BỘ GIÁO DỤC VÀ ĐÀO TẠO BỘ GIAO THÔNG VẬN TẢI
TRƯỜNG ĐẠI HỌC HÀNG HẢI VIỆT NAM

NCS. ĐẶNG QUANG VIỆT

NGHIÊN CỨU XÂY DỰNG THUẬT TOÁN NGẪU NHIÊN TÍNH TOÁN TUYẾN ĐƯỜNG VÀ KẾ HOẠCH CHẠY TÀU TỐI ƯU TRÊN CƠ SỞ ẢNH HƯỞNG CỦA CÁC YẾU TỐ THỜI TIẾT

LUẬN ÁN TIẾN SĨ KỸ THUẬT

Hải Phòng – 2023

BỘ GIÁO DỤC VÀ ĐÀO TẠO **BỘ GIAO THÔNG VẬN TẢI**

TRƯỜNG ĐẠI ĐẠI HỌC HÀNG HẢI VIỆT NAM

NCS. ĐẶNG QUANG VIỆT

NGHIÊN CỨU XÂY DỰNG THUẬT TOÁN NGẪU NHIÊN TÍNH TOÁN TUYẾN ĐƯỜNG VÀ KẾ HOẠCH CHẠY TÀU TỐI ƯU TRÊN CƠ SỞ ẢNH HƯỞNG CỦA CÁC YẾU TỐ THỜI TIẾT

LUẬN ÁN TIẾN SĨ KỸ THUẬT

NGÀNH: KHOA HỌC HÀNG HẢI

MÃ SỐ: 9840106

Người hướng dẫn khoa học: 1. PGS. TS. Nguyễn Viết Thành

2. PGS. TS. Nguyễn Minh Đức

Hải Phòng – 2023

LỜI CAM ĐOAN

Tôi xin cam đoan đây là công trình nghiên cứu của riêng tôi dưới sự hướng dẫn khoa học của PGS.TS. Nguyễn Viết Thành và PGS. TS. Nguyễn Minh Đức, không có phần nội dung nào được sao chép một cách bất hợp pháp từ công trình nghiên cứu của tác giả khác.

Kết quả nghiên cứu, nguồn số liệu trích dẫn, tài liệu tham khảo là hoàn toàn chính xác và trung thực.

Hải Phòng, ngày tháng năm 2023

Tác giả

LỜI CÁM ƠN

Tôi xin chân thành cảm ơn Trường Đại học Hàng hải Việt Nam, Viện Đào tạo sau đại học đã cho phép và tạo điều kiện cho tôi thực hiện luận án này.

Tôi xin chân thành cảm ơn hai Thầy hướng dẫn khoa học: PGS.TS. Nguyễn Viết Thành và PGS. TS. Nguyễn Minh Đức đã tận tình hướng dẫn, định hướng nghiên cứu giúp tôi hoàn thành luận án.

Tôi xin chân thành cám ơn Khoa Hàng hải, Viện đào tạo sau đại học, Trung tâm Huấn luyện thuyền viên, Trường Đại học Hàng hải Việt Nam, các thầy giáo, cô giáo, các nhà khoa học đã góp ý, phản biện và đánh giá giúp tôi từng bước hoàn thiện luận án.

Cuối cùng, tôi xin bày tỏ lòng biết ơn sâu sắc tới gia đình và bạn bè đã luôn động viên, khuyến khích, tạo điều kiện cho tôi trong suốt thời gian tôi nghiên cứu.

