Un algorithme de routage de navires générant des trajets précis et diversifiés

Alexandre Coppé [1], Nicolas Prcovic[1]

[1] Aix Marseille Univ, Université de Toulon, CNRS, LIS, Marseille, France

6 mai 2024

Résumé

Nous présentons un algorithme déterminant avec précision des trajectoires optimales d'un navire dans un contexte multi-objectif et dynamique, où il faut notamment prendre en compte le temps de trajet et la consommation de carburant, dans des conditions météorologiques qui varient pendant le trajet. Notre approche combine deux algorithmes récents, NAMOA-TD et WRM, nous permettant d'obtenir un panel de trajectoires (sous-ensemble du front de Pareto) précises et diversifiés parmi lesquels un utilisateur peut choisir. Les premières expérimentations effectuées à partir de données météorologique réelles nous permettent de montrer l'efficacité de cette approche.*

Mots-clés

Routage de navires, chemin optimal dans un graphe, recherche multi-objective

Abstract

We present an algorithm that determines optimal ship trajectories in a multi-objective and dynamic context, where factors such as travel time and fuel consumption, amidst varying meteorological conditions along the route, need to be considered. Our approach combines two recent algorithms, NAMOA-TD and WRM, enabling us to generate a set of precise and diversified trajectories (a subset of the Pareto front) from which a user can choose. Initial experiments conducted using real meteorological data demonstrate the effectiveness of this approach.*

Keywords

Ship routing, optimal path in a graph, multi-objective search.

1 Introduction

Plus de 80% du volume des échanges commerciaux mondiaux se traite par la mer. En 2018, au sein du transport maritime, 40% des frais opérationnels étaient absorbés par le coût du carburant lors d'un voyage. Une petite amélioration, aussi minime soit-elle, peut avoir de grandes répercussions au niveau des coûts. Le routage des navires en fonction de la météo est donc un domaine qui génère un grand intérêt pour des raisons écologiques et économiques (cf figure 1).

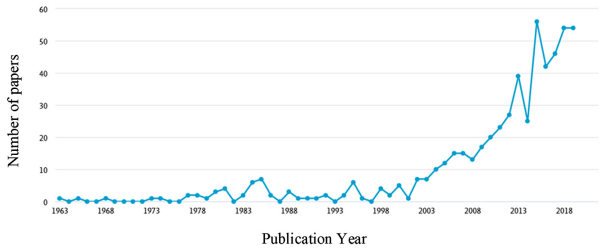

FIGURE 1 – Nombre de publication sur l'optimisation de trajet des navires. Source : Scopus February 2020.

L'objectif de nos travaux consiste à trouver un trajet de navire entre deux ports, en arrivant avant une date donnée, en optimisant la consommation de carburant et en prenant en compte les contraintes environnementales, tout ceci dans un contexte qui varie dynamiquement au cours du trajet (courants, météo).

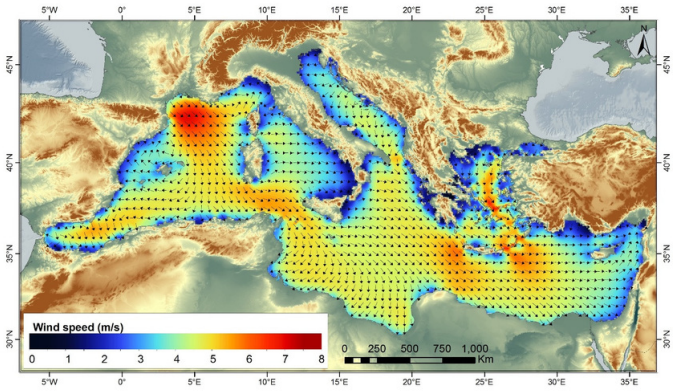

FIGURE 2 – Exemple de grille météo. Les flèches indiquent la direction du vent et les couleurs sa vitesse.

En plus de la géographie maritime, des lieux et dates de départ et d'arrivée, nous devons tenir compte des courants maritimes et de la météo. Ceux-ci nous sont données sous forme d'un pavage du globe découpé en cellules rectangulaires à l'intérieur desquelles le courant et le vent sont considérés comme étant uniformes (cf figure 2). Ces données nous permettent de calculer les différents coûts (temps,

carburant) liés au trajet qui sont fonctions du vent et des courants à un moment donné. Ces prévisions ne sont valables que pour une période donnée (typement, pendant 6h) et, pour un trajet, il nous faut récupérer les prévisions pour les périodes incluses entre la date de départ et d'arrivée du navire.

Bien que la modélisation la plus réaliste d'un trajet de navire soit une courbe continue, la numérisation des données implique leur discrétisation. L'espace et le temps numérisés seront donc un ensemble de lieux et de dates dont le nombre et la précision du repérage influeront sur l'efficacité des algorithmes et la qualité des solutions produites. Bien souvent, les lieux sont répartis uniformément sur une grille rectangulaire et forment un maillage couvrant toute la zone de déplacements possibles en se calquant sur la grille météo.

L'approche la plus classique est de représenter les lieux par un graphe dont les arcs relient les sommets correspondant aux lieux les plus proches (cf fig 3). Les arcs sont étiquetés par un vecteur de coûts (contenant au moins ceux du temps et du carburant utilisés pour joindre les deux lieux).

