

collection

DataPro

L'Intelligence Artificielle

pour les développeurs

Concepts et implémentations en **Java**

Virginie MATHIVET

Téléchargement
www.editions-eni.fr



Les exemples à télécharger sont disponibles à l'adresse suivante :
<http://www.editions-eni.fr>
Saisissez la référence ENI de l'ouvrage **DPJINT** dans la zone de recherche et validez. Cliquez sur le titre du livre puis sur le bouton de téléchargement.

Avant-propos

1. Objectifs du livre	15
2. Public et prérequis	15
3. Structure du livre	16
4. Code en téléchargement	18

Introduction

1. Présentation du chapitre	19
2. Définir l'intelligence	19
3. L'intelligence du vivant	22
4. L'intelligence artificielle	23
5. Domaines d'application	25
6. Synthèse	27

Chapitre 1 Systèmes experts

1. Présentation du chapitre	29
2. Exemple : un système expert en polygones	30
2.1 Triangles	30
2.2 Quadrilatères	32
2.3 Autres polygones	34

2 ————— L'Intelligence Artificielle

pour les développeurs - Concepts et implémentations en Java

3.	Contenu d'un système expert	34
3.1	Base de règles	35
3.2	Base de faits	36
3.3	Moteur d'inférences	37
3.4	Interface utilisateur	38
4.	Types d'inférences	39
4.1	Chaînage avant	39
4.1.1	Principe	39
4.1.2	Application à un exemple	40
4.2	Chaînage arrière	41
4.2.1	Principe	41
4.2.2	Application à un exemple	42
4.3	Chaînage mixte	43
5.	Étapes de construction d'un système	44
5.1	Extraction des connaissances	45
5.2	Création du moteur d'inférences	45
5.3	Écriture des règles	46
5.4	Création de l'interface utilisateur	46
6.	Performance et améliorations	47
6.1	Critères de performance	47
6.2	Amélioration des performances par l'écriture des règles	48
6.3	Importance de la représentation du problème	49
7.	Ajout d'incertitudes et de probabilités	51
7.1	Apport des incertitudes	51
7.2	Faits incertains	52
7.3	Règles incertaines	52
8.	Domaines d'application	53
8.1	Aide au diagnostic	54
8.2	Estimation de risques	54
8.3	Planification et logistique	55
8.4	Transfert de compétences et connaissances	55
8.5	Autres applications	56

- 9. Création d'un système expert en Java 57
 - 9.1 Détermination des besoins 57
 - 9.2 Implémentation des faits 58
 - 9.3 Base de faits 62
 - 9.4 Règles et base de règles 64
 - 9.5 Interface 66
 - 9.6 Moteur d'inférences 69
 - 9.7 Saisie des règles et utilisation 76
- 10. Utilisation de Prolog 79
 - 10.1 Présentation du langage 79
 - 10.2 Syntaxe du langage 80
 - 10.2.1 Généralités 80
 - 10.2.2 Prédicats 81
 - 10.2.3 Poser des questions 81
 - 10.2.4 Écriture des règles 82
 - 10.2.5 Autres prédicats utiles 83
 - 10.3 Codage du problème des formes géométriques 84
 - 10.4 Codage du problème des huit reines 88
 - 10.4.1 Intérêt du chaînage arrière 88
 - 10.4.2 Étude du problème 89
 - 10.4.3 Règles à appliquer 89
 - 10.4.4 Règles de conflits entre reines 90
 - 10.4.5 But du programme 92
 - 10.4.6 Exemples d'utilisation 92
- 11. Synthèse 93