Hải Phòng, ngày tháng năm 2023

Tác giả

MỤC LỤC

DANH MỤC CÁC KÝ HIỆU VÀ CHỮ VIẾT TẮT

Chữ viết tắt	Giải thích
ACOA	Ant Colony Optimization Algorithm Thuật toán tối ưu đàn kiến
BFOA	Bacterial Foraging Optimization Algorithm Thuật toán tối ưu dựa trên việc tìm kiếm thức ăn của bầy vi khuẩn
BGTVT	Bộ giao thông vận tải
CSDL	Cơ sở dữ liệu
DRT	Data Representation Template
DT	Data Template
EC	Exhausted Search Thuật toán vét cạn
ECDIS	Electronic Chart Display and Information System Hệ thống thông tin và hiển thị hải đồ điện tử
EEDI	Energy Efficiency Design Index Chỉ số thiết kế hiệu quả năng lượng
EEOI	Energy Efficiency Operational Indicator Chỉ số khai thác hiệu quả năng lượng
EGC	Enhanced Group Call Dịch vụ gọi nhóm tăng cường
FC	Fuel Consumption Mức tiêu thụ nhiên liệu
FCCC	United Nations Framework Convention On Climate Change Công ước khung của LHQ về biến đổi khí hậu
GA	Genetic Algorithm

	Thuật toán di truyền
GDT	Grid Definition Template
GHC	Green House Effect Hiệu ứng nhà kính
GRIB	Gridded Binary hay General Regularly distributed Information in Binary Form
HCA	Hill Climbing Algorithm Thuật toán leo đồi
IMCO	Intergovermental Maritime Consultative Organization Tổ chức liên chính phủ giải quyết các vấn đề Hàng hải
IMO	International Maritime Organization Tổ chức Hàng hải quốc tế
INMARSAT	Information Maritime Satellite System Hệ thống thông tin vệ tinh hàng hải
IPCC	Intergorvermental Panel On Climate Change Ủy ban liên chính phủ về biến đổi khí hậu
JIT	Just in Time "Đến kịp lúc"
MARPOL	International Convention for the Prevention of Pollution from Ships Công ước quốc tế về phòng ngừa ô nhiễm từ tàu
MEPC	Marine Environment Protection Committee Ủy ban bảo vệ môi trường biển
NCS	Nghiên cứu sinh
NetCDF	Format Network Common Data Form
OSCAR	Ocean Surface Current Analyses Real - Time Dự án nghiên cứu phân tích dòng chảy đại dương theo thời gian thực

PDT	Product Definition Template
QĐ_TTg	Quyết định Thủ tướng
RISH	Research Institute for Sustainable Humanoshere Viện nghiên cứu phát triển bền vững khí quyển nhân loại
RPM	Revolutions Per Minute Số vòng quay trên phút (đơn vị Vòng/ phút)
SEEMP	Ship Energy Efficiency Management Plan Kế hoạch quản lí hiệu quả năng lượng trên tàu
SOLAS	The International Convention for the Safety of Life at Sea Công ước quốc tế về an toàn sinh mạng trên biển
SPOS	Ship Performance Optimization System Hệ thống tối ưu hóa hoạt động tàu
UKC	Under Keel Clearance Chân hoa tiêu (độ sâu dưới đáy tàu)
UN	United Nations Liên Hợp Quốc
VB 2010	Visual Basic 2010 Ngôn ngữ lập trình Visual Basic 2010
VOC	Volatite Organic Compounds Các chất hữu cơ ở dạng bay hơi
VTS	Vessel Traffic System Hệ thống quản lý giao thông tàu thuyền
WMO	World Meteorology Organization Tổ chức khí tượng thế giới
WMO	World Meteorology Organization Tổ chức khí tượng hải dương thế giới
WPT	Waypoint

	Điểm nút (hay điểm chuyển hướng)
OSCAR	Ocean Surface Current Analyses Real – time Dự án nghiên cứu, phân tích dòng chảy đại dương theo thời gian thực
CDF	Common Data Form
ECMWF	European Centre for Medium Range Weather Forecast Trung tâm dự báo hạn vừa của Châu Âu
IEEC	International Energy Efficiency Giấy chứng nhận hiệu quả năng lượng quốc tế
SMS	Safety Management System Hệ thống quản lý an toàn
SECA	Sulphur Emission Control Area Khu vực kiểm soát phát thải So_x
BFO	Bacterial Foraging Optimization Thuật toán tối ưu dựa trên phương pháp tìm kiếm thức ăn của bầy vi khuẩn
VB 10	Visual Basic 2010 Ngôn ngữ lập trình Visual Basic 2010

