Intelligence artificielle - Planification et optimisation

GUIDE DE L'ÉTUDIANTE ET DE L'ÉTUDIANT
S7 – APP1

Module : Intelligence artificielle
Hiver 2026 – Semaines 1 à 4

Département de génie électrique et de génie informatique
Faculté de génie
Université de Sherbrooke

Historique des modifications

Date	Responsables	Description
2024-12-27	A. C. Therrien	Mises à jour. Amélioration des laboratoires.
2024-08-26	N. Zelovic	Ajout cadre d'utilisation IAG
2024-08-22	N. Zelovic	Mises à jour mineures
2023-12-29	A. C. Therrien	Mises à jour mineures
2023-08-22	A. C. Therrien	Changement de sigle
2022-08-22	A. C. Therrien	Création de l'APP

Auteur : Audrey Corbeil Therrien
Version : 1.6 (24 décembre 2025)

Ce document est réalisé avec l'aide de LaTeX et de la classe `gegi-app-guide`.

TABLE DES MATIÈRES

LISTE DES FIGURES

LISTE DES TABLEAUX

1 ÉNONCÉ DE LA PROBLÉMATIQUE

Code départ : python 3.12 (en ligne mercredi)

Jouer intelligemment

Vous êtes membre d'une équipe qui se prépare à la toute nouvelle compétition nationale AutoPlay, où les joueurs qui s'affrontent sont des joueurs artificiels programmés avec des techniques de l'intelligence artificielle. Chaque équipe ne peut soumettre qu'un seul joueur artificiel à la compétition.

Le jeu de cette première édition est un labyrinthe qu'il faut traverser pour gagner. Cependant ce labyrinthe contient aussi des obstacles, des jetons, des trésors, des portes et des ennemis. Des points sont attribués pour tous les jetons et trésors ramassés ainsi que les ennemis vaincus. En cas d'égalité, le temps requis et la longueur du parcours nécessaire pour compléter le labyrinthe déterminera l'équipe gagnante.

Une version simplifiée du jeu vous est fournie pour développer votre joueur. Dans sa version initiale, le jeu est contrôlé par le clavier. À l'initialisation du jeu, votre joueur aura accès à la carte du labyrinthe, mais pas à la position exacte des obstacles, jetons et trésors. Cependant, une fois que votre joueur est proche d'un jeton ou d'un obstacle, il peut le percevoir, tout comme les murs du labyrinthe.

stats du joueur et des ennemis.

De plus, il existe plusieurs types d'ennemis et ceux-ci requièrent différentes stratégies afin de les vaincre. Vous avez accès aux valeurs des attributs des ennemis et vous pouvez modifier les attributs de votre joueur. Toutefois, les effets de ces attributs ne sont pas définis et, après avoir essayé quelques parties, il semblerait que la valeur des attributs des ennemis change à chaque nouvelle partie, donc il faut reconfigurer le joueur selon une nouvelle collection d'ennemis à chaque ronde. Ce processus essais-erreurs prend beaucoup de temps à la main et devra faire partie du joueur artificiel pour la compétition.

Quelques membres de l'équipe tentent d'entraîner des réseaux de neurones pour le joueur, cependant la diversité des actions à prendre et la complexité de l'environnement rendent cette solution peu prometteuse. Votre coéquipier vous propose d'explorer d'autres techniques de l'intelligence artificielle.

réversible et puissant, mais fit pas avec le labyrinthe.

Beaucoup de ressources

Technique déterministique

La capacité de planifier une série d'actions pour atteindre un but est critique pour créer un joueur artificiel intelligent. Chacune des actions permet de changer d'état jusqu'à ce qu'on atteigne l'état visé. Pour trouver une série d'actions qui permet de passer de l'état initial à l'état final, il faut explorer l'espace d'état, défini par la logique du premier ordre (LPO),

Créer de manière organisée
- Limiter mon espace
- Calculer la meilleur possibilité

avec un algorithme de recherche adapté, sinon il risque d'avoir une explosion combinatoire des possibilités.

La LPO permet aussi de réaliser des systèmes experts dont la logique est définie par un ensemble de prédicats et de règles. Selon les informations fournies en entrée, un système expert est capable d'inférer une conclusion à partir de sa base de connaissances. Ce type de système a l'avantage d'avoir des règles explicites et faciles à comprendre pour les personnes opérant le système.

Plusieurs contrôleurs utilisent la logique floue pour opérer dans un environnement analogique et continu. Contrairement au monde booléen habituel de la programmation pour lequel un terme peut être vrai ou faux, la logique floue permet d'avoir des éléments qui sont en partie vrais et en partie faux. Cette approche permet de contrôler un système avec des entrées ambiguës ou bruyantes en utilisant des règles simples et très peu de calcul.

Enfin, les algorithmes génétiques proposent d'émuler les principes de l'évolution pour optimiser un système pour un objectif donné. Cette technique permet de sélectionner les configurations les plus adaptées, appelés individus, parmi une population donnée afin d'effectuer des croisements et des mutations pour améliorer l'adéquation des individus de génération en génération.

La compétition n'a aucun règlement limitant le joueur artificiel à une seule technique d'intelligence artificielle. Votre équipe souhaite donc combiner les techniques optimales afin d'obtenir le meilleur joueur artificiel et gagner la compétition !

Piste de solution

- Sortir du labyrinthe
- Récupérer les objets
- Élimination des ennemis
- Optimisation des attributs du joueur
- Optimisation de la longueur du parcours et tie breaker (temps)

Jeux

Interface qui remplace le clavier

Lire la doc

Porte à un énigme logique à résoudre

Comment faire l'architecture

Étapes :

1) Mettre en place les choix de conceptualisation et architecture
2) Implémentation
   1- Algorithme de recherche (sortir du labyrinthe)
   2- LPO (énigme porte)
   3- Logique flou pour les obstacles, ennemis (logique inversé)
   4- Algorithme génétique pour les attributs du joueur

*Quand tu vois le monstre, tu changes tes stats.

1 modules en proby ???

2 GUIDE DE LECTURE

2.1 Références essentielles

Deux références seront essentielles pour cette APP :

— Artificial Intelligence : A Guide to Intelligent Systems de Negnevitsky [1]– `IA-GUIDE`
— Artificial Intelligence : A Modern Approach de Russell et Norvig [2] – `IA-MODERN`

Des extraits du livre `IA-MODERN` sont disponibles sur le site web. Vous n'avez pas à acheter ce livre pour ce cours, cependant il s'agit d'une référence importante dans le domaine et couvre une grande variété de technique de l'IA. Il est disponible pour emprunt à la bibliothèque, ainsi qu'une version en français. Noté que la version en français n'aura pas les mêmes numéros de pages, mais les sections correspondent.

