisée pour l'évolution de la mesure de voisinage, afin de maximiser le taux de reconnaissance et de minimiser la taille de l'arbre simultanément. Cette approche présente l'avantage de favoriser l'évolution de petits arbres et donc l'émergence de modèles de données simples. En effet, il est reconnu que les modèles simples soient plus généraux. Iba et al. [67] proposent d'utiliser le nombre de nœuds des arbres de PG comme estimateur de leur complexité, selon le principe de la longueur minimale de description (*minimum description length*) [126]. Un autre avantage important de l'approche est de limiter le phénomène de surcharge (*bloat*) [9, 136] qui tend à produire des arbres de plus en plus gros, de génération en génération. L'approche utilisée restreint effectivement la croissance du code à l'aide de la sélection multi-objectif, tel que présenté dans [36, 86]. Comme pour la sélection de prototypes avec les AG, le meilleur individu de l'évolution est celui qui obtient le meilleur taux de reconnaissance dans l'ensemble de validation, parmi tous les meilleurs individus de chaque génération obtenus dans l'ensemble d'évaluation de l'adéquation.

Les paramètres utilisés pour les expérimentations sont : une population de taille 1000 ; 100 générations ; la sélection par tournois NPGA2 avec 2 individus et un rayon de niche (σ_{share}) de 1.0 ; une probabilité de croisement de 0.9 ; une probabilité de mutation réductrice (*shrink mutation*) de 0.05 ; une probabilité de mutation d'inversion de nœuds (*swap mutation*) de 0.05 ; une probabilité de mutation de sous-arbres de 0.05 et une profondeur d'arbre maximale de 17. La remarque faite à la section précédente sur le réglage des paramètres de l'algorithme s'applique toujours, c'est-à-dire que les paramètres par défauts de l'outil logiciel sont utilisés, excepté pour la taille de la population et du nombre de générations qui ont été posés suite à quelques expérimentations préliminaires.

Le tableau 3.5 présente les résultats de l'ingénierie des mesures de voisinage avec

TAB. 3.5 – Résultats de l'évolution de mesures de voisinage obtenues avec la PG sur les ensembles de données synthétiques (un voisin et aucune normalisation de l'espace des caractéristiques d'entrée).

Ensemble de données	Performance de base	Meilleur des 10 essais Gain en reconnaissance	Moyenne des 10 essais Gain en reconnaissance
Chevauchées	69.6 %	8.0 %	6.8 %
Inclinées	80.0 %	4.8 %	5.3 %
Dentelées	94.2 %	0.6 %	0.6 %
Spirales	81.4 %	8.3 %	6.1 %

la PG pour les quatre ensembles de données synthétiques. Les résultats montrent une bonne amélioration du taux de reconnaissance pour les données chevauchées, inclinées et les spirales par rapport aux performances de base. Comparativement aux résultats obtenus avec la configuration du meilleur 1-PPV, présentés au tableau 3.1, l'amélioration est significativement plus importante en utilisant les mesures de voisinage évoluées pour l'ensemble des spirales, un peu plus grande pour l'ensemble des données chevauchées, mais un peu moins grande pour l'ensemble des données inclinées. Ce dernier résultat s'explique par le fait que, dans ce cas, la transformation blanchissante utilisée par le meilleur 1-PPV standard est optimale pour cet ensemble de données.

La figure 3.6 illustre les partitions de l'espace d'entrée pour les meilleures mesures de voisinage obtenues avec la PG et présentées au tableau 3.5. Les régions en gris de chacune des partitions de l'espace s'avèrent être associées par classement à la classe des données représentées par des points noirs, alors que les régions en blanc sont associées par classement à la classe des données représentées par des points blancs. Les partitions de l'espace d'entrée pour les ensembles de données inclinées et spirales contiennent quelques artefacts dans des régions non couvertes par les ensemble d'entraînement et d'évaluation de l'adéquation. D'autre part, pour les ensembles de données chevauchées et dentelées, les espaces d'entrée semblent convenablement partitionnés en deux classes distinctes.

3.5 Co-évolution de classifieurs

Dans les sections précédentes, les classifieurs 1-PPV étaient conçus en sélectionnant d'abord les prototypes avec un AG multi-objectif et en faisant ensuite évoluer la mesure

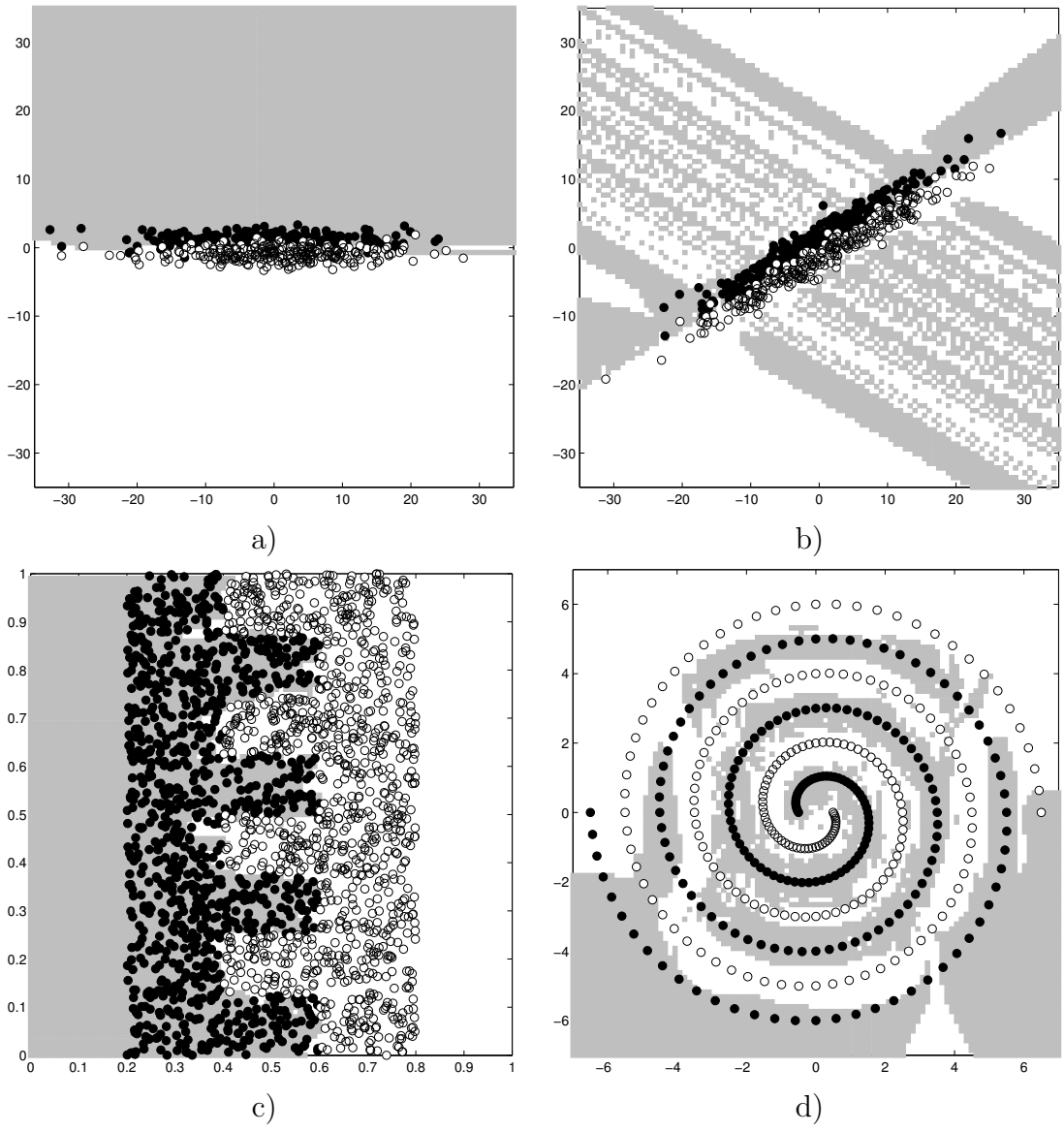


FIG. 3.6 – Espace des caractéristiques d'entrée partitionné par les meilleures mesures de voisinage obtenues avec la PG : a) ensemble des données chevauchées ; b) ensemble des données inclinées ; c) ensemble des données dentelées ; d) ensemble des spirales.

TAB. 3.6 – Résultats de la co-évolution de la sélection de prototypes et de la mesure de voisinage sur les ensembles de données synthétiques.

Ensemble de données	Performance de base	Meilleur des 10 essais		Moyenne des 10 essais	
		Gain en reconnaissance	Taux de sélection	Gain en reconnaissance	Taux de sélection
Chevauchées	69.6 %	8.8 %	55.2 %	6.6 %	54.6 %
Inclinées	80.0 %	8.0 %	65.6 %	2.2 %	65.5 %
Dentelées	94.2 %	0.6 %	64.8 %	0.9 %	64.0 %
Spirales	81.4 %	8.3 %	72.9 %	1.7 %	70.0 %

de voisinage avec la PG. Ces deux approches sont complètement indépendantes : il est possible de sélectionner des prototypes pour tout type de mesure de voisinage, et inversement. Dans ce contexte, la co-évolution (voir section 1.2.7) s'avère très intéressante pour une évolution simultanée de la sélection de prototypes et de la mesure de voisinage.

Nous examinons les deux stratégies. Premièrement, nous co-évoluons de façon coopérative la sélection de prototypes en utilisant un AG multi-objectif (tel que présenté à la section 3.3) et l'ingénierie de la mesure de voisinage avec la PG (tel que présenté à la section 3.4). On évalue l'adéquation pour l'espèce de la sélection de prototypes en calculant le taux de reconnaissance de chaque individu, en conjonction avec la meilleure mesure de voisinage de la génération précédente. De même, on évalue l'adéquation pour l'espèce de la mesure de voisinage en calculant le taux de reconnaissance de chaque individu, en conjonction avec le meilleur ensemble de prototypes de la génération précédente. Pour la première génération, les individus de référence sont sélectionnés aléatoirement dans chaque espèce. On peut qualifier cette approche d'assignation optimiste du crédit de collaboration [155].

Le tableau 3.6 présente les résultats de la co-évolution de la sélection de prototypes et de la mesure de voisinage. Pour chaque espèce, les paramètres utilisés sont les mêmes que ceux présentés aux sections précédentes. Les résultats démontrent qu'il est possible d'obtenir des taux de reconnaissance comparables à ceux du tableau 3.5 tout en réduisant jusqu'à 45 % le nombre de prototypes.

En deuxième lieu, nous avons introduit une troisième espèce qui évolue en compétition avec les deux espèces précédentes. Cette troisième espèce est composée de chaînes de bits de la même taille que l'ensemble des données d'évaluation de l'adéquation. Chaque bit de cette chaîne indique si l'échantillon correspondant dans l'ensemble de données est utilisé ou non pour l'évaluation de l'adéquation. Le nombre de uns dans chaque chaîne de bits est constant. En effet, les opérateurs d'initialisation de la chaîne de bits, de croisement uniforme et de mutation par inversion de bits sont modifiés de façon à ne

TAB. 3.7 – Résultats de la co-évolution de trois espèces sur les ensembles de données synthétiques : coopération de la sélection de prototypes et de la mesure de voisinage, en compétition avec la sélection d'un sous-ensemble de données d'évaluation, avec des classifieurs 1-PPV.

Ensemble de données	Performance de base	Meilleur des 10 essais		Moyenne des 10 essais	
		Gain en reconnaissance	Taux de sélection	Gain en reconnaissance	Taux de sélection
Chevauchées	69.6 %	7.2 %	20.8 %	5.7 %	30.3 %
Inclinées	80.0 %	6.4 %	55.2 %	4.2 %	56.8 %
Dentelées	94.2 %	1.2 %	57.0 %	0.8 %	60.3 %
Spirales	81.4 %	11.3 %	66.7 %	2.2 %	65.9 %

pas changer le nombre de uns de chaque chaîne. L'évolution possède un seul objectif, qui est de minimiser le taux de reconnaissance. Les individus de cette troisième espèce sont couplés de nouveau avec les individus des deux autres espèces selon une stratégie d'assignation maximale du crédit : chaque individu de cette troisième espèce, qui est une sélection d'échantillons difficiles de l'ensemble d'évaluation de l'adéquation, est évalué avec la meilleure combinaison de sélection de prototypes et de mesure de voisinage trouvée à la génération précédente, alors que le sous-ensemble d'évaluation le plus difficile de la génération précédente est utilisé pour évaluer l'adéquation des individus des deux autres espèces. La meilleure paire d'ensemble de prototypes et de mesure de voisinage de l'évolution est déterminée en évaluant la meilleure paire de chaque génération avec l'ensemble complet de validation et en gardant la paire la plus performante. L'utilisation de la co-évolution compétitive pour la sélection de cas d'évaluation a d'abord été abordée dans [62]. Plus tard, Panait et Luke [110] ont exploré cette idée dans le contexte de l'évolution de programmes robustes, avec des résultats concluants pour des problèmes classiques de PG.

Le tableau 3.7 présente les résultats de la co-évolution coopérative de la sélection de prototypes et de la mesure de voisinage, en compétition avec un sous-ensemble de données d'évaluation. Dans tous les cas, l'évolution des espèces en compétition se fait par une sélection de 50 % des données de l'ensemble d'évaluation comme cas d'adéquation pour chacun des individus. L'espèce compétitive évolue avec une population de taille 1000, une probabilité de croisement uniforme de 0.3 et une probabilité de mutation par inversion de bits de 0.05. Les paramètres des deux autres espèces sont les mêmes que ceux mentionnés précédemment.

Les résultats démontrent clairement que l'utilisation d'une troisième espèce en co-

évolution pour la sélection d'un ensemble d'évaluation aide à réduire le nombre de prototypes sélectionnés, particulièrement pour l'ensemble des données chevauchées. En comparaison avec l'évolution à deux espèces, les taux de reconnaissance sont légèrement plus faibles pour les ensembles des données chevauchées et des données inclinées et sont plus élevés pour les ensembles des données dentelées et des spirales. En général, on peut affirmer que l'introduction de la troisième espèce en co-évolution améliore la capacité de généralisation du système de reconnaissance. De plus, la troisième espèce réduit le nombre de cas de test d'évaluation en diminuant considérablement le nombre de prototypes, ce qui contribue à réduire les besoins en calcul de 50 % à 90 %, selon l'ensemble de données et l'évolution spécifique.

3.6 Résultats sur des données réelles

Dans cette section, nous évaluons les approches co-évolutionnaires décrites précédemment sur la base de résultats obtenus sur cinq ensembles de données du *UCI Machine Learning Repository* (ensembles de données UCI) [13], présentés au tableau 3.8. Comme pour les ensembles de données synthétiques, chaque ensemble de données a été partitionné aléatoirement en quatre sous-ensembles de même taille (données d'entraînement, d'évaluation de l'adéquation, de validation et de test). Le tableau 3.9 présente les performances de base et les résultats obtenus avec des classifieurs 1-PPV sur ces ensembles de données, en utilisant les ensembles complets des données d'entraînement comme prototypes, en sélectionnant la meilleure configuration 1-PPV avec les ensembles d'évaluation de l'adéquation et de validation et en donnant des résultats pour les ensembles de test. Le tableau 3.10 présente les résultats obtenus pour les mêmes ensembles de données avec l'application de l'édition de Wilson et de la condensation de Hart.

Le tableau 3.11 présente les résultats de la co-évolution simultanée de la sélection de prototypes avec un AG multi-objectif et de la mesure de voisinage avec la PG, sur les ensembles de données UCI. Les résultats indiquent que l'amélioration du taux de reconnaissance est significative pour les ensembles de données cmc et ionosphere, en comparaison avec les performances de base et avec les meilleures configurations classiques du tableau 3.9. Pour l'ensemble de données spambase, on note une amélioration de 13.5 % par rapport à la performance de base, mais de seulement 3 % par rapport à la meilleure configuration classique. Le taux de reconnaissance pour l'ensemble de données pid semble difficile à améliorer, avec un gain d'environ 1 %. La meilleure configuration classique semble confirmer ce point avec -1 % d'amélioration[1] par rapport à

[1]Le lecteur doit garder à l'esprit que la configuration du meilleur classifieur 1-PPV est sélectionnée à partir de l'ensemble de validation, mais que les résultats sont obtenus sur l'ensemble de test.

TAB. 3.8 – Description des ensembles de données choisis du *UCI Machine Learning Repository* (ensembles de données UCI).

Ensemble de données	Taille	Nombre d'attributs	Nombre de classes	Domaine d'application
abalone	4177	8	29	Prédiction de l'âge d'oreilles de mer à partir de mesures physiques.
cmc	1473	9	3	Choix des mesures contraceptives à partir de caractéristiques démographiques et socio-économiques.
ionosphere	351	34	2	Détection de structures dans l'ionosphère à partir de retours de signaux radars.
pid	768	8	2	Diagnostique du diabète chez des femmes de tribus amérindiennes Pima à partir de mesure médicales générales.
spambase	4601	57	2	Classement de courriels comme étant des polluriels (ou non) à partir de statistiques sur la fréquence dans les messages de caractères et de mots spécifiques.

TAB. 3.9 – Performances obtenues pour les ensembles de données UCI des classifieurs 1-PPV classiques, en faisant varier la norme L_a ($a = 1, 2, \infty$) et la normalisation (aucune, d'échelle et blanchissante) de l'espace des caractéristiques d'entrée. Les résultats donnés sont obtenus sur les ensembles de test.

		Meilleure configuration		
Ensemble de données	Performance de base (L_2)	Distance	Normalisation	Gain en reconnaissance
abalone	49.2 %	L_1	d'échelle	−0.7 %
cmc	44.2 %	L_∞	d'échelle	0.3 %
ionosphere	84.1 %	L_∞	aucune	2.3 %
pid	66.7 %	L_2	d'échelle	−1.0 %
spambase	77.8 %	L_1	d'échelle	10.4 %

TAB. 3.10 – Résultats de l'édition de Wilson et de la condensation de Hart sur les ensembles de données UCI ($k = 1$ avec la norme L_2 et aucune normalisation).

Ensemble de données	Perf. base	Édition de Wilson		Cond. de Hart		Wilson + Hart	
		Gain recon.	Taux sélect.	Gain recon.	Taux sélect.	Gain recon.	Taux sélect.
abalone	49.2 %	3.6 %	48.1 %	−0.9 %	68.7 %	3.0 %	17.2 %
cmc	44.2 %	1.6 %	39.1 %	−0.5 %	73.6 %	3.0 %	15.0 %
ionosphere	84.1 %	−4.5 %	86.2 %	1.1 %	29.9 %	0.0 %	11.5 %
pid	66.7 %	4.2 %	70.8 %	−4.7 %	46.4 %	1.6 %	14.1 %
spambase	77.8 %	−0.6 %	74.0 %	−3.8 %	44.5 %	−2.6 %	15.7 %

TAB. 3.11 – Résultats de la co-évolution de la sélection de prototypes et de la mesure de voisinage sur les ensembles de données UCI.

Ensemble de données	Performance de base	Meilleur des 10 essais		Moyenne des 10 essais	
		Gain en reconnaissance	Taux de sélection	Gain en reconnaissance	Taux de sélection
abalone	49.2 %	4.5 %	55.9 %	−3.7 %	56.7 %
cmc	44.2 %	10.0 %	59.8 %	5.6 %	60.2 %
ionosphere	84.1 %	8.0 %	60.9 %	6.0 %	54.6 %
pid	66.7 %	1.0 %	38.0 %	1.1 %	57.1 %
spambase	77.8 %	13.5 %	59.9 %	13.2 %	57.5 %

TAB. 3.12 – Résultats de la co-évolution de trois espèces sur les ensembles de données UCI : coopération de la sélection de prototypes et de la mesure de voisinage, en compétition avec la sélection d'un sous-ensemble de données d'évaluation, avec des classifieurs 1-PPV.

Ensemble de données	Performance de base	Meilleur des 10 essais		Moyenne des 10 essais	
		Gain en reconnaissance	Taux de sélection	Gain en reconnaissance	Taux de sélection
abalone	49.2 %	2.6 %	37.9 %	−2.6 %	36.4 %
cmc	44.2 %	3.3 %	27.7 %	1.2 %	27.3 %
ionosphere	84.1 %	5.7 %	50.6 %	4.4 %	49.7 %
pid	66.7 %	5.7 %	20.8 %	0.4 %	21.8 %
spambase	77.8 %	12.7 %	45.2 %	12.1 %	42.0 %

la performance de base. D'autre part, il y a une variation importante dans les taux de reconnaissance obtenus pour l'ensemble de données abalone, d'une gain moyen de −3.7 % à un gain de 4.5 % pour le meilleur résultat. Cette variation peut s'expliquer par le grand nombre de classes dans l'ensemble de données, où chaque classe est l'âge d'une oreille de mer (*abalone* en anglais). Il aurait été préférable d'utiliser une mesure d'adéquation qui favorise les classements de données qui sont près (±1 an) des classes de données réelles. Finalement, les taux de sélection obtenus pour tous les ensembles de données se situent entre 55 % et 60 %, à l'exception du meilleur résultat de l'ensemble de données pid (38 %).

Les ensembles de données UCI sont aussi testés en utilisant une co-évolution à trois espèces. Pour cette expérience, la taille de l'ensemble des données d'évaluation de l'adéquation est limitée à 50 % de l'ensemble original ou à 200 pour les plus grands ensembles de données (abalone et spambase). Le tableau 3.12 présente les résultats pour les ensembles de données. Pour les ensembles abalone, ionosphere et spambase, on note une faible diminution des taux de reconnaissance, entre −1 % et −3 %, par rapport à la co-évolution à deux espèces. Pour l'ensemble de données cmc, la diminution du taux de reconnaissance est plus importante, −6.7 % pour le meilleur résultat et −4.5 % en moyenne. À l'inverse, on observe une amélioration de 4.7 % pour le meilleur résultat avec l'ensemble de données pid, et une légère diminution, −0.7 %, des résultats de performance moyenne. Ces fluctuations du taux de reconnaissance s'expliquent par les améliorations des taux de sélection obtenus pour tous les ensembles de données, en comparaison avec la co-évolution à deux espèces. La diminution importante du taux de sélection, qui s'étend de 50 % pour l'ensemble ionosphere à 20 % pour l'ensemble pid, est symptomatique de solutions qui utilisent moins d'information pour le classement

et donc qui surapprennent moins. Cela tend à confirmer les effets de robustesse de la co-évolution compétitive qui ont déjà été observés dans [62, 110].

3.7 Discussion

Pour ces travaux, nous avons utilisé la règle du k-PPV comme système de classement de base à optimiser par les AE. D'un point de vue plus général, l'objectif de ce chapitre est d'illustrer l'idée que les AE peuvent bien performer pour optimiser finement des systèmes de reconnaissance des formes en suivant l'approche méthodologique proposée à la section 1.4. Nous ne croyons pas que les AE doivent être employés directement à la base d'un système de reconnaissance : en effet, des méthodes de classement basées sur des modèles théoriques solides et bien établis performent très bien dans la plupart des situations. Tel que mentionné par les théorèmes du *no free lunch* [158], aucune approche n'est supérieure aux autres de façon absolue, et donc une approche basée purement sur une méthode générique telle que les AE ne surpasserait probablement pas les approches spécifiques au domaine. Mais la grande généricité des AE peut être utile dans des systèmes hybrides, en collaboration avec des approches classiques de classement, de façon à tirer le meilleur des deux méthodes. En effet, la conception d'un bon système de reconnaissance des formes va habituellement au-delà de l'utilisation d'un classifieur. Les hypothèses de travail des méthodes de classement sont parfois très fortes et difficiles à obtenir en pratique. La connaissance du domaine est aussi souvent nécessaire pour extraire les caractéristiques les plus discriminantes des données brutes. Les AE peuvent être très utiles dans ces cas pour transformer le problème original sous une forme qui permettra la meilleure performance du classifieur. L'hybridation des AE avec des classifieurs peut se faire de plusieurs façons afin d'optimiser finement le système : une chaîne de bits d'AG pour la sélection d'éléments, un AG à valeurs réelles pour l'optimisation d'une fonction, la PG pour évoluer le modèle du système, etc. De plus, on peut utiliser la co-évolution pour optimiser simultanément ces différents sous-systèmes.

3.8 Conclusion

Nous avons introduit dans ce chapitre différentes façons d'optimiser un classifieur par les k-PPV en utilisant les AE. Tout d'abord, nous avons utilisé un AG à chaînes de bits pour sélectionner des prototypes dans un ensemble d'entraînement. Deux objectifs guident le processus d'optimisation : maximiser le taux de reconnaissance et minimiser

la taille de l'ensemble des prototypes. Nous avons utilisé l'opération de sélection basée sur le critère de Pareto afin d'optimiser simultanément tous les objectifs et de reporter à la fin du processus d'optimisation la décision du rapport du taux de reconnaissance à taille au nombre de prototypes. Les résultats ont démontré que l'AG multi-objectif peut sélectionner des sous-ensembles d'entraînement, pour obtenir un taux de reconnaissance égal ou supérieur à ceux obtenus avec l'ensemble complet d'entraînement comme prototypes, tout en réduisant significativement le nombre de prototypes. On obtient une meilleure réduction de l'ensemble avec des approches classiques de sélection de prototypes, mais les taux de reconnaissance sont alors moins bons que la performance de base. En observant les fronts de Pareto de la figure 3.4, on peut sélectionner, selon les besoins du problème, des solutions ayant différents rapports du taux de reconnaissance au nombre de prototypes.

Nous avons également proposé une approche basée sur la PG pour évoluer une mesure de voisinage spécifique au domaine, qui capture le modèle sous-jacent aux données. Nous proposons ici un changement de paradigme. En effet, nous optons pour l'optimisation de la configuration du modèle plutôt que seulement l'optimisation des paramètres du modèle comme dans les approches d'optimisation classiques. Les résultats sont encourageants, en ce sens qu'ils sont meilleurs ou comparables à ceux obtenus avec le meilleur classifieur par les k-PPV avec une configuration classique. Cela illustre la puissance de la PG pour construire un modèle de données.

Enfin, ces deux approches ont été intégrées ensemble par une co-évolution coopérative. Cette approche à deux espèces a donné des taux de reconnaissance comparables à ceux obtenus avec la PG et a permis de réduire le nombre de prototypes à des niveaux comparables à ceux obtenus avec un AG. Cela illustre l'intérêt de la co-évolution, qui permet d'évoluer simultanément différents éléments d'un problème, avec des résultats aussi bons que ceux obtenus avec des évolutions indépendantes. Nous avons ensuite introduit une espèce compétitive qui a provoqué une saine compétition entre l'espèce de la sélection de données d'évaluation et les deux espèces coopératives de la sélection de prototypes et de la mesure de voisinage. Ceci a eu pour effet de réduire le risque de surapprentissage, car l'ensemble d'évaluation change dans le temps afin de sélectionner des données d'évaluation difficiles à classer. En pratique, les résultats sont comparables ou un peu moins bons que ceux obtenus avec seulement les deux espèces coopératives, alors que les solutions sont significativement plus simples du fait que les tailles des ensembles de prototypes sont réduites. Donc, il est très intéressant d'effectuer une co-évolution compétitive pour résoudre des problèmes de reconnaissance des formes. Nous avons développé l'idée de l'utilisation des AE pour optimiser différents aspects des classifieurs par les k-PPV, mais cette méthodologie pourrait facilement être transposée à d'autres méthodes de classement.

Chapitre 4

Évolution de représentations de caractères cursifs

One should not increase, beyond what is necessary, the number of entities required to explain anything.
William d'Occam

Ce chapitre porte sur les composants de segmentation et d'extraction de caractéristiques d'un système de reconnaissance de caractères manuscrits en ligne [115] et sur l'automatisation partielle du développement de ces composants à l'aide de la programmation génétique (PG) [9, 76]. Dans le contexte de la reconnaissance des caractères, l'aspect le plus important à considérer lors du développement de systèmes de haute performance est possiblement la sortie du module d'extraction de caractéristiques, c'est-à-dire la représentation des caractères [144]. Dans ce qui suit, nous nous concentrons particulièrement sur une représentation en ligne spécifique et, avec l'aide de la PG, nous tentons de l'améliorer en utilisant des techniques qui requièrent un minimum d'intervention humaine.

À la section 4.1, nous résumons la représentation floue régionale de l'écriture cursive originale [60] et nous présentons des résultats obtenus précédemment à titre de référence. Nous présentons ensuite les résultats pour la recherche d'une nouvelle variation hiérarchique de la configuration de grille uniforme originale. À la section 4.2, nous introduisons une nouvelle approche guidée par les données, où la segmentation de la grille hiérarchique des caractères n'est plus statique, mais varie en fonction de la distribution spatiale des traits manuscrits. Nous présentons ensuite, à la section 4.3, la méthode de recherche avec la PG de représentations optimales de l'écriture cursive, accompagnée

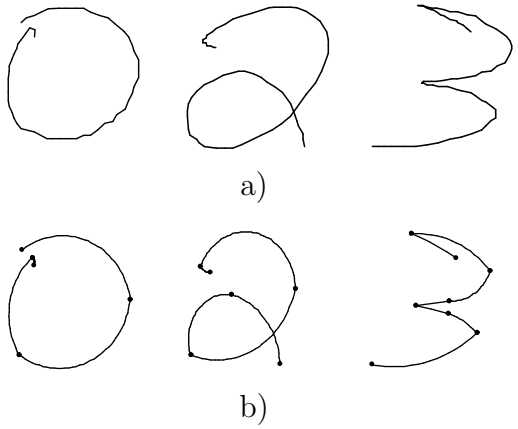


FIG. 4.1 – Trois caractères cursifs d'écriture en ligne : a) traits bruts ; b) les reconstructions en traits.

de résultats expérimentaux.

4.1 Représentations floues régionales

La représentation floue régionale en ligne [60, 111] commence avec une décomposition approximative des traits par une séquence d'arcs de cercle, tel que décrit dans [89]. Ainsi, un caractère est représenté par une séquence d'arcs de cercle $s_1, s_2, \ldots, s_i, \ldots, s_q$, où chaque arc $s_i = (\mathbf{p}_0, \mathbf{p}_1, l, c)$ est décrit par quatre paramètres : $\mathbf{p}_0$ et $\mathbf{p}_1$, les points de départ et d'arrivée de l'arc, l, la longueur curviligne et c, la courbure. On peut déterminer l'angle d'orientation d'un trait, θ, en connaissant l'orientation du vecteur $\overrightarrow{\mathbf{p}_0\mathbf{p}_1}$. La figure 4.1 illustre quelques exemples de caractères isolés avec leur reconstruction en traits. Il est à noter qu'un segment de droite est considéré comme un cas particulier d'arc de cercle de courbure nulle.

4.1.1 Extraction de caractéristiques

Le module d'extraction de caractéristiques décompose le caractère en un nombre fixe de régions rectangulaires à l'intérieur de la fenêtre d'attention du caractère. Pour chacune de ces régions, un vecteur flou est formé avec l'orientation et la courbure des traits. La figure 4.2 illustre les ensembles flous utilisés pour quantifier les traits segmentés. Les quatre ensembles flous illustrés à la figure 4.2a sont utilisés pour rendre flou l'angle d'orientation $\theta \in [-180°, 180°]$, alors que les trois ensembles flous de la figure 4.2b sont utilisés pour rendre floue la courbure $c \in [-\infty, \infty]$. Par conséquent, un trait peut être

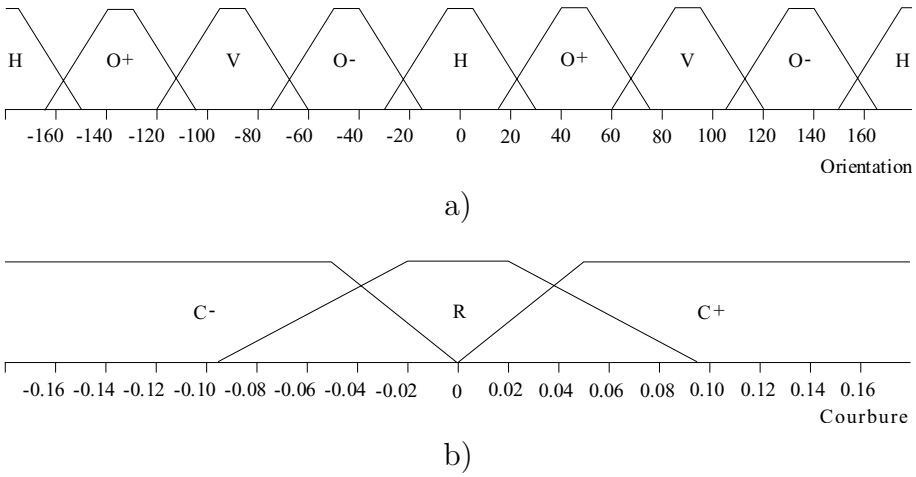


FIG. 4.2 – Ensembles flous de la représentation floue régionale originale : a) ensembles **H**, **V**, **O**+ et **O**− pour le paramètre θ ; b) ensembles **R**, **C**+ et **C**− pour le paramètre c.

considéré comme plus ou moins horizontal (**H**), vertical (**V**), oblique avec une pente positive (**O**+) ou oblique avec une pente négative (**O**−), selon l'orientation du trait, θ. De même, un trait peut être plus ou moins rectiligne (**R**), avoir une courbure positive (**C**+) ou négative (**C**−). Étant donné que chaque région d'un caractère est associée à un seul vecteur flou, lorsque plusieurs traits sont présents dans une région, on utilise la valeur maximale d'appartenance floue pour chaque ensemble flou. L'espace vectoriel complet des caractéristiques est construit par simple concaténation des différents vecteurs flous régionaux. Cette opération permet la mise en correspondance de différentes caractéristiques régionales, extraites à partir d'un nombre variable de traits dans un vecteur flou de dimension fixe, avec des valeurs dans l'intervalle $[0, 1] \in I\!R$. Cette mise en correspondance s'avère très avantageuse lorsqu'on travaille avec des algorithmes communs de classement.

Le désavantage de cette représentation est que l'orientation d'un trait y est globale et ne donne pas d'information quant à l'orientation locale des différents traits à l'intérieur de chaque région. Un raffinement possible pour contourner ce problème, proposée initialement dans [60], consiste à rogner les traits aux limites de la région avant de calculer les angles θ. Un autre raffinement consiste à normaliser chaque valeur d'appartenance floue en la multipliant par le rapport de la longueur du trait à la longueur totale de la diagonale de la région. Ces deux solutions sont utilisées pour toutes les expérimentations de ce chapitre.

En plus des sept variables floues énumérées précédemment, deux autres caractéristiques sont extraites de chaque région : les centres de masse horizontal (**MX**) et vertical (**MY**) des traits, après le rognage. Ces caractéristiques sont également normalisées dans

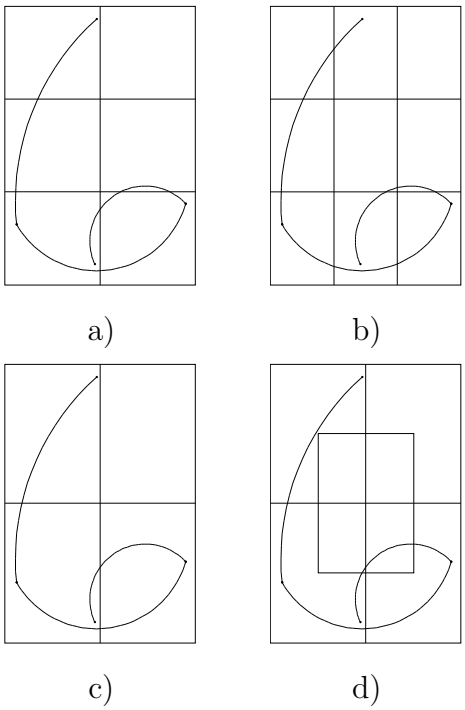


FIG. 4.3 – Différentes topologies de grille : a) 3×2 régulier (RFR-3x2) ; b) 3×3 régulier (RFR-3x3) ; c) arbre quad à 2 niveaux (RFR-quad2) ; d) arbre quin à 2 niveaux (RFR-quin2).

l'intervalle $[0, 1]$. Elles représentent les coordonnées d'une position relative dans le prolongement de la région ; $(0, 0)$ correspond au coin inférieur gauche de la région et $(1, 1)$, au coin supérieur droit.

4.1.2 Topologies de grille

Dans [60, 111], il a été proposé de diviser la fenêtre d'attention en une grille uniforme avec, par exemple, 3×2 régions ou 3×3 régions, tel qu'illustré à la figure 4.3a et 4.3b. Dans ce chapitre, nous proposons également d'utiliser une topologie de grille hiérarchique, en partant avec une fenêtre d'attention entière, et en subdivisant la région en quatre régions (arbre quad) ou cinq régions (arbre quin), récursivement. La figure 4.3c montre la décomposition en cinq régions pour un arbre quad à deux niveaux et la figure 4.3d illustre la décomposition en six régions pour un arbre quin à deux niveaux. Comme topologies de grille standards, nous avons également testé les représentations en arbre quad à trois niveaux (RFR-quad3) et en arbre quin à trois niveaux (RFR-quin3). Ces représentations hiérarchiques en arbre quin et en arbre quad sont inspirées des travaux de Park et al. [112].

4.1.3 Résultats expérimentaux

Toutes les expérimentations de classement présentées dans ce chapitre ont été effectuées en utilisant un perceptron multicouche (PMC), entraîné avec la rétro-propagation en ligne standard avec momentum [54], et avec une seule couche cachée de 50 neurones. Le taux d'apprentissage et le momentum sont fixés à 0.1 et 0.25, respectivement. L'entraînement est contrôlé en utilisant une procédure de validation où 67 % de l'ensemble d'apprentissage est utilisé pour l'entraînement et 33 % pour la validation. Le PMC est entraîné durant un minimum de 35 époques et le nombre total d'époques d'entraînement varie de 65 à 125. La stratégie de décision (post-traitement) consiste simplement à classer les données selon la sortie maximale du PMC.

Dans le présent chapitre, la configuration du PMC utilisée pour toutes les expérimentations correspond à la meilleure configuration établie préalablement [111] pour des représentations floues régionales de base (RFR-3x2 et RFR-3x3). Il est certain que si l'on modifie la représentation de caractères utilisée, par exemple en utilisant un nombre élevé de caractéristiques, la configuration du PMC risque de ne plus être optimale. En ce sens, les résultats présentés dans la suite du chapitre sont conservateurs étant donné qu'il aurait peut-être été possible d'améliorer significativement les résultats uniquement en modifiant la configuration du classifieur. Un choix délibéré a donc été fait afin de fixer au préalable la configuration du classifieur de type PMC pour limiter le problème à une recherche de représentations de caractères.

Les expérimentations sont effectuées avec la section 1a (chiffres isolés) des ensembles de données Unipen [53]. Les données d'entraînement proviennent de Unipen Train-R01/V07 et consistent en 15 953 caractères, alors que les données de test sont tirées de DevTest-R02/V02 et consistent en 8 598 caractères. Les ensembles de données sont utilisés tels quels, à l'exception des échantillons de caractères de largeur nulle et de hauteur nulle, tel que mentionné dans [111] (4 cas dans l'ensemble d'entraînement et 34 dans l'ensemble de test).

Le tableau 4.1 présente les taux de reconnaissance obtenus pour différentes topologies conçues par des humains. Les expérimentations sont effectuées sur les six topologies de grille présentées à la section 4.1.2, en extrayant, pour chaque région, les neuf caractéristiques décrites à la section 4.1.1. Donc, pour chaque topologie, la taille de l'ensemble des caractéristiques correspond au nombre de régions multiplié par neuf. La colonne *Taux moy.* donne la moyenne des taux de reconnaissance obtenus sur l'ensemble de test pour dix différents entraînements de PMC. La colonne *Taux max.* donne le meilleur taux de reconnaissance obtenus sur ces dix entraînements et la colonne *Écart-type taux* donne l'écart-type par rapport à la moyenne. Les colonnes *Gain moy. RFR-3x2* et *Gain*

Topologie de grille	Nombre caract.	Taux moy. (%)	Taux max. (%)	Écart-type taux (%)	Gain moy. RFR-3x2	Gain max. RFR-3x2
RFR-3x2	54	95.60	95.77	0.18	–	0.18
RFR-3x3	81	95.84	96.09	0.21	0.25	0.49
RFR-quad2	45	94.72	95.07	0.25	−0.87	−0.52
RFR-quad3	189	96.33	96.66	0.17	0.73	1.06
RFR-quin2	54	95.35	95.65	0.20	−0.24	0.05
RFR-quin3	279	96.37	96.66	0.21	0.77	1.06

TAB. 4.1 – Taux de reconnaissance pour différentes topologies de grille de représentations floues régionales.

max. RFR-3x2 donnent respectivement l'amélioration moyenne et maximale du taux de reconnaissance par rapport au taux moyen de la RFR-3x2, qui est utilisé ici comme référence.

Le taux de reconnaissance moyen est de 95.60 % pour la RFR-3x2 de référence. À partir de cette représentation de référence, les améliorations possibles sont obtenues principalement avec des représentations ayant de nombreuses caractéristiques supplémentaires (jusqu'à cinq fois plus) : 0.25 % pour la RFR-3x3, 0.73 % pour la RFR-quad3 et 0.77 % pour la RFR-quin3. D'autre part, avec les représentations en arbre quin et en arbre quad à 2 niveaux, on obtient une diminution de performance (−0.24 % pour la RFR-quin2 et −0.87 % pour la RFR-quad2). Ces résultats démontrent que la topologie de grille peut avoir un effet important sur la performance en utilisant de telles représentations régionales. Avec la topologie RFR-quin3, on obtient une amélioration de 0.77 % en moyenne, et de 1.06 % dans le meilleur des cas. Les résultats de la RFR-quad3 semblent un peu moins bons que ceux de la RFR-quin3 en moyenne (0.73 % et 0.77 %, respectivement), mais cet écart n'est pas statistiquement significatif. Toutefois, la RFR-quad3 utilise 90 caractéristiques en moins que la RFR-quin3, ce qui est un avantage.

