

V – Algorithme Min-Max

M. Solnon

PT, Lycée Gustave Eiffel, Dijon

6 octobre 2026

Généralités

Conventions

- J_0 : Maximilien (Max)
- J_1 : Minerva (Min)

Heuristique

- Fonction d'évaluation d'une position, dans $\mathbb{R} \cup \{-\infty, +\infty\}$.
- Plus l'heuristique est grande, plus la position est favorable à Max
- Plus l'heuristique est petite, plus la position est favorable à Min

Exemple d'heuristique : morpion 3 × 3

Pour chaque case vide, pour chaque alignement de trois cases (ex. pour la case du milieu) dont elle fait partie.

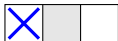
Max : croix bleues , Min : cercles rouges

Pour Max

1 point



10 points



100 points



0 point



Pour Min

-1 point



-10 points



-100 points



0 point



Exemple d'heuristique : morpion 3×3

Pour chaque case vide, si Max jouait dans la case, compter les alignements dans lesquels Min n'a pas joué :

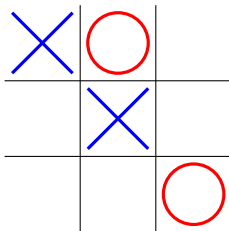
- 100 points par alignement de 3 dont fait partie la case ;
- 10 points par alignement de 2 dont fait partie la case ;
- 1 points par alignement de 1 dont fait partie la case ;

Pour chaque case vide, si Min jouait dans la case, compter les alignements dans lesquels Max n'a pas joué :

- -100 points par alignement de 3 dont fait partie la case ;
- -10 points par alignement de 2 dont fait partie la case ;
- -1 points par alignement de 1 dont fait partie la case ;

Calculer cette heuristique sur cette position

Max : croix bleues, Min : cercles rouges



Algorithme Min-Max : principe

Pour Max

Jouer sur le coup d'heuristique/évaluation maximale.

Pour Min

Jouer sur le coup d'heuristique/évaluation minimale.

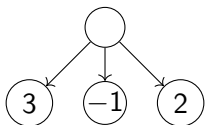
Évaluation de la position

C'est celle du coup choisi

Exemple

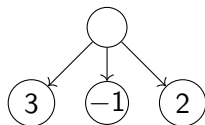
Max

Min



Min

Max



Min



Quand utilise-t-on l'heuristique ?

- On fixe une profondeur d'exploration $n \geq 1$.
- On explore les positions jusqu'à une « profondeur au plus n » de la position que l'on évalue.
- On utilise l'heuristique :
 - pour les cases où la profondeur maximale est atteinte ;
 - sans coup jouable (match nul ou victoire).

Fonctions primitives

- `coups_possibles(position)`
- `joueur_courant(position)`
- `heuristique(position)`

Récupérer le coup de score maximal

```
Fonction posScoreMax(listePosSc):  
    posMax,scoreMax ← listePosSc[0]  
    Pour chaque position,score De  
listePosSc[1:], Faire  
        Si score>scoreMax Alors  
            posMax,scoreMax ← position,score  
    Renvoyer posMax,scoreMax  
Fin Fonction
```

Récupérer le coup de score minimal

```
Fonction posScoreMin(listePosSc):  
  | posMin,scoreMin ← listePosSc[0]  
  | Pour chaque position,score De  
listePosSc[1:], Faire  
  | | Si score<scoreMin Alors  
  | | | posMin,scoreMin ← position,score  
  | Renvoyer posMin,scoreMin  
Fin Fonction
```

Algorithme du Min-Max naïf

```
Fonction minimax(position,profondeur):  
  listeCoups ← coups_possibles(position)  
  Si profondeur = 0 ou listeCoups est vide  
Alors  
    Renvoyer -1,heuristique(position)  
  Sinon
```

Algorithme du Min-Max naïf

```
listePosSc ← liste vide
Pour chaque posSuiv De listeCoups, Faire
|   _,score ← minimax(posSuiv,n-1)
|   ajouter (posSuiv,score) à listePosSc
joueur ← joueur_courant(position)
Si joueur = 0 Alors
|   Renvoyer posScoreMax(listePosSc)
Sinon
|   Renvoyer posScoreMin(listePosSc)
Fin Fonction
```

Algorithme du Min-Max mémorisé

```
Fonction minimax_memoisé(position,profondeur) :  
    memo ← dictionnaire vide  
    Fonction aux(position,profondeur) :  
        Si position ∈ memo Alors  
            cpMemo,scMemo,profMemo  
← memo[position]  
        Si profMemo ≥ profondeur Alors  
            Renvoyer (positionMemo,scMemo)
```

Algorithme du Min-Max mémorisé

```
listeCoups ← coups_possibles(position)
Si profondeur = 0 ou listeCoups est vide
Alors
    score ← heuristique(position)
    memo[position] ← -1,score,profondeur
    Renvoyer -1,score
listePosSc ← liste vide
Pour chaque posSuiv De listeCoups, Faire
    _,score ← aux(posSuiv,n-1)
    ajouter (posSuiv,score) à listePosSc
```

Algorithme du Min-Max mémorisé

```
joueur ← joueur_courant(position)
Si joueur = 0 Alors
  | coup,score ← posScoreMax(listePosSc)
Sinon
  | coup,score ← posScoreMin(listePosSc)
memo[position] ← coup,score,profondeur
Renvoyer coup,score
Fin Fonction
Renvoyer aux(position,profondeur)
Fin Fonction
```