DANH MỤC CÁC BẢNG

DANH MỤC CÁC HÌNH

TÓM TẮT

Bên cạnh việc chú trọng, định hướng phát triển kinh tế biển, đưa Việt Nam trở thành Quốc gia mạnh về biển, giàu từ biển thì vấn đề phòng ngừa ô nhiễm môi trường từ tàu, phòng ngừa ô nhiễm không khí, sử dụng tiết kiệm, hiệu quả năng lượng trên tàu biển và giảm lượng khí thải từ tàu luôn luôn được Đảng, Nhà nước và Chính phủ Việt Nam đặc biệt quan tâm.

Đề tài luận án *"Nghiên cứu xây dựng thuật toán ngẫu nhiên tính toán tuyến đường và kế hoạch chạy tàu tối ưu trên cơ sở ảnh hưởng của các yếu tố thời tiết"* mang tính thời sự, cấp thiết và thực tiễn cao.

Thông qua việc thực hiện đề tài luận án, NCS nghiên cứu và đề xuất giải pháp nhằm tăng hiệu quả sử dụng năng lượng, giảm phát thải nhà kính từ tàu biển góp phần bảo vệ môi trường biển một cách hiệu quả, tiết kiệm chi phí, dễ dàng áp dụng đó là: Tính toán tuyến đường và kế hoạch chạy tàu tối ưu. NCS thiết lập tuyến đường chạy tàu tối ưu và đưa ra phương án vận hành tàu hợp lý trên từng đoạn của tuyến tối ưu đã được thiết lập.

Trong đề tài luận án, NCS tiến hành:

- Tổng hợp thông tin thời tiết bao gồm sóng, gió và dòng chảy phục vụ tính toán tuyến đường và kế hoạch chạy tàu tối ưu (Khai thác bản tin sóng toàn cầu, bản tin gió toàn cầu của Rish và dòng chảy của Oscar);

- Ứng dụng phương pháp bình phương nhỏ nhất xác định đặc tính thay đổi tốc độ và tiêu thụ nhiên liệu của tàu biển trong từng điều kiện hành hải cụ thể phục vụ tính toán tuyến đường và kế hoạch chạy tàu tối ưu;

- Nghiên cứu xây dựng và ứng dụng thuật toán BFO cải tiến để tính toán tuyến đường và kế hoạch chạy tàu tối ưu nhiên liệu dựa trên nguyên tắc "*Tàu đến cảng kịp lúc*" (Just in time).

Từ khóa: Tuyến đường tối ưu, kế hoạch chạy tàu tối ưu; Bản tin sóng, gió của RISH; Bản tin dòng chảy OSCAR; Thuật toán BFO; Just in Time.

ABSTRACT

In addition to emphasizing the development and economic orientation towards the marine economy, making Vietnam a strong country in the sea, rich from the sea, the prevention of environmental pollution from ships, air pollution prevention, using economically and efficiently energy on ships, along with reducing emissions from vessels, are always a matter of special concern for the Communist Party, the State, and the Government of Vietnam.

The doctoral dissertation topic "Research on building a random algorithm for the optimal route calculation and optimal ship operation plan based on the influence of weather factors" is topical, urgent and highly practical.

Through the implementation of the dissertation, the author researches and proposes solutions to increase the efficiency of energy use, reduce greenhouse gas emissions from ships, effectively contribute to marine environmental protection, save costs, and facilitate practical application. The proposed solutions include calculating optimal routes and rationally operating vessel on each specific route segments.