Des outils calculant des trajets dans ce contexte existent déjà et sont utilisés par des entreprises de routage maritimes. Cependant les solutions en usage pêchent par au moins deux aspects d'après les utilisateurs avec lesquels nous avons discutés :

- Les trajectoires fournies sont trop grossières lorsqu'on est proche des côtes.
- Un seul trajet est souvent proposé alors que, dans un contexte multi-objectif, il y a en a souvent un grand nombre possibles (dites Pareto-optimales, cf figure 4). Or, un commandant de navire peut avoir ses idées préconçues sur le meilleur trajet à effectuer donc il aura du mal à accepter une unique solution qui diffère trop de ce qu'il estime a priori être le mieux. D'où l'utilité de lui offrir un panel diversifié de solutions possibles parmi lesquelles il en trouvera une qui a plus de chances de lui convenir.

Le plan de cet article est le suivant. Nous commençons par donner une définition formelle du problème de recherche d'un trajet optimal dans un contexte multi-objectif et dynamique. Puis, nous présentons les algorithmes existants permettant de traiter ce problème. Ensuite, nous présentons notre approche, basée sur deux algorithmes récents, qui nous permet d'obtenir des trajectoires diversifiées et aussi précises que l'on veut, afin de satisfaire aux demandes des utilisateurs du domaine. Enfin, nous présentons des expérimentations à partir de données réelles fournies par l'entreprise de transport maritime avec laquelle nous collaborons.

1.1 Définition formelle du problème

Considérons un graphe orienté et muni d'une fonction de coût c défini par : G = (N, A , c) où N = $\{x_1..., x_n\}$ est un ensemble fini de sommets et A $\subset$ N $\times$ N un ensemble fini de |A| arcs de la forme (x_i,x_j). à chaque arc a $\in$ A est associé un vecteur de fonction de coûts à valeurs dans $\mathbb{N}$ de la forme c(a) = (c_1(a), . . . , c_q (a)) où q est le nombre de valuations du graphe, et c_i la fonction qui attribue à un arc

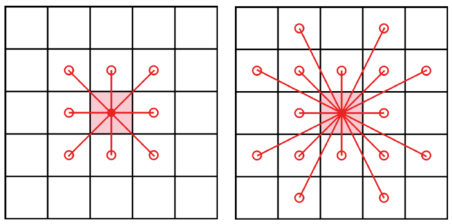

FIGURE 3 – Exemple de voisinage à 8 et 16 voisins où les sommets correspondent à des lieux répartis uniformément dans une grille rectangulaire, utilisé dans [1] .

du graphe sa i^{eme} valuation. s $\in$ N est le sommet source du graphe (départ), et p $\in$ N est le sommet puits du graphe (arrivée). Une solution est un chemin partant du sommet source et finissant par le sommet puits.

On s'intéresse aux chemins optimaux entre deux sommets du graphe. Soit CH_{ij} l'ensemble des chemins de x_i vers x_j dans le graphe G. On définit un vecteur de coût C qui à un chemin ch_{ij} associe l'évaluation des coûts pour parcourir ce chemin. On cherche à identifier le (ou les) chemin(s) $ch_{ij} \in CH_{ij}$ tels qu'il n'existe pas de chemin ch'_{ij} $\in CH_{ij}$ vérifiant C(ch'_{ij}) < C(ch_{ij}). L'opérateur $<$ compare des vecteurs, ce qui implique un ordre partiel sur le coût des chemins.

2 Recherche d'un chemin optimal dans un graphe

Il existe de nombreux algorithmes de recherche de chemins optimaux dans un graphe valué, qui se déclinent notamment selon s'ils sont mono ou multi-objectifs et si le(s) coût(s) des arêtes évoluent au cours du temps.

Tous ces algorithmes supposent que le trajet se fait à vitesse constante. La prise en compte d'un changement de vitesse ou de puissance du moteur au cours du trajet inclut une complexité encore plus grande qui n'est jamais pris en compte en pratique.

2.1 Algorithmes mono-objectifs

L'optimisation mono-objectif consiste à n'optimiser qu'un seul critère et permet d'obtenir une seule solution optimale.

L'algorithme le plus connu est celui de Dijkstra [2] qui utilise une approche gloutonne pour obtenir une solution optimale en temps $O(n \log n)$, où n est le nombre de sommets du graphe.

L'algorithme A* [4] est une variante de Dijkstra qui a été définie initialement pour traiter le cas où le graphe (trop grand voire infini) est défini en compréhension. Il évalue un chemin en construction non seulement en prenant en compte le coût du chemin déjà parcouru (comme Dijkstra) mais aussi une sous-évaluation heuristique du coût du chemin restant à parcourir.

En pratique A* permet de sélectionner plus rapidement le meilleur chemin et de diminuer notablement le temps de calcul pour obtenir le chemin le moins coûteux.

2.2 Algorithmes multi-objectifs

Les algorithmes multi-objectifs doivent optimiser plusieurs objectifs dont aucun objectif n'est dominant par rapport aux autres.
Dans le cas qui nous occupe, il nous faut prendre en compte le temps de trajet, le coût du carburant, l'impact environnemental, l'usure du navire, le confort des voyageurs (minimiser les changements de cap et de vitesse), etc.
Dès lors que l'on a plus un seul objectif, on n'a plus unicité de la solution optimale (plus précisément : une seule des solutions parmi celles qui ont le coût optimal) mais un nombre potentiellement exponentiel de vecteurs de coûts dont aucun ne domine l'autre, qu'on appelle *Front de Pareto*.