Des extraits du livre `IA-GUIDE` sont disponibles sur le site web. Vous n'avez pas à acheter ce livre pour ce cours, toutefois, il est disponible pour achat à la coopérative UdeS et est une référence très utile pour la conception de systèmes intelligents.

Pour la première pratique procédurale :

Agents intelligents

— `IA-MODERN` Chapitre 3 *Solving Problems by Searching* (prioriser la résolution par recherche et la sélection d'heuristique. Il n'est pas nécessaire de mémoriser tous les algorithmes de recherche, attarder vous seulement à profondeur d'abord, largeur d'abord, avare et A*)
— `IA-MODERN` Chapitre 7, *Logical Agents* sections 7.1 à 7.5.2 (Arrêtez à *A resolution algorithm* p. 227)
— `IA-MODERN` Chapitre 8 *First Order Logic*
— Suggéré : The Airline Industrys Problem with Absolutely Ancient IT. Identifiez les besoins de l'industrie d'aviation et les faiblesses des systèmes actuels. Quels systèmes font usage d'une forme d'IA ?
— Suggéré : Chapitre 11 de `IA-MODERN` (non disponible sur le site web - limitation du droit d'auteur)

Pour la deuxième pratique procédurale :

— `IA-MODERN` Chapitre 9, *Inference in First-Order Logic* jusqu'à la section 9.4 inclusivement.

Logique Floue

— `IA-GUIDE` Chapitre 4 *Fuzzy Expert Systems*

Pour la troisième pratique procédurale :

Algorithme génétique

— `IA-GUIDE` Chapitre 7 *Evolutionary Computation* 7.1 à 7.5

Systèmes intelligents

— `IA-GUIDE` Chapitre 9 *Knowledge Engineering* 9.1, 9.2, 9.3, 9.5.

2.2 Documents complémentaires

Les chapitres 1, 2 et 11 de `IA-MODERN` sont intéressants pour votre culture personnelle sur l'intelligence artificielle et pour vous donner un peu de perspective sur le sujet. La lecture de ces chapitre n'est pas nécessaire pour les évaluations ; la lecture de ces chapitres est donc uniquement suggérée.

— Canal YouTube Code Bullet. *Attention de ne pas y perdre 20 heures...*
— The Fastest Maze-Solving Competition On Earth.
— Site du logiciel : site web de SWI-Prolog
— `IA-GUIDE` Chapitre 3 *Frame-based expert systems*

3 LOGICIELS ET MATÉRIEL

Python

Python est utilisé dans le cadre de cette activité pédagogique et de son évaluation, autant pour la problématique que pour les examens. Il est installé dans les laboratoires du département et il est disponible gratuitement sur le web.

Les environnements de développement Python ont tous leurs particularités bien qu'il soit standardisé. Consultez toujours la documentation de l'environnement que vous utilisez afin de vous assurer de faire les choses correctement.

Les librairies nécessaires pour la problématique sont énumérés dans le fichier `README` et le fichier `requirements.txt`, disponibles avec le code de départ distribué sur le site web.

SWI-Prolog

SWI-Prolog est utilisé dans le cadre de cette activité pédagogique et de son évaluation, autant pour le rapport que pour les examens. Il est installé dans les laboratoires du département et il est disponible gratuitement sur le web. Pour l'installation, suivez les indications dans le fichier `README` distribué sur le site web.

Les environnements de développement Prolog ont tous leurs particularités, surtout en ce qui concerne le langage Prolog, bien qu'il soit standardisé. Consultez toujours la documentation de l'environnement que vous utilisez afin de vous assurer de faire les choses correctement.

4 SANTÉ ET SÉCURITÉ

Dispositions particulières

Aucune considération particulière pour cette unité.

Notez qu'un nouvelle directive entre en vigueur à l'automne 2023 pour les laboratoires au département GEGI. **Le port de lunettes de sécurité est OBLIGATOIRE** dans plusieurs laboratoires, indiqués par des affiches. Vous devez respecter cette consigne en tout temps, SAUF si la personne tutrice lève EXPLICITEMENT l'obligation pour la durée d'une activité pédagogique. En cas de non-respect de la consigne vous serez expulsé du laboratoire.

5 SOMMAIRE DES ACTIVITÉS

Semaine 1

— Première rencontre de tutorat
— Étude personnelle et exercices
— Formation à la pratique procédurale 1
— Formation à la pratique procédurale 2

Semaine 2

— Étude personnelle et exercices
— Formation à la pratique en laboratoire 1
— Révision formative du livrable
— Formation à la pratique procédurale 3

Semaine 3

— Formation à la pratique en laboratoire 2
— Consultations pour la résolution de la problématique

Semaine 4

— Validation : Défense orale
— Remise des livrables d'APP
— Évaluation formative théorique écrite et pratique
— Deuxième rencontre de tutorat

Semaine 5

— Consultation facultative
— Évaluation sommative théorique écrite et pratique

6 PRODUCTIONS À REMETTRE

- Les productions se font par **équipe de 3**, sauf lorsque indiqué autrement.
- L'identification des membres des équipes doit être faite sur la page web de l'unité avant 16h30, le **vendredi de la première semaine**.
- La date limite pour le dépôt électronique est 17h00, le jour de votre défense orale. Les retards seront pénalisés.
- Les productions soumises à l'évaluation doivent être originales pour chaque équipe, sinon l'évaluation sera pénalisée en cas de non-respect de cette consigne. L'apprentissage est collaboratif, mais les productions doivent être uniques.

6.1 Productions à remettre

Schéma-Bloc préliminaire du système

Révision formative du schéma-bloc de votre architecture. Le but de cette révision est de valider vos choix de conception et de vous aider à bien partir la problématique. Cette révision se fera en classe au début de la semaine 2. Le local est indiqué à l'horaire par une période ayant comme titre *APP1 : Révision formative du livrable*.

Défense orale

L'évaluation de votre solution à la problématique se fera via une défense orale. Vous devrez présenter votre solution et défendre vos choix de conception dans le but de démontrer votre compréhension des différentes techniques d'intelligence artificielle. Tous les membres de l'équipe doivent intervenir pendant cette défense.