In this doctoral dissertation, the author conducts the following tasks:

- Synthesizing weather information including waves, winds and currents to serve the calculation of optimal routes and optimal ship operation plans (utilizing global wave and wind forecasts from Rish and current data from Oscar);

- Applying the least squares method to determine the characteristics of changes in speed and fuel consumption of ships under specific weather condition, serving the calculation of optimal route and ship operation plan;

- Researching, developing, and applying an improved BFO algorithm to calculate optimal routes and ship operation plans based on the Just-in-Time principle - ensuring ships arrive at the port on time.

Keywords: Optimal route, optimal ship operation plan, RISH wave and wind bulletin, OSCAR current bulletin, Bacterial foraging optimization algorithm, Just in Time.

MỞ ĐẦU

1. Tính cấp thiết của đề tài luận án

Đảng Cộng sản Việt Nam đã ban hành 02 Nghị quyết Trung ương về chiến lược biển là:

- Nghị quyết số 09 - NQ/TƯ, ngày 09/02/2007 về chiến lược biển Việt Nam đến năm 2020 [1];

- Nghị quyết số 36 - NQ/TƯ, ngày 22/10/2018 về chiến lược phát triển bền vững kinh tế biển Việt Nam đến năm 2030, tầm nhìn đến 2045 nhằm đưa Việt Nam trở thành Quốc gia mạnh về biển, giàu từ biển [2].

Đặc biệt, ngày 19/03/2015, Việt Nam chính thức tham gia đầy đủ các Phụ lục của Công ước quốc tế phòng ngừa ô nhiễm từ tàu biển - MARPOL 73/78 [69] (International Convention for the Prevention of Pollution from Ships, 1973, as modified by the Protocol of 1978 thereto). Công ước này là một trong những Công ước chủ chốt về bảo vệ môi trường biển. Công ước đưa ra những quy định nhằm phòng ngừa ô nhiễm từ hoạt động của tàu biển bao gồm ngăn ngừa ô nhiễm dầu, hàng nguy hiểm, độc hại, cũng như do nước thải, rác thải và khí thải từ tàu biển.

Trong bối cảnh Việt Nam là thành viên chính thức đầy đủ 6 Phụ lục của Công ước MARPOL 73/78, Bộ Giao thông vận tải đã đưa vào thử nghiệm và triển khai các chương trình giáo dục nâng cao nhận thức về phòng ngừa ô nhiễm biển từ tàu, phòng ngừa ô nhiễm không khí, sử dụng tiết kiệm, hiệu quả năng lượng trên tàu biển và giảm lượng khí thải từ tàu. Bộ Giao thông vận tải đưa ra các khuyến cáo và ban hành các quy định bắt buộc thực hiện với mục đích nâng cao hiệu quả sử dụng năng lượng trên tàu biển, cụ thể:

- Thông tư số 40/2018/TT – BGTVT, ngày 29/6/2018 "Quy định về thu thập và báo cáo tiêu thụ nhiên liệu của tàu biển Việt Nam" [4];

- QCVN 26/2018/BGTVT "Quy chuẩn kỹ thuật quốc gia về các hệ thống ngăn ngừa ô nhiễm biển của tàu" và Thông tư số 09/2019/TT – BGTVT, ngày 01/3/2019 "Ban hành quy chuẩn kỹ thuật quốc gia về các hệ thống ngăn ngừa ô nhiễm biển của tàu" [5].

Như vậy, cùng với việc chú trọng, định hướng phát triển kinh tế biển, các vấn đề môi trường phát sinh trong thực tiễn hoạt động của ngành Hàng hải (các tai nạn tràn dầu, các vấn đề ô nhiễm mới nảy sinh, …) luôn luôn được Đảng, Nhà nước và Chính phủ Việt Nam đặc biệt quan tâm.