Il n'est pas aisé de comparer ces résultats avec ceux de la littérature. Dans une étude publiée par Ratzlaff [122], il est noté que la plupart des taux de reconnaissance rapportés pour la section 1a des données Unipen ont été obtenus en utilisant Train-R01/V07, et non DevTest-R02/V02, qui semble être une banque plus difficile. Dans [111], nous avons rapporté un taux moyen de reconnaissance allant jusqu'à 96.9 % (comparativement à 96.37 % pour le meilleur résultat du tableau 4.1), en utilisant des représentations régionales similaires, mais avec plus de caractéristiques globales. À notre connaissance, le seul autre résultat pour la banque DevTest-R02/V02 complète a été publié par Ratzlaff, qui a obtenu 98.1 % en utilisant une représentation et un classifieur différents [121].

Il note également dans l'article qu'il a dû étiqueter manuellement les données car il n'avait pas accès aux étiquettes de classes officielles[1]. Il est donc probable que ce ré-étiquetage de la banque DevTest-R02/V02 ait nettoyé les données, corrigeant ainsi tout mauvais étiquetage. De plus, il a été démontré avec une expérience de reconnaissance avec 10 lecteurs humains qu'environ 1 % des chiffres de la section 1a de la banque DevTest-R02/V02 sont complètement illisibles [60]. Cela nous mène à penser que 1 % des caractères peuvent être mal étiquetés.

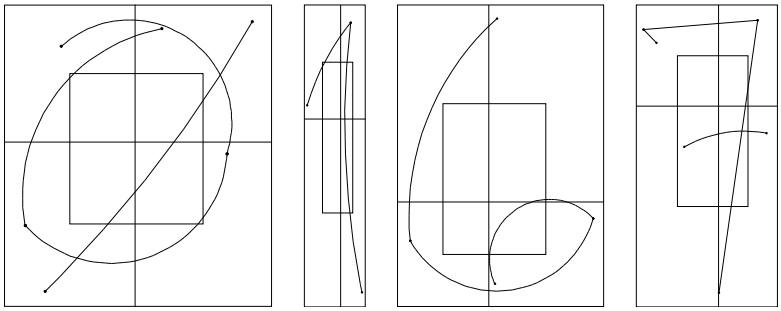
Quoi qu'il en soit, le but de cette expérimentation n'est pas de développer un système de reconnaissance optimal en tant que tel, car cela impliquerait probablement d'utiliser plusieurs classifieurs et représentations. L'objectif est plutôt d'explorer différentes topologies dans le contexte de représentations floues avec régions et d'expérimenter avec la PG pour concevoir de telles topologies.

4.2 Représentations guidées par les données

La représentation floue régionale possède plusieurs caractéristiques intéressantes, telles que la mise en correspondance d'un caractère cursif arbitraire en ligne dans un vecteur des caractéristiques normalisé et de longueur fixe. D'autre part, nous pouvons spéculer que l'utilisation d'une topologie de grille statique s'avère sous-optimale, spécialement pour les caractères déformés. En se basant sur les travaux de Park et al. [112], nous analysons maintenant l'utilisation de topologies hiérarchiques guidées par les données, dont les régions sont définies récursivement autour du centre de masse des traits, plutôt qu'autour du centre géométrique de la région du parent. Par exemple, dans le cas de la topologie en arbre quin, les quatre coins de la région centrale correspondent aux centres des quatre autres régions, qui sont elles-mêmes définies par le centre de masse du caractère entier, tel qu'illustré à la figure 4.4. Pour ces représentations en arbre quad et en arbre quin à 3 niveaux, la position de chaque région au troisième niveau est calculée avec le centre de masse de la région de son parent, du deuxième niveau. Pour les topologies de grille guidées par les données 3×2 et 3×3, la largeur et la hauteur de chaque région sont ajustées selon la position du centre de masse du caractère par rapport à son centre géométrique.

Pour toutes les topologies guidées par les données, étant donné que les centres de

[1] À l'origine, la banque DevTest-R02/V02 a été transmise aux contributeurs d'Unipen sans les étiquettes de classes en vue d'une compétition qui n'a finalement jamais été organisée. Par la suite, les étiquettes de classes ont été transmises dans une hiérarchie indépendante de fichiers qui ont dû être fusionnés avec les données originales non-étiquetées.



a) b) c) d)

FIG. 4.4 – Topologie de grille guidée par les données en arbre quin à 2 niveaux (DDR-quin2) pour différents chiffres manuscrits : a) zéro ; b) un ; c) six ; d) sept.

Topologie de grille	Nombre caract.	Taux moy. (%)	Taux max. (%)	Écart-type taux (%)	Gain moy. RFR-3x2	Gain max. RFR-3x2
DDR-3x2	48	95.43	95.62	0.12	−0.17	0.03
DDR-3x3	72	96.10	96.29	0.12	0.51	0.69
DDR-quad2	40	93.98	94.21	0.21	−1.62	−1.39
DDR-quad3	168	94.97	95.31	0.23	−0.63	−0.29
DDR-quin2	48	95.02	95.26	0.18	−0.58	−0.34
DDR-quin3	248	94.69	95.09	0.21	−0.91	−0.51

TAB. 4.2 – Taux de reconnaissance pour différentes topologies de grille de représentations guidées par les données.

masse sont déjà utilisés pour déterminer les limites des régions, les caractéristiques correspondantes (**MX** et **MY**) sont retirées de la représentation floue et remplacées par un rapport hauteur/largeur (**HW**). La longueur finale de la représentation correspond donc à 8 fois le nombre total de régions.

4.2.1 Résultats expérimentaux

Le tableau 4.2 montre les résultats obtenus avec les représentations guidées par les données. Les expérimentations sont effectuées dans les mêmes conditions que celles décrites à la section 4.1.3. Étonnamment, les résultats démontrent que les représentations guidées par les données tendent à diminuer la performance, en particulier pour les topologies en arbre quad et en arbre quin. Cela démontre que l'extraction de caractéristiques peut s'avérer contre-intuitive. Seule la DDR-3x3 semble tirer avantage de l'approche guidée par les données, avec une amélioration de 0.51 % par rapport à la référence, et de 0.26 % par rapport à la RFR équivalente. La figure 4.5 montre un

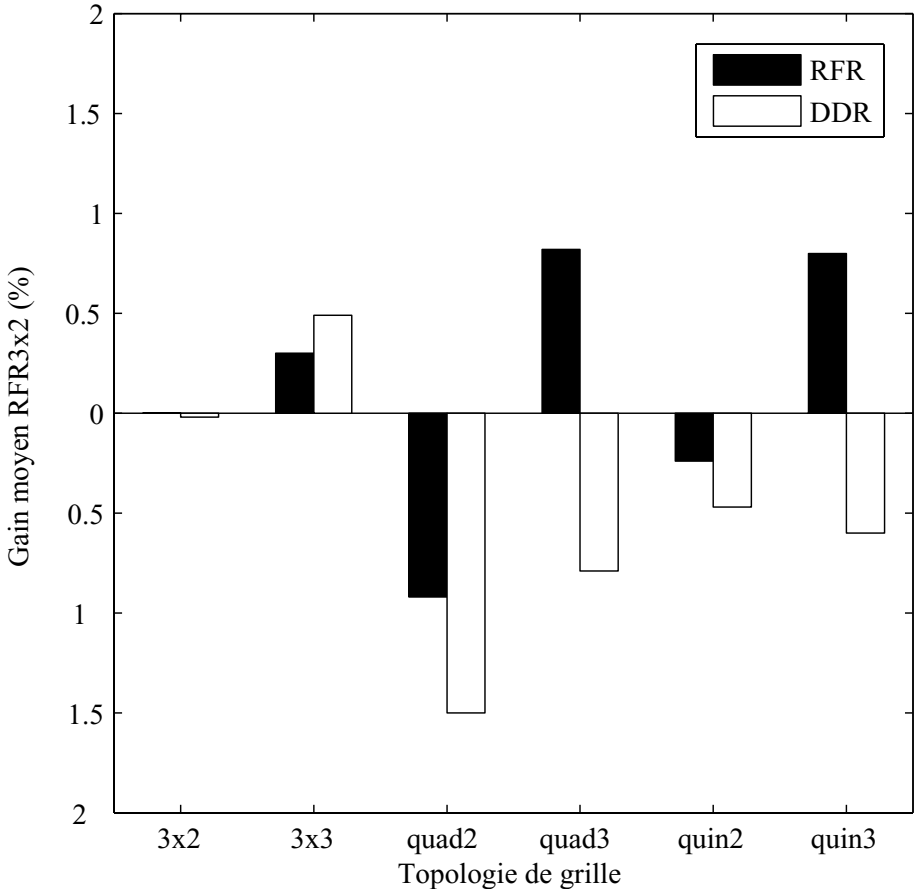


FIG. 4.5 – Gain moyen du taux de reconnaissance par rapport à la RFR-3x2 de base pour les RFR et DDR testées.

histogramme qui compare le gain moyen du taux de reconnaissance par rapport à la référence, pour différentes représentations RFR et DDR.

4.3 Ingénierie de représentations par programmation génétique

Jusqu'à maintenant, nous avons testé un ensemble fixe de topologies statiques et guidées par les données, conçues à la main, et nous avons démontré expérimentalement que certaines topologies sont plus performantes que d'autres. Aussi, un ensemble fixe de caractéristiques était systématiquement extrait de chaque région, ce qui générait parfois des représentations imposantes souffrant de la fameuse malédiction de la dimensionnalité. Finalement, les caractéristiques elles-mêmes étaient pré-définies par un ensemble fixe d'ensembles flous. L'objectif de cette section est d'établir une nouvelle procédure

automatique basée sur la programmation génétique pour explorer des topologies plus variées, avec des caractéristiques spécifiques pour chaque région.

Comme travaux passés, plusieurs articles ont exploré l'utilisation des AE pour la sélection de caractéristiques [159], la transformation de l'espace des caractéristiques [123] et la construction de caractéristiques [82, 132, 133]. Dans un contexte de reconnaissance de documents, Teredesai et Govindaraju [141, 142] ont utilisé la PG pour la conception de classifieurs actifs pour la reconnaissance de l'écriture hors ligne. Leur système est basé sur la représentation hiérarchique de Park [112], avec une sélection implicite de caractéristiques. Une autre approche pertinente a été proposée par Radtke, Wong et Sabourin [120], où un algorithme mémétique multi-objectif a été utilisé pour concevoir des représentations de caractères hors ligne selon une méthode similaire à la nôtre. Cependant, leur recherche est basée sur les algorithmes génétiques plutôt que sur la PG.

4.3.1 Évolution des représentations

Pour l'évolution des représentations de caractères cursifs, nous utilisons la PG à base d'arbre classique à deux objectifs, pour une fusion des éléments de la représentation régionale floue statique et de la représentation guidée par les données. Les données traitées par les nœuds des arbres consistent en deux paires de coordonnées, correspondant au coin inférieur gauche et au coin supérieur droit d'une région rectangulaire dans la fenêtre d'attention du caractère. Le tableau 4.3 présente les différentes primitives qui peuvent être utilisées durant le processus évolutionnaire. Cet ensemble de primitives fonctionnelles (nœuds d'arbre) peut être classé en deux catégories. La première catégorie contient les primitives qui modifient les régions (S2H, S2V, S3H, S3V, S4, S5, D2H, D2V, D3H, D3V, D4, D5 et ZM) en divisant la région actuelle en sous-régions ou en modifiant l'étendue de la région (ZM seulement). La deuxième catégorie contient les primitives d'extraction de caractéristiques (OR, CU, MX, MY et HW) qui extraient un type donné de caractéristiques de la région actuelle sans modifier sa définition. La région actuelle, pour un nœud donné, est toujours définie par son nœud parent et la racine hérite simplement de toute la matrice du caractère. Il existe une autre primitive terminale (T) qui sert seulement à fermer la structure de l'arbre. Chaque primitive est associée à un poids afin de biaiser sa probabilité de sélection durant les opérations d'initialisation et de mutation. Ce biais permet l'équilibre entre les primitives qui modifient les régions et les primitives d'extraction de caractéristiques.

Deux types de primitives modifiant les régions sont définis. Le premier type divise la région du parent en fractions pré-établies hauteur/largeur (S2H, S2V, S3H, S3V, S4

Nom	Args	Poids	Description
S2H	2	1.0	Découpage statique horizontal en deux régions.
S2V	2	1.0	Découpage statique vertical en deux régions.
S3H	3	1.0	Découpage statique horizontal en trois régions.
S3V	3	1.0	Découpage statique vertical en trois régions.
S4	4	1.0	Découpage statique en régions quad.
S5	5	1.0	Découpage statique en régions quin.
D2H	2	1.0	Découpage guidé par les données horizontal en deux régions.
D2V	2	1.0	Découpage guidé par les données vertical en deux régions.
D3H	3	1.0	Découpage guidé par les données horizontal en trois régions.
D3V	3	1.0	Découpage guidé par les données vertical en trois régions.
D4	4	1.0	Découpage guidé par les données en régions quad.
D5	5	1.0	Découpage guidé par les données en régions quin.
ZM	1	1.0	Zoom avec facteur généré aléatoirement.
OR	1	16.0	Extraction d'une orientation générée aléatoirement.
CU	1	12.0	Extraction d'une courbure générée aléatoirement.
MX	1	4.0	Extraction de la position horizontale du centre de masse.
MY	1	4.0	Extraction de la position verticale du centre de masse
HW	1	4.0	Extraction du rapport hauteur/largeur.
T	0	1.0	Terminal de fermeture.

Tab. 4.3 – Ensemble des primitives utilisées pour les évolution de PG.

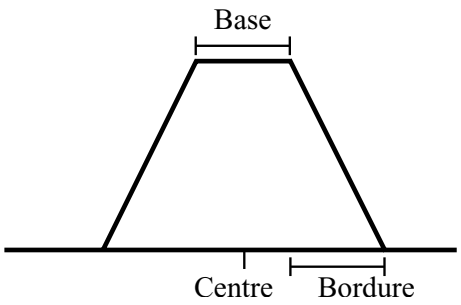


FIG. 4.6 – Forme de l'ensemble flou utilisé pour extraire les caractéristiques d'orientation et de courbure.

et S5), tel que décrit à la section 4.1. Il inclut également la primitive ZM, qui modifie l'échelle de la région actuelle en appliquant un zoom généré aléatoirement entre 20 % et 200 %. Ce zoom est généré durant l'initialisation ou la mutation et agit comme une constante éphémère aléatoire [76]. Les primitives qui modifient les régions du deuxième type (D2H, D2V, D3H, D3V, D4 et D5) divisent la région du parent selon le centroïde des traits trouvés dans cette région, d'une façon similaire aux représentations guidées par les données de la section 4.2.

Chaque primitive d'extraction de caractéristiques produit un effet de bord en ajoutant une nouvelle caractéristique à la représentation en sortie, sans affecter la région actuelle, qui est simplement passée inchangée au nœud enfant. Il existe également deux types de primitives d'extraction de caractéristiques. Le premier type comprend les primitives d'orientation et de courbure (OR et CU) qui extraient respectivement le degré de courbure et d'orientation des traits dans la région courante. Les deux primitives incluent trois paramètres générés aléatoirement (centre, base et bordure) qui définissent un ensemble flou de trapèzes symétriques, tel qu'illustré à la figure 4.6. Comme pour la primitive de zoom, ces trois paramètres sont des constantes éphémères aléatoires générées durant l'initialisation ou la mutation. L'étendue des valeurs de la primitive OR sont : centre $[-90°, 90°]$, base $[0°, 30°]$ et bordure $[5°, 50°]$. Toutes ces valeurs sont discrètes, avec des incréments de 5°. La forme floue pour l'extraction de l'orientation est répétée deux fois, à $-180°$ et à 180° par rapport à la position initiale du centre. Pour la primitive CU, les étendues des valeurs sont : centre $[-0.160, 0.160]$, base $[0, 0.160]$ et bordure $[0.005, 0.160]$. Les trois valeurs sont discrètes, avec des incréments de 0.005. Le deuxième type de primitives d'extraction de caractéristiques est composé des centres de masse horizontal et vertical (MX et MY) et du rapport hauteur/largeur de la région (HW). Ces primitives n'utilisent pas de constante éphémère aléatoire.

4.3.2 Résultats expérimentaux

Le protocole expérimental utilisé pour faire évoluer une population de représentations de l'écriture cursive nécessite une procédure à double validation afin d'éviter un surapprentissage des données. Donc, l'ensemble Train-R01/V07 est d'abord décomposé aléatoirement en un ensemble d'évaluation de l'adéquation, pour déterminer approximativement l'adéquation des représentations, et en un ensemble de validation, pour sélectionner la meilleure représentation de l'évolution. Ensuite l'ensemble d'évaluation de l'adéquation est sous-divisé une autre fois en un ensemble d'entraînement pour l'évaluation de l'adéquation, utilisé pour mettre à jour les poids du classifieur PMC, et en un ensemble de validation pour l'évaluation de l'adéquation, utilisé pour interrompre l'apprentissage lorsque le réseau de neurones commence à surapprendre. De cette façon, l'ensemble de validation final est complètement indépendant du processus évolutionnaire et favorise la sélection de la représentation qui démontre la meilleure capacité de généralisation. Finalement, le meilleur individu de l'évolution est évalué sur l'ensemble DevTest-R02/V02 complet afin de produire les taux de reconnaissance rapportés ci-dessous.

La sélection du meilleur individu de l'évolution à partir d'un ensemble de validation final, non utilisé pour l'entraînement du classifieur PMC ni pour l'évaluation de l'adéquation des individus, est dans la même veine que l'approche utilisée au chapitre précédent pour sélectionner les meilleurs individus de l'évolution. La différence entre l'approche utilisée au chapitre 3 et celle utilisée ici est que la meilleure représentation d'une évolution est déterminée à partir de tests sur l'ensemble de validation final de toutes les représentations de caractères de toutes les générations. Comme pour la coévolution de classifieurs de type plus proche voisin, la méthode de sélection des meilleurs individus de l'évolution devrait permettre de conserver la représentation de caractères manuscrits ayant la meilleure performance possible pour la reconnaissance de caractères non vus durant l'apprentissage, donc une représentation qui généralise bien.

Ce protocole a été répété quatre fois en utilisant deux tailles différentes pour l'ensemble de validation : 33 % de l'ensemble d'entraînement complet pour les deux premières expérimentations, 20% pour les deux dernières. Ces choix sont fait afin de vérifier que des configurations différentes de partitionnement n'affectent pas significativement les résultats. Dans chaque cas, l'ensemble d'évaluation de l'adéquation (respectivement 67 % et 80 % du total) a été divisé aléatoirement en deux parties égales : une première moitié pour l'ensemble d'entraînement pour l'évaluation de l'adéquation et l'autre moitié pour l'ensemble de validation pour l'évaluation de l'adéquation. De plus, les données de cette sous-division aléatoire sont mélangées à chaque évaluation de l'adéquation.

Topologie de grille	Nombre caract.	Taux moy. (%)	Taux max. (%)	Écart-type taux (%)	Gain moy. RFR-3x2	Gain max. RFR-3x2
Evol1	90	96.40	96.72	0.15	0.81	1.12
Evol2	81	96.27	96.57	0.16	0.67	0.97
Evol3	112	96.31	96.46	0.13	0.71	0.87
Evol4	105	96.37	96.51	0.10	0.77	0.91

TAB. 4.4 – Taux de reconnaissance obtenus pour les meilleures représentations des quatre évolutions distinctes de PG.

Le processus évolutionnaire est guidé par une optimisation à deux objectifs. Le premier objectif consiste à maximiser le taux de reconnaissance durant la phase d'entraînement. Les paramètres utilisés pour le réseau PMC sont : une couche cachée de 50 neurones, un taux d'apprentissage de 0.1, un momentum de 0.25, un minimum de 25 et un maximum de 100 époques d'entraînement et l'interruption de l'entraînement après 5 époques sans amélioration du taux de reconnaissance de l'ensemble de validation pour l'évaluation de l'adéquation. Le deuxième objectif consiste à minimiser le nombre total de caractéristiques de la représentation.

Chaque évolution inclut des populations de 1000 individus, 100 générations, une probabilité de croisement de 0.9, une probabilité de mutation (standard, d'inversion de nœuds (*swap*), réductrice (*shrink*)) de 0.05 et un opérateur de sélection multi-objectif NSGA-II. Les profondeurs d'arbres minimale et maximale initiales sont posées respectivement à 3 et à 7 niveaux et la profondeur d'arbre durant l'évolution est limitée à un maximum de 17 niveaux. La taille des populations et le nombre de générations de chaque évolution ont été posés afin que les temps de calcul soient acceptables tout en permettant une convergence vers des résultats intéressants. Il est certain que l'application du présent chapitre est très lourde et pas nécessairement très efficace en terme computationnel. Cependant, étant donnée que l'objectif principal de l'application est de faire une preuve de concept de l'approche, nous avons déployé peu d'effort afin de réduire significativement les temps de calculs. Les autres paramètres de l'AE correspondent aux valeurs par défaut de l'outil logiciel. L'implantation de PG est faite en C++ avec la framework Open BEAGLE, présentée au chapitre 2, et distribuée sur une grappe de calculs de type Beowulf de 26 AMD Athlons de 1.2 GHz avec Distributed BEAGLE, présenté à la section 2.3.

Le tableau 4.4 présente les taux de reconnaissance obtenus pour les meilleures représentations des quatre évolutions. Chaque meilleur individu est testé dix fois, en utilisant la même méthodologie d'entraînement que celle décrite aux sections 4.1.3 et 4.2.1. Les résultats démontrent qu'avec moins de la moitié du nombre de caractéris-

tiques, la meilleure représentation obtenue par PG affiche un taux de reconnaissance similaire à la meilleure représentation conçue par des humains de la section 4.1.3 (taux moyens de 96.40 % par rapport à 96.37 %). De plus, les écarts-types de ces taux moyens sont systématiquement plus faibles (0.15 % par rapport à 0.21 % pour le meilleur cas). Il est également intéressant de noter que le processus évolutionnaire est plutôt stable. On obtient des résultats comparables pour les quatre évolutions distinctes, même si le plus grand ensemble de validation des deux premières (Evol1 et Evol2) semblent avoir produit des représentations contenant moins de caractéristiques.

La figure 4.7 présente les fronts de Pareto à la dernière génération pour les quatre évolutions de représentations de caractères cursifs. Sur chaque graphique, une première courbe (cercles et traits discontinus) représente le front de Pareto en tant que tel, avec en abscisse le nombre de caractéristiques utilisées et en ordonnée le taux d'erreur de classement sur les ensembles d'évaluation de l'adéquation. Une deuxième courbe (étoiles et traits continus) trace le taux d'erreur sur les ensembles de validation pour les solutions présentes sur la première courbe. On remarque sur la figure que les taux de classement sur les ensembles de validation sont corrélés avec les taux de classement correspondant sur les ensembles d'évaluation de l'adéquation. Également, il semble que pour toutes les solutions du front de Pareto, plus le nombre de caractéristiques est élevé, plus le taux de reconnaissance sur les ensembles de validation est bon. Pour les quatre évolutions, on n'observe pas de dégradation des performances sur les ensembles de validation lorsque le nombre de caractéristiques est très grand. En ce sens, l'utilisation de l'optimisation multi-objectif semble avoir porté fruit, en réduisant le nombre de caractéristiques et donc en prévenant possiblement le surapprentissage. Finalement, il faut souligner que comme expliqué au début de la présente section, le meilleur individu de chaque évolution est sélectionné comme étant l'individu ayant le meilleur taux de classement sur les données de validation. Cette méthode de sélection implique que le meilleur individu de l'évolution n'est pas nécessairement présent sur le front de Pareto de la dernière génération.

La figure 4.8 montre la S-expression complète du programme en arbre Evol1. La figure 4.9 résume la structure d'arbre correspondante en représentant les primitives qui modifient la région comme des ellipses et les primitives d'extraction de caractéristiques comme des cases rectangulaires (voir le tableau 4.3 pour la description des primitives). Lorsque plusieurs caractéristiques sont extraites d'une région donnée, elles sont énumérées dans une seule case sans leurs paramètres. Par exemple, « 2OR, 1CU » signifie que deux primitives d'orientation et une primitive de courbure sont extraites de la région définie par le nœud parent. Les primitives terminales sont omises à moins que leur parent ne modifie la région.

Cette figure démontre que le processus évolutionnaire a convergé vers un mélange des

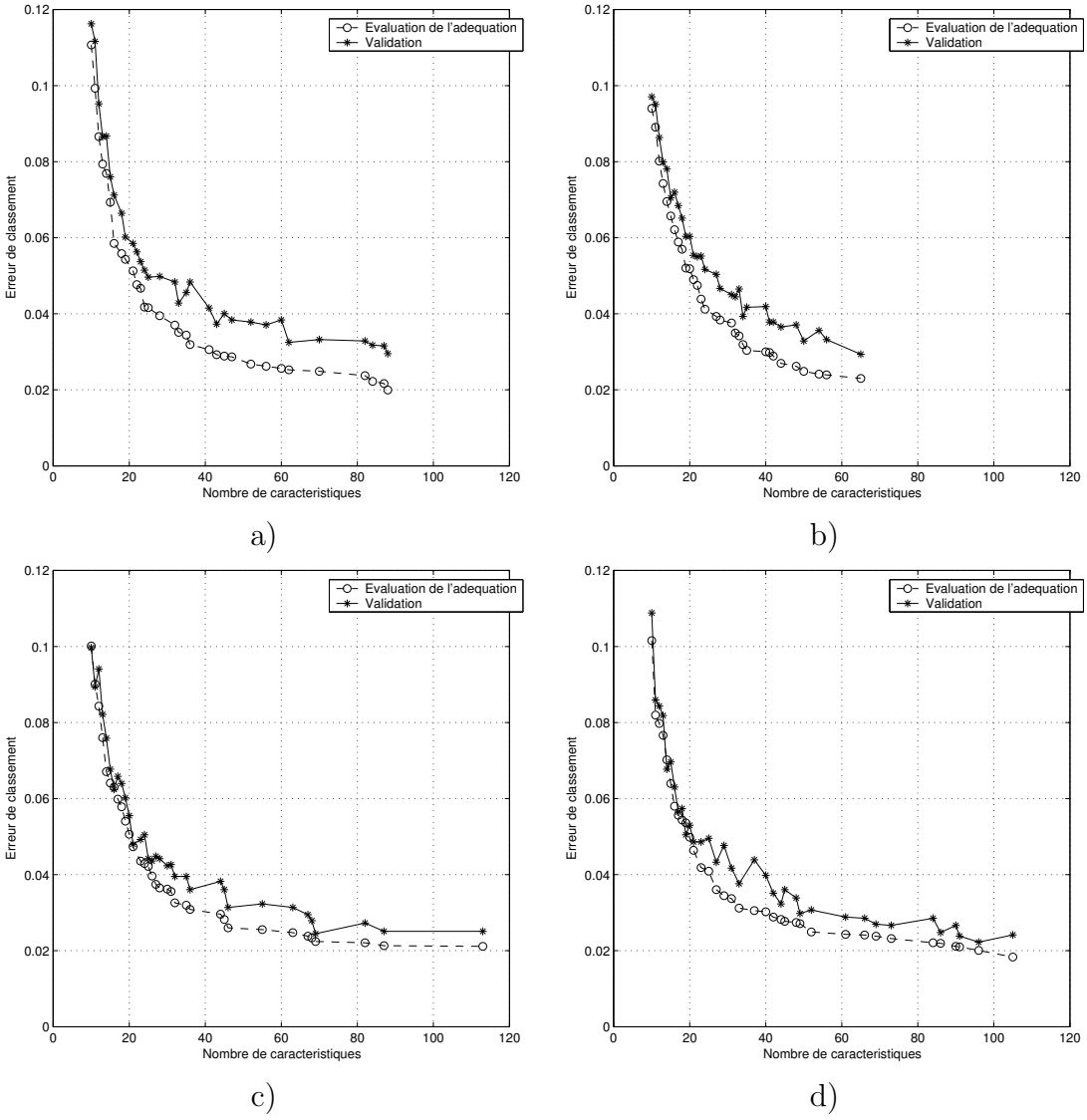


FIG. 4.7 – Fronts de Pareto pour les quatre évolutions de représentations de caractères cursifs : a) Evol1 ; b) Evol2 ; c) Evol3 ; d) Evol4.

```
(D5
  ((OR -40 5 30) ((CU 0.05 0.12 0.095) ((OR 45 50 15) ((CU 0.07 0.05 0.055) (HW
    (S3H
      ((CU 0.065 0.075 0.135) ((CU 0.125 0.11 0.02) ((OR -10 10 25) ((OR 5 20 15) T))))
      ((CU -0.075 0.05 0.07) ((OR 55 45 10) ((CU -0.005 0.05 0.05) ((OR 50 35 30)
        (MX ((OR -25 45 30) T))))))
      ((CU 0.125 0.105 0.15) ((OR 50 40 30) ((OR 20 25 25) ((CU -0.05 0.045 0.09) T)))))))))
  (S4
    (D2V
      (MX ((OR -25 45 30) T))
      (S3H
        ((CU 0.05 0.07 0.11) T)
        (MX T)
        ((CU 0.055 0.07 0.16) T)))
    ((OR -85 40 20) ((OR 20 40 25) ((OR 0 45 0) ((CU 0.015 0.145 0.15) T))))
    (S2H
      (HW ((OR 25 50 30) T))
      (MX ((OR -45 50 5) ((OR -85 40 20) ((CU 0.07 0.12 0.09) T)))))
    ((CU 0.125 0.155 0.02) ((CU 0.08 0.035 0.015) ((OR -40 20 20) ((CU -0.115 0.04 0.155)
      ((CU 0.055 0.07 0.16) T))))))
  (HW ((OR -35 30 20) (MX (MX
    (S5
      ((OR -60 35 25) ((OR 30 45 0) ((OR 50 35 30) (MX ((OR -25 45 30) T)))))
      ((CU 0.135 0.075 0.105) T)
      (MY
        (D5
          ((CU 0.075 0.12 0.07) T)
          ((CU 0.03 0.02 0.145) T)
          ((CU -0.09 0.085 0.075) T)
          T
          (MX T)))
      ((CU 0.155 0.06 0.095) ((CU 0.095 0.11 0.045) ((CU -0.095 0.04 0.11) T)))
      ((OR -40 40 30) ((OR 50 35 30) ((OR -55 40 10) T))))))))
  (HW ((OR -40 45 5) ((CU 0.125 0.105 0.15) ((OR 50 40 30) ((OR 20 25 25) ((CU -0.05 0.045 0.09)
    (D4
      ((OR -5 45 30) T)
      ((CU 0.125 0.045 0.12) T)
      ((OR -75 10 25) T)
      ((OR 80 20 10) (MY ((OR -30 45 10) ((OR 50 35 0) T))))))))))
  ((OR -40 5 30) ((CU 0.05 0.12 0.095) ((OR 45 50 15) ((CU 0.07 0.05 0.055) (HW
    (S3H
      ((CU 0.065 0.075 0.135) ((CU 0.125 0.11 0.02) ((OR -10 10 25) ((OR 15 40 25) T))))
      ((CU -0.075 0.05 0.07) ((OR 55 45 10) ((ZM 1.33357) ((OR 75 45 5) T))))
      ((CU -0.105 0.055 0.15) ((OR -75 30 30) ((OR 50 35 30) (MX ((OR -25 45 30) T))))))))))))
```

FIG. 4.8 – S-expression de l'arbre Evol1.

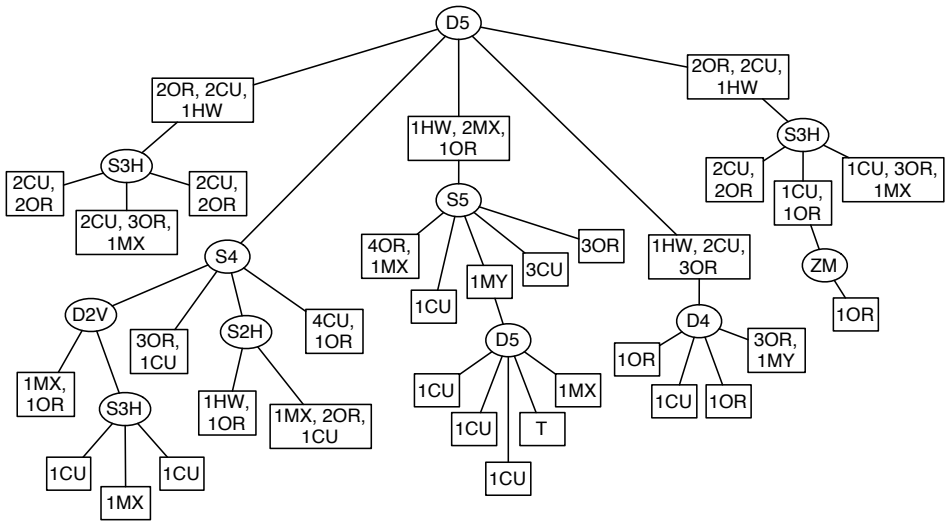


FIG. 4.9 – Structure d'arbre de la représentation Evol1 ; les ellipses représentent les primitives qui modifient la région et les cases représentent les primitives d'extraction de caractéristiques (voir le tableau 4.3).

stratégies de segmentation statique et guidée par les données. Il est intéressant de noter que, parmi les quatre représentations, le nœud modifiant la région qui est le plus près de la racine est toujours guidé par les données. Par conséquent, la stratégie guidée par les données n'est pas une si mauvaise idée, en autant qu'elle ne soit pas systématique. Pour l'une des représentations (Evol2), la racine ne modifie pas la région ; elle extrait donc des caractéristiques globales. Enfin, on peut observer que la profondeur de la segmentation hiérarchique est principalement de 2 et de 3 niveaux, mais peut parfois atteindre 4 niveaux.

Afin d'avoir une meilleure compréhension des faiblesses du système évolué de reconnaissance de caractères, nous avons examiné chaque caractère mal classé de l'ensemble de test. Plusieurs d'entre eux sont mal écrits ou mal segmentés, et quelques-uns sont manifestement mal étiquetés. La figure 4.10 montre un sous-ensemble de ces caractères. Ils sont présentés sous leur forme de reconstruction des traits par arcs de cercle, ce qui correspond à l'entrée du module d'extraction de caractéristiques de la PG. Nous supposons ici que le processus de filtrage qui produit cette décomposition en traits préserve toute les informations discriminantes contenues dans le caractère, malgré que l'on reconnaît qu'il s'agit d'une hypothèse forte. Rappelons que l'objectif de ce chapitre est d'exposer une nouvelle méthode pour optimiser le composant d'extraction de caractéristiques d'un système de reconnaissance de caractères, et non de présenter les meilleures performances.

Les caractères de la figure 4.10 sont ceux qui ont été mal classés par au moins trois

des quatre classifieurs. Ils représentent environ 2.2 % de DevTest-R02/V02 de la section 1a. Les exemples de caractères incomplets concernent surtout les chiffres 0 et 4 : par exemple, une barre oblique sans le 0 ou une barre verticale d'un 4 sans le reste du chiffre. Certains caractères ont l'air de points ou de tirets. Soit ils font partie des caractères mal segmentés, soit ils ont été mal étiquetés et ne devraient pas faire partie de la section 1a. Certains caractères sont méconnaissables, comme le dernier exemple des caractères 0 et 2, respectivement. D'autres contiennent deux caractères comme le premier exemple du caractère 7. D'autres encore sont manifestement mal étiquetés comme certains exemples des caractères 2, 3, 4, 6 et 7. En général, on peut voir que cet ensemble de test est assez difficile, même si certains caractères mal classés semblent reconnaissables. Cela peut signifier que notre ensemble de primitives aurait avantage à être augmenté avec de nouveaux types de caractéristiques.

4.4 Conclusion

Dans ce chapitre, nous avons vu à quel point la conception de caractéristiques discriminantes peut être contre-intuitive. Par exemple, l'utilisation systématique de la segmentation guidée par les données introduite à la section 4.2 a donné des taux de reconnaissance moins bons que son équivalent statique de la section 4.1, contrairement à ce qui était prévisible. Nous avons démontré à la section 4.3 que la programmation génétique peut être utilisée pour automatiser, au moins partiellement, le processus d'essais et erreurs qui accompagne le plus souvent le développement du composant d'extraction de caractéristiques des systèmes de reconnaissance des formes. Cette automatisation a été démontrée pour un cas particulier de reconnaissance de caractères manuscrits, mais l'approche générale n'est aucunement limitée à cette application.

L'approche proposée possède l'avantage de déplacer une partie du fardeau que constitue le développement coûteux fait par des humains, vers des calculs par ordinateurs, plus économiques et qui peuvent se faire sans arrêt, 24 heures par jour. Le problème à résoudre doit bien entendu être formulé convenablement afin de permettre la convergence vers des solutions intéressantes. Cela a été fait en utilisant une partition hiérarchique à base d'arbre de la matrice de caractères (*bounding box*), quelques primitives d'extraction de caractéristiques de haut niveau et un processus à deux objectifs qui minimise le nombre de caractéristiques et qui maximise le taux de reconnaissance. Il est possible de généraliser cette programmation génétique multi-objectif à plusieurs autres systèmes d'extraction de caractéristiques, en adaptant simplement les primitives au problème afin de permettre l'extraction adéquate de l'information discriminante.

Dans le cas particulier de notre système de reconnaissance de caractères, même si nous n'avons pas obtenu des taux de reconnaissance significativement plus élevés pour l'ensemble de données de test Unipen, il a été possible, avec la conception génétique de représentations de l'écriture cursive, de réduire significativement la taille de l'ensemble de caractéristiques et de favoriser un processus d'entraînement du classifieur plus stable.

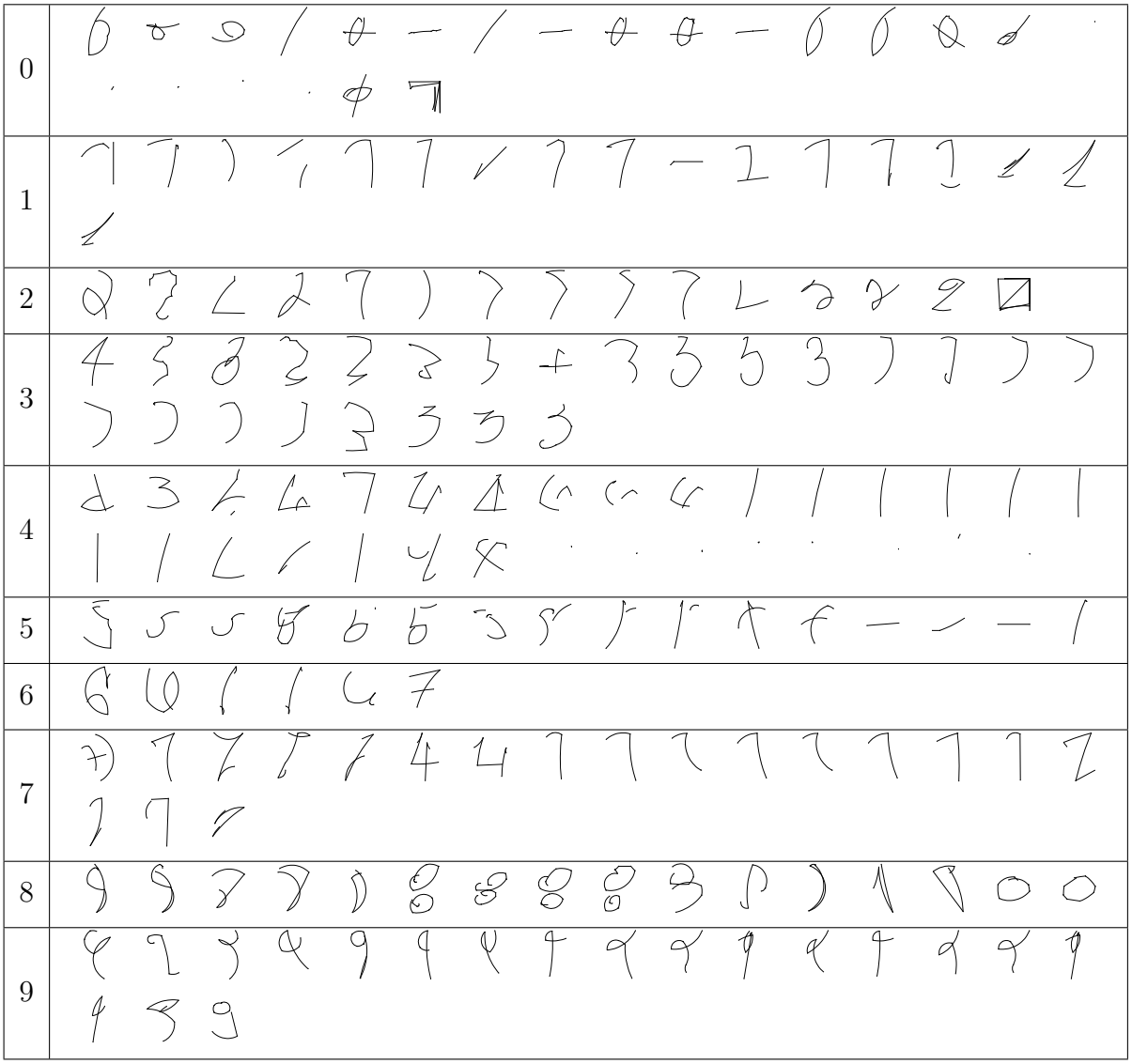


FIG. 4.10 – Caractères de DevTest-R02/V02 mal classés par au moins trois des quatre représentations évoluées.