Durée : 15 minutes
Heure et local : voir Site Web

Contenu :

- Introduction
- Schéma-bloc du système
  - Entrées-sorties
  - Format des données
  - Techniques IA utilisées
- Mise en oeuvre de chaque système IA. Pour chaque système :
  - Justification du choix
  - Sélection des règles/critères/stratégies
  - Évaluation des performances
- Démonstration

— Conclusions et recommandations
— Période de questions

Fichiers

Vous devez faire le dépôt électronique de tous les fichiers ayant servi à la résolution de la problématique dans un fichier d'archive *.zip* nommé selon les CIP des membres de l'équipe séparés par un tiret. Il y a donc un seul dépôt par équipe.

Votre archive doit être bien organisée et contenir ce qui suit.

— Le code ayant servi à la résolution, incluant la base de connaissance Prolog ;
— Votre support visuel ayant servi pendant la défense ;
— Un sous-répertoire nommé `Tests` qui contient votre plan de test/vérification, les tests principaux que vous avez faits pour valider votre code ainsi que leurs résultats.

6.2 Schéma de concept

Le schéma de concept est une production individuelle optionnelle en vue de la deuxième rencontre de tutorat. Le schéma de concepts à faire lors de l'étude personnelle cible la question suivante :

Qu'est-ce qu'un système intelligent ?
Qu'est-ce qu'un agent intelligent ?
Comment réaliser une planification automatique ?
Comment utiliser la logique floue ?
Comment utiliser les algorithmes génétique ?

7 ÉVALUATIONS

7.1 Rapport et livrables associés

L'évaluation de la défense orale et du dépôt électronique portera sur les compétences figurant dans la description des activités pédagogiques. Ces compétences ainsi que la pondération de chacune d'entre elles dans l'évaluation du rapport sont indiquées au tableau 7.1. L'évaluation est directement liée aux livrables demandés à la section 6.1 et le tableau 7.1 y réfère à l'aide d'une courte description.

Élément	GEI890-1	GEI890-2	GEI895-1	GEI895-2	GEI895-3
Schéma-bloc du système	10		20		
Mise en oeuvre - planification	10	10	5	15	
Mise en oeuvre - logique floue	10	10	5	15	
Mise en oeuvre - génétique	10	10	5	15	
Mise en oeuvre - système expert	10	10	5	15	
Démonstration					10
Conclusions et recommandations	10				10
Fonctionnement du prototype		10			10
Qualité de l'implémentation		10		15	
Plan de test et vérifications					10
Total	60	60	40	75	40

TABLEAU 7.1 Sommaire de l'évaluation de la défense orale

7.2 Évaluation sommative

L'évaluation sommative théorique est un examen écrit qui porte sur tous les éléments de compétences de l'unité. Une partie de la documentation, incluant les manuels des langages de programmation, sera mis à la disposition des personnes étudiantes sous format numérique.

L'évaluation sommative pratique porte sur l'utilisation d'une base connaissance Prolog et les outils de mise en oeuvre de la logique floue et d'algorithmes génétiques tels que vus pendant l'unité. Les questions seront répondues sur papier (pas de dépôt électronique).

7.3 Évaluation finale

L'évaluation finale théorique est un examen écrit qui porte sur tous les éléments de compétences de l'unité. Une partie de la documentation, incluant les manuels des langages de programmation, sera mis à la disposition des personnes étudiantes sous format numérique.

L'évaluation finale pratique porte sur l'utilisation d'une base connaissance Prolog et les outils de mise en oeuvre de la logique floue et d'algorithmes génétiques tels que vus pendant l'unité. Un dépôt électronique du code produit sera requis.

7.4 Utilisation de l'intelligence artificielle générative

Dans le cadre de cette unité d'APP, **l'usage des IAG est permis** pour votre apprentissage, l'aide à la rédaction et pour résoudre la problématique, conditionnellement à ce que tout élément produit par une IAG directement utilisée dans les productions remises à l'équipe professorale soit **cité et documenté dans les règles de l'art**.

L'utilisation des IAG lors des évaluations sommatives et finales n'est pas permise.

8 PRATIQUE PROCÉDURALE 1

But de l'activité

Comprendre :

- La recherche par espace d'état
- Les algorithmes de recherche non-informés et informés
- La logique du premier ordre

Afin de bien réussir cette activité, il vous est suggéré de la préparer en lisant la documentation suggérée à la section 2.1 pour ces sujets.

8.1 EXERCICES

P1.E1 Recherche par espace d'état

1. Dans le contexte d'une recherche par espace d'état, qu'est-ce qu'un état ?
2. Comment passe-t-on d'un état à un autre ?
3. Faites l'espace d'état d'un jeu des tours de Hanoï à trois disques. Dans ce jeu, l'objectif est de déplacer la tour de la tige de gauche vers la tige de droite. On ne peut déplacer qu'un disque à la fois, et un disque doit impérativement être sur un disque plus grand ou sur un emplacement vide.

   Ne pas dessiner les états redondants.

FIGURE 8.1 État de départ des tours de Hanoï

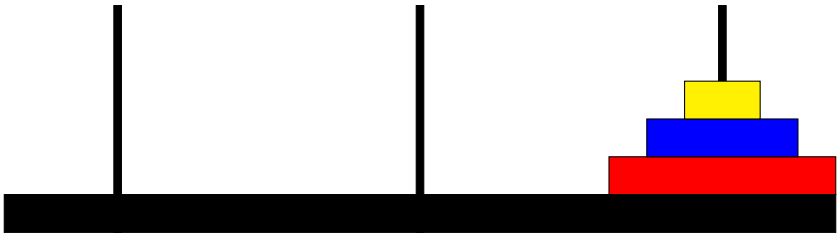

FIGURE 8.2 État final des tours de Hanoï

P1.E2 Manque d'information ?

1. Qu'est-ce qu'une action non-déterministe ? Qu'arrive-t-il à la recherche ?
2. Qu'est-ce qu'un environnement partiellement observable ? Qu'arrive-t-il à la recherche ?
3. Si l'environnement est inconnu, qu'est-ce qui arrive à la recherche ?

P1.E3 Le chemin le plus court

Trouver le chemin utilisé par les techniques suivantes pour aller d'Arad à Bucarest en utilisant les figures 3.1 (page 64) et 3.20 (page 89) de [2].

1. Recherche profondeur d'abord
2. Recherche avare (*greedy best-first search*)
3. Recherche A^*

P1.E4 Logique du premier ordre

1. Qu'est-ce que la logique du premier ordre ?
2. À quoi sert-elle ?
3. Est-ce qu'il y a un moyen rapide de discuter d'ensembles et de listes en logique du premier ordre ?

