

# L'Intelligence Artificielle pour les développeurs

Concepts et implémentations en C#

Virginie MATHIVET

Téléchargement  
[www.editions-eni.fr](http://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 **DPINT** dans la zone de recherche et validez. Cliquez sur le titre du livre puis sur le bouton de téléchargement.

## Avant-propos

## Introduction

1. Structure 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

### Systemes 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 . . . . .	33
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 . . . . .	39

# 2 ————— L'Intelligence Artificielle

pour les développeurs - Concepts et implémentations en C#

4.	Types d'inférences . . . . .	40
4.1	Chaînage avant . . . . .	40
4.1.1	Principe . . . . .	40
4.1.2	Application à un exemple . . . . .	40
4.2	Chaînage arrière . . . . .	42
4.2.1	Principe . . . . .	42
4.2.2	Application à un exemple . . . . .	42
4.3	Chaînage mixte. . . . .	44
5.	Étapes de construction d'un système . . . . .	45
5.1	Extraction des connaissances. . . . .	46
5.2	Création du moteur d'inférences . . . . .	46
5.3	Écriture des règles . . . . .	47
5.4	Création de l'interface utilisateur . . . . .	47
6.	Performance et améliorations . . . . .	48
6.1	Critères de performance. . . . .	48
6.2	Amélioration des performances par l'écriture des règles . . . . .	49
6.3	Importance de la représentation du problème . . . . .	50
7.	Domaines d'application . . . . .	52
7.1	Aide au diagnostic . . . . .	52
7.2	Estimation de risques . . . . .	53
7.3	Planification et logistique. . . . .	54
7.4	Transfert de compétences et connaissances . . . . .	54
7.5	Autres applications. . . . .	55
8.	Création d'un système expert en C# . . . . .	55
8.1	Détermination des besoins. . . . .	56
8.2	Implémentation des faits . . . . .	57
8.3	Base de faits. . . . .	61
8.4	Règles et base de règles. . . . .	62
8.5	Interface . . . . .	64
8.6	Moteur d'inférences . . . . .	67
8.7	Saisie des règles et utilisation. . . . .	74

- 9. Utilisation de Prolog . . . . . 76
  - 9.1 Présentation du langage . . . . . 77
  - 9.2 Syntaxe du langage . . . . . 78
    - 9.2.1 Généralités . . . . . 78
    - 9.2.2 Prédicats . . . . . 78
    - 9.2.3 Poser des questions . . . . . 79
    - 9.2.4 Écriture des règles . . . . . 80
    - 9.2.5 Autres prédicats utiles . . . . . 81
  - 9.3 Codage du problème des formes géométriques . . . . . 82
  - 9.4 Codage du problème des huit reines . . . . . 86
    - 9.4.1 Intérêt du chaînage arrière . . . . . 86
    - 9.4.2 Étude du problème . . . . . 86
    - 9.4.3 Règles à appliquer . . . . . 87
    - 9.4.4 Règles de conflits entre reines . . . . . 88
    - 9.4.5 But du programme . . . . . 90
    - 9.4.6 Exemples d'utilisation . . . . . 90
- 10. Ajout d'incertitudes et de probabilités . . . . . 91
  - 10.1 Apport des incertitudes . . . . . 91
  - 10.2 Faits incertains . . . . . 92
  - 10.3 Règles incertaines . . . . . 93
- 11. Synthèse . . . . . 94

**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

# 4 L'Intelligence Artificielle

pour les développeurs - Concepts et implémentations en C#

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.	Exemples d'applications. . . . .	121
7.1	Premières utilisations . . . . .	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	FuzzySet : 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	LinguisticValue : valeur linguistique	141
8.3.2	LinguisticVariable : variable linguistique	142
8.4	Règles floues	143
8.4.1	FuzzyExpression : expression floue	143
8.4.2	FuzzyValue : valeur floue	144
8.4.3	FuzzyRule : règle floue	144
8.5	Système de contrôle flou	146
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	165
3.	Exemple en cartographie	166
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	174
4.2.1	Principe et pseudo-code	175
4.2.2	Application à la carte	176