Chapitre 5

Arbres de décision flous, extensibles et évolutionnaires

If you torture the data enough, it will confess.
Ronald Coase (prix Nobel d'économie 1991)

Ce chapitre porte sur un système développé dans le cadre du doctorat pour la combinaison de classifieurs dans une structure en arbre, dont l'échange d'information est basé sur la logique floue [128]. Le système se nomme EFFECT pour *Extensible Flexible Fuzzy Evolutionary Classifier Trees*. Le système utilise la programmation génétique (PG) pour déterminer la topologie des arbres d'une part, et il effectue un entraînement de chaque topologie candidate afin de déterminer les paramètres des unités de traitement qui la forment d'autre part.

Nous présentons à la section 5.1 les principes et motivations à la base du système EFFECT. Ensuite, à la section 5.2, nous présentons une revue de la littérature. Le fonctionnement algorithmique du système EFFECT est détaillé à la section 5.3, alors que la performance du système est évaluée à la section 5.4, avec des résultats sur des ensembles de données réelles. Par la suite, nous présentons à la section 5.5 une modification au système EFFECT original, afin de geler les valeurs de paramètres d'unités de traitement en cours d'évolution. Les résultats avec le système EFFECT modifié démontrent que les taux de classement restent pratiquement les mêmes, alors que la complexité moyenne des arbres est substantiellement réduite. Finalement, nous concluons à la section 5.6 sur l'extensibilité du système avec l'intégration de classifieurs plus complexes.

5.1 Principes et motivations

Les architectures avec ensembles de classifieurs [41] connaissent une popularité grandissante. Tout particulièrement, les *mixtures d'experts* consistent à prendre une décision en tenant compte des résultats fournis par des composants experts spécialisés pour des aspects précis du problème. La prise de décision peut se faire de nombreuses façons, par exemple avec une fonction discriminante qui pondère la valeur des différents experts selon leurs « compétences » respectives par rapport à la donnée testée. On note également d'autres approches avec un ensemble de classifieurs telles que le *bagging*, où chaque composant de classement est développé à partir d'un sous-ensemble différent des données d'entraînement, et le *boosting*, où une pondération différente est accordée à chaque donnée pour l'entraînement de chaque composant de classement selon les résultats obtenus jusqu'à présent.

Les arbres de décision [41, 98, 119] sont des techniques éprouvées pour le classement. Il existe plusieurs algorithmes pour l'entraînement d'arbres de décision, la plupart effectuant un découpage selon une dimension différente de l'espace des caractéristiques pour chaque nœud. Les arbres de décision sont particulièrement intéressants lorsque les caractéristiques sont nominales et lorsqu'il est nécessaire d'interpréter facilement la fonction discriminante du classifieur. On peut voir également les arbres de décision comme un agrégat de classifieurs très simples, qui effectuent un découpage répété du problème en sous-problèmes plus simples. En ce sens, on peut affirmer qu'un arbre de décision constitue une organisation hiérarchique d'un ensemble de classifieurs.

D'autre part, la logique floue [128] généralise la logique booléenne avec la notion de vérité partielle. Alors que la logique classique utilise des termes binaires (0 ou 1, vrai ou faux), la logique floue est basée sur la notion de degré de vérité. Le degré de vérité d'un énoncé est représenté par une valeur réelle entre 0 et 1, par exemple 0 indiquant que l'énoncé est totalement faux, 1 indiquant que l'énoncé est totalement vrai ou 0.5 indiquant que l'énoncé est à moitié vrai (ou à moitié faux). La logique floue est particulièrement intéressante dans les contextes d'ingénierie de système où les notions de possibilité et d'incertitude sont omniprésentes. Les systèmes basés sur la logique floue utilisent souvent des mémoires associatives floues [128], qui sont les équivalents flous des systèmes à base de règles de la logique binaire classique. Dans ces systèmes, un traitement de type min-max flou est effectué, avec des conjonctions logiques par opérations max entre les degrés de vérité et des disjonctions logiques par opérations min sur ces mêmes degrés. Des équivalents flous aux arbres de décision classiques ont été développés il y a longtemps [22], avec un traitement flou de type min-max pour la discrimination des données à chaque nœud. Comparativement aux arbres de décision

classiques, ces arbres de décision flous sont parfois mieux adaptés au traitement de certains types de problèmes, particulièrement si les caractéristiques des données sont essentiellement numériques.

Le système EFFECT se base sur ces trois approches pour effectuer une agrégation de classifieurs en arbres de décision flous. Les nœuds des arbres de décision sont des classifieurs de natures diverses. Pour illustrer le fonctionnement du système, nous avons choisi d'utiliser ici trois principaux classifieurs : 1) une unité de classement paramétrique linéaire de type neurone ADALINE [54], 2) une unité de classement non-paramétrique de type plus proche voisin (PPV) [31, 41] et 3) une unité de classification non supervisée à K-moyennes floues [42]. Ces unités représentent un éventail d'algorithmes de classement relativement simples. L'approche proposée n'est pas limitée à ces types d'unités et de nombreux autres algorithmes de classement pourraient être intégrés au système. De plus, étant donné la nature des unités de traitement utilisées, les décisions à chaque nœud sont prises en tenant compte de toutes les variables de l'espace des caractéristiques. Cette approche contraste avec celle de la plupart des arbres de décision classiques, où chaque décision se prend selon un seul attribut des données. Cela a l'avantage d'offrir une plus grande flexibilité dans la partition hiérarchique de l'espace, mais implique que dans le cas général, les classifieurs ainsi produits sont pratiquement impossibles à interpréter par des humains.

La topologie des arbres de décision est évoluée par PG. Les paramètres des unités de traitement des arbres sont calculés par un processus d'entraînement effectué pour chaque topologie testée. L'algorithme d'entraînement général est bâti de sorte que les temps de calcul soient acceptables, étant donné que des milliers (peut-être même des millions) de topologies peuvent être entraînées pour chaque évolution. Les paramètres des nœuds de l'arbre sont déterminés à partir d'un algorithme d'entraînement propre à chaque unité de classement. L'entraînement se fait de la racine vers les feuilles, le résultat du classement de chaque donnée à un nœud étant transmis à ses nœuds fils afin d'en moduler l'entraînement.

Ce système permet la formation d'ensembles de classifieurs en arbres. La spécialisation des différents composants experts se fait par un découpage hiérarchique générant des régions d'activité de plus en plus restreintes dans l'espace des caractéristiques, selon la profondeur du nœud traité. L'utilisation de degrés d'appartenance flous permet un traitement robuste et possibiliste du découpage hiérarchique, réduisant ainsi la sensibilité au bruit par rapport aux approches avec décisions binaires ou nominales. De plus, le système comporte une séparation explicite entre la recherche de topologies par PG et l'apprentissage statistique des paramètres numériques. Si l'on se reporte à l'approche méthodologique proposée à la section 1.4, cela répond aux deux premiers principes, avec

un hybride entre la PG, parfaitement adaptée à la recherche de topologies d'arbres, et des heuristiques pour combiner des classifieurs et en évaluer les paramètres.

5.2 Revue de littérature

Depuis longtemps, l'incertitude et le flou ont été intégrés aux systèmes de classement automatique. Dès 1977, Chang et Pavlidis [22] ont proposé des algorithmes pour la génération d'arbres de décision flous. En partie grâce au développement remarquable des capacités de calcul des ordinateurs, on a assisté dans les 15 dernières années à l'élaboration de nombreux systèmes flous hybrides avec des algorithmes évolutionnaires (AE) et/ou des réseaux de neurones, capables d'apprentissage et d'adaptation [30]. Dans le contexte de ce chapitre, différents aspects de ce domaine précis sont d'intérêt. Dans un premier temps, on note quelques approches [74, 113, 130] sur l'évolution de la structure de systèmes flous accompagnée d'un apprentissage par optimisation locale de paramètres numériques. Dans le système de Pedrycz et Reformat [113], l'évolution par PG des règles d'un système flou est suivie d'une optimisation des paramètres numériques du meilleur individu de l'évolution. Dans le système FuGeNeSys de Russo [130], la procédure d'optimisation est intégrée au processus évolutionnaire, avec un apprentissage des paramètres numériques pour chaque individu ayant une adéquation supérieure au meilleur individu obtenu jusqu'à présent. L'approche de Kim et Lee [74] est particulièrement intéressante dans le contexte de ce chapitre, étant donné que l'apprentissage des paramètres du système flou par optimisation locale est fait pour chacune des topologies testées au cours de la recherche évolutionnaire. Une telle intégration de la procédure d'optimisation locale dans la boucle d'évolution devrait permettre une interaction positive entre l'apprentissage et la recherche évolutionnaire. De plus, les valeurs des paramètres appris sont conservées dans la représentation des individus, provoquant ainsi une adaptation des paramètres dans la lignée de la théorie de l'évolution de Lamarck [85].

Différentes approches évolutionnaires ont été explorées pour l'évolution de la topologie d'arbres de décision. On note trois publications [43, 94, 113] qui présentent des systèmes qui utilisent directement la PG pour découvrir des topologies performantes dans un contexte de classement. Aussi, Ishibuchi *et al.* [69] utilisent un algorithme génétique (AG) pour l'évolution de systèmes de règles floues de type Michigan [3, 30], où la population représente un système à base de règles, chaque individu la composant décrivant une règle. On note également des approches pour une co-évolution coopérative des différentes parties de systèmes à base de règles floues [94, 114].

D'autre part, Sethi et Yoo [131] présentent une heuristique pour une construc-

- Q : taille de l'ensemble des données à classer ;
- D : taille de l'ensemble des caractéristiques ;
- L : nombre d'étiquettes de classe différentes ;
- $\mathbf{l} = [l_1 \ldots l_L]^T$: vecteur des étiquettes de classe possibles ;
- $\mathbf{P} = [\mathbf{p}_1 \ldots \mathbf{p}_Q]$: ensemble des données à classer, où $\mathbf{p}_i$ est le vecteur de la donnée d'indice i ;
- $\mathbf{a} = [a_1 \ldots a_Q]^T$: étiquettes de classe obtenues par classement des données ;
- $\mathbf{d} = [d_1 \ldots d_Q]^T$: véritables étiquettes de classe des données ;
- $\mathbf{S} = [\mathbf{s}_1 \ldots \mathbf{s}_Q]$: matrice des degrés d'appartenance aux classes, où $\mathbf{s_i}$ est le vecteur des degrés d'appartenance de la donnée i aux différentes classes ;
- $\mathbf{f} = [f_1 \ldots f_Q]^T$: degrés d'appartenance flous des données ;

FIG. 5.1 – Variables générales utilisées dans la description des algorithmes du système.

tion descendante (*top-down*) d'arbres de décision avec unités de type neurone perceptron. Des similarités peuvent être tracées entre cet algorithme et l'approche utilisée dans le système EFFECT pour l'entraînement de topologies d'arbres. Finalement, les techniques pour la formation d'ensembles de classifieurs[1] sont nombreuses et variées [18, 65, 75, 147]. On doit cependant souligner plus particulièrement les travaux de Kuncheva et Jain [84], où un AG est utilisé pour la sélection de différents sous-ensembles de prototypes, afin de déterminer la configuration de quelques classifieurs formant une mixture d'experts.

5.3 Système EFFECT

Cette section présente de façon détaillée le fonctionnement du système EFFECT. En premier lieu, les algorithmes utilisés pour la propagation de l'information entre les composants des arbres de décision et l'entraînement des unités de traitement sont présentés (sous-section 5.3.1). Ensuite, nous présentons les algorithmes d'entraînement spécifiques aux différentes unités de classement (sous-sections 5.3.2 à 5.3.5) et la configuration utilisée pour l'évolution de la topologie des arbres par PG (sous-section 5.3.6).

La figure 5.1 décrit quelques variables générales utilisées dans la description des différents algorithmes présentés dans les sous-sections suivantes. Les colonnes $\mathbf{p}_i$ de la matrice $\mathbf{P}$ représentent les données à classer ou à utiliser pour l'entraînement de

[1] Il est à noter que les termes *ensemble de classifieurs* et *classifieur multiple* sont employés comme des synonymes dans ce chapitre, sans faire de distinction particulière entre les deux appellations.

l'arbre. L'ensemble $\mathbf{P}$ comprend Q données et chaque donnée comporte D valeurs qui correspondent aux caractéristiques du problème. Le vecteur $\mathbf{d}$ correspond aux véritables étiquettes de classe des données de $\mathbf{P}$, alors que le vecteur $\mathbf{a}$ correspond aux étiquettes de classe des données déterminées par classement. Le vecteur $\mathbf{l}$ de taille L comprend les différentes étiquettes de classe possibles pour les données. Le vecteur $\mathbf{f}$ représente les degrés d'appartenance flous des données en un point de l'arbre, alors que la matrice $\mathbf{S}$ représente les degrés d'appartenance aux classes de chaque donnée en un point de l'arbre (voir la prochaine sous-section pour plus d'explications).

5.3.1 Combinaison floue et entraînement des unités

Les arbres de décision flous du système EFFECT sont des combinaisons d'unités de classement qui altèrent des degrés d'appartenance flous associés aux données. Le traitement se fait en deux temps, premièrement avec une passe descendante dans l'arbre par une suite d'opérations de disjonctions floues (opérations min) des degrés d'appartenance, suivie d'une passe ascendante par des conjonctions floues (opérations max) de degrés d'appartenance aux classes. La passe descendante est effectuée de la racine vers les feuilles, avec l'utilisation d'un vecteur des degrés d'appartenance flous des données à classer. La valeur initiale du degré d'appartenance de chaque donnée à la racine est 1. Des opérations min sont effectuées à chaque niveau de l'arbre entre les degrés d'appartenance flous de ces données et la valeur résultant du classement fait par l'unité de classement. Chaque feuille possède un vecteur de scores de classement $\mathbf{m}$ des données pour chacune des classes. Pour chaque feuille de l'arbre, le degré d'appartenance f_i de chaque donnée, obtenu par le traitement descendant, est multiplié avec le vecteur des scores de classement $\mathbf{m}$ de la feuille, afin de générer un vecteur de classement $\mathbf{s}_i$ pour chaque donnée (voir équation 5.1). Cette multiplication consiste en quelque sorte à déterminer une corrélation entre le degré d'appartenance de chaque donnée et les scores de classement déterminés lors de l'entraînement. Chaque valeur de ce vecteur de classement correspond au degré d'appartenance de la donnée à la classe associée. Ensuite, on effectue un traitement ascendant par des opérations max entre les différentes valeurs des vecteurs de classement $\mathbf{s}_i$. Il en résulte à la racine un vecteur des degrés d'appartenance aux classes par données. On affecte à chaque donnée $\mathbf{p}_i$ l'étiquette de la classe de valeur maximale du vecteur de classement $\mathbf{s}_i$ (voir équation 5.4).

Le figure 5.2 présente la fonction récursive $\mathbf{S} = \text{classer}(\mathbf{P}, \mathbf{f}, n)$ utilisée par l'algorithme de classement des arbres de décision. Les arguments de cette fonction sont : l'ensemble des données à classer $\mathbf{P}$, les degrés d'appartenance des données $\mathbf{f} = [1 \ldots 1]^T$ et la racine de l'arbre comme nœud n. À la suite de cet appel, ont détermine le classement de chacune des données $\mathbf{p}_i$ de l'ensemble $\mathbf{P}$ en assignant à l'étiquette de classement $\mathbf{a}$

Soit la fonction $\mathbf{S} = \text{classer}(\mathbf{P}, \mathbf{f}, n)$, avec les variables :
- n : nœud courant ;
- H : nombre de fils du nœud n ;
- $n(k)$: $k$ième nœud fils de n, avec $k = 1, \ldots, H$ (seulement si n est une branche) ;
- $\mathbf{m} = [m_1 \ \ldots \ m_L]^T$: vecteur des scores de classement du nœud n (seulement si n est une feuille).

1. Initialiser $\mathbf{s}_i = [0 \ \ldots \ 0]^T$ pour $i = 1, \ldots, Q$;
2. **Si** le nœud n est une feuille, **alors** :
   (a) Calculer la matrice des degrés d'appartenance aux classes $\mathbf{S}$ par la multiplication du vecteur des scores de classement $\mathbf{m}$ avec les degrés d'appartenance flous des données :
   $$\mathbf{s}_i = f_i \mathbf{m}, \quad i = 1, \ldots, Q \tag{5.1}$$
3. **Si** le nœud n est une branche, **alors** :
   (a) Appeler la fonction de classement $\mathbf{F}' = \text{classer}_{\text{type}}(\mathbf{P}, \mathbf{f}, n)$ associée au type exact du nœud n pour calculer les degrés d'appartenance flous des nœuds fils (voir définition des fonctions aux sous-sections 5.3.2 à 5.3.5) ;
   (b) Calculer le minimum entre les degrés d'appartenance des nœuds fils et les degrés du nœud actuel :
   $$f'_{i,j} = \min(f'_{i,j}, f_i), \quad i = 1, \ldots, Q, \quad j = 1, \ldots, H \tag{5.2}$$
   (c) Appeler récursivement la fonction $\mathbf{S}'(k) = \text{classer}(\mathbf{P}, \mathbf{f}'_k, n(k))$ pour évaluer la matrice des degrés d'appartenance aux classes de chacun des nœuds fils $n(k)$, avec $k = 1, \ldots, H$;
   (d) Calculer le maximum des valeurs des matrices de degrés d'appartenance entre les valeurs actuelles et celles des nœuds fils :
   $$s_{i,j} = \max(s_{i,j}, s'_{i,j}(k)), \quad i = 1, \ldots, Q, \quad j = 1, \ldots, L, \quad k = 1, \ldots, H \tag{5.3}$$
4. **Retourner S** comme matrice des degrés d'appartenance de classement.

FIG. 5.2 – Fonction récursive de l'algorithme de classement général.

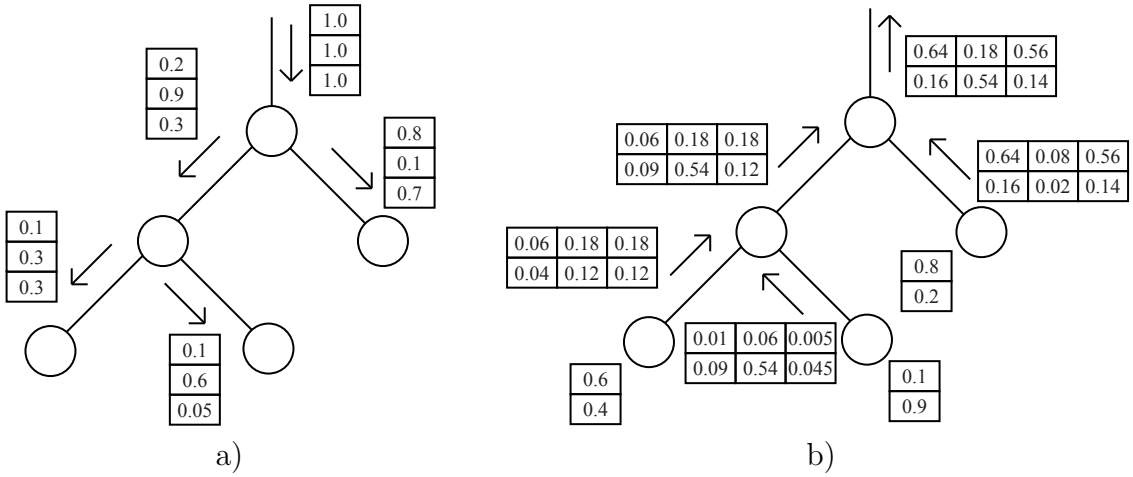


FIG. 5.3 – Illustration du classement dans les arbres de décision flous : a) passe descendante et b) passe ascendante ; avec $Q = 3$ et $L = 2$.

l'étiquette de la valeur maximale du vecteur des degrés d'appartenance $\mathbf{s}_i$, selon l'équation suivante.

$$g = \underset{j=1,\ldots,L}{\operatorname{argmax}}(s_{i,j}), \quad a_i = l_g, \quad i = 1, \ldots, Q \tag{5.4}$$

La figure 5.3 illustre par un exemple simple la passe descendante et la passe ascendante pour le classement dans les arbres de décision flous. Le traitement présenté dans cette figure est effectué sur trois données qui peuvent être classées en deux classes différentes à l'aide d'un arbre de décision de quatre nœuds. On présente la passe descendante à la figure 5.3a, avec des tableaux verticaux de trois valeurs qui représentent les degrés d'appartenance $\mathbf{f}$ des données en différents points de l'arbre. On observe que les valeurs diminuent au fur et à mesure que l'on descend dans l'arbre, étant donné la suite d'opérations min effectuées entre les degrés d'appartenance provenant du nœud parent et les degrés d'appartenance résultant du traitement caractéristique de l'unité de classement associée au nœud actuel. Pour ce qui est de la passe ascendante présentée à la figure 5.3b, on remarque deux types de tableaux. Il y a premièrement des tableaux verticaux de deux valeurs représentant les vecteurs des scores de classement $\mathbf{m}$ de chaque feuille. Deuxièmement, il y a des tableaux de deux lignes et de trois colonnes qui constituent les matrices de degrés d'appartenance aux classes $\mathbf{S}$ en chaque point de l'arbre. Les trois tableaux de deux lignes et de trois colonnes générés par les feuilles de l'arbre sont calculés par une multiplication des degrés d'appartenance $\mathbf{f}$ des données (présentés à la figure 5.3a) avec les vecteurs des scores de classement $\mathbf{m}$ des feuilles (voir équation 5.1). Les tableaux au niveau des branches de l'arbre proviennent d'opérations max entre les valeurs des niveaux inférieurs. À la racine de l'arbre, la décision pour le classement des données se prend selon le degré d'appartenance de classement pour chacune des

données, soit la première classe pour la première donnée et la troisième donnée (respectivement, degrés d'appartenance de classement de 0.64 et 0.56), alors que la deuxième donnée est assignée à la deuxième classe (degré d'appartenance de classement de 0.54).

L'entraînement des arbres de décision est similaire au classement des données, avec en plus une étape d'entraînement durant la passe descendante, afin de déterminer la valeur des paramètres des unités de classement. Pour la plupart des unités de classement de l'arbre, les valeurs des paramètres obtenues suite à l'entraînement de l'unité sont utilisées pour classer les données durant la passe descendante. Pour chaque feuille de l'arbre, le vecteur des scores de classement $\mathbf{m}$ est estimé au cours de l'entraînement par une somme quadratique normalisée des degrés d'appartenance flous (voir équation 5.5). La passe ascendante de l'entraînement reste inchangée par rapport à l'algorithme de classement général. La figure 5.4 présente de façon détaillée la fonction récursive $\mathbf{S} = \text{entrainer}(\mathbf{P}, \mathbf{d}, \mathbf{f}, n)$ utilisée par la procédure d'entraînement. L'entraînement d'un arbre de décision se fait donc en appelant cette fonction $\mathbf{S} = \text{entrainer}(\mathbf{P}, \mathbf{d}, \mathbf{f}, n)$, avec comme arguments l'ensemble des données d'entraînement $\mathbf{P}$, les véritables étiquettes des données $\mathbf{d}$, les degrés d'appartenance $\mathbf{f} = [1 \ldots 1]^T$ et la racine de l'arbre comme nœud n. Comme pour le classement, on peut déterminer l'étiquette de classement pour chacune des données de $\mathbf{P}$ en utilisant l'équation 5.4.

Les fonctions $\mathbf{S} = \text{classer}_{\text{type}}(\mathbf{P}, \mathbf{f}, n)$ et $\mathbf{S} = \text{entrainer}_{\text{type}}(\mathbf{P}, \mathbf{d}, \mathbf{f}, n)$ sont définies pour les différentes unités de classement du système EFFECT. Les algorithmes de classement et d'entraînement des différentes unités utilisées dans le système sont présentés aux quatre sous-sections suivantes.

5.3.2 Unité ADALINE

L'unité de classement de type ADALINE comprend un neurone linéaire entraîné avec une règle d'optimisation par les moindres carrés (en anglais, *least mean squares* ou LMS) [54]. La règle standard d'entraînement du neurone ADALINE est modifiée pour tenir compte des degrés d'appartenance flous f_i des données. De plus, étant donné qu'un neurone est un classifieur dichotomique, c'est-à-dire qu'il ne peut discriminer les données que selon deux classes, il faut ré-étiqueter les données avant de procéder à l'entraînement du neurone. À cette fin, une heuristique est proposée à la figure 5.5 pour un ré-étiquetage des données en deux classes. Dans cet algorithme, on calcule d'abord la somme normalisée des degrés d'appartenance des données associées à chaque classe originale (voir équation 5.6). Par la suite, on effectue une boucle itérative sur les classes originales, en ordre décroissant des valeurs des sommes normalisées. Une classe originale est ré-étiquetée à chaque itération afin d'équilibrer le plus possible la

Soit la fonction $\mathbf{S} = \text{entrainer}(\mathbf{P}, \mathbf{d}, \mathbf{f}, n)$, avec les variables :
- n : nœud courant ;
- H : nombre de fils du nœud n ;
- $n(k)$: $k$ième nœud fils de n, avec $k = 1, \ldots, H$ (seulement si n est une branche) ;
- $\mathbf{m} = [m_1 \ \ldots \ m_L]^T$: vecteur des scores de classement de n (seulement si n est une feuille).

1. Initialiser $\mathbf{s}_i = [0 \ \ldots \ 0]^T$ pour $i = 1, \ldots, Q$;
2. **Si** le nœud n est une feuille, **alors** :
   (a) Calculer les valeurs du vecteur des scores de classement $\mathbf{m}$ du nœud feuille n par une somme quadratique normalisée des degrés d'appartenance :

$$m_i = \frac{\sum_{j=1}^{Q} g_{j,i}^2}{\sum_{j=1}^{Q} f_j^2}, \quad g_{j,i} = \begin{cases} f_j & : \quad d_j = l_i \\ 0 & : \quad \text{autrement} \end{cases}, \quad i = 1, \ldots, L \tag{5.5}$$

   (b) Calculer la matrice des degrés d'appartenance $\mathbf{S}$ selon l'équation 5.1 ;
3. **Si** le nœud n est une branche, **alors** :
   (a) Déterminer les paramètres de l'unité de classement et les degrés d'appartenance flous des nœuds fils par un appel à la fonction $\mathbf{F}' = \text{entrainer}_{\text{type}}(\mathbf{P}, \mathbf{d}, \mathbf{f}, n)$ selon l'unité de classement associée au nœud n (voir la définition des fonctions aux sous-sections 5.3.2 à 5.3.5) ;
   (b) Calculer le degré d'appartenance minimal selon l'équation 5.2 ;
   (c) Appeler récursivement la fonction $\mathbf{S}'(k) = \text{entrainer}(\mathbf{P}, \mathbf{f}'_k, n(k))$ pour évaluer la matrice des degrés d'appartenance aux classes des nœuds fils $n(k)$, avec $k = 1, \ldots, H$;
   (d) Calculer la valeur maximale des valeurs des matrices de degrés d'appartenance selon l'équation 5.3.
4. **Retourner S** comme matrice des degrés d'appartenance de classement.

FIG. 5.4 – Fonction récursive de l'algorithme d'entraînement général.

Soit la fonction $\mathbf{d}' = \text{dichotomiser}(\mathbf{P}, \mathbf{d}, \mathbf{f})$, avec les variables :
– $\mathbf{d}' = [d'_1 \ \ldots \ d'_Q]^T$: résultat du ré-étiquetage des données en deux classes ;
– $\mathbf{u} = [u_1 \ \ldots \ u_L]^T$: variable temporaire contenant le ré-étiquetage des classes originales.

1. Calculer les sommes normalisées des degrés d'appartenance flous par classe :

$$h_i = \frac{\sum_{j=1}^{Q} g_{j,i}}{\sum_{j=1}^{Q} f_j}, \quad g_{j,i} = \begin{cases} f_j & : \quad d_j = l_i \\ 0 & : \quad \text{autrement} \end{cases}, \quad i = 1, \ldots, L \tag{5.6}$$

2. Sélectionner la classe $k(1)$ dont la somme normalisée est maximale :

$$k = \underset{i=1,\ldots,Q}{\text{argmax}}(h_i) \tag{5.7}$$

3. Initialiser la somme $z = h_{k(1)}$ des degrés de la première classe ;
4. Ré-étiqueter la classe de somme maximale, $u_{k(1)} = 1$;
5. Affecter une valeur négative à la somme normalisée, $h_{k(1)} = -1$;
6. **Pour tous les** $i = 2, \ldots, L$:
   (a) Déterminer avec l'équation 5.7 l'indice $k(i)$ de la classe de somme maximale ;
   (b) **Si** $|z + h_{k(i)} - 0.5| < |z - 0.5|$, **alors** : $z = z + h_{k(i)}$ et $u_{k(i)} = 1$,
   sinon : $u_{k(i)} = 2$;
   (c) Affecter une valeur négative à la somme normalisée, $h_{k(i)} = -1$;
7. **Si** $\mathbf{u} = [1 \ \ldots \ 1]^T$, **alors** : $u_{k(1)} = 2$;
8. Ré-étiqueter les données : $d'_i = u_{d_i}$ pour $i = 1, \ldots, Q$;
9. **Retourner** $\mathbf{d}'$ comme résultat du ré-étiquetage en deux classes.

FIG. 5.5 – Algorithme de ré-étiquetage en deux classes.

Soit la fonction $\mathbf{F}' = \text{classer}_{\text{ADA}}(\mathbf{P}, \mathbf{f}, n)$, avec les variables :
- n : nœud courant ;
- $\mathbf{F}' = [\mathbf{f}'(1)\, \mathbf{f}'(2)]$: matrice des degrés d'appartenance flous des nœuds fils ;
- $\mathbf{w} = [w_1 \ \ldots \ w_D]^T$ et b : poids du neurone ADALINE associé au nœud n.

1. **Pour tous les** $i = 1, \ldots, Q$:
   (a) Calculer la sortie du neurone :

$$y_i = \mathbf{w}^T \mathbf{p}_i - b \tag{5.8}$$

   (b) Calculer les degrés d'appartenance des deux nœuds fils :

$$f'_i(1) = \begin{cases} 1 & : \quad y_i > 1 \\ 0 & : \quad y_i < 0 \\ y_i & : \quad \text{autrement} \end{cases}, \quad f'_i(2) = 1 - f'_i(1) \tag{5.9}$$

2. **Retourner** $\mathbf{F}' = [\mathbf{f}'(1)\, \mathbf{f}'(2)]$ comme la matrice des degrés d'appartenance des nœuds fils.

FIG. 5.6 – Algorithme de classement des unités ADALINE.

distribution des degrés d'appartenance des données entre les deux nouvelles classes. Cet algorithme de ré-étiquetage consiste donc en une approche vorace de complexité linéaire avec équilibrage des degrés d'appartenance.

Chaque unité de classement ADALINE possède des poids $\mathbf{w}$ et b, qui sont déterminés à l'entraînement. La figure 5.6 présente l'algorithme de classement de l'unité ADALINE, qui suit une formulation standard du classement avec un neurone. D'autre part, la figure 5.7 présente l'algorithme d'entraînement de l'unité. On remarque à l'équation 5.12 que la correction des poids du neurone est faite par une descente du gradient [54], modifiée pour moduler la correction selon le degré d'appartenance normalisé $\bar{f}_i$ de la donnée traitée. Le calcul de l'erreur quadratique moyenne de l'équation 5.13 est modifié de la même façon. Avec ces modifications à l'algorithme original, on tient compte de l'intensité du signal de chaque donnée au nœud actuel de l'arbre EFFECT pour la mise à jour des poids et la mesure de l'erreur de sortie. Pour toutes les expérimentations présentées dans le présent chapitre, les paramètres utilisés pour l'entraînement de chacune des unités ADALINE sont : un taux d'apprentissage de $\eta = 0.1$, un nombre maximal de $t^{max} = 50$ itérations et une interruption de l'entraînement lorsque la racine de l'erreur quadratique moyenne est inférieure à $\varepsilon = 0.05$.

Soit la fonction $\mathbf{F}' = \text{entrainer}_{\text{ADA}}(\mathbf{P}, \mathbf{d}, \mathbf{f}, n)$, avec les variables :
- n : nœud courant ;
- $\mathbf{F}' = [\mathbf{f}'(1)\, \mathbf{f}'(2)]$: matrice des degrés d'appartenance flous des nœuds fils ;
- $\mathbf{w} = [w_1 \ \ldots \ w_D]^T$ et b : poids du neurone ADALINE du nœud n ;
- η : taux d'apprentissage du neurone (fixé à 0.1) ;
- t^{max} et ε : critères d'arrêt (fixés à $t^{max} = 50$ et $\varepsilon = 0.05$).

1. Initialiser $\mathbf{w} = [0 \ \ldots \ 0]^T$, $b = 0$ et $E(0) = \infty$;
2. Appeler la fonction $\mathbf{d}' = \text{dichotomiser}(\mathbf{P}, \mathbf{d}, \mathbf{f})$ (voir figure 5.5) ;
3. Calculer les degrés d'appartenance normalisés :

$$\bar{f}_i = f_i \Big/ \max_{j=1,\ldots,Q} (f_j) \ , \quad i = 1, \ldots, Q \tag{5.10}$$

4. **Pour tous les** $t = 1, \ldots, t^{max}$, **tant que** $E(t-1) \geq \varepsilon$:
   (a) **Pour tous les** $i = 1, \ldots, Q$, dans le désordre (permutation aléatoire) :
      i. Calculer la sortie y_i avec l'équation 5.8 et l'erreur de cette sortie :

$$e_i = \begin{cases} 1 - y_i & : \quad d'_i = 1 \quad \text{et} \quad y_i < 1 \\ -y_i & : \quad d'_i = 2 \quad \text{et} \quad y_i > 0 \\ 0 & : \quad \text{autrement} \end{cases} \tag{5.11}$$

      ii. Mettre à jour les poids du neurone :

$$\mathbf{w} = \mathbf{w} + 2\eta e_i \bar{f}_i \mathbf{p}_i, \quad b = b - 2\eta e_i \bar{f}_i \tag{5.12}$$

   (b) Calculer la nouvelle racine de l'erreur quadratique moyenne :

$$E(t) = \sqrt{\left(\sum_{i=1}^{Q} e_i^2 \bar{f}_i \right) \Big/ \left(\sum_{i=1}^{Q} \bar{f}_i \right)} \tag{5.13}$$

5. Appeler la fonction $\mathbf{F}' = \text{classer}_{\text{ADA}}(\mathbf{P}, \mathbf{f}, n)$ pour classer les données ;
6. **Retourner** $\mathbf{F}'$ comme matrice des degrés d'appartenance des nœuds fils.

FIG. 5.7 – Algorithme d'entraînement des unités ADALINE.

5.3.3 Unité plus proche voisin

La seconde unité de classement employée est basée sur la règle du PPV [31, 41]. L'unité effectue un classement à deux classes à l'aide d'un nombre restreint de prototypes de données. Chaque nœud d'un arbre de décision de type unité PPV possède un ensemble de prototypes $\mathbf{W} = [\mathbf{w}_1 \ldots \mathbf{w}_C]$. Une distance euclidienne standard est utilisée pour évaluer la distance entre les données. Les degrés d'appartenance des données à classer sont modulés entre 0 et 1 lorsque les deux plus proches voisins appartiennent à des classes différentes, à partir des distances à ces deux plus proches voisins (voir équation 5.16). L'algorithme de classement des unités PPV est détaillé à la figure 5.8.

L'entraînement est fait en bâtissant un ensemble de prototypes par une itération sur les données d'entraînement, en ordre décroissant des degrés d'appartenance. Les données qui sont mal classées par les prototypes choisis jusqu'à présent sont sélectionnées pour être ajoutées à l'ensemble des prototypes. De plus, la taille de l'ensemble des prototypes est limitée par le paramètre C^{max}. Cette sélection de prototypes est fortement inspirée de l'algorithme de condensation de Hart [58] présenté à la figure 3.3. Avant l'entraînement, un ré-étiquetage des données en deux classes se fait avec le même algorithme que celui utilisé pour les unités ADALINE (voir figure 5.5). La figure 5.9 présente l'algorithme d'entraînement des unités PPV. Il est à noter que le nombre maximal de prototypes dans chaque nœud de type PPV est de $C^{max} = 6$ pour toutes les expérimentations présentées dans le présent chapitre.

5.3.4 Unité à K-moyennes floues

Le troisième type principal d'unités de classement permet une classification non supervisée des données selon l'algorithme des K-moyennes floues (en anglais : *fuzzy K-means*) [42]. L'algorithme de classement de cette unité est d'une certaine façon similaire à l'unité PPV avec l'utilisation de « prototypes », c'est-à-dire des points considérés comme les centres des nuages de données. Les algorithmes de classement diffèrent cependant par un calcul pour les K-moyennes floues des degrés d'appartenance des données qui tient compte de la distance à tous les centres de l'unité (voir équation 5.22). Cet algorithme est présenté à la figure 5.10.

La figure 5.11 présente l'algorithme d'entraînement des unités à K-moyennes floues. À la première étape de l'algorithme, les centres sont initialisés aux K données de degré d'appartenance maximal en position actuelle de l'arbre (nœud n), avec bris aléatoire de la sélection pour les cas où il y a égalité des degrés. Ensuite, on remarque que

Soit la fonction $\mathbf{F}' = \text{classer}_{\text{PPV}}(\mathbf{P}, \mathbf{f}, n)$, avec les variables :
- n : nœud courant ;
- C : nombre de prototypes du nœud n ;
- $\mathbf{W} = [\mathbf{w}_1 \ \ldots \ \mathbf{w}_C]$: prototypes du nœud n ;
- $\mathbf{d}^{proto} = [d_1^{proto} \ \ldots \ d_C^{proto}]^T$: étiquettes de classe des prototypes du nœud n.

1. **Pour tous les** $i = 1, \ldots, Q$:
   (a) Déterminer le plus proche voisin de l'ensemble des prototypes :
   $$t = \underset{j=1,\ldots,C}{\text{argmin}} \|\mathbf{p}_i - \mathbf{w}_j\| \tag{5.14}$$
   (b) Déterminer le deuxième plus proche voisin de l'ensemble des prototypes :
   $$u = \underset{\substack{j=1,\ldots,C \\ j \neq t}}{\text{argmin}} \|\mathbf{p}_i - \mathbf{w}_j\| \tag{5.15}$$
   (c) **Si** $d_t^{proto} \neq d_u^{proto}$, **alors** :
   $$f_i'(1) = \begin{cases} \dfrac{\|\mathbf{p}_i - \mathbf{w}_u\|}{\|\mathbf{p}_i - \mathbf{w}_t\| + \|\mathbf{p}_i - \mathbf{w}_u\|} & : \quad d_t^{proto} = 1 \\ \dfrac{\|\mathbf{p}_i - \mathbf{w}_t\|}{\|\mathbf{p}_i - \mathbf{w}_t\| + \|\mathbf{p}_i - \mathbf{w}_u\|} & : \quad d_t^{proto} = 2 \end{cases} \tag{5.16}$$
   sinon :
   $$f_i'(1) = \begin{cases} 1 & : \quad d_t^{proto} = 1 \\ 0 & : \quad \text{autrement} \end{cases} \tag{5.17}$$
   (d) Calculer le degré d'appartenance du deuxième nœud fils comme le complément du degré du premier fils :
   $$f_i'(2) = 1 - f_i'(1) \tag{5.18}$$
2. **Retourner** $\mathbf{F}' = [\mathbf{f}'(1)\, \mathbf{f}'(2)]$ comme matrice des degrés d'appartenance des nœuds fils.

FIG. 5.8 – Algorithme de classement des unités PPV.

Soit la fonction $\mathbf{F}' = \text{entrainer}_{\text{PPV}}(\mathbf{P}, \mathbf{d}, \mathbf{f}, n)$, avec les variables :
- n : nœud courant ;
- C : nombre de prototypes du nœud n ;
- C^{max} : nombre maximal de prototypes utilisables (fixé à 6) ;
- $\mathbf{W} = [\mathbf{w}_1 \ldots \mathbf{w}_{C^{max}}]$: ensemble des prototypes du nœud n ;
- $\mathbf{d}^{proto} = [d_1^{proto} \ldots d_{C^{max}}^{proto}]^T$: étiquettes de classes dichotomisées de l'ensemble des prototypes $\mathbf{W}$ du nœud n ;
- $\mathbf{u}(i)$: variables temporaires contenant l'ensemble des indices des données de $\mathbf{P}$ ayant i comme étiquette dichotomisée, avec $i = 1, 2$.