4 _____ L'Intelligence Artificielle

pour les développeurs - Concepts et implémentations en Java

Chapitre 2 Logique floue

1. Présentation du chapitre	95
2. Incertitude et imprécision	96
2.1 Incertitude et probabilités	96
2.2 Imprécision et subjectivité.	96
2.3 Nécessité de traiter l'imprécision.	97
3. Ensembles flous et degrés d'appartenance	98
3.1 Logique booléenne et logique floue	98
3.2 Fonctions d'appartenance	99
3.3 Caractéristiques d'une fonction d'appartenance.	102
3.4 Valeurs et variables linguistiques	103
4. Opérateurs sur les ensembles flous	104
4.1 Opérateurs booléens.	104
4.2 Opérateurs flous	106
4.2.1 Négation.	106
4.2.2 Union et intersection	108
5. Création de règles	110
5.1 Règles en logique booléenne	110
5.2 Règles floues	110
6. Fuzzification et défuzzification.	113
6.1 Valeur de vérité.	113
6.2 Fuzzification et application des règles	115
6.3 Défuzzification.	119
7. Domaines d'application	121
7.1 Première utilisation	121
7.2 Dans les produits électroniques.	122
7.3 En automobile.	122
7.4 Autres domaines.	122

8.	Implémentation d'un moteur de logique floue	123
8.1	Le cœur du code : les ensembles flous	124
8.1.1	Point2D : un point d'une fonction d'appartenance	124
8.1.2	EnsembleFlou : un ensemble flou	125
8.1.3	Opérateurs de comparaison et de multiplication	126
8.1.4	Opérateurs ensemblistes	127
8.1.5	Calcul du barycentre	136
8.2	Ensembles flous particuliers	138
8.3	Variables et valeurs linguistiques	141
8.3.1	Valeur linguistique	141
8.3.2	Variable linguistique	142
8.4	Règles floues	143
8.4.1	Expression floue	144
8.4.2	Valeur numérique	144
8.4.3	Règle floue	145
8.5	Système de contrôle flou	147
8.6	Synthèse du code créé	150
9.	Implémentation d'un cas pratique	151
10.	Synthèse	157

Chapitre 3

Recherche de chemins

1.	Présentation du chapitre	159
2.	Chemins et graphes	160
2.1	Définition et concepts	160
2.2	Représentations	161
2.2.1	Représentation graphique	161
2.2.2	Matrice d'adjacence	161
2.3	Coût d'un chemin et matrice des longueurs	164
3.	Exemple en cartographie	166

6 ————— L'Intelligence Artificielle

pour les développeurs - Concepts et implémentations en Java

4.	Algorithmes naïfs de recherche de chemins	168
4.1	Parcours en profondeur	168
4.1.1	Principe et pseudo-code.	168
4.1.2	Application à la carte.	170
4.2	Parcours en largeur	172
4.2.1	Principe et pseudo-code.	173
4.2.2	Application à la carte.	174
5.	Algorithmes "intelligents"	177
5.1	Algorithme de Bellman-Ford	178
5.1.1	Principe et pseudo-code.	178
5.1.2	Application à la carte.	180
5.2	Algorithme de Dijkstra.	183
5.2.1	Principe et pseudo-code.	184
5.2.2	Application à la carte.	185
5.3	Algorithme A*	188
5.3.1	Principe et pseudo-code.	188
5.3.2	Application à la carte.	189
6.	Domaines d'application	196
7.	Implémentation	197
7.1	Nœuds, arcs et graphes	198
7.1.1	Implémentation des noeuds	198
7.1.2	Classe représentant les arcs	198
7.1.3	Graphes	199
7.2	Fin du programme générique	200
7.2.1	IHM	200
7.2.2	Algorithme générique	201
7.3	Implémentation des différents algorithmes	202
7.3.1	Recherche en profondeur	202
7.3.2	Recherche en largeur	204
7.3.3	Algorithme de Bellman-Ford	205
7.3.4	Algorithme de Dijkstra	206
7.3.5	Algorithme A*	208

7.4	Application à la carte	209
7.4.1	Gestion des tuiles	209
7.4.2	Implémentation de la carte	211
7.4.3	Programme principal	219
7.5	Comparaison des performances.	222
8.	Synthèse	224