Front de Pareto Lorsqu'on a plusieurs critères, on ne peut dire qu'une solution est meilleure qu'une autre que si elle l'est sur tous les critères. On dit alors qu'elle la *domine*. Certaines solutions sont incomparables : une solution peut être meilleure sur un critère et moins bonne sur un autre.

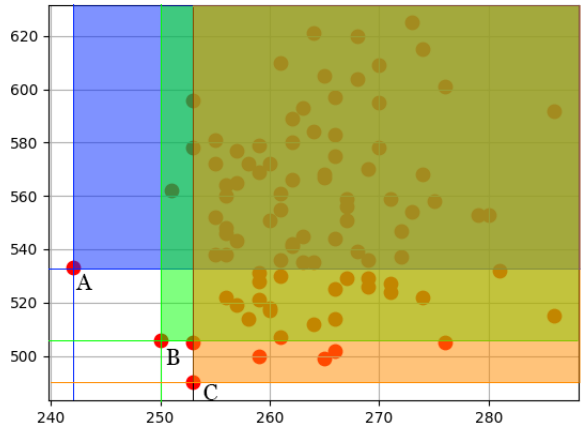

FIGURE 4 – Ensemble de solutions. On a en bleu, vert et orange les zones dominées par les sommets A, B et C, qui constituent un front de Pareto.

L'ensemble des solutions dominées par aucune autre solution s'appelle le *front de Pareto*. Une solution *pareto-optimale* a un vecteur de coût tel qu'il n'existe pas une solution alternative pour laquelle tous les éléments du vecteur de coût seraient meilleurs.
Un problème multi-objectif présente la difficulté qu'un algorithme le résolvant ne doit pas prendre en compte uniquement le coût de la meilleure solution pour éliminer d'autres candidats potentiels mais tous les vecteurs de coût du front de Pareto en cours de constitution. Le problème devient ainsi NP-difficile et donc potentiellement de complexité exponentielle.
MOA* [6] (Multi-Objective A*) et NAMOA* (New Approach to MOA*) [5] sont des variantes généralistes de A* prenant en compte le multi-objectif. MOA* prend en compte notamment le fait qu'un sommet est l'aboutissement de plusieurs chemins de vecteurs de coûts différents, qu'il faut prendre en compte pour évaluer les coûts des chemins aboutissant aux successeurs de ce sommet (ses extensions). NAMOA* améliore MOA* en considérant des extensions de chemins partiels plutôt que des extensions de sommets.

Scalarisation La scalarisation consiste à combiner linéairement différents critères pour en former un seul. Si on a trois critères c_1, c_2 et c_3, la scalarisation consiste à définir la fonction C(c_1,c_2, c_3) = $\alpha_1.c_1 + \alpha_2.c_2 + \alpha_3.c_3$, où les α_i sont des coefficients à fixer qui déterminent l'importance relative de chaque coût c_i.
En choisissant adéquatement les coefficients (par recherche dichotomique), on peut obtenir toute l'enveloppe convexe d'un front de Pareto en un temps polynomial. Il nous manquera juste les solutions qui sont dans les "cavités" de l'enveloppe convexe.
La scalarisation a l'immense avantage de permettre une résolution rapide mais, comme elle ne trouve que les solutions de l'enveloppe convexe du front de Pareto, il faut que celle-ci contienne un "bon" sous-ensemble de solutions par rapport au contexte.
Dans la suite de l'article, nous appellerons *Dijkstra scalaire*, un algorithme de Dijkstra effectuant du multi-objectif en scalarisant les coûts.

Algorithmes dépendant du temps Lorsque le coût des arêtes d'un graphe change au cours du temps (par exemple à cause de la météo), la propriété qu'un sous-chemin d'un chemin optimal est optimal devient fausse (cf figure 5). On ne peut plus s'en servir pour éliminer des chemins partiels sous-optimaux. Il faut utiliser d'autres critères plus faibles.

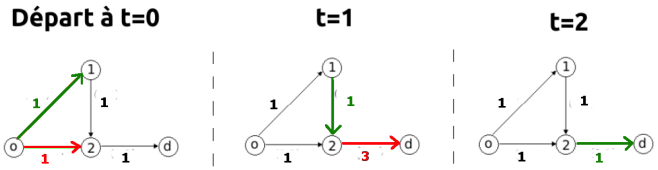

FIGURE 5 – Exemple de trajet pour lequel arriver plus tard en un sommet permet de minimiser un coût global de trajet : passer au temps 2 au sommet 1 permet de gagner en coût (1+1+1 au lieu de 1+3).

Dans le contexte du routage maritime, nous pouvons citer l'algorithme de Venetti [7] et NAMOA*-TD [1]. Pour permettre l'élimination de chemins partiels candidats, ces algorithmes ont deux critères :

- Le coût est supérieur à celui d'une solution déjà trouvée.
- Le coût est supérieur à celui d'un chemin partiel arrivant au même sommet à la même date.

