collection DataPro

L'Intelligence Artificielle
pour les développeurs
Concepts et implémentations en C#

Virginie MATHIVET

Téléchargement
www.editions-eni.fr

eni éditions

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.

Chapitre 2
Logique floue

Chapitre 4
Algorithmes génétiques

Chapitre 5
Métaheuristiques d'optimisation

Chapitre 6
Systèmes multi-agents

Chapitre 7
Réseaux de neurones

Bibliographie

Sitographie

Annexe

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.

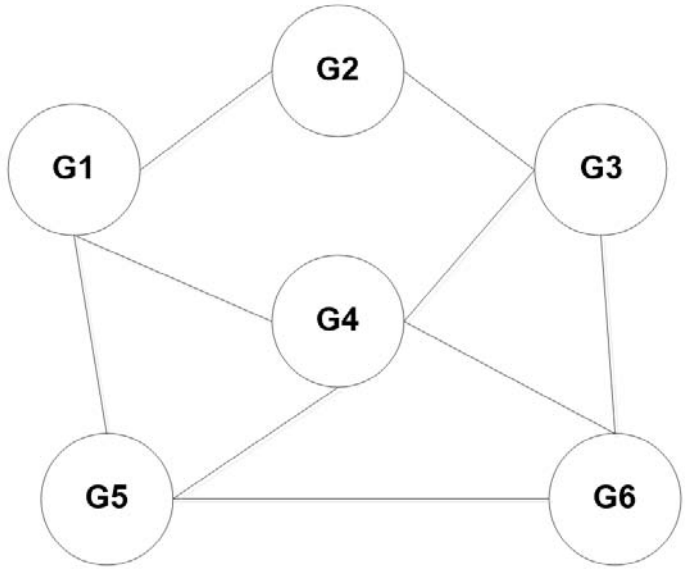
2. Chemins et graphes

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.

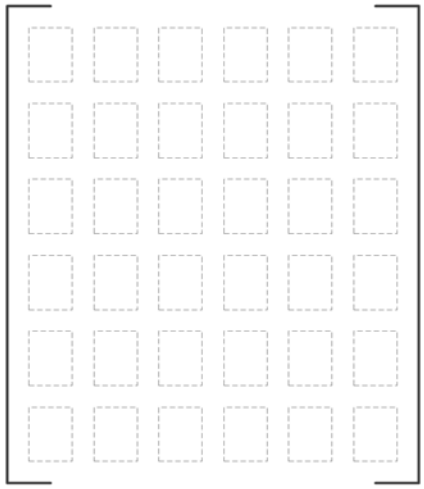
On préfère souvent utiliser une matrice, appelée **matrice d'adjacence**.

Remarque

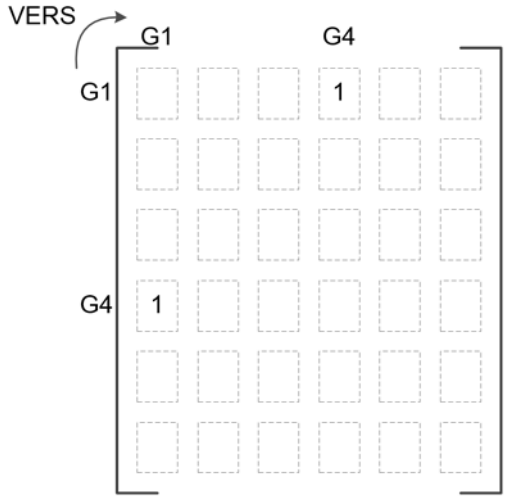
Une matrice est une structure mathématique particulière qui peut être vue plus simplement comme un tableau à deux dimensions.

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

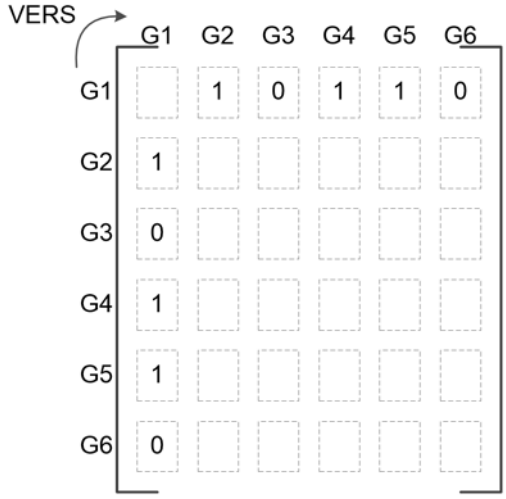
Dans l'exemple des gares, on a donc une matrice de 6 par 6 (car il y a 6 gares) :



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 :



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



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