# 6 ————— L'Intelligence Artificielle

pour les développeurs - Concepts et implémentations en C#

5. Algorithmes "intelligents" . . . . .	179
5.1 Algorithme de Bellman-Ford . . . . .	180
5.1.1 Principe et pseudo-code . . . . .	180
5.1.2 Application à la carte . . . . .	182
5.2 Algorithme de Dijkstra . . . . .	186
5.2.1 Principe et pseudo-code . . . . .	186
5.2.2 Application à la carte . . . . .	187
5.3 Algorithme A* . . . . .	190
5.3.1 Principe et pseudo-code . . . . .	190
5.3.2 Application à la carte . . . . .	192
6. Implémentations . . . . .	200
6.1 Nœuds, arcs et graphes . . . . .	200
6.1.1 Implémentation des nœuds . . . . .	200
6.1.2 Classe représentant les arcs . . . . .	201
6.1.3 Interface des graphes . . . . .	202
6.2 Fin du programme générique . . . . .	203
6.2.1 IHM . . . . .	203
6.2.2 Algorithme générique . . . . .	204
6.3 Codage des différents algorithmes . . . . .	205
6.3.1 Recherche en profondeur . . . . .	205
6.3.2 Recherche en largeur . . . . .	207
6.3.3 Algorithme de Bellman-Ford . . . . .	208
6.3.4 Algorithme de Dijkstra . . . . .	209
6.3.5 Algorithme A* . . . . .	211
6.4 Application à la carte . . . . .	212
6.4.1 Tile et Tiletype . . . . .	213
6.4.2 Implémentation de la carte . . . . .	215
6.4.3 Programme principal . . . . .	222
6.5 Comparaison des performances . . . . .	226
7. Domaines d'application . . . . .	228
8. Synthèse . . . . .	230

**Chapitre 4**  
**Algorithmes génétiques**

- 1. Présentation du chapitre ..... 233
- 2. Évolution biologique. .... 234
  - 2.1 Le concept d'évolution ..... 234
  - 2.2 Les causes des mutations ..... 235
  - 2.3 Le support de cette information : les facteurs ..... 236
  - 2.4 Des facteurs au code génétique ..... 239
  - 2.5 Le « cycle de la vie ». .... 241
- 3. Évolution artificielle ..... 242
  - 3.1 Principes ..... 242
  - 3.2 Vue d'ensemble du cycle. .... 244
    - 3.2.1 Phases d'initialisation et de terminaison. .... 244
    - 3.2.2 Phase de sélection ..... 244
    - 3.2.3 Phase de reproduction avec mutations ..... 245
    - 3.2.4 Phase de survie ..... 245
  - 3.3 Convergence ..... 245
- 4. Exemple du robinet. .... 246
  - 4.1 Présentation du problème ..... 246
  - 4.2 Initialisation de l'algorithme ..... 246
  - 4.3 Évaluation des individus ..... 247
  - 4.4 Reproduction avec mutations ..... 247
  - 4.5 Survie. .... 249
  - 4.6 Suite du processus ..... 250
- 5. Choix des représentations ..... 250
  - 5.1 Population et individus ..... 250
  - 5.2 Gènes ..... 250
  - 5.3 Cas d'un algorithme de résolution de labyrinthe ..... 251
- 6. Évaluation, sélection et survie ..... 254
  - 6.1 Choix de la fonction d'évaluation ..... 254
  - 6.2 Opérateurs de sélection ..... 255
  - 6.3 Opérateurs de survie. .... 256



# 8 \_\_\_\_\_ L'Intelligence Artificielle

pour les développeurs - Concepts et implémentations en C#

7.	Reproduction : crossover et mutation. . . . .	257
7.1	Crossover. . . . .	257
7.2	Mutation. . . . .	261
8.	Domaines d'application . . . . .	262
9.	Implémentation d'un algorithme génétique . . . . .	264
9.1	Implémentation générique d'un algorithme . . . . .	264
9.1.1	Spécifications . . . . .	264
9.1.2	Paramètres. . . . .	265
9.1.3	Individus et gènes . . . . .	267
9.1.4	IHM. . . . .	269
9.1.5	Processus évolutionnaire . . . . .	270
9.2	Utilisation pour le voyageur de commerce . . . . .	275
9.2.1	Présentation du problème . . . . .	275
9.2.2	Environnement . . . . .	276
9.2.3	Gènes. . . . .	279
9.2.4	Individus . . . . .	280
9.2.5	Programme principal . . . . .	284
9.2.6	Résultats . . . . .	286
9.3	Utilisation pour la résolution d'un labyrinthe . . . . .	287
9.3.1	Présentation du problème . . . . .	287
9.3.2	Environnement . . . . .	288
9.3.3	Gènes. . . . .	295
9.3.4	Individus . . . . .	296
9.3.5	Programme principal . . . . .	301
9.3.6	Résultats . . . . .	302
10.	Coévolution . . . . .	304
11.	Synthèse . . . . .	305