P1.E5 Proposition de logique du premier ordre

Traduisez les énoncés suivants en logique du premier ordre ou en texte.

1. Tous les chats sont des mammifères
2. Tous dans la classe sont futés
3. Spot a une soeur qui est un chat
4. Quelqu'un dans la classe est futé
5. $\exists x\ \forall y\ Aime(x, y)$ Note : c'est x qui aime y
6. $\forall y\ \exists x\ Aime(x, y)$ Note : c'est x qui aime y
7. Spot a au moins 2 soeurs

9 PRATIQUE PROCÉDURALE 2

But de l'activité

- Utiliser la logique du premier ordre pour représenter des algorithmes
- Utiliser un langage de programmation de logique formelle
- Comprendre l'inférence en logique du premier ordre
- Comprendre les problèmes de la planification dans un monde réel
- Se familiariser avec la planification par recherche de l'espace d'états
- Utiliser la logique floue

Afin de bien réussir cette activité, il vous est suggéré de la préparer. Vous pouvez la préparer en lisant la documentation qui est suggérée à la section 2.1 pour ces sujets.

9.1 EXERCICES

P2.E1 Logique du premier ordre et pseudocode

Exprimez cet algorithme en logique du premier ordre.

PROCÉDURE BonZoo
    Trouvé := faux
    Animal := premier animal de la liste des animaux du zoo
    TANT QUE trouvé est faux et animal est valide
        SI animal est un éléphant **ALORS**
            Trouvé := vrai
        SINON
            Animal := l'animal suivant de la liste des animaux du zoo
    SI trouvé est vrai **ALORS**
        BonZoo := vrai
    SINON
        BonZoo := faux

P2.E2 Logique du premier ordre et Prolog

Exprimez ce code Prolog en une seule phrase de logique du premier ordre. Ce code vérifie, s'il y a un passage valide d'une case vers une autre. Vous pouvez définir une nouvelle relation.

```
passage(X1, Y1, X2, Y1) :- X2 is X1+1, libre(X2,Y1).
passage(X1, Y1, X2, Y1) :- X2 is X1-1, libre(X2,Y1).
passage(X1, Y1, X1, Y2) :- Y2 is Y1+1, libre(X1,Y2).
passage(X1, Y1, X1, Y2) :- Y2 is Y1-1, libre(X1,Y2).
passage(X1, Y1, X2, Y2) :- X2 is X1+1, Y2 is Y1+1, libre(X2,Y2).
passage(X1, Y1, X2, Y2) :- X2 is X1+1, Y2 is Y1-1, libre(X2,Y2).
passage(X1, Y1, X2, Y2) :- X2 is X1-1, Y2 is Y1+1, libre(X2,Y2).
passage(X1, Y1, X2, Y2) :- X2 is X1-1, Y2 is Y1-1, libre(X2,Y2).
```

P2.E3 Inférence en logique du premier ordre

1. Qu'est-ce que l'inférence ?
2. Quelles sont les règles d'inférence pour la logique propositionnelle ? Pour les quantificateurs ?
3. Démontrez en utilisant l'algorithme de chaînage avant que Julie est la représentante de Jean.

   (1) $\forall x\ BilanSuperieur(x, 100000) \Rightarrow Client(x, Or)$
   (2) $\forall x\ ClientRegulier(x) \wedge BilanSuperieur(x, 100000) \Rightarrow EnvoyerPublicite(x)$
   (3) $\forall x\ ClientRegulier(x) \wedge Client(x, Or) \Rightarrow Representant(Julie, x)$
   (4) $BilanSuperieur(Jean, 100000)$
   (5) $ClientRegulier(Jean)$

4. Refaites le problème précédent, mais en utilisant le chainage arrière.

P2.E4 Planification

1. Qu'est-ce que la planification ?
2. Quels sont les problèmes liés à la planification ?
3. Qu'est-ce que la planification par recherche de l'espace d'états ?
4. Qu'est-ce que la planification hiérarchique et quand est-elle utile ?
5. Comment tenir compte du temps, d'une cédule et des ressources dans la planification ?

P2.E5 Logique floue

1. Comment déterminez-vous l'angle de rotation du volant pour effectuer un virage en voiture ?
2. Qu'est-ce qu'une règle floue ?
3. Comment concevoir un système intelligent à partir de logique floue ?

P2.E6 Contrôle flou d'un changeur d'air

Considérons un système d'échangeur d'air pour la salle de bain. Le système prend en entrée une lecture de température et une lecture d'humidité de la pièce, et doit contrôler la vitesse de fonctionnement d'un ventilateur au plafond. La température peut être lue sur une plage de 40^oF à 120^oF avec une précision de 2^oF. Les lectures d'humidité se font sur la plage 0% à 100%, par incrément de 2%. La vitesse du ventilateur est exprimée en RPM de 0 à 1000, avec une précision de 20 RPM.

À partir des fonctions d'appartenance et de la FAM (*Fuzzy Associative Memory*, définition pp. 118-119 de `IA-GUIDE` données respectivement aux figures 9.1 et 9.2 :

1. Déterminer la vitesse retournée par le système lorsque la température est de 70^oF et l'humidité de 60% si l'on choisit paramètres suivants :
   — l'opérateur ET est utiliser pour combiner les prémisses d'une même règle ;
   — l'opérateur de conjonction (ET) pour les prémisses est le minimum de l'intersection des ensembles ;
   — l'implication sera saturée (*clipped*) ;
   — le choix des règles pour une série de règles ayant une même conséquence se fait par l'opérateur maximum ;
   — la défuzzification se fait par la méthode du centre de gravité (*Center of Gravity == COG*) ;

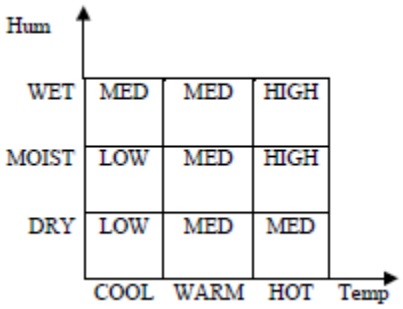

FIGURE 9.1 Règles exprimées et résumées sous forme d'une figure 2D.

2. Dans les mêmes conditions, dessiner la surface utilisée pour la défuzzification si l'opérateur de conjonction pour les prémisses des règles est le minimum et l'implication de la règle est la mise à l'échelle (*scaled*).
3. Dans les mêmes conditions, dessiner la surface utilisée pour la défuzzification si l'opérateur de conjonction pour les antécédents des règles est le produit et l'implication est la saturation (*clipped*). La sélection entre les règles partageant la même conséquence est faite par le maximum.
4. Dans les mêmes conditions, dessiner la surface utilisée pour la défuzzification si l'opérateur de conjonction pour les antécédents est le produit et l'implication est la mise à l'échelle (*scaled*).