Chapitre 4

Algorithmes génétiques

1.	Présentation du chapitre	227
2.	Évolution biologique.	228
2.1	Le concept d'évolution	228
2.2	Les causes des mutations	229
2.3	Le support de cette information : les facteurs	230
2.4	Des facteurs au code génétique	233
2.5	Le « cycle de la vie ».	235
3.	Évolution artificielle	236
3.1	Principes	236
3.2	Vue d'ensemble du cycle.	238
3.2.1	Phases d'initialisation et de terminaison	238
3.2.2	Phase de sélection	238
3.2.3	Phase de reproduction avec mutations.	239
3.2.4	Phase de survie.	239
3.3	Convergence	239
4.	Exemple du robinet.	240
4.1	Présentation du problème	240
4.2	Initialisation de l'algorithme	240
4.3	Évaluation des individus	241
4.4	Reproduction avec mutations	241
4.5	Survie.	243
4.6	Suite du processus	244

8 ————— L'Intelligence Artificielle

pour les développeurs - Concepts et implémentations en Java

5. Choix des représentations	244
5.1 Population et individus	244
5.2 Gènes	244
5.3 Cas d'un algorithme de résolution de labyrinthe	245
6. Évaluation, sélection et survie	247
6.1 Choix de la fonction d'évaluation	247
6.2 Opérateurs de sélection	249
6.3 Opérateurs de survie	250
7. Reproduction : crossover et mutation	251
7.1 Crossover	251
7.2 Mutation	254
8. Domaines d'application	256
9. Coévolution	257
10. Implémentation d'un algorithme génétique	259
10.1 Implémentation générique d'un algorithme	259
10.1.1 Spécifications	259
10.1.2 Paramètres	260
10.1.3 Individus et gènes	261
10.1.4 IHM	264
10.1.5 Processus évolutionnaire	264
10.2 Utilisation pour le voyageur de commerce	268
10.2.1 Présentation du problème	268
10.2.2 Environnement	269
10.2.3 Gènes	271
10.2.4 Individus	272
10.2.5 Programme principal	276
10.2.6 Résultats	277
10.3 Utilisation pour la résolution d'un labyrinthe	278
10.3.1 Présentation du problème	278
10.3.2 Environnement	279
10.3.3 Gènes	283
10.3.4 Individus	284

- 10.3.5 Modification de la fabrique 288
- 10.3.6 Programme principal 290
- 10.3.7 Résultats 291
- 11. Synthèse 293

Chapitre 5
Métaheuristiques d'optimisation

- 1. Présentation du chapitre 295
- 2. Optimisation et minimums 296
 - 2.1 Exemples 296
 - 2.2 Le problème du sac à dos 296
 - 2.3 Formulation des problèmes 297
 - 2.4 Résolution mathématique 298
 - 2.5 Recherche exhaustive 300
 - 2.6 Métaheuristiques 300
- 3. Algorithmes gloutons 301
- 4. Descente de gradient 304
- 5. Recherche tabou 306
- 6. Recuit simulé 309
- 7. Optimisation par essais particuliers 310
- 8. Méta-optimisation 312
- 9. Domaines d'application 313
- 10. Implémentation 314
 - 10.1 Classes génériques 314
 - 10.2 Implémentation des différents algorithmes 316
 - 10.2.1 Algorithme glouton 316
 - 10.2.2 Descente de gradient 316
 - 10.2.3 Recherche tabou 318
 - 10.2.4 Recuit simulé 319
 - 10.2.5 Optimisation par essais particuliers 320

10 _____ L'Intelligence Artificielle

pour les développeurs - Concepts et implémentations en Java

10.3	Résolution du problème du sac à dos	322
10.3.1	Implémentation du problème	322
10.3.2	Algorithme glouton	329
10.3.3	Descente de gradient	330
10.3.4	Recherche tabou	331
10.3.5	Recuit simulé	334
10.3.6	Optimisation par essais particuliers	336
10.3.7	Programme principal	339
10.4	Résultats obtenus	341
11.	Synthèse	344

Chapitre 6 **Systemes multi-agents**

1.	Présentation du chapitre	347
2.	Origine biologique	348
2.1	Les abeilles et la danse	348
2.2	Les termites et le génie civil	350
2.3	Les fourmis et l'optimisation de chemins	351
2.4	Intelligence sociale	352
3.	Systemes multi-agents	352
3.1	L'environnement	352
3.2	Les objets	353
3.3	Les agents	353
4.	Classification des agents	354
4.1	Perception du monde	354
4.2	Prise des décisions	354
4.3	Coopération et communication	355
4.4	Capacités de l'agent	356

