ETUDE COMPARATIVE DES ALGORITHMES A*

Situation : instance3.txt

Départ (464, 416)

Arrivé (61, 69)

A* sans pondération

```
********** RESULTAT A STAR **********
Point de départ : (464, 416)
Point d'arrivé : (61, 69)
 Longueur du chemin : 757
 Nombre des sommets visités : 40922
```

Résultats :

- Longueur du chemin : **757**
- Sommets visités : **40 922**

A * (pondéré) version 1 : ***f(n) = w * g(n) + (1-w)*h(n) , avec 0 < w < 1***

1. ***w = 0.2*** (proche de 0 : approche gloutone)

   ```
   ********** RESULTAT A STAR  WEIGHTED V1 **********
   Point de départ : (464, 416)
   Point d'arrivé : (61, 69)
    Longueur du chemin : 1134
    Nombre des sommets visités : 7463
   ```

   Résultats :

   - Longueur du chemin : **1 134**
   - Sommets visités : **7 463**

2. ***w = 0.5*** (équilibre entre le coût réel et l'heuristique)

   ```
   ********** RESULTAT A STAR  WEIGHTED V1 **********
   Point de départ : (464, 416)
   Point d'arrivé : (61, 69)
    Longueur du chemin : 757
    Nombre des sommets visités : 39861
   ```

   Résultats :

   - Longueur du chemin : **757**
   - Sommets visités : **39 861**

3. ***w = 0.8*** (proche de 1 : prudence de Dijkstra)

```
********************* RESULTAT A STAR  WEIGHTED V1 ***********************
Point de départ : (464, 416)
Point d'arrivé : (61, 69)
 Longueur du chemin : 757
 Nombre des sommets visités : 164174
```

Résultats :

- Longueur du chemin : **757**
- Sommets visités : **164 174**

A* (pondéré) version 2 : ***f(n) = g(n) + w * h(n) , avec w > 1***

1. ***w = 1.3*** (proche de 1 : acceptable (vitesse et optimalité du chemin))

```
********************* RESULTAT A STAR  WEIGHTED V1 ***********************
Point de départ : (464, 416)
Point d'arrivé : (61, 69)
 Longueur du chemin : 771
 Nombre des sommets visités : 4875
```

Résultats :

- Longueur du chemin : **771**
- Sommets visités : **4 875**

2. ***w = 1.7*** (proche de 2 : peu optimal mais meilleur que la *version 1* avec ***w*** proche de **0**)

```
********************* RESULTAT A STAR  WEIGHTED V1 ***********************
Point de départ : (464, 416)
Point d'arrivé : (61, 69)
 Longueur du chemin : 810
 Nombre des sommets visités : 5040
```

Résultats :

- Longueur du chemin : **810**
- Sommets visités : **5040**

3. ***w = 2.5*** (approche gloutone et sous-optimalité du chemin)

```
********************* RESULTAT A STAR  WEIGHTED V1 ***********************
Point de départ : (464, 416)
Point d'arrivé : (61, 69)
 Longueur du chemin : 1016
 Nombre des sommets visités : 7792
```

Résultats :

- Longueur du chemin : **1 016**
- Sommets visités : **7 7792**

A* (pondéré) version 3 : ***f(n) = g(n) + w * h(n) , avec w > 1 dynamique***

fonction d'ajustement de w

```julia
# Ajustement du coéfficient de pondération de l'heurisque en fonction du contexte
function ajuster(sommet::Sommet, graphe::Graphe, dist_initial::Float64)
    progression = sommet.distance / dist_initial  # taux de progression dans le graphe
    # Pondération dynamique selon la progression
    if progression <= 0.3
        # Phase initiale : priorité à une exploration rapide (glouton)
        if sonder_niveau(sommet, graphe) > 9
            return 2.5  # Aggressif dans les zones ouvertes
        else
            return 1.4  # Prudence modérée
        end
    elseif progression <= 0.7
        # Phase intermédiaire : équilibre entre vitesse et prudence
        if sonder_niveau(sommet, graphe) > 9
            return 1.25  # Balance entre exploration et objectif
        else
            return 1.15  # Progression prudente
        end
    else
        # Phase finale : priorité à l'optimalité (focus sur l'objectif)
        if sonder_niveau(sommet, graphe) > 9
            return 1.0  # Approche standard A*
        else
            return 0.6  # Avancer avec prudence
        end
    end
end
```

fonction du sondage des voisins visitables

```julia
# Sonder les voisins qui ne sont pas des obstacles
function sonder_voisins(sommet::Sommet, graphe::Graphe)
    nb_colonnes = size(graphe.matrice, 2)
    coord = number_to_coord(sommet.numero, nb_colonnes)
    voisins = 0
    # Nord
    if(coord[1] != 1 && est_visitable(graphe.matrice[coord[1] - 1, coord[2]], graphe.liste_couleurs))
        voisins += 1
    end
    # Est
     if(coord[2] != size(graphe.matrice, 2)  && est_visitable(graphe.matrice[coord[1], coord[2] + 1], graphe.liste_couleurs))
        voisins += 1
    end
    # Sud
    if(coord[1] != size(graphe.matrice, 1)  && est_visitable(graphe.matrice[coord[1] + 1, coord[2]], graphe.liste_couleurs))
        voisins += 1
    end
    # Ouest
    if(coord[2] != 1  && est_visitable(graphe.matrice[coord[1], coord[2] - 1], graphe.liste_couleurs))
        voisins += 1
    end
    return voisins
end
```

fonction du sondage des voisins visitables au niveau 2

```julia
function sonder_niveau(sommet::Sommet, graphe::Graphe)
    nb_colonnes = size(graphe.matrice, 2)
    coord = number_to_coord(sommet.numero, nb_colonnes)
    voisins = 0
    # Nord
    if(coord[1] != 1 && est_visitable(graphe.matrice[coord[1] - 1, coord[2]], graphe.liste_couleurs))
        voisins += sonder_voisins(sommet, graphe)
    end
    # Est
    if(coord[2] != size(graphe.matrice, 2)  && est_visitable(graphe.matrice[coord[1], coord[2] + 1], graphe.liste_couleurs))
        voisins += sonder_voisins(sommet, graphe)
    end
    # Sud
    if(coord[1] != size(graphe.matrice, 1)  && est_visitable(graphe.matrice[coord[1] + 1, coord[2]], graphe.liste_couleurs))
        voisins += sonder_voisins(sommet, graphe)
    end
    # Ouest
    if(coord[2] != 1  && est_visitable(graphe.matrice[coord[1], coord[2] - 1], graphe.liste_couleurs))
        voisins += sonder_voisins(sommet, graphe)
    end
    return voisins
end
```

Résultats :

```
********************* RESULTAT A STAR  WEIGHTED V3 ***********************
Point de départ : (464, 416)
Point d'arrivé : (61, 69)
 Longueur du chemin : 874
 Nombre des sommets visités : 13725
```

- Longueur du chemin : **874** - Sommets visités : **13 725**

Tolérable!