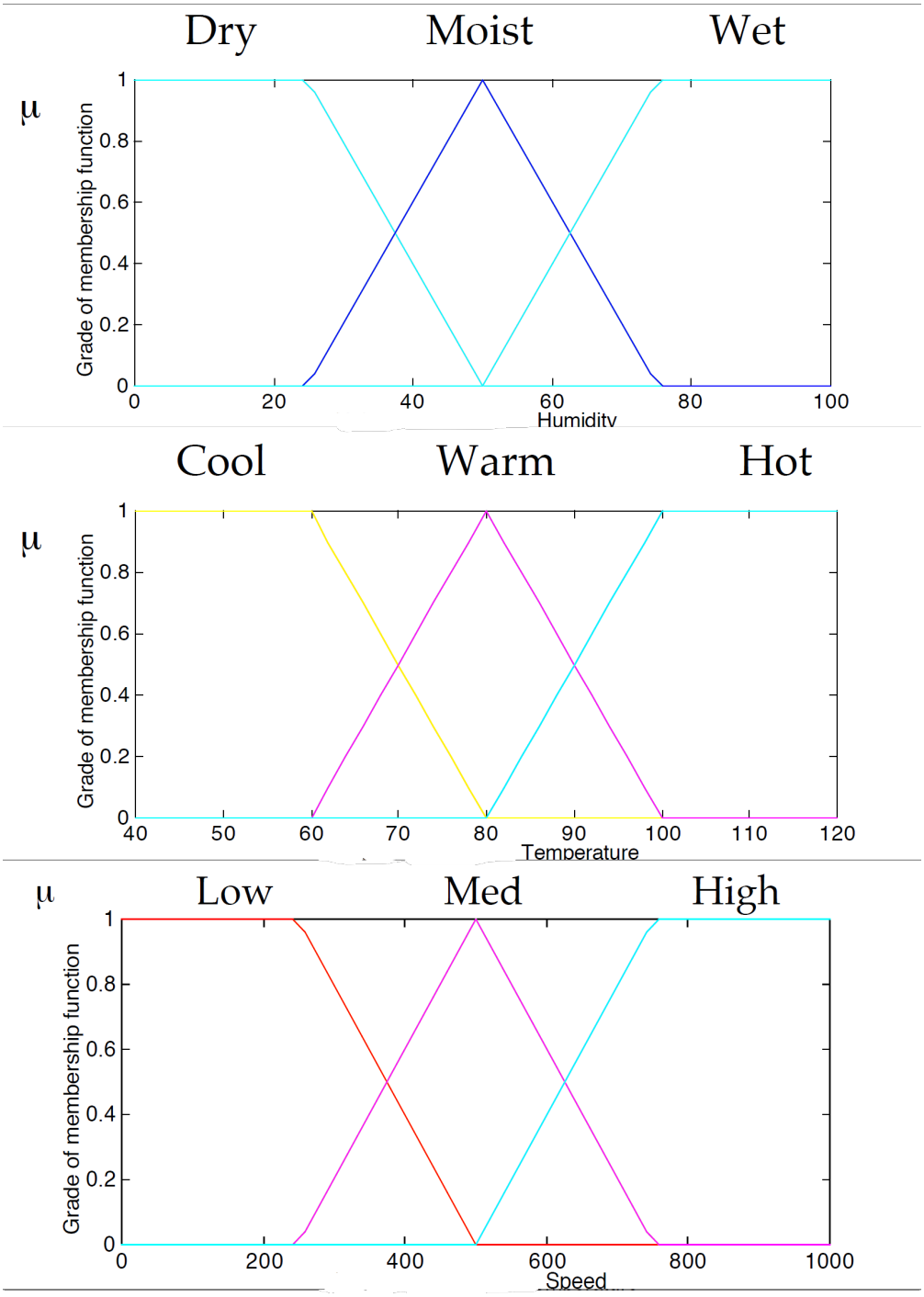

FIGURE 9.2 Définition des fonctions d'appartenances respectivement pour la température, l'humidité et la vitesse de ventilateur.

10 PRATIQUE EN LABORATOIRE

Buts de l'activité

Le but de l'activité est de se familiariser avec le langage Prolog.

- Système de programmation déclaratif
- Base de connaissances et requêtes
- Manipulation de listes
- Interaction avec Python

Installation de Prolog

Vous pouvez trouver l'installation de Prolog sur le site web de SWI-Prolog. Installez la dernière version stable et n'oubliez pas de mettre le dossier de l'exécutable dans votre `PATH`, sinon vous ne pourrez pas l'utiliser en interaction avec Python.

Il arrive que des anti-virus considèrent SWI-Prolog suspicieux. Si votre SWI-Prolog disparaît, c'est qu'il a probablement été mis en quarantaine.

Profitez-en pour installer la librairie `swiplserver` dans votre environnement Python, soit le même que vous utiliserez pour solutionner la problématique. Cette librairie permet d'interroger une base de connaissance Prolog à partir d'un programme Python. Plus d'informations sont disponibles sur le site de SWI MQI. Un exemple d'utilisation de cette librairie est inclus dans le code source distribué pour le laboratoire sur la .

Introduction à Prolog

Un court séminaire d'introduction à Prolog se déroulera au début du laboratoire. Ce séminaire est interactif, donc **préparez votre installation de SWI-Prolog avant le laboratoire**.