5. Principaux algorithmes.	357
5.1 Algorithmes de meutes.	357
5.2 Optimisation par colonie de fourmis	358
5.3 Systèmes immunitaires artificiels	360
5.4 Automates cellulaires	361
6. Domaines d'application	363
6.1 Simulation de foules.	363
6.2 Planification	363
6.3 Phénomènes complexes	364
7. Implémentation	365
7.1 Banc de poissons 2D	365
7.1.1 Les objets du monde et les zones à éviter	365
7.1.2 Les agents-poissons	367
7.1.3 L'océan	375
7.1.4 L'application graphique	378
7.1.5 Résultats obtenus	381
7.2 Tri sélectif	383
7.2.1 Les déchets	384
7.2.2 Les agents nettoyeurs	386
7.2.3 L'environnement	390
7.2.4 L'application graphique	394
7.2.5 Résultats obtenus	398
7.3 Le jeu de la vie	400
7.3.1 La grille	400
7.3.2 L'application graphique	403
7.3.3 Résultats obtenus	407
8. Synthèse	408

12 _____ L'Intelligence Artificielle

pour les développeurs - Concepts et implémentations en Java

Chapitre 7 Réseaux de neurones

1. Présentation du chapitre	411
2. Origine biologique	412
3. Le neurone formel	414
3.1 Fonctionnement général	414
3.2 Fonctions d'agrégation	415
3.3 Fonctions d'activation	416
3.3.1 Fonction heavyside	416
3.3.2 Fonction sigmoïde	416
3.3.3 Fonction gaussienne	417
3.4 Poids et apprentissage	418
4. Perceptron	418
4.1 Structure	418
4.2 Condition de linéarité	419
5. Réseaux feed-forward	421
6. Apprentissage	422
6.1 Apprentissage non supervisé	422
6.2 Apprentissage par renforcement	424
6.3 Apprentissage supervisé	424
6.3.1 Principe général	424
6.3.2 Descente de gradient	425
6.3.3 Algorithme de Widrow-Hoff	427
6.3.4 Rétropropagation	427
6.4 Surapprentissage et généralisation	429
6.4.1 Reconnaître le surapprentissage	430
6.4.2 Création de sous-ensembles de données	431
7. Autres réseaux	432
7.1 Réseaux de neurones récurrents	432
7.2 Cartes de Kohonen	432
7.3 Réseaux de Hopfield	433

- 8. Domaines d'application 433
 - 8.1 Reconnaissance de patterns 434
 - 8.2 Estimation de fonctions 434
 - 8.3 Création de comportements 434
- 9. Implémentation d'un MLP 435
 - 9.1 Points et ensembles de points 435
 - 9.2 Neurone 438
 - 9.3 Réseau de neurones 440
 - 9.4 IHM 443
 - 9.5 Système complet 444
 - 9.6 Programme principal 447
 - 9.7 Applications 449
 - 9.7.1 Application au XOR 449
 - 9.7.2 Application à Abalone 450
 - 9.7.3 Améliorations possibles 452
- 10. Synthèse 453

Bibliographie 455

Sitographie

- 1. Pourquoi une sitographie ? 459
- 2. Systèmes experts 459
- 3. Logique floue 461
- 4. Algorithmes génétiques 463
- 5. Recherche de chemins 465
- 6. Métaheuristiques 466
- 7. Systèmes multi-agents 467
- 8. Réseaux de neurones 468

14 _____ L'Intelligence Artificielle

pour les développeurs - Concepts et implémentations en Java

Annexe

1. Installation de SWI-Prolog	471
2. Utilisation de SWI-Prolog sous Windows	472
Index	475

Chapitre 3

Recherche de chemins

1. Présentation du chapitre

De nombreux domaines font face à un problème de recherche de chemins, appelé "pathfinding" en anglais. On pense tout d'abord aux GPS et aux logiciels de recherche d'itinéraires (en voiture, en train, en transport en commun...), voire aux jeux vidéo dans lesquels les ennemis doivent arriver sur le joueur par le chemin le plus court.