1. Appeler la fonction $\mathbf{d}' = \text{dichotomiser}(\mathbf{P}, \mathbf{d}, \mathbf{f})$ (voir figure 5.5) ;
2. Trier les données en deux ensembles selon les étiquettes dichotomisées :
$$\mathbf{u}(1) = \{i \in [1, \ldots, Q] \,|\, \mathbf{d}'_i = 1\}, \quad \mathbf{u}(2) = \{i \in [1, \ldots, Q] \,|\, \mathbf{d}'_i = 2\} \tag{5.19}$$
3. Sélectionner comme premier prototype de $\mathbf{W}$ la donnée de $\mathbf{u}(1)$ ayant un degré d'appartenance maximal :
$$t = \underset{i \in \mathbf{u}(1)}{\text{argmax}}(f_i), \quad \mathbf{u}(1) = \mathbf{u}(1) - \{t\}, \quad C = 1, \quad \mathbf{w}_C = \mathbf{p}_t, \quad d_C^{proto} = 1 \tag{5.20}$$
4. **Pour tous les** $i = 2, \ldots, Q$, **tant que** $C < C^{max}$ et $(|\mathbf{u}(1)| + |\mathbf{u}(2)|) > 0$:
   (a) **Si** i est impair et $|\mathbf{u}(1)| > 0$, **alors** $g = 1$, **sinon** $g = 2$;
   (b) Obtenir l'indice de la donnée de degré d'appartenance maximal et en déterminer le plus proche voisin de l'ensemble des prototypes :
$$t = \underset{j \in \mathbf{u}(g)}{\text{argmax}}(f_j), \quad h = \underset{j=1,\ldots,C}{\text{argmin}} \|\mathbf{p}_t - \mathbf{w}_j\| \tag{5.21}$$
   (c) **Si** $d_h^{proto} \neq g$, **alors** : $C = C + 1, \quad \mathbf{w}_C = \mathbf{p}_t, \quad d_C^{proto} = g$;
   (d) Retirer l'indice t de $\mathbf{u}(g)$: $\mathbf{u}(g) = \mathbf{u}(g) - \{t\}$.
5. Appeler la fonction $\mathbf{F}' = \text{classer}_{\text{PPV}}(\mathbf{P}, \mathbf{f}, n)$ pour classer les données ;
6. **Retourner** $\mathbf{F}'$ comme matrice des degrés d'appartenance des nœuds fils.

FIG. 5.9 – Algorithme d'entraînement des unités PPV.

Soit la fonction $\mathbf{F}' = \text{classer}_{\text{FKM}}(\mathbf{P}, \mathbf{f}, n)$, avec les variables :
- n : nœud courant ;
- K : nombre de centres utilisés dans l'unité de classement ;
- $\mathbf{W} = [\mathbf{w}_1 \ \ldots \ \mathbf{w}_K]$: ensemble des centres du nœud n ;
- ν : niveau de partage des degrés d'appartenance (fixé à 2).

1. Calculer la matrice $\mathbf{U}$ des degrés d'appartenance des nœuds fils :

$$u_{i,j} = \frac{1}{\sum_{k=1}^{K} \left(\frac{\|\mathbf{p}_j - \mathbf{w}_i\|}{\|\mathbf{p}_j - \mathbf{w}_k\|} \right)^{\frac{2}{\nu-1}}}, \quad i = 1, \ldots, K, \quad j = 1, \ldots, Q \tag{5.22}$$

2. **Retourner** $\mathbf{F}' = \mathbf{U}$ comme matrice des degrés d'appartenance des nœuds fils.

FIG. 5.10 – Algorithme de classement des unités à K-moyennes floues.

l'algorithme d'entraînement est le même que l'algorithme original présenté dans [42], excepté pour le calcul de la position des nouveaux centres (équation 5.24), où les degrés d'appartenance des nœuds $\mathbf{f}$ sont intégrés à l'équation originale. Les paramètres de classement et d'entraînement utilisés durant toutes les expérimentations du chapitre sont : un niveau de partage $\nu = 2$, un arrêt de l'entraînement avec une erreur de partitionnement inférieure à $\varepsilon = 0.05$ et un nombre maximal de $t^{max} = 30$ itérations. De plus, la PG est configurée afin de pouvoir utiliser trois variétés d'unités de classement à K-moyennes floues, soit des unités avec $K = 2$, $K = 3$ et $K = 4$ centres.

5.3.5 Autres unités

Le système EFFECT possède trois unités supplémentaires de classement. Ces unités ne possèdent pas de paramètre modifié par l'entraînement. La première unité effectue une duplication en « Y » du traitement dans l'arbre. Pendant la passe descendante, les degrés d'appartenance flous du nœud actuel sont directement donnés aux deux nœuds fils. Les fonctions de classement $\mathbf{F}' = \text{classer}_{\text{Y}}(\mathbf{P}, \mathbf{f}, n)$ et d'entraînement $\mathbf{F}' = \text{entrainer}_{\text{Y}}(\mathbf{P}, \mathbf{d}, \mathbf{f}, n)$ de l'unité de duplication sont donc définies très simplement en retournant $\mathbf{F}' = [\mathbf{f}\,\mathbf{f}]$ comme matrice des degrés d'appartenance.

Les deux autres unités de classement appliquent une transformation des degrés d'appartenance flous. Cette transformation se fait selon une fonction monotone croissante unique à chaque unité. Ces fonctions sont paramétrées selon une valeur α générée aléa-

Soit la fonction $\mathbf{F}' = \text{entrainer}_{\text{FKM}}(\mathbf{P}, \mathbf{d}, \mathbf{f}, n)$, avec les variables :
- n : nœud courant ;
- K : nombre de centres utilisés dans l'unité de classement ;
- $\mathbf{W} = [\mathbf{w}_1 \ \ldots \ \mathbf{w}_K]$: ensemble des centres du nœud n ;
- ν : niveau de partage des degrés d'appartenance (fixé à 2) ;
- ε : critère d'arrêt sur l'erreur de partitionnement (fixé à 0.05) ;
- t^{max} : critère d'arrêt sur le nombre maximal d'itérations (fixé à 30).

1. Initialiser les centres $\mathbf{W}$ en choisissant les K données de degré d'appartenance maximal (bris d'égalité par choix aléatoire) :

$$\hat{\mathbf{f}} = \mathbf{f} \quad \text{suivie de} \quad h = \underset{j=1,\ldots,Q}{\text{argmax}}(\hat{f}_j), \quad \hat{f}_h = -1, \quad \mathbf{w}_i = \mathbf{p}_h, \quad i = 1, \ldots, K \tag{5.23}$$

2. Calculer la partition initiale $\mathbf{U}(0)$ avec l'équation 5.22 ;
3. Initialiser $t = 1$ et $E(0) = \infty$;
4. **Tant que** $t \leq t^{max}$ et $E(t-1) \geq \varepsilon$:
   (a) Calculer les nouveaux centres $\mathbf{W}$:

$$\mathbf{w}_i = \frac{\sum_{j=1}^{Q} (u_{i,j})^{\nu} f_j \mathbf{p}_j}{\sum_{j=1}^{Q} (u_{i,j})^{\nu} f_j}, \quad i = 1, \ldots, K \tag{5.24}$$

   (b) Calculer la nouvelle partition floue $\mathbf{U}(t)$ avec l'équation 5.22 ;
   (c) Calculer l'erreur de partitionnement :

$$E(t) = \max_{\substack{i=1,\ldots,K \\ j=1,\ldots,Q}} |u_{i,j}(t) - u_{i,j}(t-1)| \tag{5.25}$$

   (d) $t = t + 1$.
5. **Retourner** $\mathbf{F}' = \mathbf{U}(t-1)$ comme matrice des degrés d'appartenance.

FIG. 5.11 – Algorithme d'entraînement des unités à K-moyennes floues.

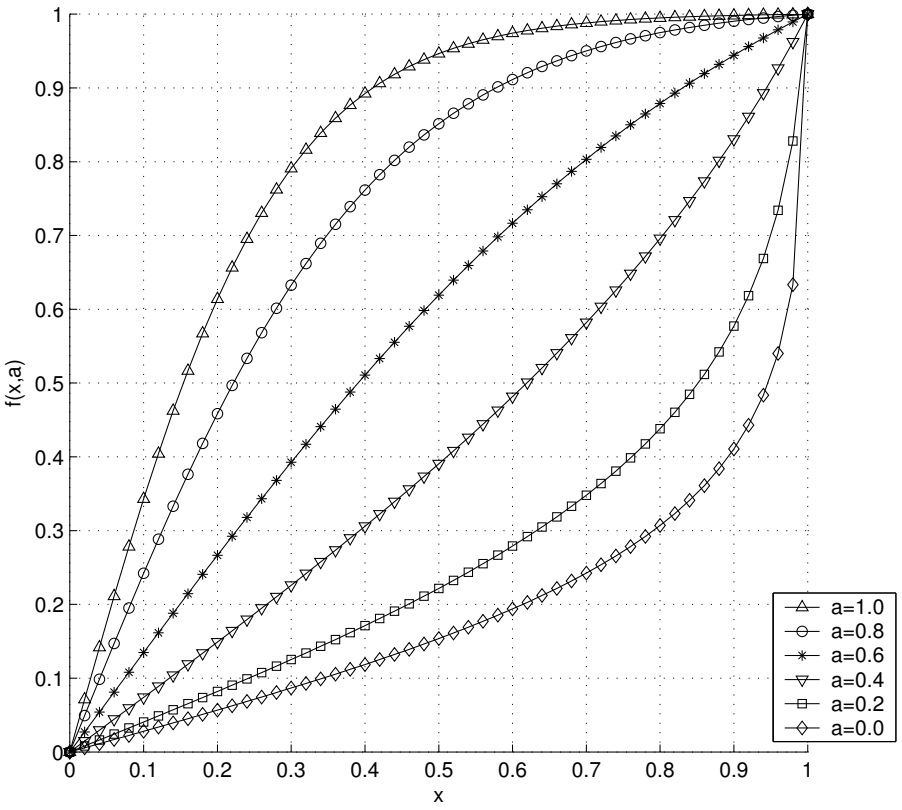


FIG. 5.12 – Tracé de la fonction d'amplification des degrés d'appartenance flous $\mathrm{f}_{amp}(x, \alpha)$ de l'équation 5.26, avec $x \in [0.0, 1.0]$ et différentes valeurs du paramètre $\alpha = \{0.0, 0.2, 0.4, 0.6, 0.8, 1.0\}$.

toirement entre 0 et 1 lors de la création ou de la mutation des arbres. La deuxième unité consiste en une fonction qui « amplifie » les degrés d'appartenance flous des données. La fonction $\mathrm{f}_{amp}(x, \alpha)$ de cette unité est définie par l'équation suivante.

$$\begin{aligned} \beta &= 12|\alpha - 0.5|^{0.75}, \\ \gamma &= 2\frac{1+\mathrm{e}^{-\beta}}{1-\mathrm{e}^{-\beta}}, \\ \mathrm{f}_{amp}(x,\alpha) &= \begin{cases} \gamma\left(\dfrac{1}{1+\mathrm{e}^{-\beta x}} - 0.5\right) & : \quad \alpha > 0.5 \\ \dfrac{-1}{\beta}\ln\left(\dfrac{0.5\gamma - x}{x + 0.5\gamma}\right) & : \quad \alpha \leq 0.5 \end{cases} \end{aligned} \tag{5.26}$$

La fonction de classement $\mathbf{F}' = \mathrm{classer}_{\mathrm{AMP}}(\mathbf{P}, \mathbf{f}, n)$ et la fonction d'entraînement $\mathbf{F}' = \mathrm{entrainer}_{\mathrm{AMP}}(\mathbf{P}, \mathbf{d}, \mathbf{f}, n)$ transforment les degrés d'appartenance au nœud actuel en retournant $\mathbf{F}' = \mathrm{f}_{amp}(\mathbf{f}, \alpha)$ comme vecteur des degrés d'appartenance du nœud fils. Un tracé de la fonction d'amplification $\mathrm{f}_{amp}(x, \alpha)$ avec différentes valeurs de α est présenté à la figure 5.12.

La troisième unité de classement consiste en une fonction de « clarification » des degrés d'appartenance flous des données. La formule mathématique de $\mathrm{f}_{sha}(x, \alpha)$ est présentée à l'équation suivante.

$$\begin{aligned}
\beta &= 24|\alpha - 0.5|^{0.75}, \\
\gamma &= \frac{(1 + \mathrm{e}^{0.5\beta})(1 + \mathrm{e}^{-0.5\beta})}{\mathrm{e}^{0.5\beta} - \mathrm{e}^{-0.5\beta}}, \\
\lambda &= \frac{1}{1 + \mathrm{e}^{0.5\beta}}, \\
\mathrm{f}_{sha}(x, \alpha) &= \begin{cases} \gamma \left(\frac{1}{1 + \mathrm{e}^{\beta(0.5 - x)}} - \lambda \right) & : \quad \alpha > 0.5 \\ \frac{-1}{\beta} \ln \left(\frac{\gamma - x - \gamma\lambda}{x + \gamma\lambda} \right) + 0.5 & : \quad \alpha \leq 0.5 \end{cases}
\end{aligned} \tag{5.27}$$

La fonction de classement $\mathbf{F} = \mathrm{classer}_{\mathrm{SHA}}(\mathbf{P}, \mathbf{f}, n)$ et la fonction d'entraînement $\mathbf{F} = \mathrm{entrainer}_{\mathrm{SHA}}(\mathbf{P}, \mathbf{d}, \mathbf{f}, n)$ sont définies en clarifiant les degrés d'appartenance actuels par le retour de $\mathbf{F} = \mathrm{f}_{sha}(\mathbf{f}, \alpha)$ comme vecteur des degrés d'appartenance du nœud fils. Le tracé de la fonction $\mathrm{f}_{sha}(x, \alpha)$ pour quelques valeurs de α est présentée à la figure 5.13.

5.3.6 Évolution de topologies d'arbres

Les topologies d'arbres de décision sont évoluées par PG. À cette fin, les différentes unités de classement du système forment l'ensemble des primitives de PG utilisables, tel que présenté au tableau 5.1. En plus des unités de classement, l'ensemble des primitives comprend une primitive T agissant comme terminal des arbres de décision. Cette primitive comporte le vecteur $\mathbf{m}$ des scores de classement afin de convertir les degrés d'appartenance flous des données en vecteurs de degrés d'appartenance de classement (voir les figures 5.2 et 5.4). Le paramètre α des primitives des fonctions d'amplification et de clarification est généré aléatoirement lors de l'initialisation et de la mutation des arbres, de façon similaire aux constantes éphémères générées aléatoirement [76]. De plus, les valeurs de la colonne « Complexité » donnent la complexité des différentes unités utilisées. Ces valeurs sont utilisées pour estimer la complexité totale des arbres de décision. Elles ont été choisies arbitrairement, en tenant compte du nombre de paramètres numériques employés par les unités et de la complexité relative des différents algorithmes d'apprentissage.

Une mesure d'adéquation à deux objectifs est utilisée pour guider la recherche darwinienne. Le premier objectif consiste à maximiser les marges floues moyennes de classement. Conceptuellement, cette approche va dans le même sens que les techniques modernes de classement basées sur la maximisation de la marge géométrique, en maximisant cette fois la marge entre les degrés d'affectation $\mathbf{s}_i$ des données aux différentes

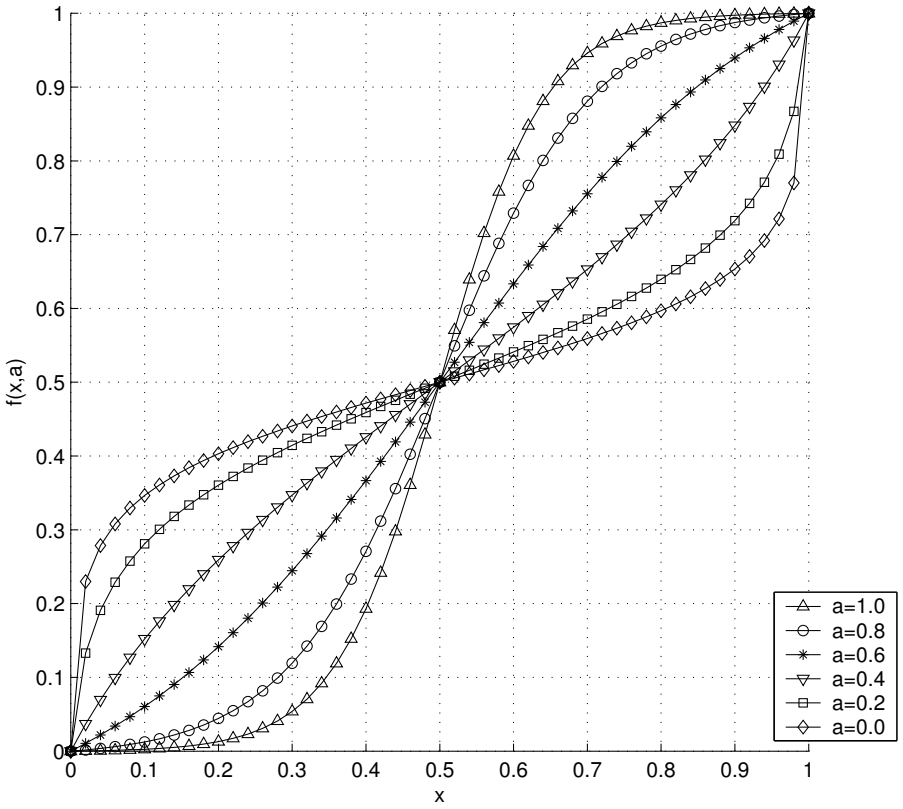


FIG. 5.13 – Tracé de la fonction de clarification des degrés d'appartenance flous $\mathrm{f}_{sha}(x, \alpha)$ de l'équation 5.27, avec $x \in [0.0, 1.0]$ et différentes valeurs du paramètre $\alpha = \{0.0, 0.2, 0.4, 0.6, 0.8, 1.0\}$.

Nom	Args	Complexité	Description
ADA	2	0.1	Classifieur linéaire neuronal de type ADALINE.
PPV	2	$0.05C$	Classifieur non paramétrique de type plus proche voisin. La variable C de la mesure de complexité correspond à la taille de l'ensemble des prototypes retenus.
F2M	2	0.1	Classificateur à K-moyennes floues, avec $K = 2$.
F3M	3	0.15	Classificateur à K-moyennes floues, avec $K = 3$.
F4M	4	0.2	Classificateur à K-moyennes floues, avec $K = 4$.
Y	2	0.01	Duplication des degrés d'appartenance flous.
AMP	1	0.02	Fonction d'amplification des degrés d'appartenance flous $f_{amp}(x, \alpha)$, avec le paramètre α généré aléatoirement.
SHA	1	0.02	Fonction de clarification des degrés d'appartenance flous $f_{sha}(x, \alpha)$, avec le paramètre α généré aléatoirement.
T	0	0.01	Terminal des arbres de décision.

TAB. 5.1 – Ensemble des primitives de PG du système EFFECT.

classes. Les équations suivantes présentent le calcul de l'adéquation liée à cet objectif.

$$t(i) = \underset{j=1,\ldots,L}{\operatorname{argmax}}(s_{i,j}), \tag{5.28}$$

$$u(i) = \underset{\substack{j=1,\ldots,L \\ j \neq t(i)}}{\operatorname{argmax}}(s_{i,j}), \tag{5.29}$$

$$v(i) = j \quad \text{tel que} \quad l_j = d_i, \tag{5.30}$$

$$w(i) = \begin{cases} s_{i,t(i)} - s_{i,u(i)} & : \quad t(i) = v(i) \\ s_{i,v(i)} - s_{i,t(i)} & : \quad \text{autrement} \end{cases} \tag{5.31}$$

$$\mathit{fit}(1) = \sqrt{\sum_{i=1}^{Q} (1 - w(i))^2} \tag{5.32}$$

Aux équations 5.28 et 5.29, on détermine les classes ayant les deux valeurs de degrés d'appartenance maximales pour la donnée i. À l'équation 5.30, on obtient le véritable indice de l'étiquette de classe de la donnée i. Ensuite, on calcule à l'équation 5.31 la marge entre les degrés d'appartenance de classement. Si la donnée i est bien classée, la marge est calculée comme l'écart entre les deux degrés d'appartenance aux classes maximaux. Si la donnée est mal classée, la marge est calculée comme l'écart entre le degré d'appartenance de la véritable classe et le degré d'appartenance de classement maximum. Finalement, l'équation 5.32 présente le calcul de l'adéquation liée à l'objectif en utilisant le complément de la marge floue entre les degrés d'appartenance aux classes. Cette mesure de qualité vise donc à bien classer les données et à maximiser l'écart entre les deux degrés d'appartenance aux classes maximums pour les données bien classées, afin de clarifier le plus possible la prise de décision.

Le deuxième objectif consiste à minimiser la complexité des arbres de décision. Ceci va dans le sens du principe du rasoir d'Occam [39], où, à qualité égale, les solutions simples sont préférées aux solutions complexes. Cette complexité est estimée par la somme des valeurs de complexité associées à chacun des nœuds de l'arbre. Les valeurs de complexité des différentes unités de classement sont présentées au tableau 5.1. Il est à noter que les valeurs de complexité des différentes unités ont été établies de sorte que la valeur de la complexité totale des arbres soit du même ordre de grandeur que la valeur des marges floues de classement.

5.4 Résultats expérimentaux

Le système EFFECT décrit à la section précédente a été testé sur sept ensembles de données différents. Ces ensembles de données, détaillés au tableau 5.2, sont tirés du

TAB. 5.2 – Description des ensembles de données utilisés pour évaluer et comparer la performance du système EFFECT.

Ensemble de données	Taille	# attr.	# classes	Domaine d'application
Wisconsin breast cancer (bcw)	699	10	2	Identification de la sévérité (bénigne ou maligne) de tumeurs au sein.
BUPA liver disorders (bld)	345	6	2	Détection de désordres au foie.
Boston housing (bos)	506	13	3	Évaluation de la valeur d'habitations de quartiers de la région de Boston. Les données sont étiquetées en trois classes, selon la valeur médiane v (en 1000 \$) des maisons (classe 1 : $v \leq 18.77$, classe 2 : $v \in]18.77, 23.74]$, classe 3 : $v > 23.74$).
Contraceptive method choice (cmc)	1473	9	3	Prédiction du choix de la mesure contraceptive à partir de caractéristiques démographiques et socio-économiques.
Pima indians diabetes (pid)	768	8	2	Diagnostique du diabète chez des femmes de tribus amérindiennes Pima à partir de mesure médicales générales.
Image segmentation (seg)	2310	19	7	Détermination de l'image d'appartenance de régions de 3×3 pixels.
Teaching assistant evaluation (tae)	151	5	3	Prédiction de la performance d'enseignement de chargés de cours.

TAB. 5.3 – Performances de différents classifieurs standards.

Ensemble de données	1-PPV		PMC		Lim *et al.*	
	Taux moy. classement	Écart-type	Taux moy. classement	Écart-type	Taux max. classement	Écart estimé
bcw	95.9 %	2.7 %	96.3 %	1.8 %	97 %	0.7 %
bld	62.7 %	6.0 %	42.0 %	0.8 %	72 %	2.5 %
bos	77.3 %	5.4 %	55.7 %	3.7 %	78 %	1.9 %
cmc	43.3 %	3.6 %	42.7 %	0.2 %	57 %	1.4 %
pid	70.7 %	5.0 %	75.4 %	2.6 %	78 %	1.6 %
seg	97.3 %	0.8 %	14.3 %	0.0 %	98 %	0.3 %
tae	63.4 %	12.0 %	31.8 %	2.4 %	77 %	4.0 %

machine learning repository de l'Université de Californie à Irvine (UCI) [13]. Une validation croisée à 10 recoupements (10-*folds cross-validation*) est effectuée pour générer les résultats du chapitre. À cette fin, chaque ensemble de données est partitionné aléatoirement en dix sous-ensembles disjoints de taille égale. Chacun de ces sous-ensembles forme alors un ensemble de test auquel est associé un ensemble d'entraînement formé des neuf sous-ensembles restants. Chaque algorithme de classement est évalué une ou plusieurs fois sur chacun de ces dix recoupements (paires d'ensembles d'entraînement et de test). Les taux de classement donnés dans le chapitre sont donc des moyennes de la performance des algorithmes entraînés et testés sur les dix recoupements de chaque ensemble de données.

Le tableau 5.3 présente la performance de différents classifieurs standards sur les ensembles de données du tableau 5.2. La colonne « 1-PPV » donne sur les résultats avec un classifieur PPV standard, en utilisant une distance euclidienne et une normalisation d'échelle dans $[-1, 1]$. Les données d'entraînement de chaque recoupement sont utilisées comme prototypes, alors que les données tests sont utilisées pour l'évaluation des taux de classement.

La colonne « PMC » donne les résultats du classement avec un réseau de neurones de type perceptron multicouche (PMC), entraîné avec l'algorithme de rétro-propagation des erreurs avec momentum [54]. Pour chaque recoupement, un réseau comprenant une couche cachée de 20 neurones est entraîné avec un taux d'apprentissage de 0.2 et un paramètre de momentum de 0.35. Pour chaque entraînement, l'ensemble d'entraînement de chaque recoupement est re-partitionné aléatoirement en deux ensembles, soit un sous-ensemble pour modifier les poids du réseau (67 % des données) et un sous-ensemble de validation pour interrompre l'entraînement et sélectionner le réseau qui

généralise le mieux (33 % des données). L'entraînement d'un réseau dure un maximum de 125 itérations, avec une interruption possible au cours des 90 dernières itérations si aucun gain sur le taux de classement des données de validation n'est observé pendant 15 itérations consécutives. Étant donné la nature stochastique de l'algorithme d'apprentissage du PMC, les résultats du tableau 5.3 proviennent de 10 répétitions pour chaque recoupement, pour un total de 100 entraînements par ensemble de données. Ici aussi, une normalisation d'échelle dans $[-1, 1]$ est appliquée aux données.

La colonne « Lim *et al.* » présente les meilleurs résultats rapportés dans [90] pour 33 algorithmes d'apprentissage différents. Ces 33 algorithmes comprennent 22 algorithmes à base d'arbre de décision ou de règles, 9 algorithmes statistiques et 2 algorithmes de type réseaux de neurones. L'écart estimé correspond au calcul $\sqrt{p(1-p)/n}$, où p est l'erreur de classement observée et n est la taille de l'ensemble d'entraînement. Il est à noter que seuls les taux de classement maximal et minimal sont donnés par Lim *et al.* L'utilisation du meilleur taux de classement des 33 algorithmes testés introduit un biais important dans les résultats. En effet, on surestime les taux de classement en utilisant les ensembles de test pour sélectionner le taux de l'algorithme de classement le plus performant. À cet effet, les taux moyens seraient certainement plus appropriés pour une comparaison équitable des performances. Malheureusement, ces taux ne sont pas donnés dans l'article. Également, les résultats présentés par Lim *et al.* comportent seulement deux chiffres significatifs, alors qu'un chiffre significatif supplémentaire aurait permis une comparaison plus fine des résultats.

En analysant brièvement les résultats du tableau 5.3, on remarque que le réseau de neurones PMC obtient des taux de classement moyens très faibles pour les ensemble de données seg et tae, et plutôt faibles pour les ensembles bld et bos. Les taux de classement moyens des classifieurs PPV et PMC sont comparables pour les ensembles de données bcw et cmc, alors que le taux du PMC pour l'ensemble pid est significativement meilleur que celui du classifieur PPV. Si l'on compare les résultats du classifieur PPV avec ceux de Lim *et al.*, on remarque un taux de classement en deçà de l'écart estimé du taux maximum pour l'ensemble bos, et très près de l'écart estimé pour les ensembles bcw et seg. Pour les autres ensembles de données, les taux du classifieur PPV sont significativement inférieurs aux taux présentés de Lim *et al.*, sans toutefois démontrer de faiblesse aussi apparente que le classifieur PMC.

5.4.1 Résultats avec le système EFFECT

La performance du système EFFECT a été testée avec les sept ensembles de données UCI présentés au tableau 5.2. Les résultats sont donnés comme des moyennes calculées

sur 100 évolutions pour chaque ensemble de données, soit 10 évolutions par recoupement. Les données ont été préalablement normalisées pour avoir des valeurs comprises dans l'intervalle $[-1, 1]$. Pour chaque évolution, l'ensemble de données d'entraînement est partitionné aléatoirement en deux sous-ensembles disjoints de taille égale. Le premier sous-ensemble, nommé ensemble d'évaluation de l'adéquation, sert à déterminer les paramètres des unités de classement par un entraînement et à calculer l'adéquation associée au premier objectif. Le deuxième sous-ensemble de données, nommé ensemble de validation, sert à déterminer le meilleur individu de l'évolution, en sélectionnant celui ayant un taux de classement maximal sur ce sous-ensemble. Cette approche est identique à celle utilisée au chapitre précédent pour sélectionner les meilleures représentations de caractères de chaque évolution. En utilisant des données non vues durant l'apprentissage évolutionnaire, on espère sélectionner un arbre de décision qui aura une bonne capacité de généralisation.

Chaque évolution est faite avec une population de 1000 individus sur 200 générations, en utilisant l'algorithme de sélection multi-objectif NSGA-II [37]. Les arbres sont initialisés en utilisant l'algorithme standard *ramped half-and-half* [76], avec une profondeur initiale qui se situe entre 2 et 5 niveaux et une profondeur maximale d'arbre de 17 niveaux. Le croisement est appliqué avec une probabilité de 0.85 et les différentes mutations (standard, réductrice et par inversion de nœuds) sont appliquées avec une probabilité de 0.05 chacune. Comme pour les applications des deux chapitres précédents, les valeurs des différents paramètres de PG utilisées sont des valeurs par défaut, excepté pour la taille des populations et le nombre de générations. Au cours des différentes expérimentations préliminaires, on a observé qu'après 100 générations, la convergence de la PG ne semblait pas complète, d'où le prolongement de la durée des évolutions sur 200 générations. Il serait certainement intéressant de faire une étude afin de déterminer avec plus de précision le nombre de générations optimal permettant d'obtenir des résultats intéressants en un minimum de générations. Mais ici aussi, l'efficacité computationnelle n'est pas un objectif très important de l'application et aucun effort particulier n'a été fait de ce côté.

Le tableau 5.4 présente les performances du système EFFECT sur les sept ensembles de données UCI choisis. En comparant les résultats à ceux du tableau 5.3, on remarque que le système EFFECT est capable de générer de meilleurs taux de classement moyens que le classifieur PMC pour tous les ensembles de données. Par rapport au classifieur PPV, on obtient pour le système EFFECT des taux de classement moyens significativement supérieurs pour les ensembles bld (+5.9 %), cmc (+7.9 %) et pid (+5.5 %), légèrement supérieur pour l'ensemble bcw (+0.6 %), légèrement inférieurs pour les ensembles bos (−1.4 %) et seg (−2.6 %), et significativement inférieur pour l'ensemble tae (−11.4 %). On explique le dernier cas par la petite taille de l'ensemble tae, qui

TAB. 5.4 – Performances du système EFFECT.

Ensemble de données	Taux de classement		Complexité des arbres	
	Moyenne	Écart-type	Moyenne	Écart-type
bcw	96.5 %	1.6 %	1.4	1.5
bld	68.6 %	6.8 %	3.2	3.5
bos	75.9 %	5.1 %	2.7	2.2
cmc	51.2 %	3.3 %	4.2	3.2
pid	76.2 %	4.0 %	2.3	2.1
seg	94.7 %	1.4 %	4.9	2.2
tae	52.0 %	11.2 %	5.1	3.8

possède 151 données au total, alors que la moitié des données sont utilisées dans le système EFFECT pour la sélection du meilleur individu de chaque évolution. Dans ces circonstances, il aurait été probablement préférable d'équilibrer différemment la taille des sous-ensembles de données utilisés pour les évolutions. L'écart-type important observé sur les taux de classement moyen est un indicateur supplémentaire qui valide cette hypothèse. D'autre part, si l'on compare le système EFFECT avec les résultats rapportés dans Lim *et al.*, on constate que pour l'ensemble de données bcw, le taux de classement se situe à l'intérieur d'un écart estimé du maximum, alors que les résultats pour les ensembles bos et pid sont très près de l'écart estimé. Les résultats pour l'ensemble cmc sont toujours significativement inférieurs à ceux rapportés par Lim *et al.*, quoiqu'ils se comparent avantageusement à ceux obtenus avec les classifieurs PPV et PMC. Finalement, les résultats pour l'ensemble tae sont significativement plus faibles que ceux de Lim *et al.*

On constate donc que le système EFFECT fournit des résultats qui, dans la plupart des cas, se comparent avantageusement aux résultats obtenus avec les classifieurs PMC et PPV. D'autre part, excepté pour l'ensemble tae, on observe que le système donne des résultats comparables ou légèrement inférieurs aux meilleurs résultats des 33 algorithmes rapportés par Lim *et al.* La contre-performance pour l'ensemble tae est probablement causée par la méthode de sélection des meilleurs solutions du système, qui n'est pas adaptée aux ensembles de petite taille. Un autre désavantage non négligeable du système EFFECT est le temps de calcul nécessaire à une évolution. En effet, selon la nature et la taille des ensembles de données, une évolution peut durer de quelques minutes à quelques heures sur un ordinateur personnel moderne, ce qui est relativement long comparativement au temps de calcul typique des autres algorithmes de classement.

À la figure 5.14, on présente les fronts de Pareto pour quatre évolutions des en-

sembles de données bld, pid, seg et tae. Comme à la figure 4.7 du chapitre précédent, chaque graphique comporte deux courbes qui présentent les performances sur les ensembles de données d'évaluation de l'adéquation et de validation. Dans le cas présent, la première courbe (cercles avec traits discontinus) présente le front de Pareto de la dernière génération, avec, en abscisse, la complexité totale des arbres de décision et, en ordonnée, l'erreur quadratique moyenne sur les marges floues calculées selon l'équation 5.32. La deuxième courbe (étoiles avec traits continus) présente en ordonnée l'erreur de classement sur les ensembles de données de validation correspondant aux solutions du front de Pareto de la première courbe. Il est important de noter que la nature des ordonnées pour les deux courbes de chaque graphique (marges floues et taux d'erreur de classement) diffèrent totalement, même si l'échelle utilisée dans les graphiques est la même et que plus une valeur est faible, plus la performance est bonne dans les deux cas. Pour chacun des quatre ensembles de données présentés à la figure 5.14, il a été choisi arbitrairement de présenter le front de Pareto de l'évolution, parmi les cent évolutions effectuées pour chaque ensemble de données, dont le meilleur individu performe le mieux sur les données de l'ensemble de test.

On remarque à la figure 5.14 que les taux d'erreur de classement sur les ensembles de validation des données bld et pid semblent constants, quoique bruités, pour toutes les valeurs de complexité. Pour les graphiques sur les ensembles de données seg et tae, les taux d'erreur de classement sur les ensembles de validation semblent comporter une certaine corrélation avec les valeurs de marges floues, même si pour des solutions de complexité élevée, les taux d'erreurs semblent équivalents pour les différentes solutions du front de Pareto. Finalement, il est important de rappeler que le meilleur individu de l'évolution correspond à l'individu ayant un taux d'erreur de classement minimum sur l'ensemble de validation. Cette méthode de sélection fait donc en sorte que le meilleur individu de l'évolution n'est pas nécessairement présent sur le front de Pareto de la dernière génération.

La figure 5.15 présente les performances de l'évolution d'arbres de décision flous avec différentes tailles de l'ensemble de validation. La taille de l'ensemble de validation en abscisse est présentée sous la forme du ratio entre la taille de l'ensemble de validation et la taille totale de l'ensemble de données d'entraînement. Trois courbes sont présentées sur la figure, soit la courbe des marges floues de classement calculée selon l'équation 5.32 (cercles avec traits discontinus), la courbe du taux d'erreur de classement sur l'ensemble de validation (étoiles avec traits continus) et la courbe du taux d'erreur de classement sur l'ensemble de test (signes + avec traits pointillés). Pour chaque position sur l'abscisse du graphique, les trois points en ordonnée représentent les résultats moyens de dix évolutions, avec une évolution par recoupement d'ensembles de données. On remarque que plus la taille de l'ensemble de validation est élevée, plus le taux d'er-

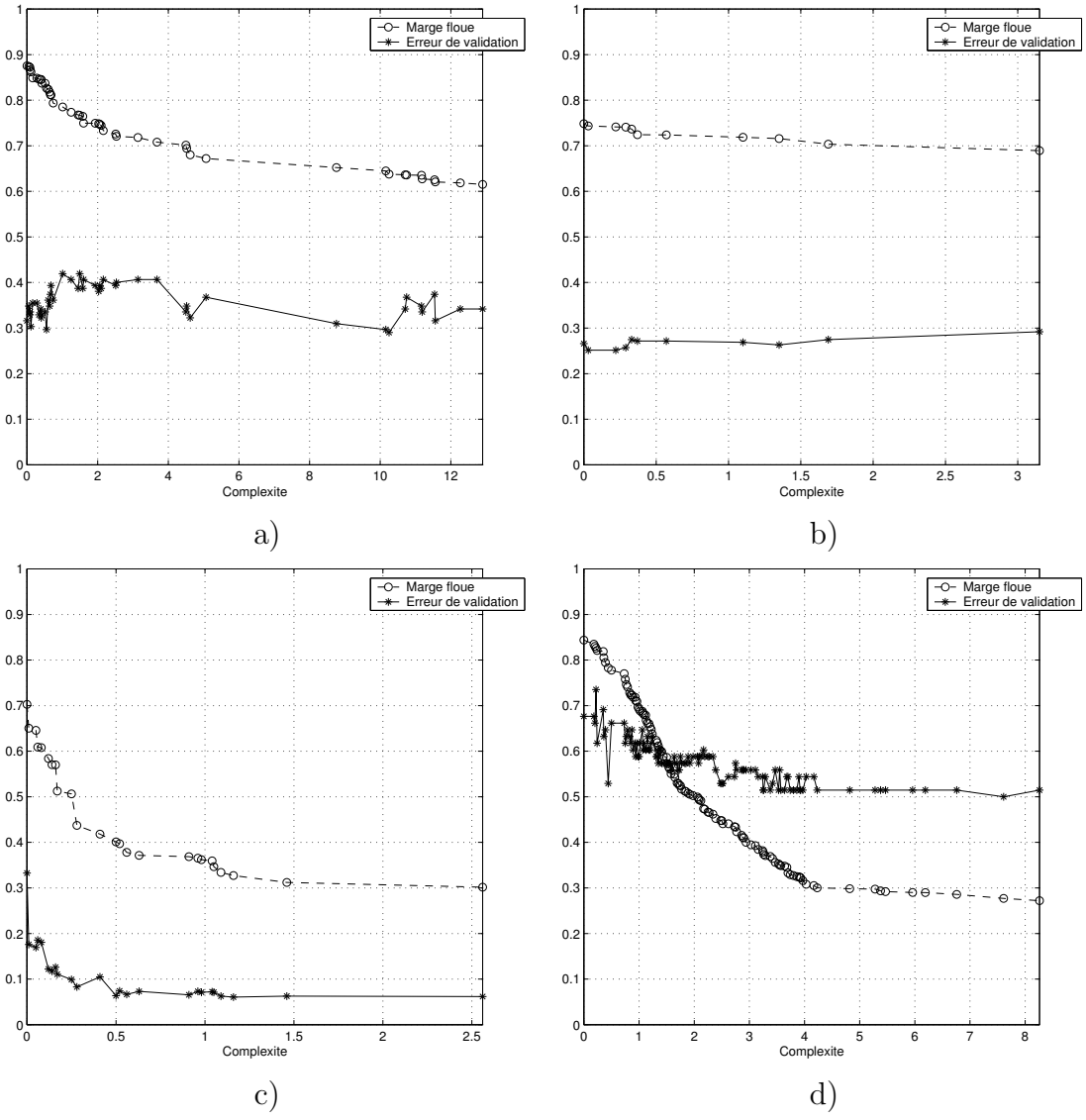


FIG. 5.14 – Fronts de Pareto de quatre évolutions d'arbres de décision flous du système EFFECT : a) bld ; b) pid ; c) seg ; d) tae.

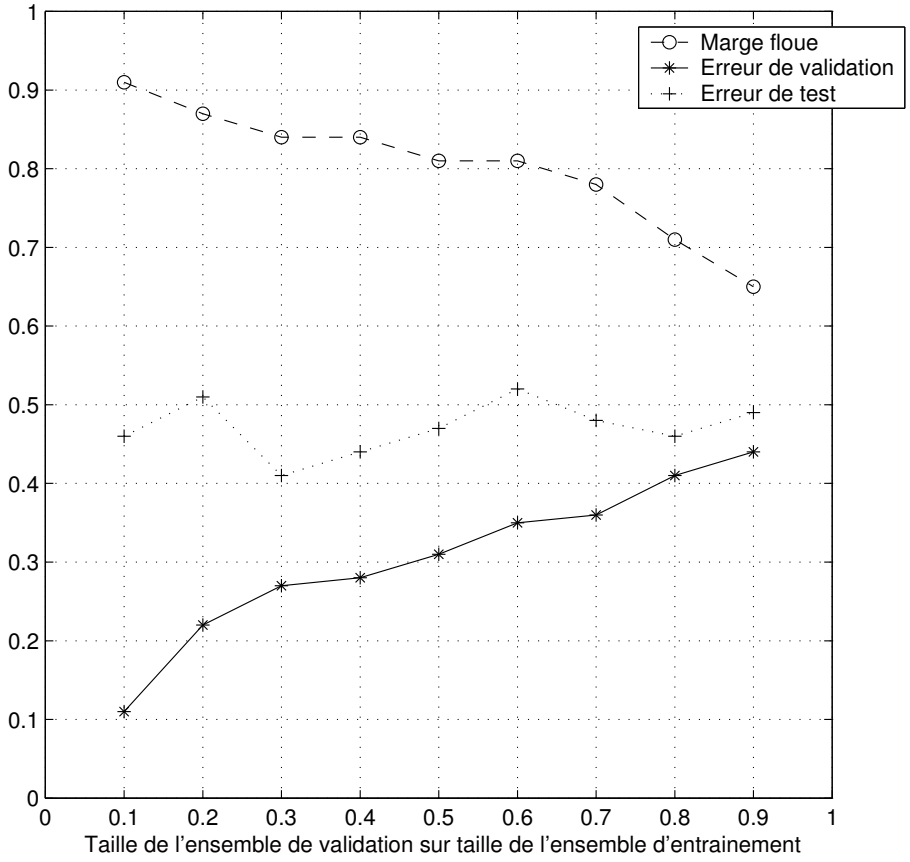


FIG. 5.15 – Performance du système EFFECT sur les données tae avec différentes tailles de l'ensemble de validation.

reur de validation est élevé. On explique ce phénomène par le fait que plus la taille de l'ensemble de validation est petite, plus les chances sont grandes que l'on obtienne un individu en cours d'évolution qui soit particulièrement adapté aux données de validation. À l'inverse, la marge floue décroît proportionnellement avec la taille de l'ensemble de validation, étant donné que l'ensemble d'évaluation de l'adéquation correspond aux données de l'ensemble d'entraînement non sélectionnées pour former l'ensemble de validation. Donc, plus la taille de l'ensemble de validation est petite, plus il y a des données dans l'ensemble d'évaluation de l'adéquation, ce qui limite le surapprentissage de ces données. Finalement, on remarque que le taux d'erreur de classement de l'ensemble des données de test semble plutôt stable, quoique bruité, pour les différentes tailles de l'ensemble de validation. En bref, pour l'ensemble de données tae, la taille de l'ensemble de validation par rapport à la taille totale de l'ensemble des données d'entraînement ne semble pas avoir une influence déterminante sur le taux de classement final sur l'ensemble de données de test du meilleur individu de chaque évolution. Cependant, ces observations ne nous permettent pas de tirer des conclusions générales étant donné que l'ensemble tae comporte seulement 151 données, ce qui est très peu dans le contexte actuel d'apprentissage.