NAMOA*-TD, une version "time-dependent" de NAMOA*, s'avère beaucoup plus efficace en pratique de l'algorithme de Venetti. Ceci est notamment dû au fait que, contrairement à Venetti, NAMOA*-TD utilise une heuristique d'estimation du coût du chemin restant à parcourir. Cette heuristique s'avère être une évaluation suffisamment proche du réel pour détecter tôt les chemins candidats qui

seront dominés.
Bien qu'apparaissant comme une méthode efficace dans le cadre du routage maritime international en permettant le calcul de trajets réalistes en moins de deux minutes, NAMOA*-TD a pourtant des limites qui l'empêchent d'être pleinement satisfaisant dans le contexte qui nous occupe. Le maillage rectangulaire et régulier qui détermine la forme des trajectoires n'est pas assez précis lors de la navigation proche des côtes ou dans des zones maritimes étroites (par exemple, la Manche) : la zone maritime peut être plus étroite que les mailles. La solution consistant à réduire la taille des mailles mène à une forte augmentation du temps de calcul. Par exemple, diviser la taille des mailles (rectangulaires) selon ses deux dimensions multiplie par quatre le nombre de sommets du graphe dans un contexte où la complexité temporelle des algorithmes est exponentielle en ce nombre de sommets.
En pratique, l'outil utilisé par l'armateur que nous avons consulté diminue les tailles des mailles dans certaines zones où il était constaté a posteriori que c'était nécessaire. L'inconvénient est que le graphe doit se construire en quelque sorte "à la main" et qu'il est difficile de juger précisément de la taille des mailles en fonction des lieux.
Or, il existe une approche récente, qui permet de fixer les lieux de passages d'un navire à un degré de précision arbitrairement grand, que nous allons pouvoir adapter à notre contexte.

2.3 L'approche de Weather Routing Metaheuristic

Weather Routing Metaheuristic (WRM) [3] constitue une approche originale du routage de navire qui ne fixe pas a priori les lieux de passages et permet de générer des trajets dont les lieux de passage peuvent se trouver n'importe où sur la surface de navigation à un degré de précision aussi élevé que l'on souhaite.
Le graphe se détermine en générant aléatoirement n lieux dans une zone donnée et en créant une arête entre deux lieux si la distance les séparant est inférieure à une constante donnée. Chaque segment liant deux lieux peut traverser plusieurs cellules rectangulaires de la grille météo indiquant les directions et forces des courants et des vents. On découpe donc le segment en sous-segments dont les extrêmités sont sur les frontières entre les cellules, chaque cellule indiquant un courant et une vitesse de vent dont la direction et la force sont constantes. On calcule le vecteur de coûts pour chacun de ces sous-segments selon un modèle donné par l'armateur en fonction du navire, puis on fait la somme de ces vecteurs pour obtenir le vecteur de coûts de tout le segment.
Le fait qu'on puisse choisir n à l'unité près permet de choisir précisément la taille de l'instance et donc le temps d'exécution de la méthode. Les coordonnées des lieux sont déterminés avec la précision que l'ont veut. Le fait que les lieux soient tirés au hasard (de manière uniforme) plutôt que fixés selon une grille régulière n'empêche pas la densité des points de rester à peu près uniforme sur toute la zone de navigation.
L'algorithme procède par itérations en cherchant un trajet dans le graphe puis en sélectionnant la zone géographique proche des lieux du trajet trouvé puis en générant aléatoirement n lieux dans cette zone restreinte pour former un nouveau graphe. Au fur et à mesure, la zone de proximité se réduit de plus en plus, ce qui affine la trajectoire progressivement (cf figure 6). Dans WRM, c'est un algorithme incomplet mono-objectif minimisant la consommation de carburant qui est utilisé mais rien n'empêche l'utilisation d'une autre procédure mono ou multi-objectif, complète ou incomplète. Si la procédure est multi-objectif, elle génère plusieurs chemins à chaque itération et il faut en choisir un seul pour l'itération suivante.

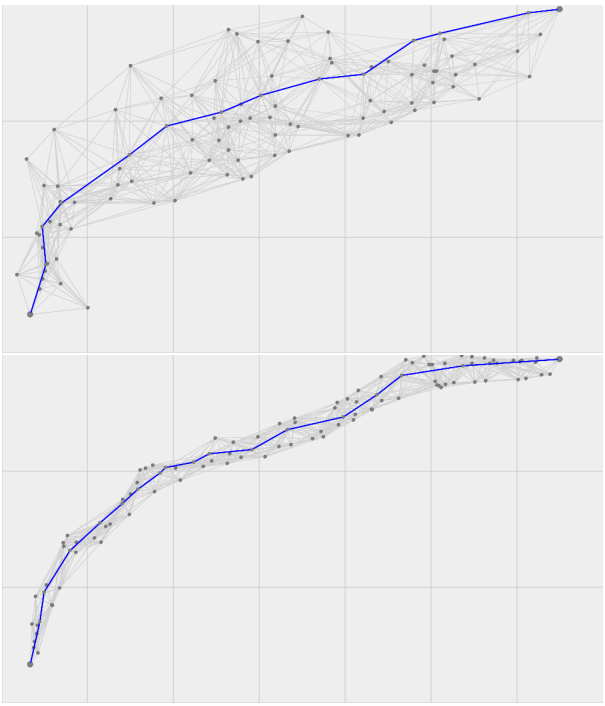

FIGURE 6 – Étape n (en haut) et étape n + 1 (en bas). La zone géographique des lieux possibles de l'étape n+1 est déterminée par la trajectoire trouvée à l'étape n. Images extraites de [3]

3 Recherche de trajets précis et diversifiés

Nous avons vu que NAMOA*-TD permettait d'obtenir un front de Pareto et donc tous les trajets possibles correspondant à une instance de problème donné mais avec une précision insuffisante dans un contexte de temps de calcul limitée. D'un autre côté, WRM permet de ne générer qu'un seul trajet mais avec un degré de précision important. Pour obtenir des trajets précis et diversifiés dans un temps court, nous proposons l'approche simple suivante.