La recherche de chemins est en réalité un domaine bien plus vaste. En effet, de nombreux problèmes peuvent être représentés sous la forme d'un graphe, comme l'enchaînement des mouvements dans un jeu d'échecs.

La recherche d'un chemin dans ce cas-là peut être vue comme la recherche de la suite des mouvements à faire pour gagner.

Ce chapitre commence par présenter les différents concepts de théorie des graphes, et les définitions associées. Les algorithmes fondamentaux sont ensuite présentés, avec leur fonctionnement et leurs contraintes.

Les principaux domaines dans lesquels on peut utiliser cette recherche de chemins sont alors indiqués et un exemple d'implémentation des algorithmes en Java est présenté et appliqué à une recherche de chemins dans un environnement en 2D.

Le chapitre se termine par une synthèse.

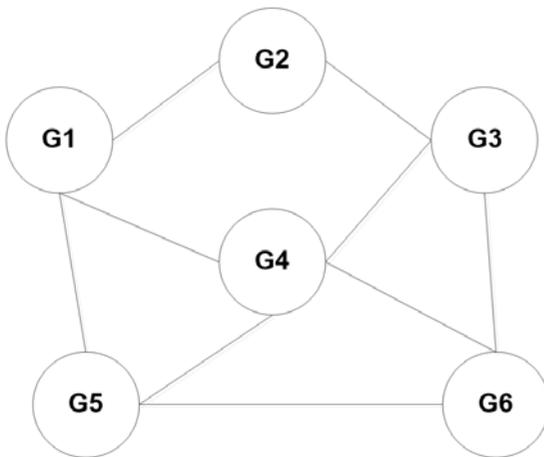
2. Chemins et graphes

Un chemin peut être vu comme un parcours dans un graphe. Les principaux algorithmes se basent donc sur la **théorie des graphes**.

2.1 Définition et concepts

Un **graphe** est un ensemble de **nœuds** ou **sommets** (qui peuvent représenter par exemple des villes) liés par des **arcs**, qui seraient alors des routes.

Voici un graphe qui représente des gares et les liens qui existent entre ces gares (en train, sans changement) :



Les gares de G1 à G6 sont donc les nœuds. L'arc allant de G5 à G6 indique la présence d'un lien direct entre ces deux gares. Il est noté (G5, G6) ou (G6, G5) selon le sens voulu.

Par contre pour aller de G1 à G6, il n'y a pas de lien direct, il faudra passer par G4 ou G5 si on ne souhaite qu'un changement, ou par G2 puis G3 avec deux changements.

Un **chemin** permet de rejoindre différents sommets liés entre eux par des arcs. Ainsi, G1-G2-G3-G6 est un chemin de **longueur 3** (la longueur est le nombre d'arcs suivis).

On parle de **circuit** lorsqu'on peut partir d'un nœud et y revenir. Ici, le graphe contient de nombreux circuits, comme G1-G4-G5-G1 ou G4-G5-G6-G4.

L'**ordre** d'un graphe correspond au nombre de sommets qu'il contient. Notre exemple contient 6 gares, il s'agit donc d'un graphe d'ordre 6.

Deux nœuds sont dits **adjacents** (ou voisins) s'il existe un lien permettant d'aller de l'un à l'autre. G5 est donc adjacent à G1, G4 et G6.

2.2 Représentations

2.2.1 Représentation graphique

Il existe plusieurs façons de représenter un graphe. La première est la **représentation graphique**, comme celle vue précédemment.

L'ordre et le placement des nœuds ne sont pas importants, cependant on va chercher à toujours placer les sommets de façon à rendre le graphe le plus lisible possible.