**Chapitre 5**  
**Métaheuristiques d'optimisation**

- 1. Présentation du chapitre ..... 307
- 2. Optimisation et minimums ..... 308
  - 2.1 Exemples ..... 308
  - 2.2 Le problème du sac à dos ..... 308
  - 2.3 Formulation des problèmes ..... 309
  - 2.4 Résolution mathématique ..... 311
  - 2.5 Recherche exhaustive ..... 312
  - 2.6 Métaheuristiques ..... 312
- 3. Algorithmes gloutons ..... 313
- 4. Descente de gradient ..... 316
- 5. Recherche tabou ..... 319
- 6. Recuit simulé ..... 321
- 7. Optimisation par essais particuliers ..... 323
- 8. Méta-optimisation ..... 325
- 9. Domaines d'application ..... 325
- 10. Implémentation ..... 327
  - 10.1 Classes génériques ..... 327
  - 10.2 Implémentation des différents algorithmes ..... 329
    - 10.2.1 Algorithme glouton ..... 329
    - 10.2.2 Descente de gradient ..... 329
    - 10.2.3 Recherche tabou ..... 331
    - 10.2.4 Recuit simulé ..... 332
    - 10.2.5 Optimisation par essais particuliers ..... 333
  - 10.3 Résolution du problème du sac à dos ..... 335
    - 10.3.1 Implémentation du problème ..... 335
    - 10.3.2 Algorithme glouton ..... 343
    - 10.3.3 Descente de gradient ..... 344
    - 10.3.4 Recherche tabou ..... 346
    - 10.3.5 Recuit simulé ..... 348

# 10 \_\_\_\_\_ L'Intelligence Artificielle

pour les développeurs - Concepts et implémentations en C#

10.3.6	Optimisation par essais particulières . . . . .	351
10.3.7	Programme principal . . . . .	354
10.4	Résultats obtenus . . . . .	356
11.	Synthèse . . . . .	360

## Chapitre 6 Systèmes multi-agents

1.	Présentation du chapitre . . . . .	363
2.	Origine biologique . . . . .	364
2.1	Les abeilles et la danse . . . . .	364
2.2	Les termites et le génie civil . . . . .	366
2.3	Les fourmis et l'optimisation de chemins . . . . .	367
2.4	Intelligence sociale . . . . .	368
3.	Systèmes multi-agents . . . . .	368
3.1	L'environnement . . . . .	368
3.2	Les objets . . . . .	369
3.3	Les agents . . . . .	369
4.	Classification des agents . . . . .	370
4.1	Perception du monde . . . . .	370
4.2	Prise des décisions . . . . .	370
4.3	Coopération et communication . . . . .	371
4.4	Capacités de l'agent . . . . .	372
5.	Principaux algorithmes . . . . .	373
5.1	Algorithmes de meutes . . . . .	373
5.2	Optimisation par colonie de fourmis . . . . .	374
5.3	Systèmes immunitaires artificiels . . . . .	376
5.4	Automates cellulaires . . . . .	377
6.	Domaines d'application . . . . .	379
6.1	Simulation de foules . . . . .	379
6.2	Planification . . . . .	380
6.3	Phénomènes complexes . . . . .	380

7.	Implémentation	381
7.1	Banc de poissons	381
7.1.1	Les objets du monde et les zones à éviter	382
7.1.2	Les agents-poissons	384
7.1.3	L'océan	392
7.1.4	L'application graphique	395
7.1.5	Résultats obtenus	399
7.2	Tri sélectif	401
7.2.1	Les déchets	401
7.2.2	Les agents nettoyeurs	404
7.2.3	L'environnement	408
7.2.4	L'application graphique	412
7.2.5	Résultats obtenus	416
7.3	Le jeu de la vie	417
7.3.1	La grille	418
7.3.2	L'application graphique	421
7.3.3	Résultats obtenus	424
8.	Synthèse	426

**Chapitre 7**

**Réseaux de neurones**

1.	Présentation du chapitre	429
2.	Origine biologique	430
3.	Le neurone formel	432
3.1	Fonctionnement général	432
3.2	Fonctions d'agrégation	433
3.3	Fonctions d'activation	434
3.3.1	Fonction "heavyside"	434
3.3.2	Fonction sigmoïde	435
3.3.3	Fonction gaussienne	435
3.4	Poids et apprentissage	436