— Utiliser un algorithme multi-objectif (NAMOA*-TD ou un Dijkstra scalaire) afin de générer un front de Pareto pour avoir des trajets diversifiés (mais peu précis).

— Sélectionner un petit nombre k de trajets du front de Pareto qui diffèrent le plus possible les uns des autres.
— Pour chacun des k trajets obtenus précédemment : lancer WRM en générant le premier graphe avec des lieux pris dans la zone de proximité du trajet. On espère ainsi raffiner progressivement une solution pour abaisser ses coûts.

Pour que le temps de calcul reste court, il faut que la grille de lieux utilisée par NAMOA*-TD sur une grille rectangulaire aient des mailles plus grosses que si NAMOA*-TD était utilisé seul afin de laisser du temps au reste de l'algorithme. Nous obtenons des trajets encore moins précis qu'avant mais ce n'est pas gênant car ils sont destinés à être raffinés juste après.

Pour la sélection des k trajets du front de Pareto, nous nous basons sur les vecteurs de coûts et faisons en sorte que leurs valeurs soient le plus uniformément réparties.

Concernant la recherche de chemins pareto-optimaux pendant une itération de WRM, nous testerons deux algorithmes :

— NAMOA*-TD pour être complet (au détriment du temps d'exécution)
— Dijkstra scalaire avec un coût égal à une scalarisation des différents coûts afin d'être rapide (au détriment de la complétude).

WRM est censé resserrer progressivement la zone où peuvent apparaître les sommets du prochain graphe autour du trajet actuel. Mais dans les cas où le trajet actuel n'est pas meilleur que le précédent, nous nous autorisons à élargir ponctuellement cette zone afin d'augmenter la chance de trouver un meilleur trajet lors de l'itération suivante.

4 Expérimentations

Les données des courants et météo sont issus de fichiers GRIB correspondant à des données réelles. Les algorithmes ont été testés sur une plage météo de 10 jours en prenant plusieurs trajets différents commençant à différents horaires. Nous avons utilisé un ordinateur 64 bits cadencé à 2.1 Ghz et ayant 192 Go de RAM.

Nous n'avons pour l'instant considéré que deux critères : le temps et le carburant consommé. Le modèle de consommation nous a été fourni par un armateur et ne constitue qu'une approximation pour un seul type de ses navires.

Les résultats des tests que nous présentons maintenant concernent un trajet de Boston vers Lisbonne à une date donnée. Les autres résultats que nous avons obtenus à d'autres dates et entre d'autres ports ne sont pas qualitativement différents.

Les tests sont tous fait avec la même graine aléatoire pour obtenir une reproductibilité des résultats et une comparaison équitable entre les différents paramétrages de l'algorithme.

Les distances sont en degrés. Pour déterminer la zone de recherche autour d'un trajet par WRM, nous fixons une distance initiale d autour du trajet initial, puis nous multiplions cette distance par 0.75 quand la solution a été améliorée et par 1.25 quand il n'y a pas eu d'amélioration. Nous ne faisons que 10 itérations.

La figure 7 présente le résultat de la première partie de l'algorithme qui consiste à exécuter NAMOA*-TD à partir d'une grille rectangulaire.

Pour la deuxième partie, WRM utilise à nouveau NAMOA*-TD.

Les figures 8, 9 et 10 indiquent le résultat de l'application de WRM avec différentes valeurs initiales de d.

Nous constatons que plus d est grand, meilleurs sont les résultats.

Sur les différents tests, nous constatons que les solutions ont tendance à converger vers des minima locaux qui sont différents de ceux obtenus par NAMOA* lors de la première partie. Par ailleurs, plus on laisse d'espace pour diverger autour d'une solution (ie, plus d est grand), plus elles sont au final de bonne qualité (cf figure 11).

Par contre, les temps de calcul de NAMOA*-TD sont trop long (plusieurs centaines voire milliers de secondes) pour que l'approche soit acceptable en pratique. C'est pourquoi nous avons alors voulu vérifier dans un deuxième temps si la substitution de NAMOA*-TD par un Dijkstra scalaire sur chacune des parties de l'algorithme permettait d'obtenir de bons résultats (cf figure 12).

Il s'avère que les résultats obtenus sont similaires en termes de qualité des solutions mais beaucoup plus rapides à obtenir.

5 Conclusion et perspectives

Nous avons proposé une méthode de génération de trajets maritimes diversifiés et précis dans un contexte dynamique de météo et de courants changeant au cours du temps.

Notre approche est en deux phases : une phase de génération de trajets diversifiés mais peu précis et donc pas optimaux, suivie d'une phase de raffinement des lieux de passage des trajets permettant d'améliorer les objectifs (temps et consommation de carburant).