Le graphe est dit **orienté** si les arcs ont un sens, représentant par exemple des rues à sens unique dans une ville. Si tous les arcs peuvent être pris dans les deux sens, on dit alors que le graphe est **non orienté**, ce qui est généralement le cas de ceux utilisés pour la recherche de chemins.

2.2.2 Matrice d'adjacence

Les représentations graphiques ne sont pas toujours très pratiques, en particulier quand il s'agit d'y appliquer des algorithmes ou de les rentrer dans un ordinateur.

On préfère souvent utiliser une matrice, appelée **matrice d'adjacence**.

■ Remarque

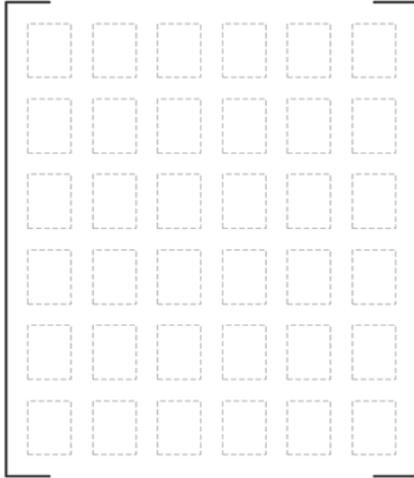
Une matrice est une structure mathématique particulière qui peut être vue plus simplement comme un tableau à deux dimensions.

Dans cette matrice, l'absence d'arc est représentée par un 0, et sa présence par un 1.

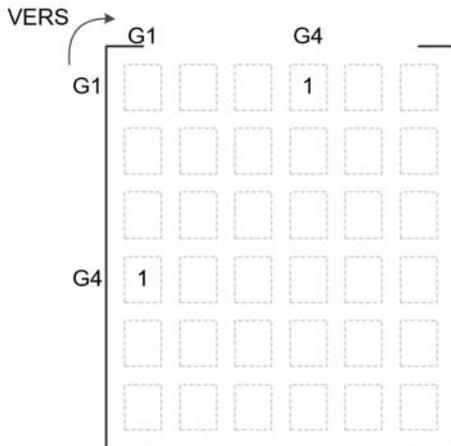
162 ————— L'Intelligence Artificielle

pour les développeurs - Concepts et implémentations en Java

Dans l'exemple des gares, on a donc une matrice de 6 par 6 (car il y a 6 gares) :



On voit sur le graphe qu'il existe un lien entre G1 et G4. La case correspondant au trajet de G1 vers G4 contient donc un 1, tout comme celle de G4 à G1 (le trajet est à double sens). On a alors la matrice suivante :



De même, il existe un arc de G1 vers G2 et G5 mais pas vers G3 ou G6. On peut donc compléter notre matrice :

VERS →

	G1	G2	G3	G4	G5	G6
G1		1	0	1	1	0
G2	1					
G3	0					
G4	1					
G5	1					
G6	0					

On fait de même pour tous les autres nœuds et les autres arcs :

VERS →

	G1	G2	G3	G4	G5	G6
G1		1	0	1	1	0
G2	1		1	0	0	0
G3	0	1		1	0	1
G4	1	0	1		1	1
G5	1	0	0	1		1
G6	0	0	1	1	1	

Il ne reste que la diagonale. Elle représente la possibilité d'aller d'un nœud à lui-même, c'est ce qu'on appelle une boucle. Ici il n'y a pas de trajet direct allant d'une gare à elle-même, on remplit donc par des 0 cette diagonale.

VERS →

	G1	G2	G3	G4	G5	G6
G1	0	1	0	1	1	0
G2	1	0	1	0	0	0
G3	0	1	0	1	0	1
G4	1	0	1	0	1	1
G5	1	0	0	1	0	1
G6	0	0	1	1	1	0

La matrice d'adjacence est alors complète.

2.3 Coût d'un chemin et matrice des longueurs

Dans le cadre de la recherche du chemin le plus court, le moins cher ou le plus rapide, on doit rajouter des informations. Celles-ci sont appelées de manière arbitraire **longueurs**, sans préciser l'unité. Il peut donc s'agir de kilomètres, d'euros, de minutes, de kilos...