5.5 Gel des paramètres appris

Dans le système EFFECT, on effectue l'évolution de topologies d'arbres de décision par PG jumelée à une modification par apprentissage des paramètres des unités de classement. Avec un tel système, on peut s'attendre à une interaction positive entre les deux mécanismes, afin de générer des topologies qui permettent un apprentissage performant et relativement stable. Cependant, il devrait toujours y avoir des variations entre plusieurs entraînements d'une même topologie. En ce sens, il serait intéressant de pouvoir fixer les paramètres numériques de certaines unités de classement en cours d'évolution, afin de permettre une plus grande stabilité du processus d'apprentissage.

À cet effet, il est proposé de modifier le système EFFECT original afin d'ajouter un bit à chaque unité de classement pour geler les paramètres appris. Lors de la création des arbres de décision, ce bit de gel des paramètres est inactif pour toutes les unités de classement du système. Un opérateur spécial de mutation est défini pour inverser ce bit aléatoirement, selon une certaine probabilité. Tant que ce bit est activé pour une unité de classement, les paramètres appris de cette unité restent inchangés d'un apprentissage à l'autre. Dans le cas particulier des feuilles des arbres, l'activation du bit de gel des paramètres implique que le vecteur des scores de classement $\mathbf{m}$ est gelé.

TAB. 5.5 – Performances du système EFFECT avec gel des paramètres appris.

Ensemble de données	Taux de classement		Complexité des arbres	
	Moyenne	Écart-type	Moyenne	Écart-type
bcw	96.5 %	1.9 %	1.1	1.1
bld	68.2 %	7.5 %	2.2	2.2
bos	74.5 %	5.5 %	2.2	2.0
cmc	50.6 %	3.8 %	2.2	1.9
pid	76.3 %	3.9 %	1.4	1.4
seg	94.5 %	1.3 %	3.0	1.5
tae	51.8 %	11.5 %	3.2	2.6

Cette modification du système EFFECT cadre dans la théorie (biologiquement peu plausible) de l'évolution de Lamarck, par une transmission de caractères acquis aux descendants. En effet, en gelant les paramètres appris des individus, on se trouve à greffer aux génotypes des caractères acquis qui seront transmis intacts aux individus des générations suivantes. L'approche devrait permettre au processus évolutionnaire de fixer dynamiquement certains paramètres. Ceci peut atténuer les effets négatifs liés à l'instabilité de l'apprentissage des paramètres, tout en permettant une configuration des arbres de décision qui seraient difficile à obtenir par apprentissage seul. Donc, on permet au processus évolutionnaire d'insérer une certain rigidité dans la configuration des arbres de décision comparativement au système standard, qui est plus malléable et donc sujet à des variations plus importantes d'un entraînement à l'autre.

5.5.1 Résultats avec gel des paramètres

Le système EFFECT avec gel des paramètres appris a été testé dans les mêmes conditions qu'à la section 5.4.1. Seule l'opération de mutation par inversion de bit de gel des paramètres a été ajoutée à la configuration originale de l'AE. Pour les expérimentations de la présente section, cette opération de mutation est appliquée à tous les individus de toutes les générations avec une probabilité d'inversion des bits de 0.1 pour chaque unité de classement. Les résultats de ces expérimentations sont présentés au tableau 5.5.

Comparativement aux résultats obtenus avec le système EFFECT sans gel des paramètres appris (tableau 5.4), on remarque que le système avec gel génère des taux de classement moyens égaux ou légèrement inférieurs pour tous les ensembles de don-

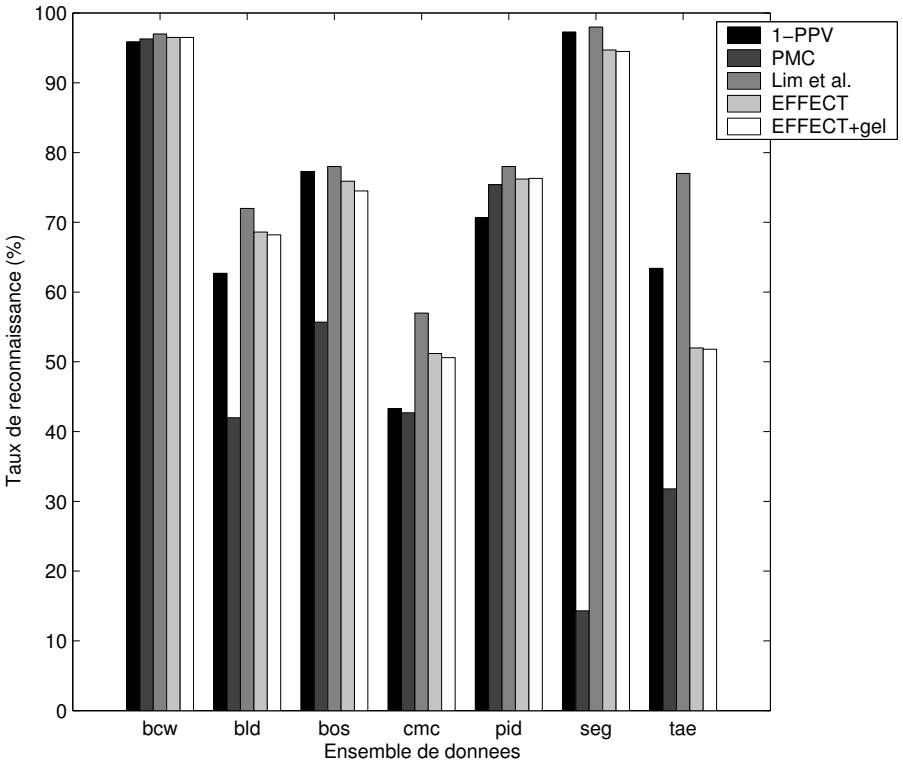


FIG. 5.16 – Comparaison des taux de reconnaissance pour des algorithmes testés.

nées, avec une complexité des arbres significativement réduite (près de la moitié dans certains cas). On peut supposer qu'en permettant un gel des paramètres, on évite une convergence des évolutions vers des topologies complexes d'arbres de décision qui permettraient d'amortir l'instabilité de l'apprentissage. La dégradation légère des taux de classement peut s'expliquer par une certaine rigidité permise dans les topologies avec gel des paramètres, ce qui pourrait possiblement favoriser le sur-apprentissage.

Finalement, la figure 5.16 présente un histogramme synthétisant les différents taux de classements obtenus pour tous les algorithmes d'apprentissage testés dans le chapitre. On remarque que pour tous les ensembles de données (excepté tae), la performance du système EFFECT avec et sans gel des paramètres est toujours assez près des taux de classement maximaux sur 33 algorithmes obtenus par Lim *et al.* [90].

5.6 Conclusion

Dans ce chapitre, nous avons proposé un système nommé EFFECT qui permet une combinaison hiérarchique de diverses unités de classement par une évolution de topo-

logies d'arbres jumelée à un apprentissage statistique de paramètres numériques. Les algorithmes d'apprentissage utilisés sont spécifiques à chaque unité de classement. Un échange flou d'information s'effectue entre les nœuds formant les arbres, par des disjonctions floues des degrés d'appartenance des données durant la passe descendante, suivies de conjonctions floues des degrés d'appartenance de classement durant la passe ascendante. En plus des unités simples de duplication, d'amplification et de clarification, trois unités de classement principales sont utilisées : une unité de classement paramétrique de type neurone ADALINE, une unité de classement non paramétrique par le PPV et une unité de classification non supervisée à K-moyennes floue. Chacune de ces trois dernières unités de classement utilise un algorithme d'apprentissage particulier. L'évolution de topologies est guidée par deux objectifs, soit la maximisation des marges floues de classement et la minimisation de la complexité des arbres.

À la section 5.4, le système EFFECT a été testé sur sept ensembles de données tirés du *machine learning repository* de l'UCI. Les résultats obtenus avec le système EFFECT se comparent avantageusement à ceux obtenus avec un classifieur PPV et un réseau de neurones PMC. Ces résultats sont également comparés à ceux rapportés par Lim *et al.* [90] pour 33 algorithmes d'apprentissage automatique différents. Excepté pour un ensemble de données, la comparaison démontre que le système EFFECT est capable de performances soutenues, avec des taux de classement près des meilleurs taux obtenus sur les 33 algorithmes testés par Lim *et al.* La contre-performance sur le dernier ensemble peut s'expliquer par la procédure de sélection des meilleurs individus de l'évolution, qui n'est pas adaptée à des ensembles de données de petite taille.

En un deuxième temps, le système EFFECT est modifié par l'ajout d'un bit à chaque unité de classement des arbres, afin de geler les paramètres appris. Le bit de chaque unité de classement peut être inversé aléatoirement durant les évolutions, par une opération de mutation spécialisée. Cette modification au système EFFECT original permet au processus évolutionnaire d'ajouter une certaine rigidité dans les arbres de décision et de possiblement pallier à l'instabilité du processus d'entraînement. Les résultats démontrent que le système avec gel des paramètres fournit des taux de classement comparables ou très légèrement inférieurs au système sans gel, tout en générant des arbres d'une complexité moyenne significativement réduite.

Comme travaux futurs, une avenue à explorer serait de développer de nouvelles stratégies pour la sélection des meilleurs individus de chaque évolution. L'approche actuelle a l'avantage de permettre le choix d'individus qui semble bien généraliser par une sélection effectuée à l'aide d'un ensemble de données qui n'est pas utilisé pour l'entraînement ni le test. Cette méthode de sélection d'individus a cependant un prix, soit de réduire significativement le nombre de données utilisées pour l'entraînement et

l'évaluation de l'adéquation, et donc de retirer de l'information précieuse qui pourrait mieux guider l'apprentissage des topologies et des paramètres. De plus, le processus de sélection est légèrement biaisé du fait que chaque individu de chaque génération est testé avec le même ensemble de données. En effet, il est fort probable que le processus fasse parfois une sélection d'individus particulièrement performants sur l'ensemble de validation, mais ayant autrement de faibles capacités de généralisation. Différentes méthodes pour la sélection des meilleurs individus de chaque évolution devraient donc être explorées afin de conserver un maximum d'information pour l'entraînement, tout en sélectionnant les individus qui généralisent le mieux le données.

Le système EFFECT est conçu pour être extensible en permettant l'intégration de différents types d'unités de classement. Des classifieurs relativement simples et représentatifs de différents algorithmes d'apprentissage communément utilisés ont été choisis pour illustrer l'utilisation du système. Dans une perspective future, il serait intéressant de tester le système EFFECT avec des unités de classement plus complexes, par exemple des unités basées sur des classifieurs de type *support vector machines* (SVM) [33]. On pourrait ainsi spécialiser des classifieurs complexes en employant différents sous-ensembles des caractéristiques d'entrées, en utilisant différents regroupements de classes ou en sélectionnant différents sous-ensembles des données d'entraînement, afin de générer de véritables mixtures d'experts hiérarchiques.

Chapitre 6

Conception automatisée de systèmes de lentilles

Science is what we understand well enough to explain to a computer. Art is everything else we do.
Donald Knuth

La conception d'un système de lentilles est une tâche complexe d'ingénierie pour laquelle il n'existe aucune démarche ou méthode définie pour la résolution de problèmes non triviaux. La conception optique moderne repose sur des ingénieurs expérimentés, qui utilisent des outils spécialisés de conception assistée par ordinateur (CAO). En se basant sur les spécifications, le processus débute habituellement avec un système de lentilles initial, souvent tiré de catalogues de designs connus, choisi selon l'intuition et l'expérience du concepteur. Puis, dans le but de répondre aux spécifications du problème, des modifications sont apportées à ce système initial par le concepteur, aidé d'algorithmes de recherche locale.

Ce chapitre porte sur l'utilisation des AE pour la conception de systèmes de lentilles. L'objectif consiste à démontrer que cette approche peut mener à des résultats comparables à ceux obtenus par le processus actuel, nécessitant de l'expertise humaine. À la section 6.1, nous introduisons les AE qui sont utilisés dans ce chapitre. À la section 6.2, nous présentons brièvement la théorie relative à la conception de systèmes de lentilles. À la section 6.3, nous passons en revue les différentes techniques d'optimisation locale et globale existantes pour la conception de systèmes de lentilles. Ensuite, à la section 6.4, nous introduisons un problème étalon de conception de systèmes de lentilles. Ce problème a été initialement présenté dans le cadre d'une compétition amicale entre des

experts humains. Les résultats obtenus avec les AE pour ce problème sont présentés à la section 6.5. Ils démontrent clairement que les AE utilisés sont capables de trouver des solutions meilleures que celles présentées à la compétition. Par la suite, nous présentons à la section 6.6 un problème réel de conception d'un système d'imagerie. Les résultats obtenus pour ce problème démontrent une fois de plus que les AE peuvent découvrir des systèmes de lentilles de qualité comparable à ceux conçus par des experts humains, après un effort raisonnable. À la section 6.7, nous reprenons le même problème d'imagerie en utilisant un critère d'optimisation à deux objectifs afin d'améliorer la qualité de l'image tout en réduisant le coût du système. Enfin, nous concluons ce chapitre avec quelques considérations sur la conception automatisée de systèmes de lentilles avec les AE.

6.1 Stratégies d'évolution

Les stratégies d'évolution (SE) (voir la section 1.2.3) sont un ensemble d'AE particulièrement adaptés à l'optimisation de fonctions avec paramètres à valeur réelle. Dans ce paradigme, chaque individu consiste en un ensemble de caractéristiques d'une solution potentielle. Cet ensemble prend habituellement la forme d'un vecteur de nombres réels de dimension fixe. Une SE, appliquée à une population de parents de taille $\mu \geq 1$, sélectionne aléatoirement des individus pour générer une population d'enfants de taille λ. Pour engendrer une nouvelle population, les μ meilleurs individus sont choisis soit parmi les λ enfants (approche (μ, λ), où $\lambda \gg \mu$), soit parmi les μ parents et les λ enfants (approche $(\mu + \lambda)$, où $\lambda \gg 1$). Chaque enfant est généré par une mutation adaptative d'un parent, qui consiste généralement à additionner des valeurs générées aléatoirement selon une fonction de densité de probabilité paramétrée. Dans les SE modernes, les paramètres de cette fonction de densité, aussi appelés *paramètres de stratégies*, sont associés à chaque individu et évoluent dans le temps selon les mêmes principes de mutation. Ces algorithmes sont qualifiés de SE auto-adaptatives (SA-ES pour *Self-Adaptive Evolution Strategies* en anglais). Les trois principales variations de SA-ES sont : les SA-ES isotropes, où le paramètre de stratégie est une valeur unique d'écart-type pour toutes les composantes durant les mutations gaussiennes ; les SA-ES anisotropes, où le paramètre de stratégie consiste en une valeur d'écart-type pour chacune des composantes durant les mutations gaussiennes ; enfin, les SA-ES corrélées, où le paramètre de stratégie de chaque individu est une matrice de covariance qui paramètre la fonction de densité de probabilité des mutations gaussiennes.

Pour toutes les expérimentations mentionnées dans ce chapitre, nous avons utilisé les SA-ES anisotropes. Il s'agit d'un compromis intéressant entre les SA-ES isotropes,

qui restreignent à une échelle uniforme les dimensions de l'espace des caractéristiques, et les SA-ES corrélées, qui requièrent des populations de très grandes tailles pour estimer les matrices de covariances. Soit $\mathbf{x}_t = [x_1^t \; x_2^t \; \cdots \; x_n^t]^T$, la représentation vectorielle d'un parent à l'instant t, et $\Sigma_t = [\sigma_1^t \; \sigma_2^t \; \cdots \; \sigma_n^t]$, les paramètres de la stratégie qui lui sont associés. Les équations décrivant la mutation des SA-ES anisotropes sont :

$$\left.\begin{aligned} \sigma_i^{t+1} &= \sigma_i^t \, e^{\tau' \mathcal{N}(0,1)} \, e^{\tau \mathcal{N}_i(0,1)} \\ x_i^{t+1} &= x_i^t + \sigma_i^{t+1} \, \mathcal{N}_i'(0,1) \end{aligned}\right\} i = 1 \ldots n \tag{6.1}$$

où $\mathcal{N}(0,1)$, $\mathcal{N}_i(0,1)$ et $\mathcal{N}_i'(0,1)$ sont des nombres aléatoires normalement distribués, indépendants, de moyenne nulle et dont l'écart-type vaut un. Les recommandations de Schwefel pour les valeurs de τ et τ' sont $1/\sqrt{2n}$ et $1/\sqrt{2\sqrt{n}}$ respectivement, où n est la dimensionnalité des individus.

Les SA-ES anisotropes sont reconnus pour être des AE très efficaces pour résoudre des problèmes d'optimisation de paramètres à valeur réelle. Mais pour obtenir des résultats appréciables, il faut utiliser des populations relativement grandes afin de générer de bonnes estimations stochastiques des paramètres de la stratégie. Récemment, des SE dérandomisées[1] ont été développées afin de pallier plusieurs problèmes inhérents aux SA-ES. Actuellement, la SE avec adaptation de la matrice de covariance (CMA-ES pour *Covariance Matrix Adaptation Evolution Strategy* en anglais) [56, 57] est probablement la SE dérandomisée la plus complète. Elle consiste à utiliser une matrice de covariances globale pour paramétrer les mutations gaussiennes et à adapter cette matrice de covariances à partir des chemins cumulatifs de mutations fructueuses. Les CMA-ES utilisent également un algorithme (μ_W, λ), qui consiste à générer les λ enfants à partir d'un seul parent moyenné avec une somme pondérée des μ parents. Comme avec l'approche (μ, λ), la sélection des individus de la nouvelle génération se fait en prenant les μ meilleurs parmi les λ enfants, avec $\lambda \gg \mu$. La CMA-ES ainsi que la SA-ES anisotrope sont utilisées dans ce chapitre pour la conception de systèmes de lentilles. Pour une description plus détaillée ainsi qu'une présentation des fondements mathématiques des CMA-ES, le lecteur est prié de se référer à [55, 56].

6.2 Conception de systèmes de lentilles

Un système de lentilles est un arrangement de lentilles ayant des indices de réfraction, des courbures, des épaisseurs et des espacements précis. La figure 6.1 illustre un exemple de système à deux lentilles. La fonction du système de lentilles est de produire

[1] Traduction libre de l'anglais *derandomized*.

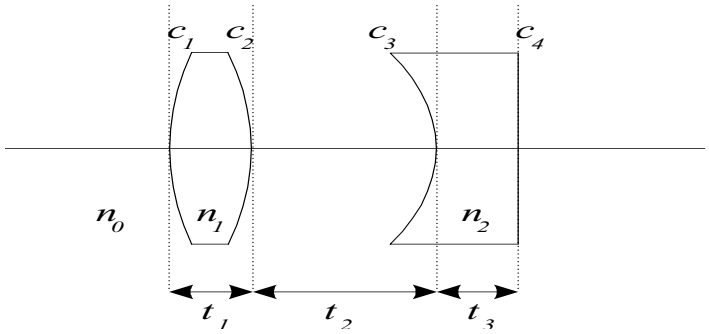


FIG. 6.1 – Paramètres d'un système de lentilles. n_i représente l'indice de réfraction du milieu, c_i est la courbure de la surface de la lentille, t_1 et t_3 sont les épaisseurs des lentilles et t_2 est l'espacement entre les lentilles.

l'image d'un objet de dimensions données et situé à une certaine distance. Bien que plusieurs arrangements de lentilles puissent produire une image de même taille et à la même distance, la problématique de la conception de systèmes de lentilles consiste essentiellement à découvrir celui qui cause le moins d'aberrations.

Les aberrations sont la différence entre l'image réelle produite par un système et l'image correspondante, une approximation, calculée avec l'optique de Gauss [107]. L'optique de Gauss est une méthode de calcul rapide pour caractériser un système optique avec différentes constantes telles que la longueur focale effective (en anglais, *effective focal length* – EFL), le diaphragme (*stop*), le f : chiffre (f*-number*), la distance de l'image et le grandissement. Les aberrations existent du fait que c'est l'optique de Gauss qui est utilisée durant le processus de design optique ; la physique exacte des systèmes de lentilles étant trop complexe pour être utilisable en pratique.

Pour caractériser un système de lentilles, on effectue un tracé de rayons. Partant d'un point donné de l'objet avec un angle initial donné, le tracé d'un rayon est le calcul de la trajectoire de la lumière dans le système optique jusqu'au plan image. Le tracé de rayons exact (réel) s'obtient par la première loi de la réfraction (Snell-Descartes), qui régit le comportement de la lumière entrant en contact avec l'interface entre deux milieux d'indices de réfraction différents. La trajectoire d'un rayon passant du milieu 1 au milieu 2 est régie par l'équation :

$$n_1 \sin\theta_1 = n_2 \sin\theta_2 \tag{6.2}$$

où n_1 et n_2 sont les indices de réfraction des milieux 1 et 2 et θ_1 et θ_2 sont les angles incident et réfracté par rapport à la normale de l'interface entre les deux milieux. La figure 6.2 illustre la première loi de la réfraction. D'autre part, l'approximation paraxiale consiste à supposer que tous les rayons sont à proximité de l'axe optique. En utilisant

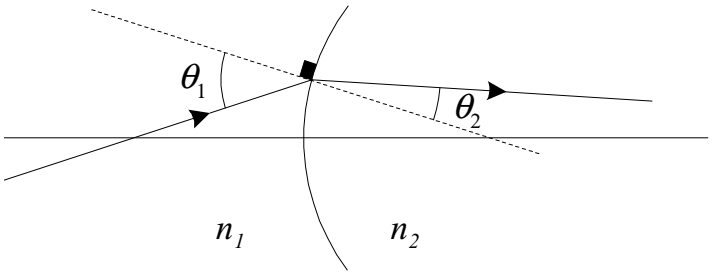


FIG. 6.2 – La première loi de la réfraction de Snell-Descartes.

l'expansion du sinus :

$$\sin\phi = \phi - \frac{\phi^3}{3!} + \frac{\phi^5}{5!} - \cdots \tag{6.3}$$

et en posant $\phi \approx 0 \Longrightarrow \sin\phi \approx \phi$, l'équation 6.2 devient :

$$n_1\theta_1 \approx n_2\theta_2 \tag{6.4}$$

Cette approximation est à la base de la théorie du premier ordre, ou optique de Gauss.

On quantifie les aberrations d'un système optique en calculant la différence entre l'image réelle, résultant de l'équation 6.2, et l'image approximée résultant de l'optique de Gauss. Autrement dit, deux tracés de rayons partant du même point de l'objet et avec le même angle, l'un exact et l'autre approximé[2], atteindront le plan image à des points différents. Ces différences, moyennées sur un ensemble représentatif de rayons, permettent de construire une mesure de qualité pour les systèmes de lentilles.

Il est intéressant de mentionner que si l'on considère également le deuxième terme de l'expansion du sinus de l'équation 6.3, on améliore notre approximation et l'on obtient alors la théorie du troisième ordre. La différence entre la théorie du premier ordre et la théorie du troisième ordre donne naissance aux cinq coefficients d'aberration primaire de Seidel : l'aberration sphérique, le coma, l'astigmatisme, la courbure de champ et la distorsion [107]. À titre d'exemple, la figure 6.3 illustre l'aberration sphérique et la distorsion. L'aberration sphérique (figure 6.3a) est due au fait que, pour les lentilles sphériques, les rayons provenant de l'infini et parallèles à l'axe optique ne convergent pas au même point, et cette divergence dépend de leur distance à l'axe optique. Ce type d'aberration cause une image floue. La distorsion, qui peut être en coussinet (distorsion positive) ou en barillet (distorsion négative), produit les formes illustrées à la figure 6.3b, la forme de référence (non distorsionnée) étant le carré en pointillé.

Enfin, l'indice de réfraction d'un matériau donné n'est pas constant mais varie en fonction de la longueur d'onde du rayon qui le traverse. À titre de référence, les valeurs

[2]Le tracé de rayon approximatif est virtuel et calculé avec l'optique de Gauss.

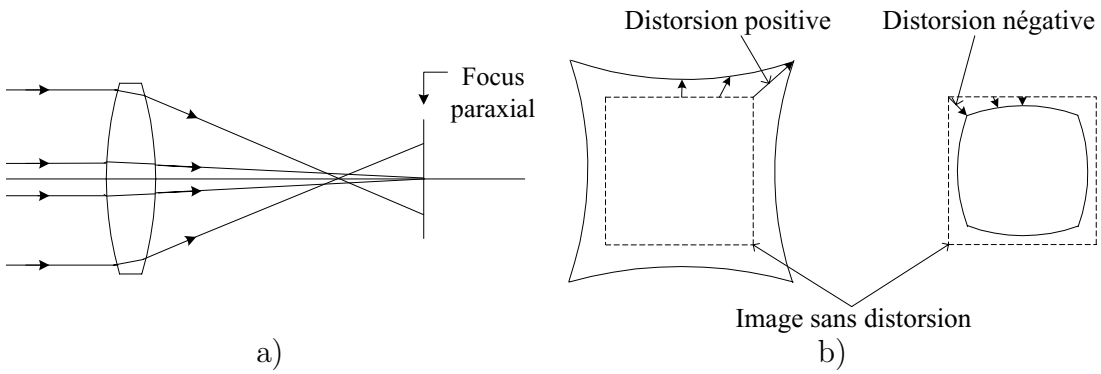


FIG. 6.3 – Deux exemples d'aberration de Seidel : a) l'aberration sphérique et b) la distorsion.

d'indices de réfraction des différents types de verre qui sont répertoriées dans les tables correspondent habituellement à la longueur d'onde de la raie D de l'hélium ($\lambda = 587.6$ nm). De plus, le taux de variation de l'indice de réfraction en fonction de la longueur d'onde est différent d'un verre à l'autre. Il est courant de caractériser le pouvoir dispersif d'un verre donné avec une mesure appelée nombre d'Abbe (*v-number*). Cette valeur correspond simplement au taux de variation relatif de l'indice de réfraction, calculé avec l'indice de réfraction du verre à trois longueurs d'ondes données. Il faut tenir compte du nombre d'Abbe des verres utilisés lors de la conception de systèmes de lentilles polychromatiques.

6.3 Optimisation de systèmes de lentilles

Le design moderne de systèmes de lentilles se fait en utilisant des logiciels de CAO spécialisés permettant de visualiser un système de lentille, d'évaluer sa qualité selon des critères particuliers et d'optimiser localement ses variables. Cette optimisation est effectuée par des algorithmes de recherche locale tels que la méthode des moindres carrés amortis (DLS pour *Damped Least Squares* en anglais). Cependant, lors de la conception de systèmes de lentilles, l'espace de recherche est le plus souvent un espace multidimensionnel complexe, non linéaire et comprenant plusieurs optimums ainsi qu'une forte corrélation entre les paramètres [137]. Par conséquent, une recherche locale permet d'explorer uniquement le voisinage immédiat des paramètres de départ. Il s'ensuit que le résultat dépend largement de la solution initiale. Mais depuis la fin des années 1980, de nombreuses expériences sur des méthodes de recherche globale ont été effectuées en design optique. Quelques chercheurs ont utilisé avec succès la méthode du recuit simulé [47, 59]. D'autres ont modifié des algorithmes d'optimisation locale, tels que le DLS,

pour permettre une exploration au-delà des optimums locaux [70]. Ces deux approches ont été récemment intégrées dans quelques outils de CAO d'optique.

L'idée d'utiliser des AE comme méthode d'optimisation globale pour la conception de systèmes de lentilles a été étudiée indépendamment par plusieurs chercheurs. Les travaux novateurs de Walk et Niklaus [153] présentent une application d'une SE de base pour la conception de systèmes de lentilles. Betensky [11] propose une méthode originale incluant un ensemble d'opérateurs de *puissance nulle*, c'est-à-dire des opérateurs qui ne modifient pas de façon significative les propriétés de premier ordre des systèmes de lentilles. Il a choisi d'utiliser un algorithme génétique (AG) à chaîne de bits, chaque position de la chaîne représentant l'application ou la non-application de l'opérateur de puissance nulle correspondant. Chaque système ainsi modifié par ces opérateurs est réoptimisé avec des algorithmes de recherche locale traditionnels. Il s'agit là d'une façon plutôt inventive d'utiliser les AE pour la conception de systèmes de lentilles. Depuis 1996, on répertorie plusieurs travaux sur l'utilisation des AG à chaîne de bits [7, 23], des AG à valeurs réelles [105, 150, 151] ou des SE [149, 150] pour l'optimisation d'un nombre fixe de paramètres à valeur réelle. Les résultats présentés dans [105] sont particulièrement impressionnants, car ils décrivent la conception automatisée fructueuse de systèmes de lentilles composés de plus de dix lentilles avec un AG à valeurs réelles et des expérimentations d'optimisation multi-objectif [26]. D'autres [24, 100] ont expérimenté une approche en deux étapes, soit l'application d'un AG à chaîne de bits qui fait une recherche globale afin de découvrir une bonne solution de départ suivie de l'application de la méthode du DLS qui complète l'optimisation afin d'améliorer cette solution.

Dans [10], nous avons présenté des résultats sur l'utilisation d'AE pour résoudre automatiquement un problème étalon en conception de systèmes de lentilles. Depuis la publication de cet article, Nagata [103] a présenté quelques résultats, pour ce même problème et d'autres problèmes tirés de [105], obtenus en utilisant l'algorithme CMA-ES. Dans cet article, Nagata utilise sa propre mesure de qualité, sans vraiment comparer ses résultats à ceux de [108] ni à ceux de [10]. Dans ce chapitre, nous apportons de nouveaux résultats pour le problème étalon mentionné ci-haut et pour un nouveau problème d'imagerie, en utilisant des techniques d'optimisation différentes. Pour les deux problèmes, les résultats obtenus avec les AE et présentés dans les sections qui suivent sont comparables et même meilleurs que les résultats obtenus par des experts humains, ce qui démontre la compétitivité de l'approche avec l'humain. De plus, les résultats présentés dans ce chapitre ont été générés de façon répétitive sur plusieurs évolutions indépendantes, ce qui démontre que la méthode proposée est fiable pour résoudre un problème de conception optique de façon constante.

6.4 Problème du *monochromatic quartet*

Afin d'évaluer la capacité des AE pour concevoir automatiquement des systèmes de lentilles, nous avons d'abord choisi un problème énoncé à la *1990 International Lens Design Conference* (ILDC 1990). Cette conférence, tenue tous les quatre ans, propose aux participants une compétition amicale de conception de systèmes de lentilles. Le problème énoncé dans le cadre de la compétition de 1990 [108] est devenu une application étalon pour évaluer la performance d'algorithmes d'optimisation de systèmes de lentilles. En effet, les onze meilleures solutions proposées formant seulement deux classes de solutions similaires, les organisateurs en ont conclu que ces classes constituaient les optimums globaux de l'espace des solutions.

Ce problème étalon se nomme le *monochromatic quartet*. Essentiellement, il consiste à trouver un système formé de quatre lentilles sphériques. Voici l'énoncé formel du problème (traduction de [108]).

> Concevoir un système de quatre composants, avec un f : chiffre de 3 ($f/3$), une longueur focale effective de 100 mm, en utilisant du verre BK7 et en tenant compte seulement de la raie D de l'hélium (c'est-à-dire utiliser un verre d'indice de réfraction $n = 1.51680$). L'objet est à l'infini et couvre 30° de plein champ (15° de demi-champ), et le plan image est plat.
>
> De plus, il faut tenir compte des contraintes suivantes : utiliser seulement des surfaces sphériques, pas de lentilles non asphériques, à gradient d'indice, de Fresnel, binaires, optiques holographiques, etc. L'épaisseur minimale des composants est de 2 mm, mais il n'y a aucune limite supérieure quant aux dimensions des lentilles. La distorsion doit être inférieure à 1 % et il ne doit pas y avoir de vignetage. Le dernier critère sert à éviter que le vignetage soit utilisé pour améliorer la performance dans les bords du système. La position du diaphragme n'est pas spécifiée.
>
> La fonction de mérite consiste à la moyenne du *RMS blur spot* pour trois angles de champ : sur l'axe, à 10.5° et à 15°, les trois angles ayant le même poids dans la fonction.

Le f : chiffre (qui peut aussi s'écrire $f/\#$) est un indicateur de la force du système de lentilles et de sa capacité à laisser passer la lumière. La longueur focale effective d'un système de lentilles est l'équivalent de la longueur focale pour une lentille seule. La longueur focale d'une lentille ou d'un système de lentilles correspond à l'inverse de sa puissance, qui est sa capacité de faire converger les rayons sur de courtes distances. Le verre BK7 est un type de verre ordinaire et couramment utilisé dans la fabrication

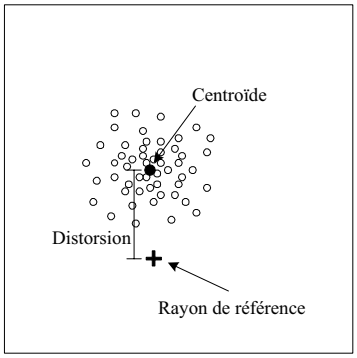
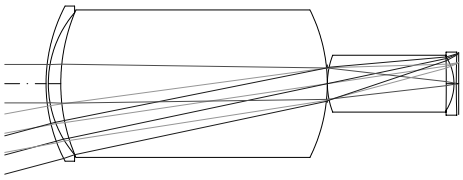


FIG. 6.4 – Mesure de la distorsion avec le centroïde du *blur spot*.

des lentilles. La contrainte sur la longueur d'onde (raie D de l'hélium) signifie que le problème est de type monochromatique, c'est-à-dire que la longueur d'onde est fixe, ce qui a pour conséquence que l'indice de réfraction du verre est également constant (autrement, il faudrait considérer différents indices de réfraction pour différentes longueurs d'onde). Il ne doit pas y avoir de vignetage, cela signifie que l'image ne doit pas être tronquée par le système de lentilles. Il est possible d'ajouter un diaphragme, c'est-à-dire une ouverture dans le système optique qui limite la quantité de lumière qui passe dans le système, permettant de réduire les aberrations. Son diamètre influence directement la longueur focale effective et le f : chiffre.

Le problème est formulé de sorte que l'on considère la distorsion séparément des autres types d'aberrations. La distorsion ne doit pas excéder 1 %, ce qui signifie que sous ce seuil, la priorité doit être mise sur la minimisation des autres aberrations. Le calcul de la taille du *RMS blur spot* consiste à tracer plusieurs rayons parallèles pour un angle d'incidence donné, en utilisant le tracé exact (équation 6.2). L'angle est posé à 0°, 10.5° et 15° successivement, tel que demandé dans l'énoncé du problème. Avec l'approximation paraxiale, tous les rayons ayant le même angle d'incidence convergent au même point. Mais avec le tracé exact, les rayons atteindront le plan image à différents points, généralement près du point où convergent les rayons approximés, formant ainsi ce qu'il est convenu d'appeler le « blur spot » , tel qu'illustré à la figure 6.4. La taille du *RMS blur spot* est calculée avec la variance de la position, au plan image, de différents rayons exacts ayant le même angle d'incidence. Un rayon de référence, tracé avec l'approximation paraxiale, est utilisé pour évaluer la distorsion, en mesurant la distance entre sa position et la position du centroïde des rayons exacts au plan image.

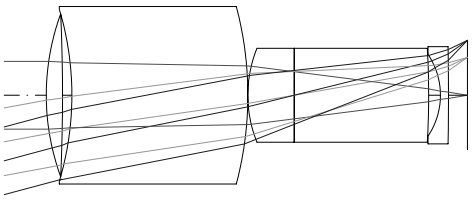
La figure 6.5 présente deux des meilleurs systèmes de lentilles présentés à la ILDC 1990, un pour chaque classe de solutions similaires. Le premier est la solution #14, le meilleur système présenté, et le deuxième est la solution #7, le meilleur système de la deuxième classe de solutions (le quatrième meilleur système présenté). La taille du



Rayon (mm)	Épaisseur (mm)	Demi-ouverture	Verre
0.0	∞	–	AIR
145.6875	2.0	64.9691	BK7
94.7116	10.8039	61.6892	AIR
162.7274	231.4971	61.6778	BK7
−143.6539	0.0	15.3583	AIR
0.0	**0.1**	**13.5674**	**AIR**
68.4004	103.2294	14.3693	BK7
−1480.2792	6.7215	23.2095	AIR
−43.0337	2.0	23.2221	BK7
858.3420	2.0046	25.8911	AIR
0.0	–	26.7948	–

(La surface en gras est le diaphragme.)

a)



Rayon (mm)	Épaisseur (mm)	Demi-ouverture	Verre
0.0	∞	–	AIR
116.753325	8.0	41.3577	BK7
−896.302517	4.6	41.5202	AIR
−148.252719	86.666308	41.7531	BK7
−170.784756	0.0	23.9590	AIR
61.852545	22.856198	21.7151	BK7
0.0	**66.187961**	**11.5910**	**BK7**
−406.404293	5.8	19.5193	AIR
−36.156468	3.965152	19.5915	BK7
−1470.14649	0.0	22.4919	AIR
0.0	–	26.7948	–

(La surface en gras est le diaphragme.)

b)

FIG. 6.5 – Le meilleur système de lentilles de chacune des deux classes de solutions, présentés par des experts humains à la ILDC 1990 : a) la solution #14, qui a un *RMS blur spot* de 0.00218 mm, et b) la solution #7, qui a un *RMS blur spot* de 0.00250 mm.

RMS blur spot de chacun des deux systèmes, calculée avec l'outil commercial de CAO d'optique CODE V [106], est de 0.00218 mm et de 0.00250 mm, respectivement. Ces valeurs sont légèrement différentes de celles reportées dans [108] (0.0021 mm et 0.0024 mm). Cela est dû à l'utilisation d'un logiciel différent pour le calcul des *RMS blur spots*.

6.5 Conception de systèmes de lentilles par algorithmes évolutionnaires

Dans le contexte de problèmes d'optimisation, il faut considérer le dilemme *exploration versus exploitation*. La configuration de l'algorithme de recherche devrait considérer un bon compromis entre l'exploration des régions de l'espace de recherche, afin de découvrir de meilleurs solutions, et l'exploitation des régions connues riches en bonnes solutions. Pour le cas particulier de la conception de systèmes de lentilles, les algorithmes de recherche locale tels que le DLS sont très performants pour découvrir les optimums locaux. Les AE devraient donc servir essentiellement à la tâche d'exploration, en laissant à des méthodes numériques bien établies la tâche d'exploitation. À l'instar de [11], mais contrairement à [24, 100], où une optimisation locale est effectuée à la fin des évolutions seulement, nous appliquons une optimisation locale à chaque individu de chaque génération, tout en s'assurant que les contraintes du problème sont encore respectées. Il s'agit là d'une approche hybride réduisant la tâche d'exploitation pour l'AE.

Pour le problème du *monochromatic quartet*, un individu est représenté par un vecteur de 15 paramètres à valeur réelle. Durant le processus d'initialisation, toutes les valeurs sont uniformément générées dans l'intervalle $[-1, 1]$ et évoluent par la suite sans contrainte. Au moment d'évaluer l'adéquation, chaque valeur est mise à l'échelle et parfois transformée afin de cadrer avec son domaine. Le tableau 6.1 présente les différents paramètres et les transformations qui y sont associées. La courbure de la dernière surface (c_8), la distance entre la dernière surface et le plan image (t_i) et l'ouverture du diaphragme sont des variables dépendantes, calculées pour chaque système au cours de l'évolution afin d'obtenir une image paraxiale nette sur le plan image et de respecter le f : chiffre (3.0) et la longueur focale effective (100.0) spécifiés. La mesure de la position du diaphragme donne sa position relative entre la première et la dernière lentille du système. La valeur de courbure d'une surface de lentille correspond à l'inverse du rayon de courbure ($c_j = 1/r_j$). Cette valeur a été utilisée durant le processus d'optimisation plutôt que le rayon étant donné que les variations de la courbure sont plus cohérentes avec les modifications physiques qui en résultent.