Sur nos premières expérimentations, nous avons constaté que pour chacune des deux phases, un algorithme rapide mais incomplet (Dijkstra scalaire) était plus pertinent qu'un algorithme complet. En effet, les temps de calculs étaient grandement réduits tandis que la qualité des solutions étaient préservées.

Alors que nous pensions au départ utiliser un algorithme incomplet et rapide (Dijkstra scalaire) pour améliorer les solutions d'un algorithme complet mais trop lent (NAMOA*-TD), nous nous sommes rendu compte qu'il était plus efficace de générer rapidement des solutions très approximatives car la qualité de celles-ci n'influaient pas sur la qualité des solutions finales.

Les résultats que nous avons obtenus demandent à être encore consolidés par un meilleur modèle de consommation de carburant et par la prise en compte d'autres critères (notamment environnementaux). L'applicabilité de notre approche dépend aussi de certaines contraintes réelles qui nous restent à connaître précisément : le temps maximum alloué pour l'ensemble des calculs et les ressources infor-

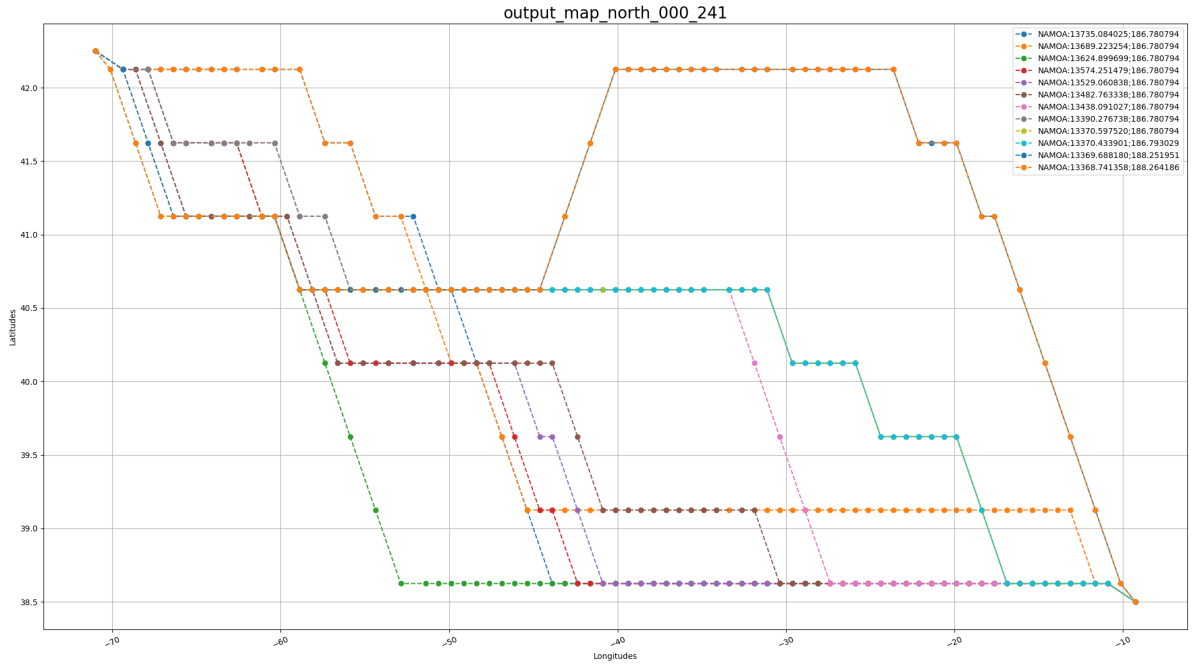

FIGURE 7 – Ensemble initial de solutions trouvées par NAMOA* dans une grille de 800 sommets. Les couples de valeurs apparaissant en haut à droite indiquent, pour chaque trajet, sa consommation et sa durée (en heures).

matiques utilisés pour ces calculs. De ceux-ci dépendent le degré de précision final des trajets que notre algorithme produit et donc de son acceptabilité.

Par ailleurs, comme pour tous les algorithmes de recherche de trajets maritimes, nous avons supposé que la vitesse était constante. Or, on sait que la consommation de carburant augmente quand la puissance du moteur change, ce qui advient quand on maintient une vitesse constante alors que les courants ou la météo évoluent. Si on veut minimiser la consommation de carburant, c'est la puissance du moteur qu'il faut rendre constante, ce qui peut impliquer un changement de vitesse. Prendre en compte cet aspect est une voie d'amélioration possible de notre approche.

6 Remerciements

Ce travail est soutenu par Bpifrance dans le cadre du projet PIA Transformation Numérique du Transport Maritime (TNTM).

Références

[1] Estelle Chauveau. *Optimisation des routes maritimes : un système de résolution multicritère et dépendant du temps*. Theses, Aix-Marseille Université (AMU), April 2018.

[2] E.W. Dijkstra. *A short introduction to the art of programming*. 1971.

[3] Stéphane Grandcolas. A metaheuristic algorithm for ship weather routing. *SN Operations Research Forum*, 2022.

[4] Peter E. Hart, Nils J. Nilsson, and Bertram Raphael. A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions on Systems Science and Cybernetics*, 4(2) :100–107, 1968.

[5] Lawrence Mandow and José-Luis Pérez de-la Cruz. Multiobjective a* search with consistent heuristics. *J. ACM*, 57 :27 :1–27 :25, 2010.

[6] Bradley S. Stewart and Chelsea C. White. Multiobjective a*. *J. ACM*, 38 :775–814, 1991.