Plan du séminaire :

- Programmation déclarative
- Syntaxe
- Listes
- Fonctions récursives

L.E1 Assistant de choix de menu

Réalisez en Prolog un aide aux clients à la sélection de repas dans un restaurant. Le programme doit suggérer aux clients des repas ou des repas légers. Un repas est composé d'un hors-d'oeuvre, d'un plat et d'un dessert. Un plat est une assiette de viande ou de poisson. Un repas est léger si le total des points pour le repas est de moins de 10 points. Les choix au menu aujourd'hui sont les suivants :

Type	Contenu	Points
hors d'oeuvre	salade	1
hors d'oeuvre	pâté	6
poisson	sole	2
poisson	thon	4
viande	porc	7
viande	boeuf	3
dessert	glace	5
dessert	fruit	1

L'interrogation de l'assistant pour un repas ou un repas léger doit se faire avec des questions du type : `repas(H,P,D)` ou encore `repasLeger(H,P,D)` avec `H` un hors-d'oeuvre, `P` un plat et `D` un dessert. Qu'est-ce qui est demandé et quelles seraient les réponses des questions suivantes ?

```
repas(pate,porc,X). repasLeger(pate,porc,X). repasLeger(X,Y,glace).
```

L.E2 Un sur deux

Sans utiliser les fonctionnalités de la librairie de liste de l'environnement Prolog, réalisez un prédicat qui permet d'imprimer une valeur sur deux d'une liste, donc les éléments 2, 4, 6, etc. Le prédicat est appelé `un_sur_deux` et a un seul paramètre, la liste à imprimer. Le prédicat devrait être capable d'imprimer toutes les listes et il devrait donc toujours être évalué à vrai (**`true`**). Testez au moins avec la liste vide et les listes avec 1, 2, 3, 4, 5 et 10 éléments.

L.E3 Se rendre à Bucharest

Pour cet exercice, il s'agit de d'intégrer une base de connaissance Prolog dans un script `Python` afin de trouver un chemin de Arad à Bucarest en utilisant une des techniques suivantes (ou toutes !) : profondeur d'abord, largeur d'abord, recherche avare et recherche A^*. Une fois le chemin trouvé, le programme doit afficher son coût total et l'ordre des villes pour arriver à destination. Afin de réaliser cet exercice, une base de connaissances partielles vous est fournie qui inclut les informations de la carte de la page 64 (figure 3.1) et les informations du tableau 3.16 de la page 85 de `IA-MODERN`.

Avec les exercices précédents, vous maîtrisez les notions de base du fonctionnement de Prolog. Vous maîtrisez aussi certains des éléments importants de la planification par espace d'état : trouver les actions possibles et donc trouver des états successeurs. Cet exercice se rapproche encore plus de la problématique.

Afin de réaliser votre solution de cet exercice (ou de la problématique), basez vous sur la planification par recherche dans l'espace d'état vu dans les procéduraux. Cet algorithme utilise des structures de données (pile, queue, queue avec priorité et *set*) pour prioriser le

prochain état dans la recherche. Ces structures de données seront gérées via Python qui fera des requêtes à Prolog pour obtenir les actions possibles et les états successeurs à chaque noeud.

Un code de départ de ce problème est fourni sur le site de session. Cela inclut un script Python qui fait une requête à Prolog et la base de connaissances des villes de Roumanie.

L.E4 Trouver les actions possibles

Prolog a la capacité de traiter des termes complexe comme atomes uniques. Généralement, on utilise des termes comme "`A`" ou "`Mere`" comme variables, qui peuvent prendre des valeurs comme "`42`" ou "`alice`". Mais il est aussi possible d'utiliser directement un prédicat comme "`trouve(X,Y)`" dans un prédicat plus complexe comme on le ferait avec une variable simple. Cette caractéristique sera très utile pour l'exercice suivant, où l'on cherche une liste de prédicats de forme "move(X,Y)" et "moveToTable(X)". Ces termes peuvent être utilisées dans une définition de règle, de format "`action(Env, move(X,Y)) :- [...]`" où X et Y seront unifiées avec des constantes selon la règle définie, résultant en une réponse de forme `move(c, b)`.

Cet exercice vous fait faire un pas vers la compréhension de la planification pour des problèmes complexes avec plusieurs types d'actions possibles. Il s'agit de trouver la liste des actions qui peuvent être posées pour un état donné d'un environnement, comme celui de la figure 10.1. L'utilisation des prédicats `member` et `findall` de Prolog sont très utiles.

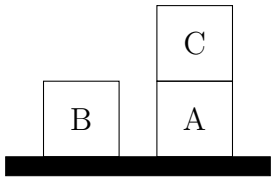

FIGURE 10.1 Le monde des blocs

L'état de l'environnement est donné sous forme de liste et l'ordre des éléments n'a pas d'importance. Pour la figure 10.1, l'état serait :

```
[on(b, table), on(a, table), on(c, a), clear(b),
               clear(c), block(a), block(b), block(c)]
```

- `on(X,Y)` signifie que le bloc `X` est sur `Y`, `Y` pouvant être un autre bloc ou la table.
- `clear(X)` signifie que le bloc `X` est libre de recevoir un autre bloc ou d'être déplacé.
- `block(X)` signifie que `X` est un bloc.

Il y a deux actions possibles dans le monde des blocs :

— `move(B,X,Y)` : Signifie de déplacer le bloc `B`, de l'endroit X à l'endroit `Y`. Pour que cette action soit valable, il faut que `B` et `Y` soient libres (`clear`), que `B` soit sur `X` (`on`) et que `B` et `Y` soient des blocs (`block`). Afin de ne pas considérer des actions inutiles, `B`, `X` et `Y` doivent être tous différents.

— `moveToTable(B,X)` : Signifie de déplacer le bloc `B` de l'endroit `X` et de le mettre sur la table. Pour que cette action soit valable, il faut que `B` soit libre (`clear`), que `B` soit sur `X` (`on`) et que `B` et `X` soient des blocs (`block`). Évidemment, `B` et `X` doivent être différents. Une action dédiée pour un déplacement sur la table (`moveToTable`) est réalisée, car moins de vérifications sont nécessaires, par exemple, la table est toujours libre.

Vous devez réaliser le prédicat nommé `actionsPossibles` qui permet de trouver, sous forme de liste, toutes les actions possibles pour un état donné. Un exemple d'appel serait le suivant :

```
actionsPossibles([on(b, table), on(a, table), on(c,a),
        clear(b), clear(c), block(a), block(b), block(c)], R).
```

Notez que `actionPossibles` doit appeler à son tour le prédicat `findall`, de sorte que :

```
actionsPossibles(Env, R) :- findall(Action, action(Env, Action), R).
```

Le résultat donné dans `R` serait alors :

```
R = [move(b, table, c), move(c, a, b), moveToTable(c, a)]
```

Validez et testez votre prédicat dans différentes situations, et non pas uniquement celle de la figure 10.1. Le prédicat `actionsPossibles` est toujours évalué à vrai (**`true`**), car dans le pire des cas, il n'y aurait aucune action possible et le résultat serait alors la liste vide.

Suggestions :

1. Commencer par faire une fonction `action(Env, move(B, X, Y))` qui valide si un environnement `Env` donnée permet une action `move(B, X, Y)`. Le prédicat `member` est très utile ici.
2. Surcharger le prédicat `action` en répétant l'opération précédente avec **moveToTable**.
3. Maintenant vous pouvez trouver toutes les actions possibles en utilisant le prédicat `findall`.