Type	# de param.	Transformation de valeurs
Courbure	7	$c_j = 0.025\tilde{c_j}$ mm^{-1}
Épaisseur	4	$t_j = (\lvert 10\tilde{t_j}\rvert + 2)$ mm
Distance	3	$t_j = \lvert 10\tilde{t_j}\rvert$ mm
Position du diaphragme	1	$s = \begin{cases} \lvert\tilde{s}\rvert & \tilde{s} \in [-1, 1] \\ 1 & \text{autrement} \end{cases}$

TAB. 6.1 – Nombre de paramètres et transformations de valeurs pour le *monochromatic quartet* ; les variables comprenant un tilde ($\tilde{x}$, par exemple) représentent une valeur non transformée du vecteur.

Type	Contrainte	Pénalité
distorsion	$\lvert\%_{\texttt{dist}}\rvert \leq 1$	$P_{\texttt{dist}} = \lvert\%_{\texttt{dist}}\rvert$
distance de l'image	$t_i \geq 0$	$P_{t_i} = -t_i$
vignetage	$l_{\texttt{vign}} = 0$	$P_{\texttt{vign}} = 1000\ l_{vign}$
RMS blur spot	Calculable à 0°, 10.5° et 15°	$P_{\texttt{RMS}} = 1000$

TAB. 6.2 – Contraintes physiques du *monochromatic quartet* ; une pénalité est nulle lorsque la contrainte associée est respectée. $\%_{\texttt{dist}}$ est le pourcentage de distorsion mesuré au plan image et $l_{\texttt{vign}}$ est le nombre de surfaces où il y a du vignetage.

La mesure d'adéquation utilisée lors des expérimentations est composée d'une valeur scalaire indiquant la qualité du système et d'une valeur booléenne spécifiant si la solution est physiquement possible ou non. Si un système est physiquement impossible, l'adéquation est calculée comme la somme des coefficients d'aberration de Seidel et de différentes pénalités :

$$\begin{aligned} F_1 &= 1000 + B + F + C + PI^2 + E \\ &\quad + P_{\texttt{dist}} + P_{t_i} + P_{\texttt{vign}} + P_{\texttt{RMS}} \end{aligned} \tag{6.5}$$

où B, F, C, PI^2 et E sont les cinq coefficients d'aberration de Seidel, l'aberration sphérique, le coma, l'astigmatisme, la courbure de champ et la distorsion, respectivement, et les P_x sont différentes pénalités calculées selon un ensemble de contraintes physiques, tel que présenté au tableau 6.2.

Si le système est physiquement possible et respecte les contraintes du problème, alors la mesure d'adéquation dépend seulement de la taille moyenne du *RMS blur spot*, calculé pour trois angles d'incidence :

$$F_2 = \sum_{\theta=\{0,10.5,15\}} \frac{\texttt{RMS}_\theta^{\texttt{orig}}}{3} \tag{6.6}$$

L'optimisation locale est ensuite appliquée au système durant six secondes. Si la solution optimisée respecte toujours les contraintes, la taille du *RMS blur spot* est calculée de nouveau.

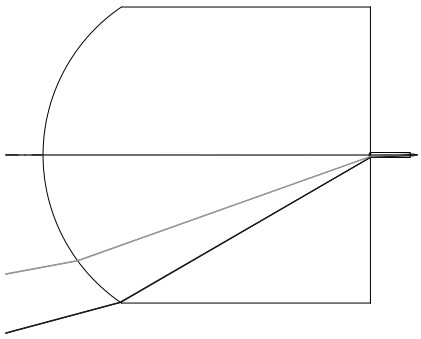
$$F_3 = \sum_{\theta=\{0,10.5,15\}} \frac{\mathtt{RMS}_\theta^{\mathtt{opt}}}{3} \tag{6.7}$$

La valeur finale de l'adéquation est donc F_1, F_2 ou F_3, selon le contexte.

La framework Open BEAGLE est utilisée pour l'implantation de toutes les expérimentations présentées dans ce chapitre. Les caractéristiques du système de lentilles (coefficients d'aberration, taille du *RMS blur spot*, etc.) sont calculées avec l'outil commercial de CAO CODE V [106]. Cet outil est aussi utilisé pour appliquer les six secondes d'optimisation locale avec l'algorithme DLS[3]. Il a été observé dans des expériences préliminaires que l'optimisation locale semblait converger en quelques secondes. Il a donc été estimé qu'une période d'optimisation locale de six secondes serait amplement suffisante dans la plupart des cas pour raffiner les solutions brutes générées par l'AE, tout en limitant les besoins en ressources computationnelles.

En guise de première expérimentation, une SA-ES anisotrope $(\mu+\lambda)$ a été appliquée au problème du *monochromatic quartet*. Cinq populations de $\mu = 50$ individus chacune ont été évoluées, avec $\lambda = 350$ enfants générés à chaque génération pour chaque population, en utilisant seulement la mutation adaptative de la SA-ES anisotrope de l'équation 6.1. La migration en anneaux unidirectionnelle aléatoire a également été appliquée à deux individus choisis dans chaque population, à chaque génération. Les populations ont évoluées durant 250 générations, avec des paramètres de stratégie initiaux de $\sigma_i = 2.0$ et une borne inférieure de 0.05. Les paramètres de l'algorithme SA-ES anisotrope ont été posés après quelques expérimentations préliminaires. Les tailles des populations sont relativement élevées comparativement aux valeurs que l'on retrouve généralement dans la littérature. Cela s'est avéré nécessaire afin d'initialiser les populations avec au moins quelques solutions qui respectent dès la première génération les contraintes de base du problème. Pour ce qui est du nombre de générations des évolutions, il a été observé que les évolutions pour le problème du *monochromatic quartet* semblaient avoir convergées vers un optimum après environ 200 générations. Cette expérience incluant cinq populations a été répétée cinq fois et le meilleur résultat obtenu est présenté à la figure 6.6. Le *RMS blur spot* a un rayon de 0.00167 mm, c'est-à-dire 23 % plus petit que le meilleur design humain présenté à la ILDC 1990. Toutefois, il faut mentionner que le système obtenu est assez excentrique. En effet, sa longueur totale est de plus de 23 m et la première lentille est disproportionnée relativement aux autres. Néanmoins, le système respecte toutes les spécifications du problème. À notre connaissance, il s'agit du meilleur design proposé à ce jour pour ce problème, ce qui contredit l'idée voulant

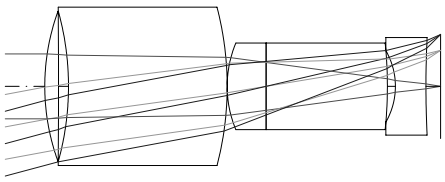
[3]En utilisant la commande `AUT` avec l'optimisation globale désactivée (`GS NO`).



Rayon (mm)	Épaisseur (mm)	Demi-ouverture	Verre
0.0	∞	–	AIR
11141.72266	20296.860683	9185.5313	BK7
184.34870	110.757080	153.1824	AIR
−487.56700	2438.208956	151.9749	BK7
0.0	**5.996627**	**49.4441**	**BK7**
−206.83954	8.574215	49.4441	AIR
224.63126	4.437700	47.4147	BK7
1803.40918	5.638265	47.4147	AIR
−226.65711	200.040559	47.4147	BK7
−234.83769	211.117756	43.9963	AIR
0.0	–	26.7949	–

(La surface en gras est le diaphragme.)

FIG. 6.6 – Meilleur système de lentilles obtenu avec la SA-ES anisotrope $(\mu + \lambda)$ pour le problème du *monochromatic quartet* ; la taille du *RMS blur spot* est de 0.00167 mm.



Rayon (mm)	Épaisseur (mm)	Demi-ouverture	Verre
0.0	∞	–	AIR
112.73243	7.052794	38.8061	BK7
−4616.30240	5.132246	38.8061	AIR
−144.83956	81.537799	38.8061	BK7
−160.81318	−0.000000	23.0086	AIR
57.83538	20.009066	20.8839	BK7
0.0	**62.313838**	**12.2913**	**BK7**
−216.40247	4.115601	18.3684	AIR
−36.90077	15.687862	18.3684	BK7
492.37558	7.557047	23.6293	AIR
0.0	–	26.7949	–

(La surface en gras est le diaphragme.)

FIG. 6.7 – Meilleur système de lentilles obtenu avec la CMA-ES pour le problème du *monochromatic quartet* ; la taille du *RMS blur spot* est de 0.00393 mm.

que les deux classes de designs proposées à la ILDC 1990 constituent les optimums globaux. Il est intéressant de mentionner que les cinq exécutions indépendantes de la SA-ES anisotrope $(\mu + \lambda)$ ont convergé vers la même classe de designs.

Comme deuxième expérimentation, une CMA-ES a été appliquée au problème du *monochromatic quartet*, en utilisant cette fois une seule population de $\mu = 10$ individus, $\lambda = 50$ enfants et 500 générations. Les valeur de μ et λ respectent les directives données dans [55] alors que le nombre de générations utilisé est posé comme étant suffisamment grand pour permettre une convergence complète de l'algorithme pour le problème à résoudre. La figure 6.7 présente le meilleur système de lentilles obtenu sur les cinq exécutions. Ce résultat obtenu avec la CMA-ES appartient à la seconde classe des solutions présentées à la ILDC 1990 (figure 6.5b). À 0.00393 mm, la taille du *RMS blur spot* est 57 % plus grande que le meilleur design humain de la même classe. Par contre,

Spécification	Exigence
Nombre de lentilles	≤ 5
Longueur totale du système	$\sum t_j \leq 120$ mm (de l'objet jusqu'à l'image)
Distance de l'objet	$t_o = 75$ mm
Champ de vision complet	80 mm ($y_o \in [-40, 40]$)
Grandissement	$m = -0.36$
f : chiffre	$f/4$ (à déterminer)
Vignetage	Aucun vignetage toléré.
Demi-ouverture des lentilles	$a_j \leq 15 \quad \forall j$
Longueurs d'onde et poids	865 nm – 0.5, 890 nm – 1.0, 915 nm – 0.5
Format de l'image	Détecteurs CCD de 15×15 μm
Qualité de l'image	Cercle contenant 75% de l'énergie d'un diamètre inférieur à 15 μm

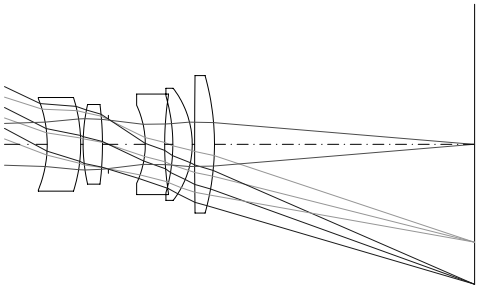
TAB. 6.3 – Spécifications pour le problème du système d'imagerie.

les ressources computationnelles nécessaires pour atteindre cette solution sont de moins de un dixième de celles requises par la SA-ES (environ 15 heures par rapport à 167 heures sur un PC Pentium 4, 3 GHz). La différence de temps entre les deux approches s'explique par la grande taille de la population nécessaire à la SA-ES anisotrope ($\mu+\lambda$), ce qui augmente les besoins en calcul mais permet une exploration plus détaillée de l'espace de recherche.

6.6 Problème du système d'imagerie

Dans cette section, nous tentons de résoudre un problème réel que nous avons appelé le *problème du système d'imagerie*. Ce problème a été soumis à l'équipe de concepteurs en optique de l'Institut national d'optique (INO), à Québec. Les experts de l'INO sont arrivés à une solution conforme aux spécifications tout en respectant les limites budgétaires imposées par le client, soit d'environ 5 personne-jours. Il ne fait aucun doute qu'avec plus de ressources, les experts auraient pu obtenir un meilleur système.

Le problème consiste à concevoir un système d'imagerie de longueur limitée. Le tableau 6.3 présente les spécifications. Ce problème inclut des contraintes physiques difficiles à respecter. Le critère de qualité est le diamètre du cercle comprenant 75 % de l'énergie, l'objectif étant que ce cercle lumineux puisse passer entièrement dans un



Rayon (mm)	Épaisseur (mm)	Demi-Ouverture	Verre (indice : Abbe)
0.0	76.13582	–	AIR
−12.59783	3.453205	4.208	1.762 : 26.5
−16.10987	0.249998	3.922	AIR
18.36991	2.0	3.541	1.734 : 51.8
−32.45907	0.608283	3.110	AIR
0.0	**3.805474**	**2.497**	**AIR**
−9.53878	2.0	3.523	1.626 : 35.7
32.59135	0.841538	4.545	AIR
−17.88445	1.999995	4.503	1.669 : 57.4
−9.49337	0.25	5.103	AIR
584.59130	2.046863	5.984	1.670 : 39.2
−26.44450	26.74464	6.327	AIR
0.0	–	14.4	–

(La surface en gras est le diaphragme.)

FIG. 6.8 – Solution présentée par les experts de l'INO pour le problème du système d'imagerie.

capteur CCD de 15×15 μm. Plus le diamètre du cercle lumineux est petit, plus la qualité de l'image est grande. Le diamètre du cercle contenant 75 % de l'énergie est évalué avec trois longueurs d'onde de différents poids, tel que spécifié au tableau 6.3. Le f : chiffre mentionné est plus une indication qu'une spécification : sa valeur peut être modifiée si nécessaire.

La figure 6.8 présente le design effectué par les experts de l'INO. Le système respecte toutes les contraintes mécaniques du problème. Le diamètre du cercle contenant 75 % de l'énergie est de 13.3 μm pour les points situés à 0 mm sur le plan objet, 33.0 μm pour les points situés à 28 mm et 21.8 μm pour les points situés à 40 mm. Étant donné que les spécifications sur la qualité de l'image sont quelque peu floues et que le cercle lumineux contenant 75 % de l'énergie a un diamètre inférieur à 15 μm pour les points situés à 0 mm sur l'objet, on peut considérer que le design est conforme à la qualité demandée. Toutefois, les diamètres mesurés pour les points situés à 28 mm et à 40 mm sur l'objet indiquent que l'image est probablement moins bien définie sur les bords que prévu.

Pour les expérimentations avec les AE, les individus sont représentés sous forme de vecteurs composés de 30 paramètres à valeur réelle. Le tableau 6.4 présente ces paramètres ainsi que les transformations appliquées. Pour ce problème, il n'y a aucune spécification sur les indices de réfraction et les nombres d'Abbe. Les valeurs initiales pour la position du diaphragme ($\tilde{s}$), les indices de réfraction ($\tilde{n_i}$) et les nombres d'Abbe ($\tilde{v_i}$) ont été normalisées dans l'étendue $[0, 1]$ à l'aide d'une fonction triangulaire. En plus de cette normalisation, les indices de réfraction et les nombres d'Abbe ont été confinés à un espace délimité par un quadrilatère dont les coins, si l'on utilise la notation (n_j, v_j), sont

Type	# de param.	Transformation de valeurs
Courbure	10	$c_i = 0.025\tilde{c_i}$
Épaisseur	5	$t_i = \lvert 20\tilde{t_i}\rvert + 2$
Distance	4	$t_i = \lvert 20\tilde{t_i}\rvert$
Indice de réfraction	5	$\mathring{n_i} = \begin{cases} \lvert\tilde{n_i}\rvert - \lfloor\lvert\tilde{n_i}\rvert\rfloor & \lceil\lvert\tilde{n_i}\rvert\rceil \text{ impair} \\ \lceil\lvert\tilde{n_i}\rvert\rceil - \lvert\tilde{n_i}\rvert & \lceil\lvert\tilde{n_i}\rvert\rceil \text{ pair} \end{cases}$ $n_i = 0.135\mathring{n_i} - 0.133\mathring{v_i} + 0.122\mathring{n_i}\mathring{v_i} + 1.620$
Nombre d'Abbe	5	$\mathring{v_i} = \begin{cases} \lvert\tilde{v_i}\rvert - \lfloor\lvert\tilde{v_i}\rvert\rfloor & \lceil\lvert\tilde{v_i}\rvert\rceil \text{ impair} \\ \lceil\lvert\tilde{v_i}\rvert\rceil - \lvert\tilde{v_i}\rvert & \lceil\lvert\tilde{v_i}\rvert\rceil \text{ pair} \end{cases}$ $v_i = -32.7\mathring{n_i} + 10.1\mathring{v_i} + 7.0\mathring{n_i}\mathring{v_i} + 60.3$
Position du diaphragme	1	$s = \begin{cases} \lvert\tilde{s}\rvert - \lfloor\lvert\tilde{s}\rvert\rfloor & \lceil\lvert\tilde{s}\rvert\rceil \text{ impair} \\ \lceil\lvert\tilde{s}\rvert\rceil - \lvert\tilde{s}\rvert & \lceil\lvert\tilde{s}\rvert\rceil \text{ pair} \end{cases}$

Tab. 6.4 – Paramètres à optimiser et transformation des valeurs pour le problème du système d'imagerie. Les variables avec un tilde ($\tilde{x}$, par exemple) représentent le paramètre original optimisé, avant la transformation. Les variables avec un point ($\mathring{x}$, par exemple) de même que la position du diaphragme sont normalisées dans l'étendue $[0, 1]$ en utilisant une fonction triangulaire.

situés à $(1.487, 70.4)$, $(1.620, 60.3)$, $(1.744, 44.7)$ et $(1.755, 27.6)$. Ce domaine inclut les indices de réfraction et les nombres d'Abbe des verres les plus courants. Contrairement aux expérimentations sur le *monochromatic quartet*, l'AE est contraint de positionner le diaphragme sur une surface de lentille, plutôt que n'importe où dans le système. Ainsi, la position du diaphragme correspond à la première surface située à gauche de la position relative du diaphragme donnée par le paramètre s. Le distance entre la dernière surface et le plan image est calculée plutôt qu'évoluée afin d'obtenir une image paraxiale nette sur le plan image. Enfin, même si la valeur du f : chiffre n'est pas imposée, nous l'avons fixée à 4.0 pour toutes les exécutions d'AE.

La mesure d'adéquation est similaire à celle utilisée pour résoudre le *monochromatic quartet*, avec une valeur réelle indiquant la qualité du système et une variable booléenne spécifiant si les contraintes du problème sont respectées. Si le système ne respecte pas les contraintes, alors la mesure d'adéquation est une somme de pénalités :

$$F_1 = 1000 + \sum P_{a_j} + P_{y_i} + P_{tt} + P_{vign} + P_{t_i} \tag{6.8}$$

où les P_x sont les pénalités associées aux contraintes du problème et sont présentées au tableau 6.5. Étant donné que les contraintes physiques sont assez difficiles à respecter, certaines d'entre elles, c'est-à-dire le la demi-ouverture des lentilles, la taille de l'image et la longueur totale du système, ont été assouplies afin de ne pas restreindre le processus évolutionnaire en pénalisant trop largement les individus. Si le système obtenu respecte

Type	Contrainte	Pénalité
Demi-ouverture†	$a_j \leq 15 \quad \forall j$	$P_{a_j} = 2a_j - 30$
Taille de l'image†	$\lvert y_i \rvert \in [12.4, 16.4]$	$P_{y_i} = \begin{cases} (144/y_i) - 10 & \lvert y_i \rvert < 12.4 \\ 0.694 y_i - 10 & \lvert y_i \rvert > 16.4 \end{cases}$
Longueur totale†	$\sum t_j \leq 119$	$P_{tt_{max}} = \sum t_j - 119$
Vignetage	$l_{vign} = 0$	$P_{vign} = 1000\, l_{vign}$
Distance de l'image	$t_i \geq 0$	$P_{t_i} = -10\, t_i$

TAB. 6.5 – Contraintes physiques pour le problème du système d'imagerie. Les différentes pénalités, P_x, ont une valeur nulle lorsque la contrainte associée est respectée. Les contraintes marquées du symbole † n'affectent pas la faisabilité du système de lentilles lorsqu'elles ne sont pas respectées. a_i est la demi-ouverture (le rayon) de la lentille i. l_{vign} correspond au nombre de surfaces affectées par le vignetage.

les contraintes, alors le diamètre du cercle contenant 75 % de l'énergie est calculé en prenant trois points sur le plan objet ($y_o = 0$ mm, $y_o = 28$ mm et $y_o = 40$ mm) et en gardant la valeur maximale obtenue. La mesure d'adéquation inclut également des pénalités pour les trois contraintes assouplies qui n'auront pas été respectées.

$$F_2 = \frac{\max\limits_{j=\{0,28,40\}} EE^{orig}_{y_o=j}}{0.015} + \sum P_{a_j} + P_{y_i} + P_{tt} \tag{6.9}$$

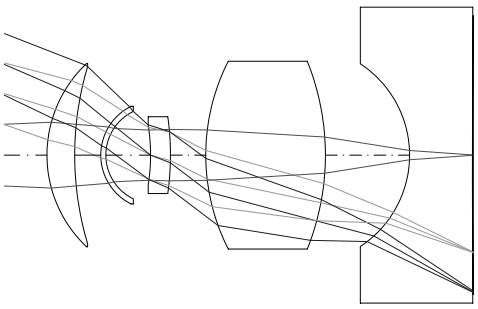
Ensuite, le système est optimisé localement durant six secondes. S'il respecte toujours les contraintes, l'adéquation est recalculée :

$$F_3 = \frac{\max\limits_{j=\{0,28,40\}} EE^{opt}_{y_o=j}}{0.015} + \sum P_{a_j} + P_{y_i} + P_{tt} \tag{6.10}$$

La première expérimentation pour le problème du système d'imagerie a été faite avec la SA-ES anisotrope $(\mu + \lambda)$. Sauf pour les paramètres de mutation de la stratégie, qui sont initialisés à $\sigma_i = 1.0$, les paramètres sont les mêmes que ceux utilisés pour le *monochromatic quartet* : cinq populations de $\mu = 50$ individus, $\lambda = 350$, 250 générations, deux individus migrant à chaque génération et un paramètre minimal de stratégie de $\sigma_i = 0.05$. Les tailles des populations et le nombre de générations de chaque évolution sont encore une fois plutôt élevés, pour les mêmes raisons que celles exposées à la section 6.5. La figure 6.9 présente l'algorithme de mutation et de croisement utilisé pour générer les individus de la nouvelle génération. L'algorithme est différent de celui utilisé pour le problème du *monochromatic quartet*, où seulement la mutation SA-ES anisotrope générait des variations. La figure 6.10 présente le meilleur système obtenu sur les cinq exécutions de la SA-ES anisotrope $(\mu + \lambda)$ avec cinq populations de $\mu = 50$ individus chacune. Le diamètre du cercle contenant 75 % de l'énergie est de 11.68 μm.

1. Soit $X = (x_1, x_2, \cdots, x_n)$ et $Y = (y_1, y_2, \cdots, y_n)$, une copie de deux individus choisis au hasard parmi la population de parents de taille μ, chaque individu étant composé de n valeurs réelles ;
2. Appliquer la mutation SA-ES anisotrope (équation 6.1) à X et à Y pour générer les individus mutés $\tilde{X}$ et $\tilde{Y}$;
3. Créer l'individu $Z = (z_1, z_2, \cdots, z_n)$ avec un croisement uniforme en affectant chaque composante à $z_j = \tilde{x_j}$, si $\mathcal{U}_j(0,1) < 0.5$ ou $z_j = \tilde{y_j}$, si $\mathcal{U}_j(0,1) \geq 0.5$, où $\tilde{x_j}$ et $\tilde{y_j}$ sont les composantes des individus mutés $\tilde{X}$ et $\tilde{Y}$, et $\mathcal{U}_j(0,1)$ sont des nombres générés aléatoirement selon une distribution uniforme dans $[0,1]$;
4. Retourner l'individu Z comme l'enfant nouvellement généré.

FIG. 6.9 – Algorithme de mutation et de croisement utilisé pour générer la nouvelle population dans les expérimentations avec la SA-ES anisotrope $(\mu+\lambda)$ pour le problème du système d'imagerie.



Rayon (mm)	Épaisseur (mm)	Demi-ouverture	Verre (indice : Abbe)
0.0	75.0	–	AIR
12.98357	2.861395	9.3063	1.744 : 44.7
32.13223	2.690679	9.1445	AIR
5.62296	0.524639	4.9694	1.755 : 27.6
5.00620	4.615829	4.509	AIR
−19.54992	**2.049896**	**2.589**	**1.755 : 27.6**
−28.86507	3.646095	3.4002	AIR
21.42485	12.345714	7.2616	1.744 : 44.7
−26.02713	8.684502	8.7982	AIR
−11.27961	6.481251	8.919	1.755 : 27.6
3078.62227	0.100000	14.1306	AIR
0.0	–	14.4	–

(La surface en gras est le diaphragme.)

FIG. 6.10 – Meilleur système de lentilles obtenu avec la SA-ES anisotrope $(\mu + \lambda)$ pour le problème du système d'imagerie ; le diamètre du cercle contenant 75 % de l'énergie est de 11.68 μm.

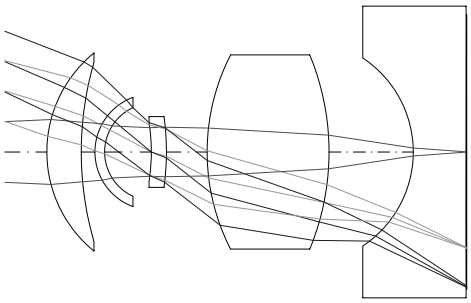
Le diamètre moyen des cinq meilleurs systèmes trouvés (sur les cinq exécutions) est de 12.19 μm, avec un écart-type de 0.48 μm. Le temps moyen de calcul pour chacune des évolutions est d'un peu moins de 8 jours (190 heures).

La qualité de l'image du meilleur système de lentilles obtenu avec la SA-ES anisotrope $(\mu + \lambda)$ est significativement meilleure que la valeur cible donnée dans les spécifications (15 μm). Les contraintes mécaniques du problème du système d'imagerie sont très grandes et compliquent probablement l'optimisation de la qualité de l'image. D'un point de vue pratique, la deuxième lentille du système est un ménisque concentrique mince et fragile, qui causerait probablement des difficultés lors de l'assemblage. L'AE ne tient pas compte de ce type de considérations, à moins bien sûr qu'il soit configuré à cet effet, afin d'éviter ce genre de lentilles. Un concepteur humain, par contre, considère naturellement ces caractéristiques. Sachant cela, un éventuel logiciel commercial de conception automatisée de systèmes de lentilles devrait certainement inclure plusieurs règles empiriques, afin d'intégrer quelques notions de savoir-faire sur la conception optique.

Il importe de noter que pour toutes les exécutions de la SA-ES anisotrope $(\mu+\lambda)$ sur le problème du système d'imagerie, la meilleure solution obtenue est toujours dans la même classe de designs. Donc, la SA-ES anisotrope $(\mu + \lambda)$ possède une bonne capacité de répétabilité pour le problème du système d'imagerie.

Pour la deuxième série d'expérimentations, l'algorithme CMA-ES a été utilisé pour résoudre le problème du système d'imagerie. Les paramètres utilisés pour les expérimentations sont : une population de $\mu = 10$ individus, $\lambda = 50$ enfants, 500 générations et des individus générés avec des valeurs aléatoires uniformément distribuées dans l'étendue $[-1, 1]$. La figure 6.11 présente le meilleur système d'imagerie obtenu avec la CMA-ES sur cinq exécutions. Le diamètre moyen du cercle contenant 75% de l'énergie du meilleur système de lentilles obtenu pour chacune des cinq évolutions avec la CMA-ES est de 14.09 μm, avec un écart-type de 1.52 μm. Le temps moyen de calcul pour une évolution avec la CMA-ES est de 41 heures.

Le diamètre du cercle contenant 75 % de l'énergie du meilleur système d'imagerie obtenu avec la CMA-ES est de 12.05 μm, ce qui est un peu plus grand que le diamètre du meilleur design obtenu avec la SA-ES anisotrope $(\mu + \lambda)$. Un examen visuel de ce système indique qu'il appartient à la même classe de designs que les meilleurs systèmes obtenus avec la SA-ES anisotrope $(\mu + \lambda)$.



Rayon (mm)	Épaisseur (mm)	Demi-ouverture	Verre (indice : Abbe)
0.0	75.0	–	AIR
13.3777	3.605079	9.4359	1.744 : 44.7
33.78061	1.434992	8.8859	AIR
6.10523	1.014729	5.4339	1.487 : 70.4
5.11550	4.891598	4.6416	AIR
−19.45778	**1.584712**	**2.5459**	**1.755 : 27.6**
−27.05155	4.264199	3.1678	AIR
22.29296	12.773908	7.6798	1.744 : 44.7
−26.63388	8.824164	9.2583	AIR
−11.81669	5.506620	9.3422	1.755 : 27.6
3950.09610	0.100008	14.1489	AIR
0.0	–	14.4	–

(La surface en gras est le diaphragme.)

FIG. 6.11 – Meilleur système de lentilles obtenu avec la CMA-ES pour le problème du système d'imagerie ; le diamètre du cercle contenant 75 % de l'énergie est de 12.05 μm.

6.7 Optimisation multi-objectif

À la section précédente, le problème de conception était formulé afin d'optimiser seulement la qualité de l'image, une fois les contraintes physiques respectées, en minimisant le diamètre du cercle contenant 75 % de l'énergie. Mais dans un contexte réel, le concepteur de systèmes de lentilles peut vouloir, après avoir atteint la qualité de l'image spécifiée, optimiser d'autres caractéristiques telles que le coût du système ou la tolérance à l'assemblage (désaxage des lentilles, etc.). D'autre part, les AE sont reconnus pour être très efficaces pour l'optimisation multi-objectif, en utilisant le concept d'optimalité de Pareto pour une recherche à base de population [26]. Cette section se veut une étude sur l'utilisation d'un AE multi-objectif pour optimiser simultanément la qualité de l'image et le coût relatif du système de lentilles.

Comme dernière série d'expérimentations, des optimisations avec des AE à deux objectifs sont effectuées dans le but de résoudre le problème du système d'imagerie. Le premier objectif consiste à minimiser la mesure d'adéquation présentée aux équations 6.8, 6.9 et 6.10 et le deuxième objectif consiste à minimiser le coût du système de lentilles. Le prix est déterminé en évaluant approximativement le coût de chaque lentille à partir d'une liste réelle, présentée au tableau 6.6, de coûts relatifs par gramme pour différents types de verre. Le coût relatif par gramme pour un verre donnée est évalué à partir de la valeur du verre le plus près dans le tableau 6.6, en utilisant cette mesure de distance :

$$d = \sqrt{(n_a - n_l)^2 + \frac{(v_a - v_l)^2}{10\,000}} \tag{6.11}$$

où n_a et v_a sont respectivement l'indice de réfraction et le nombre d'Abbe du verre et

Nom du verre	Indice de réfraction	Nombre d'Abbe	Prix relatif
F2	1.62004	36.37	0.16
F4	1.61659	36.63	0.25
F5	1.60342	38.03	0.2
K10	1.50137	56.41	0.25
K7	1.51112	60.41	0.2
KZFSN4	1.6134	44.29	0.3
KZFSN5	1.65412	39.63	0.3
LAFN7	1.7495	34.95	0.5
LF5	1.58144	40.85	0.2
LLF1	1.54814	45.75	0.25
NBAF10	1.67003	47.11	0.3
NBAF3	1.58272	46.64	0.35
NBAF4	1.60568	43.72	0.4
NBAF51	1.65224	44.96	0.3
NBAF52	1.60863	46.6	0.3
NBAK1	1.5725	57.55	2
NBAK2	1.53996	59.71	2
NBAK4	1.56883	55.98	1.5
NBALF4	1.57956	53.87	3
NBALF5	1.54739	53.63	2.5
NBASF2	1.66446	36	3
NBASF64	1.704	39.38	3
NBK10	1.49782	66.95	2
NBK7	1.5168	64.17	1
NF2	1.62005	36.43	2
NK5	1.52249	59.48	2
NKF9	1.52346	51.54	2
NKZFS4	1.61336	44.49	11
NLAF2	1.74397	44.85	3.5
NLAF3	1.717	47.96	6
NLAF7	1.7495	34.82	5.5

Nom du verre	Indice de réfraction	Nombre d'Abbe	Prix relatif
NLAK10	1.72003	50.62	4.5
NLAK12	1.6779	55.2	3
NLAK14	1.6968	55.41	3
NLAK21	1.64049	60.1	3.5
NLAK22	1.65113	55.89	3.5
NLAK33	1.75398	52.43	11.5
NLAK7	1.6516	58.52	3.5
NLAK8	1.713	53.83	3
NLAK9	1.691	54.71	5
NSF1	1.71736	29.62	4
NSF10	1.72828	28.53	4
NSF15	1.69892	30.2	3.5
NSF5	1.67271	32.25	3.5
NSF64	1.70591	30.23	3.5
NSF8	1.68894	31.31	3.5
NSK10	1.62278	56.98	4
NSK11	1.56384	60.8	3
NSK14	1.60311	60.6	3.5
NSK15	1.62296	58.02	3.5
NSK16	1.62041	60.32	3
NSK2	1.60738	56.65	2
NSK4	1.61272	58.63	6
NSK5	1.58913	61.27	5
NSSK2	1.62229	53.27	5
NSSK5	1.65844	50.88	4
NSSK8	1.61773	49.83	3.5
NZK7	1.50847	61.19	3
SF1	1.71736	29.51	2.5
SF10	1.72825	28.41	2
SF15	1.69895	30.07	2.5
SF5	1.6727	32.21	5

Tab. 6.6 – Liste des coûts relatifs, en unités par gramme, pour des verres courants utilisés dans la fabrication de lentilles ; le verre NBK7 est utilisé comme référence avec un coût de 1.0 unité monétaire par gramme.

n_l et v_l sont l'indice de réfraction et le nombre d'Abbe du verre tiré de la liste. En ce qui a trait à l'adéquation multi-objectif, il faut mentionner qu'une solution qui respecte les contraintes du système est considérée comme dominant une autre solution qui ne respecte pas les contraintes, peu importe leurs valeurs respectives d'adéquation pour chacun des objectifs.

Une variante multi-objectif d'un SA-ES anisotrope est utilisée dans les expérimentations. La stratégie de remplacement $(\mu + \lambda)$ est modifiée pour le NSGA-II (*Non-Dominated Sort Genetic Algorithm 2*) [37], plus couramment utilisé dans le paradigme des AG. Cette méthode est quelque peu similaire à l'approche $(\mu+\lambda)$, avec les particularités que $\mu = \lambda$ et que les individus sont sélectionnés pour la nouvelle génération à l'aide d'un tri multi-objectif sophistiqué de la population des parents et des enfants, basé sur la dominance et le nichage. La taille de la population est de 500 individus, les évolutions durent 500 générations et les enfants sont générés avec l'algorithme de croisement et de mutation présenté précédemment, à la figure 6.9. Avec cette configuration d'évolution de systèmes d'imagerie, le nombre d'individus testés durant chaque évolution de SA-ES multi-objectif correspond approximativement au nombre d'individus testés avec les évolutions basés sur l'algorithme du SA-ES anisotrope standard. Les paramètres de la mutation de la SA-ES anisotrope sont les mêmes que ceux utilisés pour la résolution précédente du problème du système d'imagerie, c'est-à-dire que le paramètre de la stratégie initial est de $\sigma_i = 1$ et sa valeur minimale est fixée à 0.05. Cinq évolutions différentes ont été lancées avec ces paramètres pour l'optimisation multi-objectif du système d'imagerie.

La sélection du meilleur individu de chaque évolution pour l'optimisation multi-objectif n'est pas aussi triviale qu'elle l'était pour l'optimisation à un seul objectif, où l'individu ayant la meilleure valeur d'adéquation était choisi. Dans le cas présent, il a été établi que le meilleur individu de l'évolution est celui, sur le front de Pareto, ayant le coût relatif minimal et dont le diamètre du cercle contenant 75 % de l'énergie est inférieur à 15 μm. Le meilleur individu de toutes les évolutions est choisi comme étant celui, parmi les cinq meilleurs, qui possède le coût relatif le plus bas. La figure 6.12 illustre une partie du front de Pareto de la dernière génération d'individus de l'évolution ayant produit la meilleure solution. La figure 6.13 présente le meilleur individu de toutes les évolutions, dont le cercle contenant 75 % de l'énergie a un diamètre de 13.8 μm et dont le prix relatif est de 29.61 unités par gramme. Le diamètre moyen du cercle contenant 75 % de l'énergie du meilleur système de lentilles obtenu pour chacune des cinq évolutions est de 13.6 μm, avec un écart-type de 0.8 μm et le coût relatif moyen est de 51.05 unités par gramme (écart-type de 23.57 unités par gramme). Le temps moyen de calcul pour une évolution est d'environ 63 heures.

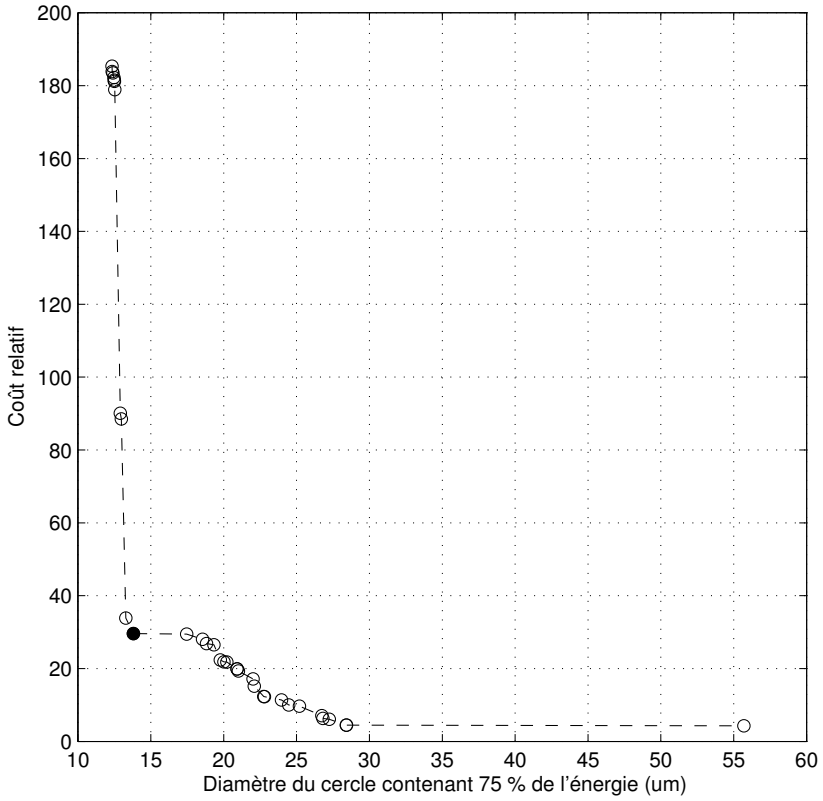
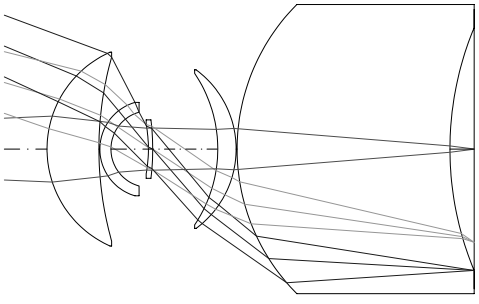


FIG. 6.12 – Front de Pareto de l'évolution ayant produit la meilleure solution. Les individus dont la valeur du premier objectif correspond à un diamètre de cercle contenant 75 % de l'énergie est supérieur à 60 μm (lorsque non-pénalisé) ont été omis. Le choix de la meilleure solution est identifié par le point noir.



Rayon (mm)	Épaisseur (mm)	Demi-ouverture	Verre (indice : Abbe)
0.0	75.0	–	AIR
10.92403	5.344778	9.8573	1.744 : 44.7
34.92101	0.1	9.4783	AIR
5.01065	1.139958	4.8137	1.755 : 27.6
3.94322	3.859566	3.8026	AIR
−14.01765	0.44	2.5340	1.755 : 27.6
−25.68777	**6.703711**	**2.1720**	**AIR**
−13.98103	1.891109	7.2912	1.745717 : 40.7141
−10.15616	0.1	7.6734	AIR
21.10594	21.929404	13.7703	1.626338 : 35.7022
32.67623	2.491357	12.5086	AIR
0.0	–	14.4	–

(La surface en gras est le diaphragme.)

FIG. 6.13 – Meilleur système de lentilles obtenu en utilisant la SA-ES anisotrope avec NSGA-II pour le problème du système d'imagerie ; le diamètre du cercle contenant 75 % de l'énergie est de 13.8 μm et le coût relatif du système est de 29.61 unités par gramme.

Le système d'imagerie conçu par les experts de l'INO a un coût relatif de 6.54 unités par gramme, ce qui est presque cinq fois plus petit que le prix du meilleur système trouvé avec l'algorithme NSGA-II. Cependant, il doit être souligné que le diamètre maximum du cercle contenant 75 % de l'énergie est de 33.3 μm, ce qui correspond à deux fois le diamètre du meilleur système évolué. En examinant de plus près le front de Pareto, on peut voir que la solution ayant une qualité d'image comparable, avec un diamètre du cercle contenant 75 % de l'énergie de 28.4 μm, a un coût relatif de 4.48 unités par gramme. Ce coût est de 32 % inférieur au coût relatif du système de l'INO. D'autre part, le système ayant le coût relatif le plus près sur le front de Pareto (6.31 unités par gramme) a un diamètre du cercle contenant 75 % de l'énergie de 26.8 μm. Cette qualité d'image est significativement supérieure à celle du système de l'INO. Cela illustre clairement l'avantage d'utiliser de l'optimisation évolutionnaire à plusieurs objectifs basée sur l'optimalité de Pareto. Cette approche permet la sélection de la solution la plus intéressante pour un certain problème en connaissance des différentes possibilités de compromis.