[7] Aphrodite Veneti, Charalampos Konstantopoulos, and G. Pantziou. Continuous and discrete time label setting algorithms for the time dependent bi-criteria shortest path problem. 2015.

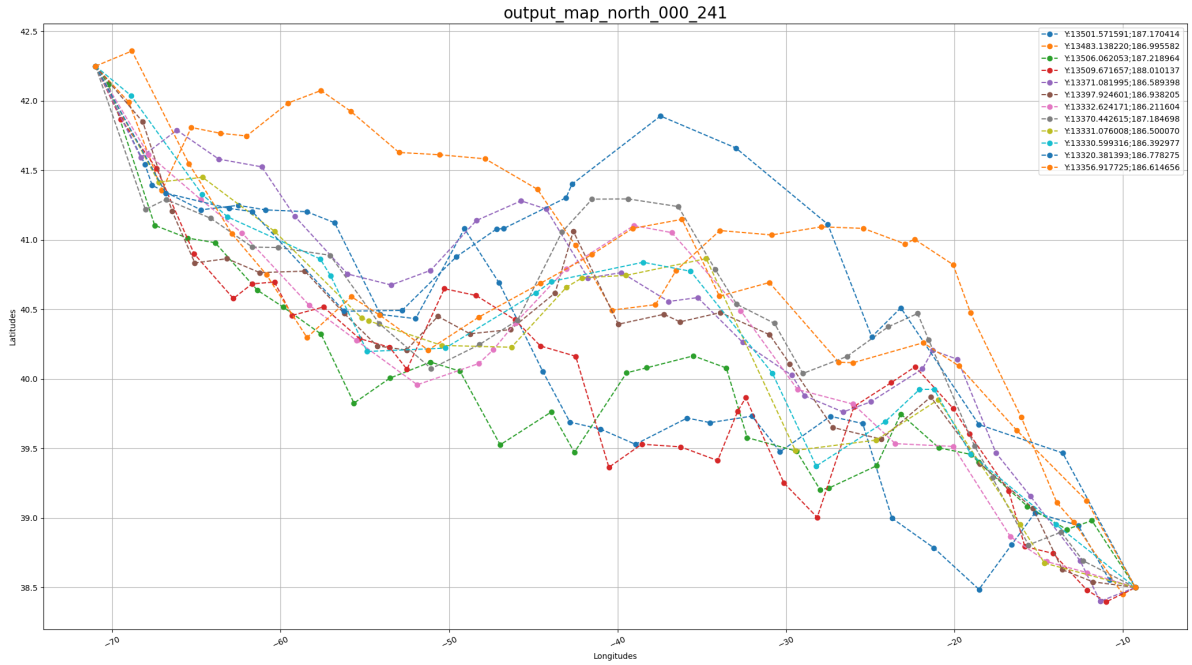

FIGURE 8 – Amelioration sur 10 iterations de la solution initiale avec d=0.5 et 200 sommets en utilisant NAMOA*-TD.

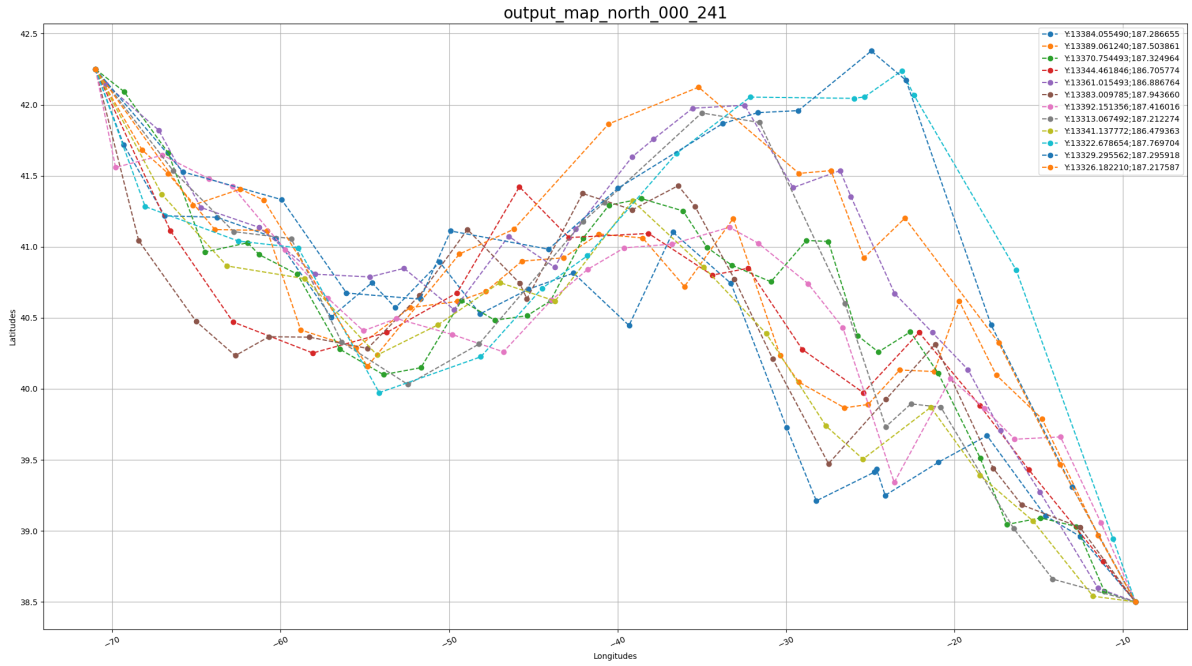

FIGURE 9 – Amelioration sur 10 iterations de la solution initiale avec d=1.0 et 200 sommets en utilisant NAMOA*-TD.

