**Etat de l’art :   
plus court chemin, algorithme de résolution**

Le problème du plus court chemin consiste, en théorie des graphes, à trouver un chemin d’un sommet (soit n le nombre de sommet) à un autre où la somme des poids des arcs (soit a le nombre d’arc) intermédiaires est minimale.

Plusieurs algorithmes permettent de résoudre cette problématique. Chaque problème a ses particularités : graphe orienté, poids négatif, basse ou haute densité, calcul entre 2 points, entre 1 point et tous les autres ou entre toutes les paires de points. Pour entrer le graphe en paramètre de l’algorithme, on peut le représenter par une liste d’adjacence ou d’une matrice d’adjacence. La taille de la liste d’adjacence est comprise entre n et n² (avec n le nombre de sommet) alors que la matrice d’adjacence est de taille n². Dans le cas d’un graphe peu dense la matrice aurait beaucoup de cases vides, ainsi il est plus intéressant d’utiliser la liste. En revanche si le graphe est dense, la matrice sera plus remplie, la matrice semblera plus appropriée.

L’algorithme de Dijkstra porte le nom de son inventeur (1959) l’informaticien Néerlandais Edsger Dijkstra. Dijkstra permet de calculer le plus court chemin depuis une source vers tous les autres points d’un graphe orienté et pondéré par des réels positifs. En prenant en entrée une liste d’adjacence et en utilisant la file de priorité par un tas binaire on a une complexité en O((n+a)\*log(n)), si la file de priorité est un tas de Fibonacci, on obtient une complexité en O(a+n\*log(n))

L’algorithme de A\* (A star) est issu de Dijkstra, il introduit une nouvelle notion, la notion d’heuristique. Il permet d’obtenir un résultat rapidement pour déterminer le plus court chemin entre 2 points mais dont l’exactitude n’est pas certaine. Il va parcourir le graphe « en direction » de la destination en priorisant selon une heuristique. Chaque sommet a un poids qui est la somme des poids des arcs entre lui et la source. L’heuristique du sommet est le cout du sommet plus une valeur donnant une indication sur la proximité avec la destination. Par exemple : le cout est la somme des distances entre les étapes d’un chemin et la valeur ajoutée serait la distance à vol d’oiseau avec l’arrivée. La file de priorité est alors gérée selon cette heuristique. L’algorithme permet de parcourir seulement partiellement le graphe, c’est un gain de temps mais une perte en fiabilité.

L’algorithme de Bellman-Ford calcule le plus court chemin entre deux points (single source) en utilisant le principe de programmation dynamique (ordonnancement). Ses poids peuvent être négatifs. Il permet de détecter les circuits absorbants mais ne résout pas le problème s’il en rencontre un. Sa complexité est de O(n\*a). Avec n < a < n², sa complexité est forcément supérieure à n² et peut aller jusqu’à n^3.

Les algorithmes de Floyd-Warshall et de Johnson permettent la résolution du plus court chemin all pair, c’est-à-dire du plus court chemin entre chaque paire de sommets. Leur complexité sont respectivement n^3 et O(n²\*log(n)+n\*a).

Après cette recherche, on estime que l’algorithme de Dijkstra est le plus adapté à notre problématique, il calcule le plus court chemin depuis une source avec la meilleure complexité. Son instance A\* pourrait permettre de gagner en temps au dépend d’une fiabilité du résultat. Ici les temps de résolutions restant faibles, on préfère obtenir LE chemin le plus court grâce à Dijkstra.