# 12 \_\_\_\_\_ L'Intelligence Artificielle

pour les développeurs - Concepts et implémentations en C#

4.	Perceptron . . . . .	437
4.1	Structure . . . . .	437
4.2	Condition de linéarité . . . . .	438
5.	Réseaux feed-forward . . . . .	439
6.	Apprentissage . . . . .	440
6.1	Apprentissage non supervisé . . . . .	440
6.2	Apprentissage par renforcement . . . . .	442
6.3	Apprentissage supervisé . . . . .	442
6.3.1	Principe général . . . . .	442
6.3.2	Descente de gradient . . . . .	443
6.3.3	Algorithme de Widrow-Hoff . . . . .	445
6.3.4	Rétropropagation . . . . .	445
6.4	Surapprentissage et généralisation . . . . .	447
6.4.1	Reconnaître le surapprentissage . . . . .	448
6.4.2	Création de sous-ensembles de données . . . . .	449
7.	Autres réseaux . . . . .	450
7.1	Réseaux de neurones récurrents . . . . .	450
7.2	Cartes de Kohonen . . . . .	450
7.3	Réseaux de Hopfield . . . . .	451
8.	Domaines d'application . . . . .	451
8.1	Reconnaissance de patterns . . . . .	452
8.2	Estimation de fonctions . . . . .	452
8.3	Création de comportements . . . . .	452
9.	Implémentation d'un MLP . . . . .	453
9.1	Points et ensembles de points . . . . .	453
9.2	Neurone . . . . .	457
9.3	Réseau de neurones . . . . .	459
9.4	IHM . . . . .	463
9.5	Système complet . . . . .	463
9.6	Programme principal . . . . .	467

9.7 Applications .....468  
    9.7.1 Application au XOR.....468  
    9.7.2 Application à Abalone .....470  
    9.7.3 Améliorations possibles .....472  
10. Synthèse du chapitre .....472

**Bibliographie**

1. Bibliographie .....475

**Sitographie**

1. Pourquoi une sitographie ? .....479  
2. Systèmes experts .....479  
3. Logique floue.....481  
4. Algorithmes génétiques .....483  
5. Recherche de chemins .....484  
6. Métaheuristiques .....485  
7. Systèmes multi-agents .....486  
8. Réseaux de neurones .....488

**Annexe**

1. Installation de SWI-Prolog.....491  
2. Utilisation de SWI-Prolog .....492

Index .....495

## 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 C# est présenté et appliqué à une recherche de chemins dans un environnement en 2D.

Le chapitre se termine par une synthèse.

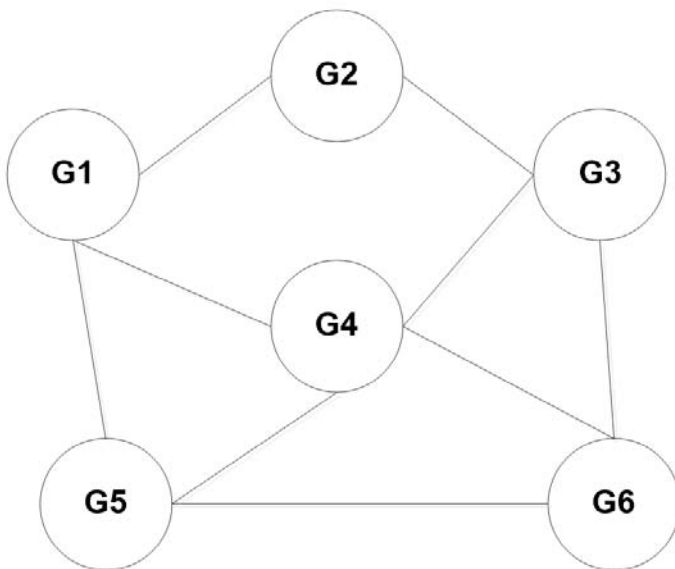
## 2. Chemins et graphes

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**.

# 162 \_\_\_\_\_ L'Intelligence Artificielle

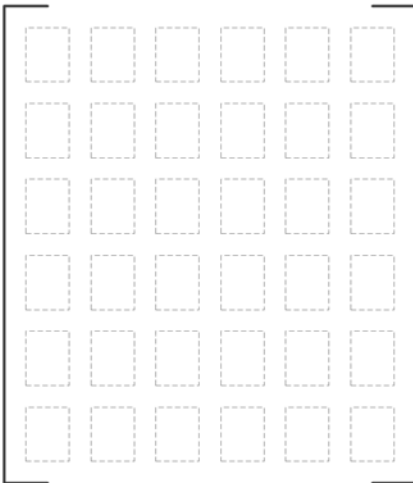
pour les développeurs - Concepts et implémentations en C#

## ■ 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.

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 :

VERS →

	G1		G4			
G1				1		
G4	1					

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					

# 164 L'Intelligence Artificielle

pour les développeurs - Concepts et implémentations en C#

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