Dans un contexte pratique, la méthode utilisée pour sélectionner le meilleur individu de chacune des évolutions multi-objectif semble logique. Pour le problème du système d'imagerie, une fois que le diamètre du cercle contenant 75 % de l'énergie a atteint le seuil de 15 μm, d'autres priorités peuvent être considérées durant le processus de design. Dans le cas présent, la minimisation du coût du système est considérée comme deuxième objectif.

En examinant le meilleur système de lentilles obtenu, on peut constater que le critère du coût a été considéré durant l'évolution. En effet, la dernière lentille, de loin la plus imposante du système, est composée d'un verre près du verre F2, qui est le verre le moins dispendieux de la liste des verres utilisables du tableau 6.6 (0.16 unités par gramme).

6.8 Conclusion

La conception optique moderne repose grandement sur des outils de CAO qui permettent une visualisation du système optique, une évaluation de la qualité de l'image et une optimisation d'un design brut avec des algorithmes de recherche locale. Toutefois, afin d'obtenir un système de lentilles performant, le concepteur doit fournir au logiciel un bon système initial. Plusieurs chercheurs ont tenté, avec un succès limité, d'éliminer cette nécessité à l'aide d'algorithmes d'optimisation globale tels que les AE afin d'automatiser complètement la tâche de conception de systèmes de lentilles. Nous avons présenté ici une approche pour la conception automatisée de systèmes de lentilles avec

les AE, avec des résultats substantiels pour deux problèmes différents de conception. Plus précisément, deux algorithmes de la famille des SE, la SA-ES anisotrope $(\mu + \lambda)$ et la CMA-ES, ont été hybridés avec un algorithme d'optimisation locale d'un outil de CAO d'optique. L'objectif de cette approche est de tirer profit à la fois de la nature exploratoire des AE et des capacités d'exploitation des algorithmes de recherche locale spécialisés.

La méthode hybride proposée consiste à effectuer, durant un temps limité, une optimisation locale sur des solutions générées avec les AE, tout en tenant compte des contraintes du problème. Bien que cette approche demande beaucoup de temps de calcul, elle semble efficace pour explorer l'espace des solutions potentielles avec les AE et pour améliorer ces solutions avec des algorithmes d'optimisation locale classiques. À notre connaissance, cela a été fait seulement dans [11], où les solutions trouvées avec un AG représentent des transformations de puissance nulle d'un système de lentilles simple initial.

Les résultats obtenus avec cette méthode hybride « AE + optimisation locale » sont assez impressionnants. Dans un premier temps, l'approche a été testée sur un problème étalon, nommé le *monochromatic quartet*, qui a été soumis dans le cadre d'une compétition amicale entre experts humains lors de la conférence scientifique ILDC 1990 [108]. Deux séries d'expérimentations sont présentées pour le *monochromatic quartet*, avec respectivement la SA-ES anisotrope $(\mu + \lambda)$ et la CMA-ES. Dans la première série d'expérimentations, la meilleure solution obtenue pour chacune des cinq exécutions est supérieure au meilleur système présenté à la compétition humaine, ce qui contredit l'idée voulant que les solutions présentées à la conférence constituaient des optimums globaux. Le *RMS blur spot* de la meilleure solution obtenue par la SA-ES anisotrope $(\mu + \lambda)$ est 23 % plus petit que celui du meilleur design humain présenté à la ILDC 1990. Toutefois, bien que cette solution respecte toutes les spécifications du problème, la disproportion de la première lentille nous laisse à penser que les concepteurs humains évitent ce genre de systèmes de lentilles pour des raisons pratiques évidentes. Pour ce qui est de la deuxième série d'expérimentations, la meilleure solution trouvée avec la CMA-ES semble appartenir à l'une des deux classes de designs obtenues à la compétition humaine. Le *RMS blur spot* de cette solution a un rayon qui est 57% plus grand (0.00393 mm comparativement à 0.00250 mm) que celui du meilleur design humain de la même classe.

Le deuxième problème est un problème réel. Il s'agit d'une application d'imagerie qui a été soumise aux concepteurs optiques de l'INO. Le problème comporte des contraintes mécaniques importantes et l'objectif de qualité de l'image consiste à minimiser le diamètre du cercle contenant 75 % de l'énergie lumineuse, la valeur cible étant de 15 μm.

Le système proposé par les experts de l'INO respecte les contraintes mécaniques et le cercle contenant 75 % de l'énergie a un diamètre qui se situe à l'intérieur des spécifications près du centre de l'image, et un diamètre légèrement supérieur à 15 μm sur les bords de l'image.

Comme avec le *monochromatic quartet*, deux séries d'expérimentations sont présentées avec la même méthodologie d'AE. La meilleure solution obtenue avec la SA-ES anisotrope $(\mu+\lambda)$ respecte toutes les spécifications du problème et le diamètre du cercle contenant 75% de l'énergie est de moins de 11.7 μm sur tout le plan image. Les résultats obtenus avec la CMA-ES sont très similaires. Le diamètre du cercle contenant 75 % de l'énergie pour le meilleur système obtenu est inférieur à 12.1 μm sur tout le plan image, et le système respecte également toutes les contraintes du problème. Il est intéressant de mentionner que la meilleure solution obtenue avec la CMA-ES semble faire partie de la même classe de designs que la meilleure solution obtenue avec la SA-ES anisotrope $(\mu+\lambda)$. Comparé au système conçu par les experts humains de l'INO, le système évolué produit une image de meilleure qualité et ce, sur tout le plan image. Pour les deux séries d'expérimentations, le seul désavantage notable des solutions conçues automatiquement est qu'elles possèdent un composant fragile qui pourrait compliquer l'assemblage. Par contre, on ne peut discréditer l'approche par AE du fait qu'elle a convergé vers ces solutions, étant donné qu'aucune spécification de cette nature n'a été clairement énoncée. Pour obtenir des systèmes de lentilles qui satisfont ce genre de considérations pratiques, il faudrait implanter dans l'AE quelques notions de bon sens liées au design optique, sous la forme d'un ensemble de contraintes générales.

Une troisième série d'expérimentations consiste à appliquer une optimisation multi-objectif au même problème d'imagerie. En plus de l'objectif de respecter les contraintes mécaniques tout en minimisant le diamètre du cercle contenant 75 % de l'énergie lumineuse, on ajoute un second objectif qui consiste à minimiser le coût relatif du système de lentilles. L'algorithme SA-ES anisotrope $(\mu + \lambda)$ est modifié afin d'inclure l'algorithme de sélection multi-objectif basé sur l'optimalité de Pareto NSGA-II, en remplacement de l'approche $(\mu + \lambda)$. À la fin du processus évolutionnaire, on obtient un ensemble de solutions non dominées en guise de résultat final. À partir de ces solutions, qui offrent différents compromis entre la qualité de l'image et le coût relatif du système, la meilleure solution choisie est celle dont le coût est minimal parmi toutes celles dont le diamètre du cercle contenant 75 % de l'énergie est inférieur à 15 μm. La meilleure solution obtenue avec cette approche multi-objectif semble être une excellente solution au problème, avec un diamètre du cercle contenant 75 % de l'énergie inférieur à 13.8 μm sur tout le plan image et un coût relatif plutôt faible.

Tous les résultats générés par les AE de ce chapitre ont été comparés avec ceux

obtenus par des experts humains en conception optique. Dans la plupart des situations, les résultats évolués sont de qualité meilleure ou comparable, ce qui démontre la compétitivité des AE avec les humains pour la conception de systèmes de lentilles. En effet, l'approche évolutionnaire utilisée dans le cadre de cette compétition homme-machine satisfait le critère de compétitivité avec l'humain énoncé par Koza et al. [80]. Il s'agit d'une contribution importante car, à notre connaissance[4], aucune autre publication ne propose un système informatique clairement apte à concevoir des systèmes de lentilles de façon entièrement autonome.

Une autre réalisation importante de ce chapitre est la répétabilité observée dans les résultats. Toutes les expérimentations avec la SA-ES anisotrope $(\mu + \lambda)$ ont convergé vers la même classe de designs pour les deux problèmes, ce qui est très étonnant vu la nature stochastique des AE. Toutefois, ces résultats ont été obtenus au prix de temps de calcul très élevés dus aux populations de grande taille, chaque expérimentation ayant nécessité environ une semaine de calcul sur un ordinateur personnel moderne. Dans le cas de la CMA-ES, les résultats obtenus sont un peu moins bons que ceux obtenus avec la SA-ES anisotrope $(\mu + \lambda)$, mais les besoins en calcul sont significativement moins élevés, en moyenne une journée par expérimentation. Avec des populations de plus petite taille et une convergence relativement rapide vers de bonnes solutions, la CMA-ES semble donc plus efficace. Ainsi, le choix de l'AE à utiliser dépend du degré de qualité recherché et des ressources (temps et équipement) disponibles.

Finalement, comme voie future de ces travaux, on pourrait tirer avantage de la programmation génétique (PG) [9, 76] afin de concevoir automatiquement des systèmes de lentilles. Plutôt que d'évoluer directement les paramètres du système de lentilles, il serait possible d'utiliser une approche développementale où la PG pourrait introduire de nouveaux composants et appliquer des transformations spécialisées au système. Chaque solution générée par la PG serait alors utilisée comme système initial auquel on appliquerait des algorithmes d'optimisation locale afin d'améliorer les paramètres du système.

[4]Excepté [10], qui rapporte des résultats préliminaires aux travaux du présent chapitre, et [103] qui a essentiellement refait les travaux présenté dans l'article précédemment mentionné, en appliquant cette fois l'algorithme CMA-ES au *monochromatic quartet*.

Chapitre 7

Conclusion

Computers are useless. They can only give you answers.
Pablo Picasso

La thèse présente plusieurs applications concrètes des algorithmes évolutionnaires (AE) qui illustrent une même approche méthodologique. Au chapitre 2, nous avons présenté les avantages d'utiliser des outils logiciels génériques d'AE, nous avons exposé différents principes sous-jacents à de tels outils et, enfin, nous avons décrit l'outil logiciel Open BEAGLE qui nous a permis de réaliser l'ensemble des études présentées dans les chapitres subséquents. Le chapitre 3 porte sur la co-évolution de classifieurs basés sur la règle du plus proche voisin (PPV). Le chapitre 4 porte sur l'évolution par PG de représentations pour la reconnaissance de caractères cursifs. Quoique spécifique à un domaine applicatif particulier, cette approche est un exemple d'utilisation des AE pour la construction de caractéristiques dans des systèmes d'apprentissage. Nous avons ensuite présenté au chapitre 5 une troisième application en reconnaissance des formes qui consiste à construire des arbres de décision flous, extensibles et évolutionnaires. Complémentaire au deux précédentes, l'approche vise à combiner par évolution plusieurs classifieurs dans des arbres flous. Finalement, nous avons proposé au chapitre 6 une méthode pour la conception de systèmes de lentilles par un algorithme hybride basé sur des stratégies d'évolution (SE) et sur une technique d'optimisation locale adaptée au problème.

La thèse comporte des contributions à plusieurs niveaux. La section 7.1 porte sur les contributions au développement logiciel d'outils génériques d'AE. La section 7.2 présente les quatre grandes contributions applicatives qui composent la thèse ainsi que différentes approches techniques mises de l'avant pour réaliser ces applications. La sec-

tion 7.3 résume les principes de l'approche méthodologique proposé pour l'ingénierie de systèmes intelligents et récapitule la façon dont elle a été appliquée pour différents problèmes. Finalement, la section 7.4 expose des perspectives de recherche future.

7.1 Contributions au développement logiciel

Un aspect important et original du projet de recherche porte sur le développement d'outils logiciels génériques d'AE. La première contribution qui en découle est la proposition de six critères de généricité pour les outils d'AE. Une autre contribution, plus pratique, est incarnée par Open BEAGLE, la framework C++ générique d'AE libre développée durant le projet de recherche.

Les six critères de généricité proposés pour les outils logiciels d'AE sont les suivants :

1. Une représentation générique ;
2. Une adéquation générique ;
3. Des opérations génériques ;
4. Un modèle évolutionnaire générique ;
5. Une gestion extensible des paramètres ;
6. Une sortie configurable.

Ces critères couvrent un ensemble de caractéristiques souhaitables dans des outils logiciels d'AE dits génériques. Il tendent à caractériser la généricité sur trois fronts, soit une représentation et une mesure d'adéquation génériques, des algorithmes suffisamment configurables et modifiables pour implanter tout AE et une gestion extensible des paramètres des algorithmes et de la sortie de l'outil. Les critères de généricité sont utilisés à la section 2.1.5 pour évaluer la généricité de six outils logiciels d'AE relativement populaires. De ces outils, trois respectent ces critères de généricité, dont Open BEAGLE.

Open BEAGLE est une framework générique d'AE codée en C++. Elle est formée de trois couches logicielles distinctes, avec à la base de l'architecture une fondation orientée objet qui consiste en une extension du C++ et de la STL. Sur cette fondation s'ajoute la framework générique d'AE qui structure l'ensemble et fournit des mécanismes de généricité permettant l'implantation de tout AE. Finalement, la couche des frameworks spécialisées inclut les implantations des différents algorithmes et représentations spécifiques aux différentes saveurs d'AE. Open BEAGLE est disponible sur le Web[1] comme

[1] `http://beagle.gel.ulaval.ca`

logiciel à code ouvert. L'outil connaît un succès relatif, avec quelques dizaines de milliers de visiteurs distincts sur le site Web depuis son lancement et plusieurs dizaines d'utilisateurs connus répartis partout dans le monde. Il existe également deux projets logiciels connexes à la framework, soit Distributed BEAGLE, une architecture maître-esclave pour la distribution des calculs sur des réseaux d'ordinateurs, et BEAGLE Visualizer, qui permet de générer des rapports Web d'analyse des évolutions à partir des fichiers de résultats XML.

Il est important de noter que toutes les applications présentées dans la thèse ont été implantées avec Open BEAGLE. Cela tend à démontrer toute la généricité de l'outil, étant donné la variété des algorithmes et des représentations utilisés.

7.2 Contributions applicatives

Le projet de recherche est plutôt varié et comprend plusieurs applications de natures différentes qui tendent à illustrer une approche commune. Les chapitres 3 à 6 présentent des applications distinctes qui sont, en soi, des contributions importantes à la reconnaissance des formes et à la conception optique. Par ailleurs, il faut noter que chaque chapitre du développement a été (ou sera) soumis pratiquement tel quel à une revue scientifique (voir annexe A pour la liste des publications).

Le doctorat comprend également quelques contributions secondaires plus techniques. Premièrement, on identifie deux contributions techniques à l'intersection des AE et de la reconnaissance des formes, soit la minimisation de la complexité des modèles évolués par PG et l'utilisation d'un ensemble de données pour la sélection de la meilleure solution d'une évolution. On rapporte également deux contributions techniques supplémentaires aux AE, soit l'utilisation de primitives de PG avec valeurs générées aléatoirement et une méthode pour des SA-ES multi-objectif basée sur l'algorithme NSGA-II.

7.2.1 Application : co-évolution de classifieurs

La co-évolution de classifieurs basés sur la règle du PPV, présentée au chapitre 3, illustre clairement l'approche méthodologique proposée. Les résultats de la sélection de prototypes par AG sur des données synthétiques démontrent la capacité de l'approche à réduire la taille de l'ensemble des prototypes sans altérer le taux de reconnaissance relativement à des classifieurs PPV classiques. Par la suite, l'évolution de la mesure de

voisinage par PG est évaluée sur ces mêmes données synthétiques, avec cette fois des gains significatifs sur le taux de reconnaissance. La sélection de prototypes est alors combinée à l'évolution de mesures de voisinage par une co-évolution coopérative. Cette approche permet des gains sur les taux de reconnaissance des données synthétiques du même ordre que ceux obtenus avec l'évolution de mesures de voisinage par PG seule, tout en ayant des taux de sélection de prototypes comparables à l'approche avec un AG seul. En conjonction avec cette co-évolution coopérative, une co-évolution compétitive des cas de tests est effectuée par un AG à chaîne de bits modifié pour sélectionner un nombre pré-établi de données tests. Les résultats démontrent que l'ajout de la co-évolution de cas de tests permet une réduction importante du taux de sélection, au coût d'une dégradation légère du taux de reconnaissance. Finalement, la co-évolution coopérative de la sélection de prototypes et de la mesure de voisinage, avec et sans co-évolution compétitive des cas de tests, est évaluée sur quelques ensembles de données réelles. Les résultats sont cohérents avec ceux obtenus sur les données synthétiques, tout en démontrant des gains parfois plus marqués sur les taux de reconnaissance.

Cette première application à la reconnaissance des formes illustre un principe important, soit d'utiliser les AE pour optimiser finement différentes parties des systèmes de reconnaissance des formes, tout en basant le système sur des méthodes de classement bien établies. En effet, les AE ne sont pas particulièrement adaptés à la reconnaissance et ne devraient donc pas être utilisés directement à cette fin. Il existe de nombreuses techniques de classement qui ont une base théorique solide et qui performent bien en pratique. En ce sens, il est préférable d'utiliser les AE pour optimiser les parties du problème de la reconnaissance des formes pour lesquelles il n'existe pas de méthode universelle, par exemple la sélection des prototypes ou la construction des mesures de voisinage pour le cas spécifique de la règle du PPV, ou la sélection et la construction des caractéristiques pour le cas général. Les AE peuvent donc être très utiles pour transformer un problème de reconnaissance en un problème de classement plus facile. L'application démontre aussi l'intérêt d'utiliser la co-évolution pour optimiser différentes parties du système simultanément, sans diminution de performance.

7.2.2 Application : ingénierie évolutionnaire de représentations de l'écriture manuscrite

Le chapitre 4 porte sur l'utilisation de la PG pour une recherche automatisée de représentations permettant la discrimination de caractères cursifs. La méthode proposée pour l'ingénierie évolutionnaire de représentations est bâtie sur un système de reconnaissance de l'écriture en ligne qui a été développé il y a quelques années au Laboratoire

de vision et systèmes numériques. Ce système comprend un module de segmentation qui effectue une décomposition approximative de traits cursifs en séquences d'arcs de cercle [89]. À la sortie du module de segmentation, on utilise une représentation floue régionale [60] afin d'extraire différentes caractéristiques des arcs de cercle à fournir au module de classement. Dans le système original, quelques topologies régulières et statiques de décomposition de la fenêtre d'attention en régions ont été examinées. Une première contribution de l'application dans le cadre du projet de doctorat consiste à modifier le système pour utiliser des topologies hiérarchiques guidées par les données, en positionnant les frontières des régions selon les centres de masses des traits dans la région parent, au lieu d'utiliser les centres géométriques. Étonnamment, les résultats avec ce type de représentations sont moins bons qu'avec les représentation statiques, illustrant l'aspect parfois contre-intuitif du développement de représentations de caractères propices au classement automatique.

Par la suite, une recherche automatique de représentations basée sur la PG est proposée. Dans cette recherche, les programmes génétiques font d'une part un découpage hiérarchique de la fenêtre d'attention, avec des primitives divisant les régions en sous-régions selon les centres géométriques ou les centres de masses, et font d'autre part une extraction de différentes caractéristiques des régions, avec un deuxième type de primitives. Une optimisation à deux objectifs consistant à minimiser le taux d'erreur de classement et le nombre de caractéristiques est utilisée. Les résultats démontrent qu'avec moins de la moitié du nombre de caractéristiques, la meilleure représentation obtenue par PG affiche un taux de reconnaissance similaire à la meilleure représentation conçue par des humains. Il est aussi important de noter que les quatre évolutions faites pour cette application ont convergé vers des résultats de qualité comparable, démontrant une certaine stabilité de la méthode.

Cette application illustre une approche pour la construction de caractéristiques dans un domaine applicatif très particulier, soit la reconnaissance de caractères cursifs. Cependant, l'approche a une portée relativement générale en illustrant l'utilisation de la PG pour construire des ensembles de caractéristiques pour la reconnaissance de formes. La PG est particulièrement intéressante à cet effet, car elle permet des encodages avec un nombre variable de caractéristiques et avec une décomposition hiérarchique des données fournies par le module de segmentation. Le développement du module d'extraction des caractéristiques est souvent fait par des humains, avec un processus coûteux d'essais et d'erreurs. L'approche démontre qu'il est possible d'automatiser cette tâche et donc de déplacer une partie du fardeau de développement vers des calculs sur ordinateurs, qui sont plus économiques et qui peuvent se faire sans arrêt, 24 heures par jour.

7.2.3 Application : évolution d'arbres de décision flous

Une tendance importante des dernières années en apprentissage automatique consiste à combiner plusieurs classifieurs pour résoudre un problème donné. Le chapitre 5 propose une architecture où des classifieurs sont combinés dans une structure en arbre afin d'effectuer un découpage hiérarchique de l'espace d'entrée. L'information se propage dans ces arbres selon une mécanique inspirée de la logique floue. Pour les expérimentations présentées dans la thèse, trois types d'unité de classement simples sont utilisées : des classifieurs paramétriques linéaires neuronaux de type ADALINE, des classifieurs non-paramétriques de type PPV et des classeurs non-supervisés de type K-moyennes floues. Le système inclut également quelques autres unités effectuant des modifications simples des degrés d'appartenance flous des données. La topologie des arbres de décision est évoluée par PG, alors que les paramètres des unités de classement sont déterminés par des algorithmes d'apprentissage spécifiques à chaque type d'unité de classement. La mesure d'adéquation comporte deux objectifs, soit la maximisation de la marge floue des degrés d'appartenance aux différentes classes et la minimisation d'une mesure de complexité des arbres de décision. Les résultats démontrent que l'approche est capable de rivaliser, en termes de taux de classement, avec des classifieurs classiques sur plusieurs ensembles de données réelles. Une modification à l'algorithme est également proposée afin de geler en cours d'évolution les paramètres appris d'unités de classement. Les résultats démontrent que les taux de classement avec l'approche modifiée sont pratiquement inchangés par rapport à l'approche originale, alors que la complexité moyenne des arbres s'avère substantiellement réduite.

Le système proposé pour combiner des classifieurs dans des arbres flous est testé avec trois types de classifieurs simples. Ces classifieurs sont modifiés pour intégrer le degré d'appartenance flou de chaque donnée dans l'algorithme d'apprentissage et pour ajuster les degrés d'appartenance lors de la prise de décision. L'approche n'est cependant pas restreinte à l'utilisation de ces trois types de classifieurs simples ; elle pourrait très bien être étendue à d'autres types de classifieurs qui pourraient, ou non, être modifiés afin d'intégrer le degré d'appartenance flou de chaque donnée. En ce sens, l'approche est générale et peut être utilisée pour faire une combinaison complexe de pratiquement tous les types de classifieurs.

7.2.4 Application : conception automatisée de systèmes de lentilles

Finalement, le chapitre 6 présente une méthode pour automatiser la conception optique à l'aide d'AE. L'approche repose sur l'utilisation de SA-ES pour une recherche de solutions brutes, qui sont par la suite raffinées par l'application d'une méthode d'optimisation locale de type moindres carrés amortis adaptée à la conception optique. L'approche est testée sur deux problèmes difficiles de conception optique, soit un problème qui a été proposé dans une conférence scientifique et qui est devenu un problème étalon pour évaluer la performance d'algorithmes d'optimisation globale, et un deuxième problème, moins académique, qui a été soumis à des experts en conception optique de l'INO. Les résultats obtenus avec les AE sont comparables ou supérieurs à ceux obtenus par des experts humains. De plus, une expérimentation avec l'optimisation simultanée de la qualité de l'image et du coût du système démontre que l'approche proposée peut efficacement intégrer une optimisation multi-objectif.

De nombreux chercheurs se sont attaqués à la conception automatisée de systèmes optiques. Bien que les outils de CAO d'optique incluant des algorithmes d'optimisation locale soient au coeur de la conception optique moderne, l'intervention humaine reste nécessaire. L'approche proposée semble prometteuse pour permettre une conception optique entièrement automatisée, sans intervention humaine. En effet, les résultats obtenus sont très encourageants étant donné la qualité des systèmes obtenus et la répétabilité des résultats. Bien que quelques résultats affichent certaines lacunes d'un point de vue pratique, il serait facile d'ajouter des règles de savoir-faire dans les spécifications, par exemple en limitant plus précisément la forme et la taille des lentilles, ce qui rendrait la méthode d'autant plus applicable dans la réalité. Enfin, la conception optique pour des problèmes concrets implique souvent que plusieurs critères soit respectés ou optimisés. Dans ces circonstances, il est possible d'inclure le nombre nécessaire d'objectifs à optimiser, selon le besoin.

7.2.5 Technique : minimisation de la complexité des modèles évolués par programmation génétique

Un critère visant la minimisation de la complexité des modèles évolués par PG est utilisé pour les trois applications en reconnaissance des formes. Pour l'évolution de mesures de voisinage par le PPV, le nombre de nœuds est minimisé selon le principe de la description de longueur minimale [126]. Pour l'ingénierie de représentations de l'écriture

manuscrite par PG, le nombre de caractéristiques extraites est directement minimisé afin de réduire l'effet de la malédiction de la dimensionnalité sur le classifieur. En ce qui a trait à la troisième application visant l'évolution d'arbres de décision flous, une mesure de complexité particulière est minimisée. Cette mesure est égale à la somme de la valeur de la complexité de tous les nœuds, chaque valeur étant proportionnelle au nombre de paramètres numériques contenus dans chaque nœud. Ces trois approches pour la minimisation de la complexité des modèles évolués par PG s'inscrivent dans la lignée des approches basées sur le rasoir d'Occam, où à valeur égale, les solution simples sont préférées aux solutions complexes. Bien que le rôle du rasoir d'Occam reste controversé pour l'apprentissage automatique [39], la préférence pour des modèles plus simples semble particulièrement intéressante pour l'utilisation de la PG. En effet, la surcharge (*bloat*) [8] est un phénomène courant en PG qui est caractérisé par une explosion de la dimension des solutions en cours d'évolution. La contribution du doctorat consiste à utiliser explicitement différentes mesures de complexité comme critère à minimiser dans une adéquation multi-objectif, dans le contexte de l'évolution de modèles de reconnaissance des formes par PG.

7.2.6 Technique : utilisation d'un ensemble de validation distinct pour la sélection de la meilleure solution d'une évolution

Différentes méthodes sont utilisées pour la validation et la sélection des meilleurs individus dans les différentes applications de la thèse. Pour la co-évolution de classifieurs PPV, les ensembles de données sont divisés en quatre sous-ensembles, avec un ensemble de prototypes, un ensemble pour l'évaluation de l'adéquation des individus, un ensemble pour la sélection des meilleures solutions de l'évolution et un ensemble pour le test final. Pour l'ingénierie évolutionnaire de représentations de caractères cursifs, les données sont déjà divisées en deux ensembles : l'ensemble d'entraînement et l'ensemble de test. L'ensemble d'entraînement est re-divisé pour l'application en trois sous-ensembles, soit l'ensemble pour l'entraînement du classifieur, l'ensemble pour la validation du classifieur et l'évaluation de la performance des représentations et l'ensemble pour choisir la meilleure représentation d'une évolution. Quant à l'évolution d'arbres de décision flous, on rapporte des résultats sur une validation croisée à dix recoupements (*10-folds validation* en anglais) des données. L'ensemble d'entraînement pour chaque évolution est séparé en deux sous-ensembles égaux, le premier étant utilisé pour entraîner les unités de classement et calculer l'adéquation des arbres, alors que la meilleure solution de l'évolution est sélectionnée à partir des performances sur le deuxième sous-ensemble de données.

Pour les trois applications à la reconnaissance des formes, la meilleure solution d'une évolution est sélectionnée à partir des performances sur un ensemble de données distinct des ensembles utilisés pour l'entraînement et l'évaluation de l'adéquation. Le rôle de cet ensemble de données correspond à celui de l'ensemble de validation des algorithmes d'apprentissage plus classiques, même s'il n'est pas utilisé pour interrompre les évolutions. Cet ensemble de validation n'est également pas employé pour guider la recherche évolutionnaire. Il devrait donc permettre de faire le choix de la meilleure solution de l'évolution par une validation de la capacité de généralisation. Il faut cependant noter que le nombre de solutions testées sur l'ensemble de validation est élevé, particulièrement pour les applications des chapitres 4 et 5, où le test est fait sur tous les individus de toutes les générations. Ce processus de validation risque donc de mener à la sélection de solutions qui ne sont que *par chance* adaptées aux données de l'ensemble. Il faut comprendre qu'étant donné le nombre élevé de solutions testées, la sélection de la meilleure solution d'une évolution peut être assimilée à une recherche aléatoire biaisée par les données de l'ensemble de validation. C'est pourquoi la taille de l'ensemble de validation doit être relativement grande pour fournir un échantillonnage permettant d'obtenir un bon estimé de la capacité de généralisation des solutions.

7.2.7 Technique : primitives de programmation génétique avec valeurs générées aléatoirement

Les trois applications basées sur la PG comportent des primitives avec arguments qui incluent des valeurs générées aléatoirement. Bien que conceptuellement très similaires aux constantes éphémères générées aléatoirement de Koza [76], nous n'avons pas connaissance de l'utilisation de telles primitives dans la littérature. Pourtant, ce type de primitives peut être très intéressant pour déterminer les paramètres des fonctions qui ne nécessitent pas de grande précision. Dans plusieurs situations, la technique proposée se compare avantageusement à d'autres techniques telles que la PG fortement typée (*Strongly Typed GP*) [99] ou l'insertion de primitives avec toutes les combinaisons de valeurs de paramètres possibles dans l'ensemble de primitives de l'évolution.

7.2.8 Technique : stratégies d'évolutions multi-objectif avec NSGA-II

Dans le chapitre 6 sur la conception automatisée de systèmes de lentilles, un nouvel algorithme est proposé pour effectuer de l'optimisation à plusieurs objectifs avec les SA-

ES. La méthode consiste simplement à remplacer la stratégie de remplacement $(\mu + \lambda)$ par l'algorithme de sélection multi-objectif NSGA-II [37]. En effet, l'algorithme NSGA-II correspond en quelque sorte à la stratégie de remplacement $(\mu + \lambda)$ des SE en posant λ égal à μ et en utilisant une procédure de tri basée sur l'optimalité de Pareto et le nichage.

Il est également intéressant de noter que la méthode proposée de SA-ES anisotrope avec NSGA-II est implantée avec Open BEAGLE seulement par le remplacement de l'opérateur $(\mu+\lambda)$ par l'opérateur du NSGA-II de l'algorithme de SA-ES anisotrope. Ce remplacement est typique d'une framework de type *black box*, avec un simple échange du composant (μ, λ) par le composant NSGA-II dans la configuration XML, et l'utilisation d'une mesure d'adéquation à plusieurs objectifs dans l'opérateur d'évaluation de l'adéquation.

7.3 Contributions méthodologiques

Nous résumons ici une approche méthodologique pour la réalisation de systèmes intelligents. Cette approche repose sur cinq principes :

1. Utiliser des algorithmes et des représentations adaptés au problème ;
2. Développer des hybrides entre des AE et des heuristiques du domaine d'application ;
3. Optimiser avec plusieurs objectifs ;
4. Co-évoluer des composants et des cas de test ;
5. Utiliser des outils logiciels génériques d'AE.

En soi, ces principes ne sont pas nouveaux. Le développement d'hybrides entre des AE et des heuristiques, l'optimisation évolutionnaire multi-objectif ou la co-évolution sont des approches reconnues et couramment utilisées dans la pratique. La nouveauté réside plutôt dans la réunion de ces cinq principes en une seule approche méthodologique et dans la présentation de quatre applications concrètes pour en illustrer la pertinence.

L'idée d'utiliser des algorithmes et des représentations adaptés au problème est illustrée dans les applications de la thèse par l'utilisation de trois algorithmes et représentations différents. Pour la co-évolution de classifieurs PPV (chapitre 3), une sélection de prototypes et de cas de tests est faite par des AG à chaîne de bits. La PG, avec l'évolution de programmes représentés par des structures en arbre, permet l'évolution de modèles pour les applications en reconnaissance des formes. Pour la co-évolution de classifieurs

PPV, les programmes génétiques représentent des mesures de voisinage entre deux données adaptées au problème à résoudre. Dans le cas de l'évolution de représentations (chapitre 4), les arbres de PG sont à la fois un découpage hiérarchique de la représentation en régions d'intérêt ainsi qu'une configuration de différentes caractéristiques extraites de ces régions pour la reconnaissance. Pour ce qui est de l'évolution des arbres de décision flous (chapitre 5), la PG est utilisée pour faire évoluer directement la topologie des arbres représentant une combinaison de classifieurs. Finalement, la conception de systèmes de lentilles (chapitre 6) peut être exprimée comme un problème d'optimisation de fonctions avec paramètres à valeur réelle. Les SE auto-adaptatives (SA-ES) sont reconnues comme étant des AE très performants pour ce type de problème d'optimisation. Pour cette application, deux SE sont employées, les SA-ES anisotropes et les CMA-ES.

Le développement d'hybrides entre des AE et des heuristiques du domaine d'application est un principe relativement reconnu dans le domaine des AE, particulièrement dans le contexte de la résolution de problèmes d'optimisation combinatoire et de recherche opérationnelle. Dans le cadre des applications en reconnaissance des formes, ce principe est illustré par l'utilisation de méthodes de classement bien établies, soit la règle du PPV (chapitre 3), le perceptron multicouche (chapitre 4) et l'application de trois classifieurs simples de natures différentes (chapitre 5). Pour la conception de systèmes optiques, les SE sont hybridées à une méthode de minimisation par moindres carrés amortis d'un outil commercial de CAO d'optique.

L'optimisation évolutionnaire multi-objectif s'est développée grandement depuis le début des années 1990. Les algorithmes modernes basés sur l'optimalité de Pareto, l'élitisme et le nichage sont relativement efficaces pour résoudre des problèmes difficiles à plusieurs objectifs. Chacune des applications de la thèse comporte des optimisations évolutionnaires multi-objectif. Pour les applications en reconnaissance des formes, les objectifs consistent d'une part à minimiser l'erreur sur le classement, soit l'erreur sur le taux de classement ou l'erreur quadratique moyenne sur les marges flous de classement, et d'autre part à minimiser la complexité des solutions, soit le nombre de prototypes sélectionnés, la longueur des mesures de voisinage, le nombre de caractéristiques ou une mesure particulière de complexité d'arbres de décision. Pour l'application en conception optique, l'expérimentation consiste à maximiser la qualité des images générées par les systèmes de lentilles tout en minimisant leur coût relatif.

La co-évolution peut être intéressante pour une résolution simultanée de plusieurs sous-problèmes d'une application ou pour une recherche dynamique des cas de tests. Dans la thèse, la co-évolution est utilisée uniquement pour l'évolution de classifieurs PPV, les autres applications n'étant pas propices à l'utilisation de cette technique.

L'évolution de classifieurs PPV illustre cependant clairement ces deux façons d'utiliser la co-évolution, avec la co-évolution coopérative de deux sous-problèmes, soit la sélection de prototypes et la construction d'une mesure de voisinage, et la co-évolution compétitive des cas de tests. Une autre façon intéressante d'utiliser la co-évolution serait de conjuguer des applications, par exemple l'ingénierie de représentation du chapitre 4 et l'évolution d'arbres de décision flous du chapitre 5.

Les principes de l'approche méthodologique proposée impliquent un développement logiciel avec des outils flexibles permettant des représentations et des algorithmes divers qui peuvent être adaptés au besoin. Les quatre applications de la thèse sont développées avec l'outil générique d'AE Open BEAGLE, illustrant par des cas concrets l'utilité et la généricité de cet outil.

7.4 Perspectives

Le projet de doctorat est structuré selon trois axes principaux. Le premier axe porte sur l'approche méthodologique proposée pour la réalisation de systèmes intelligents basés sur les AE. Le deuxième axe porte sur les avantages d'utiliser un outil logiciel générique d'AE. Ce deuxième axe est également présent dans les applications, l'outil logiciel générique Open BEAGLE ayant servi à l'implantation de toutes les expérimentations. Finalement, le troisième axe porte sur quatre applications indépendantes des AE à la reconnaissance des formes et la conception optique. Une analyse rapide de l'organisation de la thèse laisse voir que le chapitre 2 est orienté essentiellement selon le deuxième axe, alors que les autres chapitres du développement s'orientent plus particulièrement selon le troisième axe. Quant au premier axe, celui-ci est présent dans tout le projet, le chapitre 2 appuyant le principe de l'approche méthodologique sur l'utilisation d'outils logiciels génériques et les quatre chapitres suivants étant des exemples d'application de l'approche méthodologique.

Les travaux de doctorat présentés dans la thèse sont de nature exploratoire. Chacun des thèmes abordés pourrait être plus approfondi dans des projets subséquents. Par exemple, dans un contexte de reconnaissance des formes, il serait intéressant d'étudier plus amplement l'effet de la co-évolution compétitive de cas de tests sur la capacité de généralisation de classifieurs évolués. Dans le cadre plus précis de la reconnaissance de l'écriture manuscrite, il serait également intéressant de développer par PG des représentations spécifiques pour chaque type de caractères cursifs. Quant à la combinaison de classifieurs dans des arbres de décision flous, l'approche proposée semble très prometteuse pour la génération d'ensembles de classifieurs. À cet effet, il serait pertinent

d'utiliser d'autres unités de classement, possiblement plus complexes, et d'appliquer le système à des problèmes de classement difficiles. Finalement, l'approche évolutionnaire hybride pour la conception entièrement automatisée de systèmes de lentilles donne des résultats prometteurs. La suite logique de cette application serait de développer une extension d'un outil de CAO d'optique existant, afin d'y implanter l'algorithme évolutionnaire hybride proposé. En un premier temps, cette extension pourrait permettre le raffinement de la méthode par son application à de nombreux problèmes réels. Il serait alors propice de tester quelques règles de sens commun de conception optique. À moyen terme, on pourrait espérer que cette approche hybride soit acceptée comme un nouvel instrument de la « boîte à outil » des concepteurs optiques.

D'autres pistes de recherches seront très probablement explorées à plus court terme. Entre autre, nous envisageons d'étudier la construction de caractéristiques, abordée dans le contexte particulier de la reconnaissance de l'écriture cursive au chapitre 4, dans un contexte plus général de reconnaissance et d'apprentissage automatique. Dans un autre registre, les *support vector machines* (SVM) [33] forment une nouvelle famille de classifieurs qui suscitent un intérêt marqué en apprentissage automatique depuis quelques années. La performance de ces classifieurs dépend en partie du choix de la fonction noyau. À cette fin, on envisage d'utiliser la PG pour l'évolution des noyaux de SVM adaptés aux données. Également, la plupart des SVM sont des classifieurs dichotomiques, c'est-à-dire qu'ils ne permettent qu'une discrimination de données étiquetées en deux classes. En ce sens, on pourrait très bien utiliser une co-évolution coopérative de noyaux de classifieurs SVM, avec un nombre d'espèces égal au nombre de classes. Ces trois projets de recherche constituent des avenues que nous prévoyons explorer dans un avenir rapproché.

Annexe A

Liste des publications

La thèse a mené à la production de nombreux articles, soit soumis à des revues scientifiques au cours de la dernière année, soit publiés dans des actes de conférences.

Article de revue scientifique en préparation

- Christian Gagné, Simon Thibault, Julie Beaulieu et Marc Parizeau. Human-Competitive Lens System Design with Evolutionary Algorithms, 2005.

Articles soumis à des revues scientifiques

- Christian Gagné, et Marc Parizeau. Genetic Engineering of Hierarchical Fuzzy Regional Representations for Handwritten Character Recognition. Soumis à International Journal on Document Analysis and Recognition, 2005.
- Marc Dubreuil, Christian Gagné, et Marc Parizeau. Analysis of a Master-Slave Architecture for Distributed Evolutionary Computations. Soumis à IEEE Transactions on Systems, Man, and Cybernetics B, 2005.
- Christian Gagné, et Marc Parizeau. Genericity in Evolutionary Computation Software Tools : Principles and Case-Study. Soumis à International Journal on Artificial Intelligence Tools, 2005.
- Christian Gagné, et Marc Parizeau. Co-evolution of Nearest Neighbor Classifiers Soumis à Pattern Recognition, 2004.

Articles publiés dans des actes de conférences

- Christian Gagné, Marc Parizeau, et Marc Dubreuil. The master-slave architecture for evolutionary computations revisited. Dans *Genetic and Evolutionary Computations COnference (GECCO) 2003*, pages 1578–1579, Chicago, IL, USA, 2003.
- Christian Gagné, Marc Parizeau, et Marc Dubreuil. Distributed BEAGLE : an environment for parallel and distributed evolutionary computations. Dans *Proc. of the 17th Annual International Symposium on High Performance Computing Systems and Applications (HPCS)*, pages 201–208, Sherbrooke, QC, Canada, 2003.
- Christian Gagné, et Marc Parizeau. Open BEAGLE : a new C++ evolutionary computation framework. Dans *Genetic and Evolutionary Computations COnference (GECCO) 2002*, pages 888, New York, NY, USA, 2002.
- Julie Beaulieu, Christian Gagné, et Marc Parizeau. Lens system design and re-engineering with evolutionary algorithms. Dans *Genetic and Evolutionary Computations COnference (GECCO) 2002*, pages 155–162, New York, NY, USA, 2002.
- Alexandre Lemieux, Christian Gagné, et Marc Parizeau. Genetical engineering of handwriting representations. Dans *Proc. of the International Workshop on Frontiers in Handwriting Recognition (IWFHR)*, pages 145–150, Niagara-on-the-Lake, ON, Canada, 2002.
- Marc Parizeau, Alexandre Lemieux, et Christian Gagné. Character recognition experiments using Unipen data. Dans *Proc. of the International Conference on Document Analysis and Recognition (ICDAR)*, pages 481–485, Seattle, WA, USA, 2001.