11 PRATIQUE PROCÉDURALE 3

But de l'activité

— Comparer la logique floue selon Mamdani et Sugeno
— Utiliser les algorithmes génétiques
— Comprendre où se trouve la connaissance d'un agent intelligent

Afin de bien réussir cette activité, il vous est suggéré de la préparer. Vous pouvez la préparer en lisant la documentation qui est suggérée à la section 2.1 pour ces sujets.

11.1 EXERCICES

P3.E1 Inférence selon Sugeno

On cherche à comprendre la différence entre les systèmes flous Mandani (manipulation de variables logiques) et Sugeno (manipulation de fonctions numériques). En effet, Sugeno permet de trouver une fonction continue non-linéaire de contrôle d'un processus (moteur, etc.) même si on n'a accès qu'à des dépendances linéaires entre les variables de contrôle en entrée du système et les réponses de celui-ci. Suivant la valeur des variables d'état du système à contrôler, on doit recourir à des fonctions de commandes différentes. **Les règles définissent quelle fonction de commande appliquer selon les valeurs des variables d'état du système.**

Soit x la variable d'entrée qui prend ses valeurs dans l'ensemble $\{1, 2, 3, 4, 5\}$ et y le paramètre de sortie obtenu après l'application d'une règle. x peut appartenir aux sous-ensembles flous G et P définis par[1] :

G = (0/1, 0/2, 0.3/3, 0.8/4, 1/5) ;

P = (1/1, 1/2, 0.8/3, 0.6/4, 0/5) ;

Si l'inférence se fait selon l'approche de Sugeno et que les seules deux règles sont :

Si $x \in G$ ALORS $y = 9 - x$

Si $x \in P$ ALORS $y = x + 3$

Pour chaque valeur différente des entrées x, trouver la sortie de la logique de Sugeno si celle-ci correspond aux moyennes pondérées *Weighted Average* du paramètre y.

1. On désigne par la notation μ/c la valeur de la fonction d'appartenance μ associée à la valeur de $x = c$.

P3.E2 Algorithme génétiques

1. Qu'est-ce qu'un algorithme génétique ?
2. Qu'est-ce qui change dans un algorithme génétique ?
3. Quels sont les mécanismes d'hérédité ? Mécanismes d'évolution ?
4. Comment on conçoit un algorithme génétique ?

P3.E3 Une première évolution

On dispose d'une population de quatre individus, chacun représenté par un seul chromosome de 8 gènes encodés en binaire.

Individu	# 1	# 2	# 3	# 4
Chromosome	10101010	10101000	00101101	11110101
Fitness résultante	80	20	60	40

1. Déterminer la roulette de sélection ;
2. Déterminer les enfants des individus #1 et #3 en assumant un croisement de 50% des gènes ;
3. Effectuer une mutation sur le troisième gène du premier enfant des individus #1 et #2 avec un croisement à 50% de la longueur du chromosome, i.e. en ne transmettant à l'enfant que la moitié des gènes du parent.

P3.E4 Choix des paramètres et représentations

Une difficulté majeure dans l'utilisation des algorithmes génétiques repose sur le choix d'encodage des gènes et des chromosomes pour les individus d'une population. Proposer et justifier une population et son codage sous forme de chromosomes et gènes pour les deux problèmes suivants :

1. Un voyageur de commerce désire optimiser le parcours entre les différentes villes qu'il doit visiter selon la distance totale parcourue (figure 11.1).
2. On doit dimensionner les pinces d'un robot illustrées à la figure 11.2 de façon à ce que la pression appliquée aux objets saisis et modélisée par les moments ou couples $\mathbf{M}$ soient suffisament faibles pour ne pas détruire les objets, mais suffisamment forts pour les retenir. Les forces F_x et F_y sont les entrées de la pince (sorties des moteurs). Les dimensions des segments a à g ainsi que y doivent être optimisés.

P3.E5 Où est l'intelligence de l'IA ?

Les différentes techniques de l'intelligence artificielle permettre de résoudre des problèmes variés. Lors de la conception d'un agent intelligent ou d'un algorithme d'optimisation, il est

FIGURE 11.1 Ensemble des villes que doit parcourir le voyageur de commerce. Source : les contributeurs dOpenStreetMap.

important de comprendre les principes, avantages et désavantages de chaque techniques pour choisir les plus adaptés à notre problèmes.

Pour chaque technique vue pendant l'APP, répondez aux questions suivantes :

1. Où se trouve la connaissance ?
2. Comment est-elles représentée ?
3. Par quel mécanisme est mis en oeuvre "l'intelligence" ?
4. Quelle transformation doit subir les données en entrée ?
5. Comment se fait la sélection et la validation de la solution retenue ?
6. Nommez trois applications pour laquelle cette technique est bien adaptée.

