Routage maritime pour les voiliers de course

Accès aux programmes / données : https://github.com/ErwanFagnou/Routage-maritime-pour-les-voiliers-de-course

Dans le cadre du TIPE, projet de recherche à préparer pour les concours aux grandes écoles, j'ai choisi le routage maritime comme sujet. Il s'agit d'un problème d'optimisation consistant à calculer le meilleur trajet pour un navire entre deux points, en s'appuyant sur les estimations des vents, des courants et des vagues notamment. Selon les critères choisis, on peut chercher à minimiser le temps du trajet ou le carburant utilisé, éviter les intempéries, limiter les risques, etc. La difficulté réside dans le fait que la vitesse est variable, et que les voiliers, n'ayant pas de moteur, ne contrôlent pas leur trajectoire en l'absence de vent. Dans mon cas je me suis placé dans le contexte d'une course, cherchant donc le chemin le plus rapide.

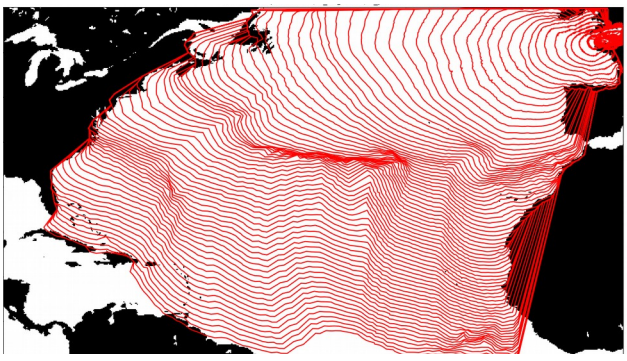
J'ai commencé par me demander comment estimer fidèlement la vitesse d'un voilier selon la météo, tout en cherchant des données que je pourrais utiliser pour tester mes algorithmes. J'ai ainsi obtenu des cartes contenant la direction et la force du vent pendant l'édition de 2018 de la Route du Rhum, une carte du monde permettant de distinguer terre et mer, et des fichiers appelées polaires de vitesse, qui donnent la vitesse d'un voilier selon la force et la direction du vent. Mon premier modèle ne prenait ainsi que le vent en compte.

Le premier algorithme que j'ai conçu était plutôt simple : on considère que le bateau évolue dans une grille, et on « propage » le bateau de case en case. A chaque itération, on regarde les cases voisines des cases où le bateau peut se trouver, et on les marque comme accessibles elles aussi, tout en mémorisant le temps écoulé et la case précédente. Lorsque la case où se situe l'arrivée est accessible, cela signifie qu'on a trouvé un chemin. Cette approche était plutôt peu précise car le chemin était restreint à la grille, et malgré une complexité moyenne correcte, comme chaque case peut être traitée plusieurs fois, le temps de calcul était mauvais.

La méthode par isochrones s'est avérée la plus efficace, et m'a permis de prendre en compte les courants marins, ce qui était impossible avec le premier algorithme. Le principe est de délimiter, par une courbe formée de points, la zone accessible pendant une durée t_1 depuis le point de départ. À partir de cette isochrone on peut alors calculer l'isochrone suivante, correspondant à une durée t_1+dt, et ainsi de suite jusqu'à l'arrivée. Plus précisément, pour chaque point de l'isochrone, on essaie d'aller pour une durée de dt dans toutes les directions. On obtient un nuage de points dont il faut déterminer le contour pour obtenir l'isochrone suivante.

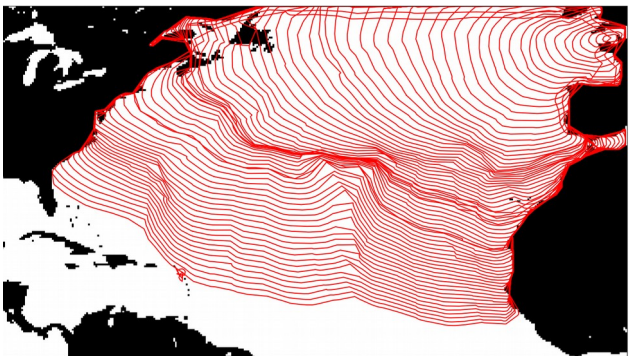
C'est cette étape là qui m'a le plus intéressé : trouver le contour d'un nuage de points – sur une sphère qui plus est – est un problème difficile à résoudre en un temps raisonnable. J'ai ainsi implémenté 3 algorithmes différents :