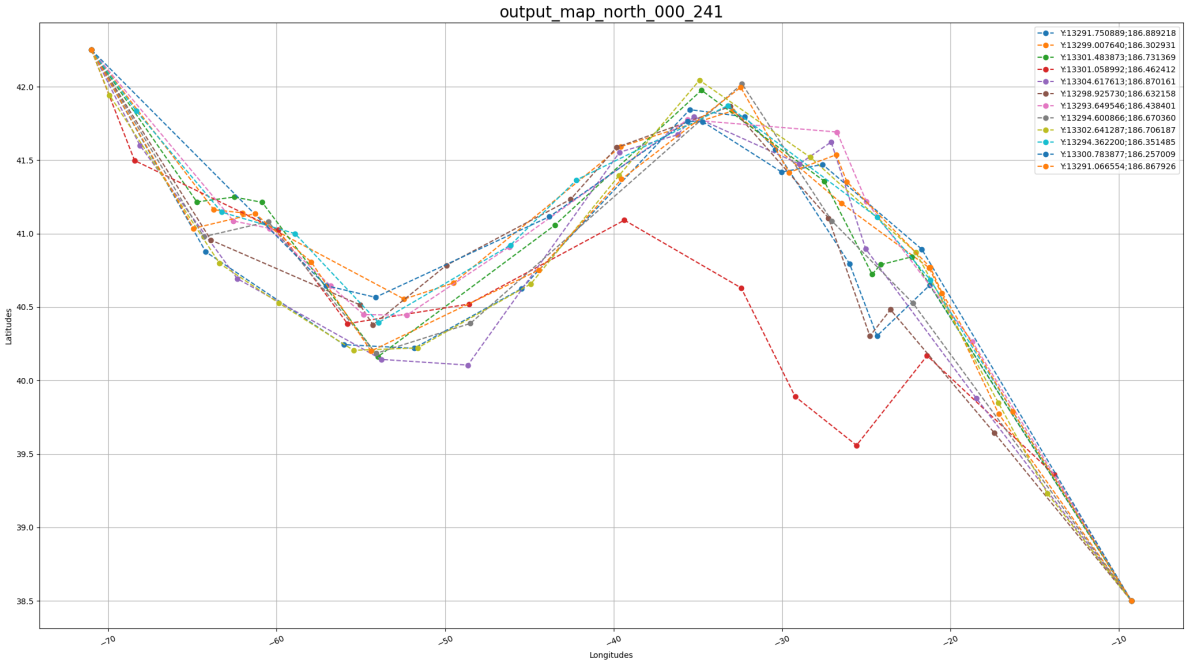

FIGURE 10 – Amelioration sur 10 iterations de la solution initiale avec d=5.0 et 200 sommets en utilisant NAMOA*-TD.

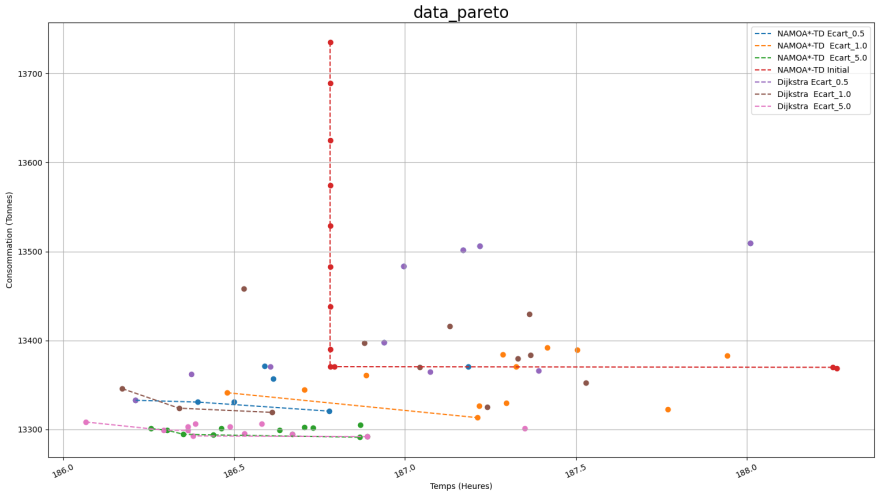

FIGURE 11 – Le front de Pareto pour l'étape 1 et ceux de l'étape 2 avec les trois valeurs de d et les deux types d'algorithmes (NAMOA*-TD et Dijkstra scalaire).

Première partie	NAMOA* (12 chemins)		Scalarisation (2 chemins)
Temps (s)	1610.12		7.743
Seconde partie	NAMOA*	Dijkstra	Dijkstra
D = 0.5	493.284	17.892	7.112
D = 1.0	1169.74	34.475	8.647
D = 5.0	$>$ 20 min.	565.256	127.749

FIGURE 12 – Comparaison des temps d'exécution pour chacune des parties selon la méthode utilisée (NAMOA*-TD ou Dijkstra scalaire)