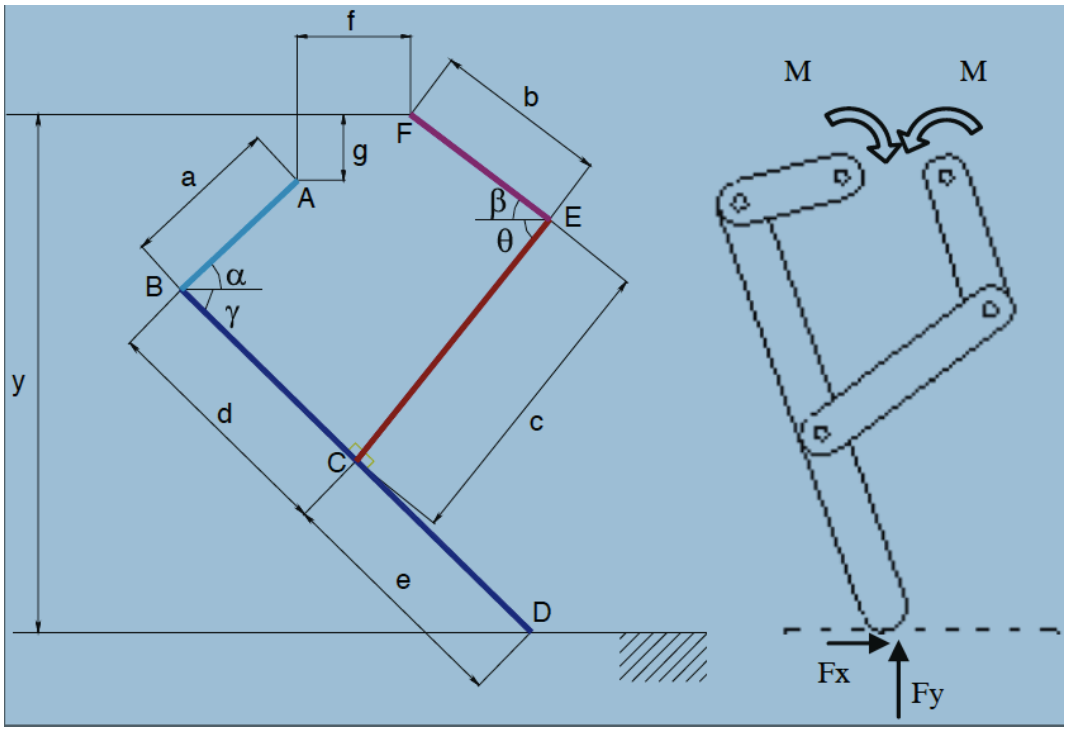

FIGURE 11.2 Paramètres à optimiser pour que les pinces puissent retenir les pièces saisies sans les détruire. Source : Rayes & Gonzalez, *Mechanical Design Optimization of a Walking Robot Leg...*, Springer, 2005.

12 PRATIQUE EN LABORATOIRE

Buts de l'activité

Le but de l'activité est de se familiariser avec les techniques de la logique floue et des algorithmes génétiques.

- Logique floue
- Algorithmes génétiques

L.E1 Pendule inversé

Télécharger le code **CartPoleFuzzy** disponible sur le site de session. **IMPORTANT** : Assurez-vous de lire les notes fournies avec le code pour installer les bonnes librairies et effectuer les corrections nécessaires.

La classe "cartpole.py" simule un pendule inversé en deux dimensions, soit une tige sur une plateforme mobile. La fonction `createFuzzyController` dans le script "main.py" génère un controlleur basé sur la logique floue pour le maintenir en équilibre.

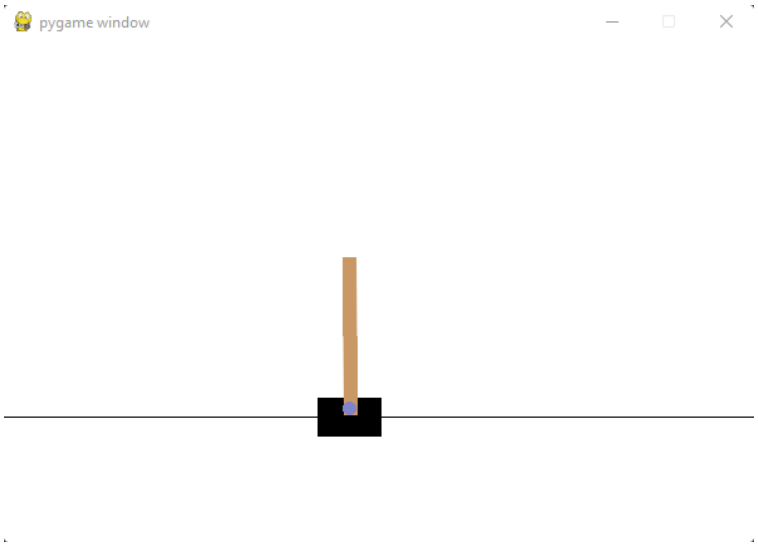

FIGURE 12.1 Illustration du pendule inversé simulé par la classe `CartPoleEnv`

1. Familiarisez-vous avec les principes du pendule inversé et la classe qui la simule. Quelles sont les entrées ? Les sorties ?
2. Identifiez les sections du code qui sont à modifier, marquées par le terme `TODO` ;
3. Familiarisez-vous avec l'API de la librairie *scikit-fuzzy* ;
4. Cherchez la solution qui vous permet de **stabiliser le pendule**. Commencer par étudier le problème sur papier - comment balancieriez vous une tige sur votre main ? Déterminez quelles catégories floues pourraient servir à la stabilisation.

5. Modifier le règle et les paramètres du système de contrôle (conjonction pour les antécédents, implication par mise à l'échelle, etc.) ;
6. Définir les fonctions d'appartenance et les règles.
7. Après chaque modification des paramètres du système flou, observez le comportement du pendule. Changez la configuration à votre guise ainsi que les règles afin de comprendre le comportement du système flou ;
8. Si vous réussissez à stabiliser le pendule pour 100 itérations, augmenter la durée à 1000 itérations.
9. S'il y a lieu, visualiser la surface de contrôle. Qu'observez-vous ?

L.E2 Optimisation avec un algorithme génétique

Télécharger le code **GeneticAlgorithm** disponible sur le site de session. La classe `Genetic` renferme le squelette d'un algorithme génétique pour l'optimisation d'une fonction numérique. Ajoutez-y les fonctions d'encodage et de décodage, ainsi que les opérations de sélection, de croisement et de mutation. Déterminez la meilleure configuration d'optimisation. Répondez aux questions suivantes :

1. Quel est l'effet de la taille de la population sur la rapidité de convergence ?
2. Quel est l'effet de la probabilité de mutation et de croisement sur la qualité de la convergence vers la solution ? Que remarquez-vous sur la distribution des individus autour de la solution optimale ?
3. Est-ce que l'augmentation du nombre de bits pour encoder un individu permet de trouver une meilleure solution ?

L.E3 Travail sur la problématique

Vous avez maintenant en main tous les outils pour planifier, contrôler et optimiser votre joueur automatisé. Rappelez-vous que la conception de systèmes intelligents est un processus itératif qui exige une analyse approfondie des données disponibles.

Bon succès !

13 VALIDATION AU LABORATOIRE

La validation de votre solution se fait via une défense orale, dont les détails sont présentés à la section 6.1 du présent document.

Un horaire de passage sera publié sur le site de session au moins 48h avant la période de validation. Vous devez vous présenter 5 minutes à l'avance afin d'être prêts à commencer à l'heure assignée.

Vous devez envoyer votre présentation aux tuteurs par courriel AVANT l'heure assignée à votre présentation afin de mettre la présentation sur l'ordinateur du local et accélérer les transitions entre les équipes.

LISTE DES RÉFÉRENCES

[1] M. Negnevitsky. Artificial Intelligence - A Guide to Intelligent Systems. Addison Wesley, 3 edition, 2011.

[2] S. J. Russell and P. Norvig. Artificial Intelligence - A Modern Approach. Pearson Education, Upper Saddle River, NJ, USA, 4 edition, 2021.