- H. Hagiwara propose dans sa thèse de découper la sphère (i.e. la Terre) en segments partants du point de départ. Pour obtenir le contour du nuage de points, on sélectionne pour chaque segment le point le plus proche de l'arrivée. Cette approximation marche très bien lorsque le chemin est relativement droit, mais supporte mal les obstacles. Son point fort est la complexité est en O(n), où n est le nombre de points, qui permet un calcul très rapide.



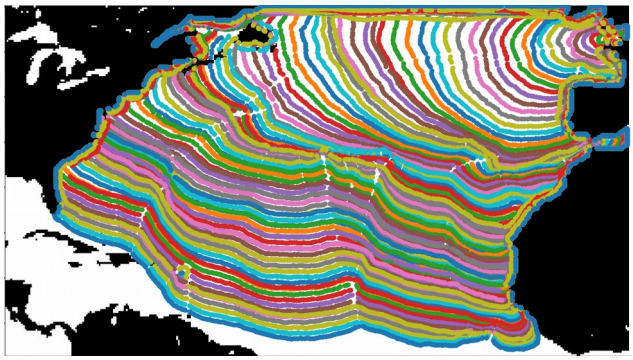
Isochrones (Hagiwara)

- Inspiré de la marche de Jarvis, algorithme de calcul d'enveloppe convexe, j'ai développé un algorithme qui fait le tour du nuage de points. En imposant une distance maximale entre deux points consécutifs, l'algorithme choisit à chaque tour le point le plus « à droite» du point précédent, et construit ainsi le contour point par point. Mais la complexité est en $O(h^2 \log(n) + hn)$ (avec h le nombre de points appartenant au contour), ou encore $O(n^{3/2})$ en supposant $h^2=O(n)$. Le problème principal est qu'une seule erreur peut bloquer tout le programme, ce qui peut arriver lorsque des angles sont très petits.



Isochrones (Jarvis)

- J'ai ensuite découvert l'*alpha-shape* d'un nuage de points, qui est un contour défini de manière plus rigoureuse. L'avantage est que l'alpha-shape est un ensemble de segments unique, et que l'on peut vérifier indépendamment pour chaque segment s'il appartient à l'alpha-shape ou non. L'algorithme que j'ai implémenté utilise la triangulation de Delaunay, liée directement à l'alpha-shape. Via une projection stéréographique, on peut projeter le nuage de points sur un plan, appliquer un algorithme de triangulation de Delaunay sur ce plan, puis en déduire l'alpha-shape. La complexité est cette fois en $O(n \log(n))$, ce qui est optimal étant donné que les algorithmes de calcul d'enveloppe convexe sont au mieux en $O(n \log(n))$.



Isochrones (alpha-shape)

	Réalité	1er algorithme (sur grille)	Isochrones (Hagiwara)	Isochrones (Jarvis)	Isochrones (alpha-shape)
Temps de trajet du voilier	12j 11h	12j 15h	10j 12h	9j 18h	9j 6h
Temps de calcul	/	357 s	30 s	285 s	189 s
Prise en compte des courants	Oui	Non	Non	Oui	Oui

Tableau comparatif des différents algorithmes

On peut remarquer que plus l'algorithme est précis, plus le temps de trajet est faible. Comme je n'ai pris en compte que le vent et les courants (en négligeant les vagues ou autres facteurs influençant la vitesse du voilier), il est normal d'obtenir un temps plus court qu'en réalité.

De plus le temps d'exécution dépend des paramètres choisis pour chaque algorithme, donc il n'est que partiellement représentatif.

L'algorithme isochrones (Hagiwara) est de loin le plus rapide, et s'avère le plus efficace pour des trajets simples sans gros obstacles. Cependant l'algorithme isochrones (alpha-shape) est à mon avis le plus précis et le plus prometteur : en effet, de nombreuses optimisations peuvent encore être effectuées, comme imposer une distance minimale entre deux points (et donc réduire drastiquement le temps de calcul), ou encore privilégier les directions qui vont vers l'extérieur du nuage de points.