Bibliographie

[1] David Andre et John R. Koza. Parallel genetic programming : A scalable implementation using the transputer network architecture. Dans Peter J. Angeline et Kenneth E. Kinnear, Jr., éditeurs, *Advances in Genetic Programming 2*, chapitre 16, pages 317–338. MIT Press, Cambridge, MA, USA, 1996.

[2] Anne Auger. *Contributions théoriques et numériques à l'optimisation continue par algorithmes évolutionnaires*. Thèse de doctorat, Université Paris 6, France, décembre 2004.

[3] Thomas Bäck, David B. Fogel, et Zbigniew Michalewicz, éditeurs. *Evolutionary Computation 1 : Basic Algorithms and Operators*. Institute of Physics Publishing, Bristol, UK, 2000.

[4] Thomas Bäck, David B. Fogel, et Zbigniew Michalewicz, éditeurs. *Evolutionary Computation 2 : Advanced Algorithms and Operators*. Institute of Physics Publishing, Bristol, UK, 2000.

[5] Thomas Bäck, Ulrich Hammel, et Hans-Paul Schwefel. Evolutionary computation : comments on the history and current state. *IEEE transactions on Evolutionary Computation*, 1(1) :3–17, 1997.

[6] J. Mark Baldwin. A new factor in evolution. *The American Naturalist*, 30 :441–451, 1896.

[7] Saswatee Banerjee et Lakshminarayan Hazra. Experiments with a genetic algorithm for structural design of cemented doublets with prespecified aberration targets. *Applied Optics*, 40(34) :6265–6273, décembre 2001.

[8] Wolfgang Banzhaf et William B. Langdon. Some considerations on the reason for bloat. *Genetic Programming and Evolvable Machines*, 3(1) :81–91, mar 2002.

[9] Wolfgang Banzhaf, Peter Nordin, Robert E. Keller, et Frank D. Francone. *Genetic Programming – An Introduction ; On the Automatic Evolution of Computer Programs and its Applications*. Morgan Kaufmann, dpunkt.verlag, 1998.

[10] Julie Beaulieu, Christian Gagné, et Marc Parizeau. Lens system design and re-engineering with evolutionary algorithms. Dans *Genetic and Evolutio-*

nary Computations COnference (GECCO) 2002, pages 155–162, New York, NY, USA, 2002.

[11] Ellis Betensky. Postmodern lens design. *Optical Engineering*, 32(8) :1750–1756, août 1993.

[12] Hans-Georg Beyer et Hans-Paul Schwefel. Evolution strategies : a comprehensive introduction. *Natural Computing*, 1(1) :3–52, 2002.

[13] Catherine L. Blake et Christopher J. Merz. UCI repository of machine learning databases, 1998.

[14] Alessandro Bollini et Marco Piastra. Distributed and persistent evolutionary algorithms : a design pattern. Dans *Proceedings of EuroGP'99*, volume 1598 de *LNCS*, pages 173–183, Goteborg, Sweden, 1999. Springer-Verlag.

[15] Martijn C. J. Bot. Feature extraction for the k-nearest neighbour classifier with genetic programming. Dans *Genetic Programming, Proceedings of EuroGP'2001*, volume 2038 de *LNCS*, pages 256–267. Springer-Verlag, 2001.

[16] Tim Bray, Jean Paoli, et C. M. Sperberg-McQueen. Extensible Markup Language (XML) 1.0 - W3C recommendation 10-february-1998. Vu le 29 mars 2005 à l'adresse `http://www.w3.org/TR/1998/REC-xml-19980210`, 1998.

[17] Henry Brighton et Chris Mellish. Advances in instance selection for instance-based learning algorithms. *Data mining and knowledge discovery*, 6(2) :153–172, 2002.

[18] Gavin Brown, Jeremy Wyatt, Rachel Harris, et Xin Yao. Diversity creation methods : a survey and categorisation. *Information Fusion*, 6(1) :5–20, 2005.

[19] Mary Campione et Kathy Walrath. *The Java Tutorial*. Addison-Wesley, Reading, MA, USA, 2 édition, 1998.

[20] José Ramòn Cano, Francisco Herrera, et Manuel Lozano. Using evolutionary algorithms as instance selection for data reduction in KDD : an experimental study. *IEEE Transactions on Evolutionary Computation*, 7(6) :561–575, décembre 2003.

[21] Erick Cantú-Paz. *Efficient and accurate parallel genetic algorithms*. Kluwer Academic Publishers, Boston, MA, USA, 2000.

[22] Robin L. P. Chang et Theodosios Pavlidis. Fuzzy decision tree algorithms. *IEEE Transactions on Systems, Man, and Cybernetics*, 7(1) :28–35, jan 1977.

[23] Sanghamitra Chatterjee et Lakshminarayan Hazra. Structural design of cemented triplets by genetic algorithm. *Optical Engineering*, 43(2) :432–440, février 2004.

[24] Xiaogang Chen et Kimiaki Yamamoto. An experiment in genetic optimization in lens design. *Journal of Modern Optics*, 44(9) :1693–1702, 1997.

[25] James Clark. XSL transformations (XSLT) 1.0 - W3C recommendation 16-november-1999. Vu le 29 mars 2005 à l'adresse `http://www.w3.org/TR/1999/REC-xslt-19991116`, 1999.

[26] Carlos A. Coello Coello, David A. Van Veldhuizen, et Gary B. Lamont. *Evolutionary Algorithms for Solving Multi-Objective Problems*. Kluwer Academic Publishers, 2002.

[27] Pierre Collet. EASEA : EAsy Specification of Evolutionary Algorithms. Vu le 29 mars 2005 à l'adresse `http://fractales.inria.fr/evo-lab/EVO-easea-engl.html`, 2001.

[28] Pierre Collet, Evelyune Lutton, Marc Schoenauer, et Jean Louchet. Take it EASEA. Dans *Parallel Problem Solving from Nature - PPSN VI 6th International Conference*, volume 1917 de *LNCS*, Paris, France, septembre 16-20 2000. Springer-Verlag.

[29] John Cona. Developing a genetic programming system. *AI Expert*, pages 20–29, février 1995.

[30] Oscar Cordòn, Fernando Gomide, Francisco Herrera, Frank Hoffmann, et Luis Magdalena. Ten years of genetic fuzzy systems : current framework and new trends. *Fuzzy Sets and Systems*, 141 :5–31, 2004.

[31] Thomas M. Cover et Peter E. Hart. Estimation by the nearest neighbor rule. *IEEE Transactions on Information Theory*, 14(1) :50–55, 1968.

[32] B.G.W. Craenen et A.E. Eiben. Computational intelligence. Dans *Encyclopedia of Life Support Sciences*. EOLSS Co. Ltd., 2002.

[33] Nello Cristianini et John Shawe-Taylor. *An introduction to support vector machines*. Cambridge University Press, Cambridge, UK, 2000.

[34] Jason M. Daida, Adam M. Hilss, David J. Ward, et Stephen L. Long. Visualizing tree structures in genetic programming. Dans *Genetic and Evolutionary Computation – GECCO-2003*, volume 2724 de *LNCS*, pages 1652–1664, Chicago, 12-16 juillet 2003. Springer-Verlag.

[35] Edwin de Jong et Jordan Pollack. Ideal evaluation from coevolution. *Evolutionary Computation*, 12(2) :159–192, 2004.

[36] Edwin D. de Jong, Richard A. Watson, et Jordan B. Pollack. Reducing bloat and promoting diversity using multi-objective methods. Dans *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2001)*, pages 11–18, San Francisco, California, USA, 7-11 juillet 2001. Morgan Kaufmann.

[37] Kalyanmoy Deb, Amrit Pratap, Sameer Agarwal, et T. Meyarivan. A Fast and Elitist Multiobjective Genetic Algorithm : NSGA–II. *IEEE Transactions on Evolutionary Computation*, 6(2) :182–197, apr 2002.

[38] Gülsen Demiröz et H. Altay Güvenir. Genetic algorithms to learn feature weights for the nearest neighbor algorithm. Dans *Proceedings of the 6th Belgian-Dutch Conference on Machine Learning (BENELEARN96)*, pages 117–126, 1996.

[39] Pedro Domingos. The role of occam's razor in knowledge discovery. *Data Mining and Knowledge Discovery*, 3 :409–425, 1999.

[40] Marc Dubreuil, Christian Gagné, et Marc Parizeau. Analysis of a master-slave architecture for distributed evolutionary computations. Soumis à IEEE Transactions on Systems, Man, and Cybernetics B, 2005.

[41] Richard O. Duda, Peter E. Hart, et David G. Stork. *Pattern Classification*. John Wiley & Sons, Inc., New York, second édition, 2001.

[42] Joseph C. Dunn. A fuzzy relative of the ISODATA process and its use in detecting compact well-separated clusters. *Journal of Cybernetics*, 3(3) :32–57, 1973.

[43] Jeroen Eggermont. Evolving fuzzy decision trees withgenetic programming and clustering. Dans *Proceedings of the Fifth European Conference on Genetic Programming (EuroGP-2002)*, volume 2278 de *LNCS*, pages 71–82, Kinsale, Ireland, 2002. Springer Verlag.

[44] Mark Erickson, Alex Mayer, et Jeffrey Horn. The niched pareto genetic algorithm 2 applied to the design of groundwater remediation systems. Dans *Proceedings of the First International Conference on Evolutionary Multi-Criterion Optimization (EMO 2001)*, pages 681–695, 2001.

[45] José L. Ribeiro Filho, Philip C. Treleaven, et Cesare Alippi. Genetic algorithm programming environments. *IEEE Computer*, 27(6) :28–43, 1994.

[46] Lawrence J. Fogel, A. J. Owens, et M. J. Walsh. *Artificial Intelligence through Simulated Evolution*. John Wiley & Sons, New York, 1966.

[47] Gregory W. Forbes et Andrew E. W. Jones. Towards global optimization with adaptive simulated annealing. Dans *Proceedings of International Lens Design Conference 1990 ILDC'90*, volume 1354, pages 144–153. Proc. of SPIE, 1990.

[48] Richard Forsyth. BEAGLE A Darwinian approach to pattern recognition. *Kybernetes*, 10 :159–166, 1981.

[49] James A. Foster. Evolutionary computation. *Nature Genetics Reviews*, 2(6) :428–436, juin 2001.

[50] Christian Gagné et Marc Parizeau. Open BEAGLE : A new versatile C++ framework for evolutionary computation. Dans *Genetic and Evolutionary Computations COnference (GECCO) 2002, Late-Breaking Papers*, New York, NY, USA, 2002.

[51] Christian Gagné, Marc Parizeau, et Marc Dubreuil. Distributed BEAGLE : An environment for parallel and distributed evolutionary computations. Dans

Proc. of the 17th Annual International Symposium on High Performance Computing Systems and Applications (HPCS) 2003, May 11-14 2003.

[52] Erich Gamma, Richard Helm, Ralph E. Johnson, et John M. Vlissides. *Design Patterns : Elements of Reusable Object-Oriented Software*. Addison-Wesley, Reading, MA, USA, 1994.

[53] Isabelle Guyon, Lambert Schomaker, Réjean Plamondon, Mark Liberman, et Stan Janet. Unipen project of on-line data exchange and recognizer benchmarks. Dans *Proc. of the 14th Int. Conf. on Pattern Recognition (ICPR)*, pages 29–33, 1994.

[54] Martin T. Hagan, Howard B. Demuth, et Mark H. Beale. *Neural Network Design*. PWS Publishing Company, 1995.

[55] Nikolaus Hansen. The CMA evolution strategy : A tutorial. Vu le 29 mars 2005 à l'adresse `http://www.bionik.tu-berlin.de/user/niko/cmatutorial.pdf`, 2005.

[56] Nikolaus Hansen, Sibylle D. Mueller, et Petros Koumoutsakos. Reducing the time complexity of the derandomized evolution strategy with covariance matrix adaptation (CMA-ES). *Evolutionary Computation*, 11(1) :1–18, 2003.

[57] Nikolaus Hansen et Andreas Ostermeier. Completely derandomized self-adaptation in evolution strategies. *Evolutionary Computation*, 9(2) :159–195, 2001.

[58] Peter E. Hart. The condensed nearest neighbor rule. *IEEE Transactions on Information Theory*, 14 :515–516, 1968.

[59] Gregory K. Hearn. Pratical use of generalized simulated annealing optimization on microcomputers. Dans *Proceedings of International Lens Design Conference 1990 ILDC'90*, volume 1354, pages 186–193. Proc. of SPIE, 1990.

[60] Jean-François Hébert, Marc Parizeau, et Nadia Ghazzali. Une représentation floue régionale pour la reconnaissance en-ligne de chiffres manuscrits. Dans *Actes du 1er Colloque International Francophone sur l'Écrit et le Document (CIFED'98)*, Québec, Canada, 1998.

[61] Michi Henning et Steve Vinoski. *Advanced CORBA Programming with C++*. Addison-Wesley, Reading, MA, USA, 1999.

[62] W. Daniel Hillis. Co-evolving parasites improve simulated evolution as an optimization procedure. *Physica D*, 42 :228–234, 1990.

[63] Geoffrey E. Hinton et Stephen J. Nowlan. How learning can guide evolution. *Complex Systems*, 1(1) :495–502, juin 1987.

[64] Shinn-Ying Ho, Chia-Cheng Liu, et Soundy Liu. Design of an optimal nearest neighbor classifier using an intelligent genetic algorithm. *Pattern Recognition Letters*, 23(13) :1495–1503, nov 2002.

[65] Tin Kam Ho, Jonathan J. Hull, et Sargur N. Srihari. Decision combination in multiple classifier systems. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16(1) :66–75, 1994.

[66] John M. Holland. *Adaptation in Natural and Artificial Systems*. University of Michigan Press, Ann Arbor, MI, 1975.

[67] Hitoshi Iba, Hugo de Garis, et Taisuke Sato. Genetic programming using a minimum description length principle. Dans Kenneth E. Kinnear Jr., éditeur, *Advances in Genetic Programming*, Complex Adaptive Systems, pages 265–284, Cambridge, 1994. MIT Press.

[68] D. C. Ince, éditeur. *Collected Works of A. M. Turing : Mechanical Intelligence*. North-Holland, Amsterdam, The Netherlands, 1992.

[69] Hisao Ishibuchi, Tomoharu Nakashima, et Tadahiko Murata. Performance evaluation of fuzzy classifier systems for multidimensional pattern classification problems. *IEEE Transactions on Systems, Man and Cybernetics, Part B*, 29(5) :601–618, 1999.

[70] Masaki Isshiki. Global optimization with escape function. Dans *Proceedings of International Optical Design Conference 1998*, volume 3482 de *Proc. of SPIE*, pages 104–109, 1998.

[71] Anil K. Jain, Robert P.W. Duin, et Jianchang Mao. Statistical pattern recognition : A review. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(1) :4–37, 2000.

[72] Maarten Keijzer, Juan J. Merelo, Gustavo Romero, et Marc Schoenauer. Evolving objects : a general purpose evolutionary computation library. Dans *EA-01, Evolution Artificielle, 5th International Conference in Evolutionary Algorithms*, 2001.

[73] Mike J. Keith et Martin C. Martin. Genetic programming in C++ : Implementation issues. Dans Kenneth E. Kinnear, Jr., éditeur, *Advances in Genetic Programming*, chapitre 13, pages 285–310. MIT Press, 1994.

[74] Daijin Kim et Han-Pyul Lee. An optimal design of neuro-FLC by Lamarckian co-adaptation of learning and evolution. *Fuzzy Sets and Systems*, 118 :319–337, 2001.

[75] Josef Kittler, Mohamad Hatef, Robert P.W. Duin, et Jiri Matas. On combining classifiers. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(3) :226–239, 1998.

[76] John R. Koza. *Genetic Programming : On the Programming of Computers by Means of Natural Selection*. MIT Press, Cambridge, MA, USA, 1992.

[77] John R. Koza. *Genetic Programming II : Automatic Discovery of Reusable Programs*. MIT Press, Cambridge, MA, USA, 1994.

[78] John R. Koza, David Andre, Forrest H Bennett III, et Martin Keane. *Genetic Programming 3 : Darwinian Invention and Problem Solving*. Morgan Kaufman, 1999.

[79] John R. Koza, Forrest H. Bennett III, David Andre, et Martin A. Keane. Genetic programming : Turing's third way to achieve machine intelligence. Dans *Evolutionary Algorithms in Engineering and Computer Science*, pages 185–197, Jyväskylä, Finland, 1999. John Wiley & Sons.

[80] John R. Koza, Martin A. Keane, Matthew J. Streeter, William Mydlowec, Jessen Yu, et Guido Lanza. *Genetic Programming IV : Routine Human-Competitive Machine Intelligence*. Kluwer Academic Publishers, 2003.

[81] Natalio Krasnogor et Jim Smith. MAFRA : A java memetic algorithms framework. Dans *Genetic and Evolutionary Computation 2000 Workshops*, pages 125–131, Las Vegas, Nevada, USA, 2000.

[82] Krzysztof Krawiec. Genetic programming-based construction of features for machine learning and knowledge discovery tasks. *Genetic Programming and Evolvable Machines*, 3(4) :329–343, décembre 2002.

[83] Ludmila I. Kuncheva et Lakhmi C. Jain. Nearest neighbor classifier : Simultaneous editing and feature selection. *Pattern Recognition Letters*, 20(11-13) :1149–1156, nov 1999.

[84] Ludmila I. Kuncheva et Lakhmi C. Jain. Designing classifier fusion systems by genetic algorithms. *IEEE Transactions on Evolutionary Computation*, 4(4) :327–336, septembre 2000.

[85] Jean-Baptiste Lamarck. *Philosophie zoologique*. Dentu, Paris, 1809.

[86] William B. Langdon. Data structures and genetic programming. Dans Peter J. Angeline et K. E. Kinnear, Jr., éditeurs, *Advances in Genetic Programming 2*, chapitre 20, pages 395–414. MIT Press, Cambridge, MA, USA, 1996.

[87] William B. Langdon. *Data Structures and Genetic Programming : Genetic Programming + Data Structures = Automatic Programming !* Kluwer Academic Publishers, Boston, MA, USA, 1998.

[88] Tom Lenaerts et Bernard Manderick. Building a genetic programming framework : The added-value of design patterns. Dans *Proceedings of EuroGP'98*, pages 196–208, 1998.

[89] Xiaolin Li, Marc Parizeau, et Réjean Plamondon. Segmentation and reconstruction of on-line handwritten scripts. *Pattern Recognition*, 31(6) :675–684, 1998.

[90] Tjen-Sien Lim, Wei-Yin Loh, et Yu-Shan Shih. A comparison of prediction accuracy, complexity, and training time of thirty-three old and new classification algorithms. *Machine Learning*, 40(3) :203–228, 2000.

[91] Sean Luke, Liviu Panait, Gabriel Balan, Zbigniew Skolicki, Jeff Bassett, Robert Hubley, et Alexander Chircop. ECJ 12 : A java-based evolutionary computation and genetic programming research system. Vu le 29 mars 2005 à l'adresse `http://cs.gmu.edu/~eclab/projects/ecj`, 2005.

[92] Robert M. MacCallum. Introducing a perl genetic programming system – and can meta-evolution solve the bloat problem ? Dans *Proceedings of the Sixth European Conference on Genetic Programming (EuroGP-2003)*, volume 2610 de *LNCS*, pages 364–373, Essex, UK, 2003. Springer Verlag.

[93] Nicholas Freitag McPhee, Nicholas J. Hopper, et Mitchell L. Reierson. Sutherland : An extensible object-oriented software framework for evolutionary computation. Dans *Genetic Programming 1998 : Proceedings of the Third Annual Conference*, page 241, Madison, Wisconsin, USA, 1998. Morgan Kaufmann.

[94] Roberto R. F. Mendes, Fabricio B. Voznika, Alex A. Freitas, et Julio C. Nievola. Discovering fuzzy classification rules with genetic programming and co-evolution. Dans *5th European Conference on Principles and Practice of Knowledge Discovery in Databases (PKDD'01)*, volume 2168 de *LNAI*, pages 314–325, Freiburg, Germany, 3-7 septembre 2001.

[95] Juan J. Merelo, Maarten Keijzer, Marc Schoenauer, Jeroen Eggermont, Sébastien Cahon, et Olivier König. Evolving Objects : Evolutionary computation framework. Vu le 29 mars 2005 à l'adresse `http://eodev.sourceforge.net`, 2004.

[96] Zbigniew Michalewicz. *Genetic Algorithms + Data Structures = Evolution Programs*. Springer-Verlag, third édition, 1996.

[97] Melanie Mitchell. *An Introduction to Genetic Algorithms*. Complex Adaptive Systems. MIT-Press, Cambridge, 1996.

[98] Tom M. Mitchell. *Machine Learning*. McGraw-Hill, 1997.

[99] David J. Montana. Strongly typed genetic programming. *Evolutionary Computation*, 3(2) :199–230, 1995.

[100] Kenneth E. Moore. Algorithm for global optimization of optical systems based upon genetic competition. Dans *Proceedings on Optical Design and Analysis Software*, volume 3780 de *Proc. of SPIE*, pages 104–109, 1999.

[101] Pablo Moscato. On Evolution, Search, Optimization, Genetic Algorithms and Martial Arts : Towards Memetic Algorithms. Rapport technique Caltech Concurrent Computation Program, Report. 826, California Institute of Technology, Pasadena, California, USA, 1989.

[102] David R. Musser et Atul Saini. *STL Tutorial and Reference Guide : C++ Programming with the Standard Template Library*. Addison-Wesley, Reading, MA, USA, 1996.

[103] Yuichi Nagata. The lens design using the CMA-ES algorithm. Dans *Proceedings of the GECCO 2004*, volume 3103 de *LNCS*, pages 1189–1200, 2004.

[104] Luiz S. Oliveira, Robert Sabourin, Flávio Bortolozzi, et Ching Y. Suen. Feature selection using multi-objective genetic algorithms for handwritten digit recognition. Dans *International Conference on Pattern Recognition 2002*, Québec, QC, Canada, 2002.

[105] Isao Ono, Shigenobu Koboyashi, et Koji Yoshida. Global and multi-objective optimization for lens design by real-coded genetic algorithms. Dans *Proceedings of International Optical Design Conference 1998 IODC'98*, volume 3482, pages 110–121. Proc. of SPIE, 1998.

[106] Optical Research Associates. CODE V : Optical design, fabrication, and analysis software. Vu le 29 mars 2005 à l'adresse `http://www.opticalres.com/cv/cvprodds_f.html`, 2005.

[107] Donald C. O'Shea. *Elements of Modern Optical Design*. John Wiley and Sons, 1985.

[108] Donald C. O'Shea. The monochromatic quartet : A search for the global optimum. Dans *Proceedings of International Lens Design Conference 1990*, volume 1354 de *Proc. of SPIE*, pages 548–554, 1990.

[109] Ben Paechter, Thomas Bäck, Marc Schoenauer, Michèle Sebag, A. E. Eiben, Juan J. Merelo, et Terence C. Fogarty. A distributed resource evolutionary algorithm machine (DREAM). Dans *Proceedings of CEC'00*, pages 951–958. IEEE Press, 2000.

[110] Liviu Panait et Sean Luke. Methods for evolving robust programs. Dans *Genetic and Evolutionary Computation – GECCO-2003*, volume 2724 de *LNCS*, pages 1740–1751, Chicago, 12-16 juillet 2003. Springer-Verlag.

[111] Marc Parizeau, Alexandre Lemieux, et Christian Gagné. Character recognition experiments using unipen data. Dans *Proc. of 6th Int. Conf. on Document Analysis and Recognition (ICDAR)*, pages 481–485, 2001.

[112] Jaehwa Park, Venu Govindaraju, et Sargur N. Srihari. OCR in hierarchical feature space. *IEEE transactions on Pattern Analysis and Machine Intelligence*, 22(4) :400–407, 2000.

[113] Witold Pedrycz et Marek Reformat. Evolutionary fuzzy modeling. *IEEE transactions on Fuzzy Systems*, 11 :652–665, octobre 2003.

[114] Carlos Andrés Pena-Reyes et Moshe Sipper. Fuzzy CoCo : a cooperative-coevolutionary approach to fuzzy modeling. *IEEE transactions on Fuzzy Systems*, 9 :727–737, octobre 2001.

[115] Réjean Plamondon et Sargur N. Srihari. Online and off-line handwriting recognition : a comprehensive survey. *IEEE transactions on Pattern Analysis and Machine Intelligence*, 22(1) :63–84, 2000.

[116] Mitchell A. Potter et Kenneth A. De Jong. Cooperative coevolution : An architecture for evolving coadapted subcomponents. *Evolutionary Computation*, 8(1) :1–29, 2000.

[117] Lutz Prechelt. An empirical comparison of seven programming languages. *IEEE Computer*, 33(10) :23–29, 2000.

[118] Bill Punch et Douglas Zongker. lil-gp 1.1 beta. Vu le 29 mars 2005 à l'adresse `http://garage.cps.msu.edu/software/lil-gp/index.html`, 1998.

[119] J. Ross Quinlan. *C4.5 : Programs for Machine Learning*. Morgan Kaufmann, San Mateo, CA, 1993.

[120] Paulo V. W. Radtke, Tony Wong, et Robert Sabourin. A multi-objective memetic algorithm for intelligent feature extraction. Dans *Proceedings of the Third International Conference on Evolutionary Multi-Criterion Optimization (EMO 2005)*, Guanajuato, Mexico, 2005.

[121] Eugene H. Ratzlaff. A scanning n-tuple classifier for online recognition of handwritten digits. Dans *Proc. of 6th Int. Conf. on Document Analysis and Recognition (ICDAR)*, pages 18–22, 2001.

[122] Eugene H. Ratzlaff. Methods, report and survey for the comparison of diverse isolated character recognition results on the unipen database. Dans *Proc. of 7th Int. Conf. on Document Analysis and Recognition (ICDAR)*, pages 623–628, 2003.

[123] Michael L. Raymer, William F. Punch, Erik D. Goodman, Leslie A. Kuhn, et Anil K. Jain. Dimensionality reduction using genetic algorithms. *IEEE Transactions on Evolutionary Computation*, 4(2) :164, jul 2000.

[124] Eric S. Raymond. *The cathedral and the bazaar : musings on Linux and open source by an accidental revolutionary*. O'Reilly & Associates, Inc., revised édition, 2000.

[125] Ingo Rechenberg. *Evolutionsstrategie*. Friedrich Frommann Verlag (Günther Holzboog KG), Stuttgart, 1973.

[126] Jorma Rissanen. Modeling by shortest data description. *Automatica*, 14 :465–471, 1978.

[127] Don Roberts et Ralph E. Johnson. Evolving frameworks : A pattern language for developing object-oriented frameworks. Dans *Pattern Languages of Program Design 3*. Addison Wesley, 1997.

[128] Timothy J. Ross. *Fuzzy Logic with Engineering Applications*. John Wiley and Sons, 2 édition, 2004.

[129] Andreas Rummler et Gerd Scarbata. ealib – a java framework for implementations of evolutionary algorithms. Dans *Computational Intelligence. Theory and Applications (Fuzzy Days 2001)*, volume 2206 de *LNCS*, pages 92–102, Dortmund, Germany, 2001. Springer Verlag.

[130] Marco Russo. FuGeNeSys - A fuzzy genetic neural system for fuzzy modeling. *IEEE transactions on Fuzzy Systems*, 6(3) :373–388, août 1998.

[131] Ishwar K. Sethi et Jae H. Yoo. Design of multicategory multifeature split decision trees using perceptron learning. *Pattern Recognition*, 27(7) :939–947, 1994.

[132] Jamie Sherrah, Robert E. Bogner, et Abdesselam Bouzerdoum. Automatic selection of features for classification using genetic programming. Dans *Proceedings of the 4th Australian and New Zealand Intelligent Information Systems Conference (ANZIIS) 1996*, pages 284–287, 1996.

[133] Jamie Sherrah, Robert E. Bogner, et Abdesselam Bouzerdoum. The evolutionary pre-processor : Automatic feature extraction for supervised classification using genetic programming. Dans *Genetic Programming 1997 : Proceedings of the Second Annual Conference*, pages 304–312, Stanford University, CA, USA, 13-16 juillet 1997. Morgan Kaufmann.

[134] Sara Silva. GPLAB : A genetic programming toolbox for MATLAB. Vu le 29 mars 2005 à l'adresse `http://gplab.sourceforge.net`, 2004.

[135] Andy Singleton. Genetic programming with C++. *BYTE*, pages 171–176, février 1994.

[136] Terence Soule et James A. Foster. Effects of code growth and parsimony pressure on populations in genetic programming. *Evolutionary Computation*, 6(4) :293–309, Winter 1998.

[137] Doron Sturlesi et Donald C. O'Shea. Future of global optimization in optical design. Dans *Proceedings of International Lens Design Conference 1990 ILDC'90*, volume 1354, pages 54–68. Proc. of SPIE, 1990.

[138] K. C. Tan, Tong H. Lee, D. Khoo, et E. F. Khor. A multiobjective evolutionary algorithm toolbox for computer-aided multiobjective optimization. *IEEE Transactions on Systems, Man, and Cybernetics – Part B : Cybernetics*, 31(4) :537–556, août 2001.

[139] Astro Teller et Manuela Veloso. PADO : A new learning architecture for object recognition. Dans *Symbolic Visual Learning*, pages 81–116. Oxford University Press, 1996.

[140] Astro Teller et Manuela Veloso. PADO : A new learning architecture for object recognition. Dans Katsushi Ikeuchi et Manuela Veloso, éditeurs, *Symbolic Visual Learning*, pages 81–116. Oxford University Press, 1996.

[141] Ankur Teredesai et Venu Govindaraju. OCR in hierarchical feature space. *Pattern Recognition*, 38(4) :500–512, 2005.

[142] Ankur Teredesai, Jaehwa Park, et Venu Govindaraju. Active handwritten character recognition using genetic programming. Dans *Proceedings of EuroGP'01*, volume 2038 de *LNCS*, pages 371–379. Springer-Verlag, 2001.

[143] Zoltán Tóth et Gabriella Kókai. An experimental evaluation of the generic evolutionary algorithms programming library. Dans *FGML 2001, Treffen der Fachgruppe maschinelles Lernen der Gesellschaft fur Informatik*, Dortmund, Germany, 2001.

[144] Øivind Due Trier, Anil K. Jain, et Torfinn Taxt. Feature extraction methods for character recognition – a survey. *Pattern Recognition*, 29(4) :641–662, 1996.

[145] Alan M. Turing. Computing machinery and intelligence. *Mind*, 59(236) :433–460, 1950.

[146] Peter Turney. How to shift bias : Lessons from the Baldwin effect. *Evolutionary Computation*, 4(3) :271–295, 1997.

[147] Giorgio Valentini et Francesco Masulli. Ensembles of learning machines. Dans *13th Italian Workshop on Neural Nets (WIRN 2002)*, volume 2486 de *LNCS*, pages 3–20, Vietri sul Mare, Italy, 2002.

[148] Róbert Ványi. Object oriented design and implementation of a general evolutionary algorithm. Dans *Genetic and Evolutionary Computation – GECCO-2004*, volume 3103 de *LNCS*, pages 1275–1286, Seattle, 2004. Springer-Verlag.

[149] Darko Vasiljević. Optimization of the Cooke triplet with various evolution strategies and damped least squares. Dans *Proceedings of Optical Design and Analysis Software*, volume 3780, pages 207–215. Proc. of SPIE, 1999.

[150] Darko Vasiljević. *Classical and Evolutionary Algorithms in the Optimization of Optical Systems*. Kluwer Academic Publishers, 2002.

[151] Darko Vasiljević et Janez Golobić. Comparison of the classical dumped least squares and genetic algorithm in the optimization of the doublet. Dans *First Online Workshop on Soft Computing*, 1996.

[152] Christian Veenhuis, Katrin Franke, et Mario Köppen. A semantic model for evolutionary computation. Dans *6th International Conference on Soft Computing*, Fukuoka, Japan, 2000.

[153] M. Walk et J. Niklaus. Some remarks on computer-aided design of optical lens systems. *Journal of Optimization Theory and Applications*, 59(2) :173–181, novembre 1988.

[154] Matthew Wall. GAlib : A C++ library of genetic algorithm components. Vu le 29 mars 2005 à l'adresse `http://lancet.mit.edu/ga`, 2005.

[155] R. Paul Wiegand, William C. Liles, et Kenneth A. De Jong. An empirical analysis of collaboration methods in cooperative coevolutionary algorithms. Dans *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2001)*, pages 1235–1242, San Francisco, California, USA, 7-11 juillet 2001. Morgan Kaufmann.

[156] Dennis L. Wilson. Asymptotic properties of nearest neighbor rules using edited data. *IEEE Transactions on Systems, Man, and Cybernetics*, 2(3) :408–421, jul 1972.

[157] Garnett C. Wilson, A. McIntyre, et Malcolm I. Heywood. Resource review : Three open source systems for evolving programs – lilgp, ECJ and grammatical evolution. *Genetic Programming and Evolvable Machines*, 5(1) :103–105, mars 2004.

[158] David H. Wolpert et William G. Macready. No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation*, 1(1) :67–82, avril 1997.

[159] Jihoon Yang et Vasant Honavar. Feature subset selection using A genetic algorithm. *IEEE Intelligent Systems*, 13 :44–49, 1998.

[160] Tina Yu, Shu-Heng Chen, et Tzu-Wen Kuo. Discovering financial technical trading rules using genetic programming with lambda abstraction. Dans *Genetic Programming Theory and Practice II*, pages 11–30, Ann Arbor, MI, 2004.

Index

aberration, **135**, 136
  de Seidel, 136, 143
  distorsion, 136, 140
  sphérique, 136
adéquation, xviii, 5, 10, 11, 23, 25, 26, 34
AE, voir algorithme évolutionnaire
AG, voir algorithmes génétiques
aide à la décision, 3
algorithme évolutionnaire, xviii, xix, 2, 3, 5–11, 21
  distribué, 44, 45
algorithme mémétique, xviii, 10, 14
algorithmes génétiques, xviii, **5**, 13, 21, 38, 162
allocateur, 33
apprentissage automatique, xviii, **3**, 4, 5, 10, 15
approche développementale, 159
approche hybride, 14, 74, 142, 157
approche méthodologique, **13**, 100, 169
approche vorace, 108
arbre de décision, 3, 98–101, 121

bagging, 98
BEAGLE Visualizer, 46, 162
blur spot, **140**, 144
boosting, 98
breeder, 37

CAO, 132, 137
classifieur dichotomique, 105
CMA-ES, **134**, 138, 145, 151
co-évolution, xix, **10**, 15, 41, 68, 162
  compétitive, 10, **11**, 15, 68, 163
  coopérative, 10, **11**, 16, 68, 100, 163
coût relatif, 152
CODE V, 142, 144
compétitivité avec l'humain, 159
conception assistée par ordinateur, voir CAO
conception automatisée, 5
condensation de Hart, 56, 57, 60, 70, 110
constante éphémère générée aléatoirement, 65, 87, 116, 168
conteneur, 33
contexte, 34
CORBA, 32, 44
courbure, 134
Covariance Matrix Adaption Evolution Strategy, voir CMA-ES
critère d'arrêt, 5
critères de généricité, **25**, 28, 161
  adéquation générique, 26
  gestion des paramètres, 27
  modèle évolutionnaire générique, 26
  opération générique, 26
  représentation générique, 25
  sortie configurable, 27
croisement, xviii, 5, 7

désengagement, 11
dème, xix, 34, 44, 45
Damped Least Squares, voir DLS
data mining, voir forage de données
datum, 40
descente du gradient, 108
design pattern, xix, 23, 32, 35, 44
diaphragme, 135, **140**

Distributed BEAGLE, 42–45, 89, 162
DLS, 137, 142, 144
dominance, voir optimalité de Pareto
données UCI, 70, 120

EAML, 33
EASEA, 29
ECJ, 22, **28**
édition de Wilson, 56, 57, 60, 70
effet de Baldwin, 10
énergie encerclée, 147, 149
ensemble d'opérateurs
    bootstrap, 35, 37
    main-loop, 35, 37
ensemble de classifieurs, 98, 99
EO, **29**
équilibrage de charge, 44
évoluateur, 35
exploration versus exploitation, **9**, 14, 142
extraction de caractéristiques, 4, 14, 76, 77, 80, 93, 164

f : chiffre, 135, 139
feature extraction, voir extraction de caractéristiques
fonction de densité de probabilité, 7
fonctions définies automatiquement, 40
fondations orientées objet, 32–33
forage de données, 3
framework, xix, 22, 23, 27, 28, 161
    black box, 24, 25, 29, 31, 38
    white box, 24
framework générique, 32–38
frameworks spécialisées, 38–41
FuGeNeSys, 100

génotype, xix, 34
GAlib, 29
GECCO, 21
GPLAB, 30
Grammatical Evolution, 22

hall-of-fame, 34
hard computing, 3

ILDC, 139, 140, 144
individu, xix, 34
INO, 146
Institut national d'optique, voir INO
intelligence artificielle, xix, **2**
intelligence computationnelle, xix, **3**
International Lens Design Conference, voir ILDC
intransitivité, 11

Java, 29, 32

K-moyennes floues, 99, 110–113, 165

Lamarck, 10, 100, 128
lil-gp, 22, **30**
logger, 35
logique floue, **98**
longueur focale effective, 135, **139**
longueur minimale de description, 65

mémoire associative floue, 98
machine à états finis, 8
malédiction de la dimensionnalité, 84
marge floue, 116, 118, 165
mesure de voisinage, 50–52, 65, 68, 163
minimum description length, voir longueur minimale de description
mixture d'experts, xix, **98**, 101, 131
modèle maître-esclave, 44, 45
monochromatic quartet, **139**
mutation, xix, 5, 7, 8, 26

neurone ADALINE, 99, 105–108, 165
neurone perceptron, 101
Niched Pareto Genetic Algorithm 2, voir NPGA2
no free lunch, 74
nombre d'Abbe, 137, 147

Non-Dominated Sorting Genetic Algorithm 2, voir NSGA-II
NPGA2, 59
NSGA-II, **9**, 89, 122, 154, 169

OneMax, 41–42
opérateur, 35
opérateur de puissance nulle, 138
opération génétique, xix, 7, 8, 13, 23, 26, 44
Open BEAGLE, 31–41, 45, 46, 89, 144, 161
optimalité de Pareto, 8, 15, 169
optimisation, 5, 8
optimisation multi-objectif, **8**, 15, 59, 65, 152, 164, 166, 167
optique de Gauss, 135, 136
outil logiciel d'AE, 16, **21**, 22, 23, 27, 161

PE, voir programmation évolutionnaire
perceptron multicouche, 80, 120
perte du gradient, voir désengagement
PG, voir programmation génétique
phénotype, xx
plus proche voisin, 50–53, 55, 56, 63, 99, 110, 120, 162, 165, 166
PMC, voir perceptron multicouche
pointeur intelligent, 33
polymorphisme, xx
population, xx
PPV, voir plus proche voisin
primitive, 40, 63
  ensemble, 40, 116
  super-ensemble, 40
programmation évolutionnaire, xx, **8**, 21
programmation génétique, xx, **7**, 13, 38, 85, 116, 163, 164
programmation logique inductive, 3
prototypes, 51

réfraction, 134
  indice, 136, 147
  première loi, 135
réseau de neurones, xix, xx, 3, 121
randomizer, 35
rasoir d'Occam, 118, 167
recherche opérationnelle, 3
reconnaissance des formes, xx, **3**, 13, 14, 50, 162
recuit simulé, 137
registre, 35
représentation, 7, 13, 23, 25
représentation guidée par les données, 82, 85
représentation régionale floue, 77–82, 85, 164
RMS blur spot, voir *blur spot*

sélection de prototypes, 50, 51, 56, 57, 59, 60, 68, 162
sélection naturelle, xviii, xx, 5, 7, 8, 13, 15, 23, 44
SA-ES, **133**, 144, 149, 154, 169
score de classement, 102, 105
SE, voir stratégie d'évolution
segmentation, 4, 14, 76, 77, 164
Self-Adaptive Evolution Strategies, voir SA-ES
Snell-Descartes, voir réfraction (première loi)
soft computing, 3
SQL, 44
steady-state, 37, 38
STL, 32, 33
stratégie d'évolution, xxi, **7**, 13, 21, 38, 133–134
  auto-adaptative, voir SA-ES
stratégie d'appariement, 11
stratégie de remplacement, 37, 154
support vector machines, 131, 172
sur-spécialisation, 11
SVM, voir support vector machines

système de lentilles, **134**
système EFFECT, 99, 101–105
système flou, xix, xxi, 3, 77, 98, 100, 165
système intelligent, xxi, 2–5

TCP-IP, 44
tolérance, 152
transformation
    blanchissante, 53
    d'échelle, 52

UCI Machine Learning Repository, voir
        données UCI
Unipen, 80, 81

validation croisée à 10 recoupements, 120
variation génétique, xviii
vignetage, 140
vivarium, 34

XML, 33, 35, 44, 46
XSLT, 33