Để giải quyết các vấn đề về phòng ngừa ô nhiễm biển từ tàu, phòng ngừa ô nhiễm không khí, sử dụng tiết kiệm, hiệu quả năng lượng trên tàu biển và giảm lượng khí thải từ tàu cho đến nay đã có rất nhiều giải pháp được nghiên cứu và áp dụng như:

- Nhóm giải pháp về thiết kế tàu: Tăng tải trọng, tối ưu hóa thiết kế thân tàu, tối ưu hóa hình dạng khí động học, tối ưu hóa dòng theo, sử dụng năng lượng gió, sử dụng năng lượng mặt trời, sử dụng năng lượng hạt nhân, …;

- Nhóm giải pháp về cải tiến công nghệ:

+ Thay đổi thiết kế và công nghệ máy nhằm mục đích tăng hiệu quả, giảm khí thải độc hải;

+ Thay đổi chất liệu sơn phủ vỏ tàu để giảm sức cản;

+ Tối ưu hóa thiết kế hệ thống chân vịt, bánh lái;

+ Thu hồi nhiệt thải hiệu quả.

- Nhóm giải pháp về khai thác tàu: Giảm tốc độ tàu nhằm giảm tiêu thụ nhiên liệu và giảm phát thải Co_2, xây dựng tuyến đường hàng hải tối ưu, mớn nước tối ưu, độ chúi tối ưu, vệ sinh thân vỏ tàu để giảm sức cản, …

- Nhóm giải pháp về nhiên liệu: Sử dụng nhiều loại nhiên liệu mới, thân thiện với môi trường, …

Một trong những giải pháp được xem là hiệu quả, tiết kiệm chi phí, dễ dàng áp dụng là: Tính toán tuyến đường và kế hoạch chạy tàu tối ưu (thiết lập tuyến đường chạy tàu tối ưu và việc vận hành tàu một cách hợp lí trên từng đoạn tuyến cụ thể đã được xây dựng).

Thực tế cho thấy tối ưu hóa kế hoạch chạy tàu thông qua việc vận hành các hệ thống hỗ trợ hàng hải tính toán tuyến đường khí tượng ngày càng trở nên phổ biến trên thế giới. Một số hệ thống được sử dụng rộng rãi như: Hệ thống Chart

Co, Hệ thống tối ưu hóa hoạt động tàu (SPOS – Ship Performance Optimization System), AMI Seaware Routing, Sea Planner, phần mềm thời tiết Interactive weather của Clearpoint, …

Tuy nhiên, các hệ thống hỗ trợ hàng hải này chưa phổ biến, chưa được sử dụng rộng rãi cho đội tàu biển Việt Nam bởi các nguyên nhân chủ yếu như sau:

- Chi phí cao;

- Là phần mềm thương mại thuyền viên không biết được thuật toán cũng như các dữ liệu thời tiết mà nhà cung cấp sử dụng (thuyền viên khó nắm bắt và làm chủ);

- Việc tiếp cận gặp nhiều khó khăn do rào cản ngôn ngữ;

- Thiếu hướng dẫn khai thác do thiếu thông tin về đặc điểm khai thác tàu.

Ngoài ra:

- Báo cáo tình hình sử dụng nhiên liệu đã trở thành bắt buộc (Bộ Giao thông vận tải đã đưa ra nhiều khuyến cáo và quy định bắt buộc phải thực thi);

- Yêu cầu về kiểm soát CO_2 ngày một gắt gao;

- Đội ngũ thuyền viên Việt Nam ở các công ty Vận tải biển đã nhận thức được yêu cầu nhưng chưa có công cụ hỗ trợ.

Thêm vào đó, ngay các kết quả đưa ra từ các dịch vụ tư vấn tuyến đường chạy tàu còn chưa thực sự đáng tin cậy và chuẩn xác do:

- Chưa tính toán được đầy đủ, chi tiết đặc điểm điều động của tàu và ảnh hưởng của các yếu tố thời tiết đến tàu;

- Hầu hết chưa tính toán tới vấn đề hiệu quả nhiên liệu khi tàu thay đổi chế độ máy;

- Chưa tính toán tới nguy cơ có thể gặp phải, ví dụ như: nguy cơ đến muộn, gặp vùng thời tiết xấu, v.v … khi các yếu tố đầu vào thay đổi.

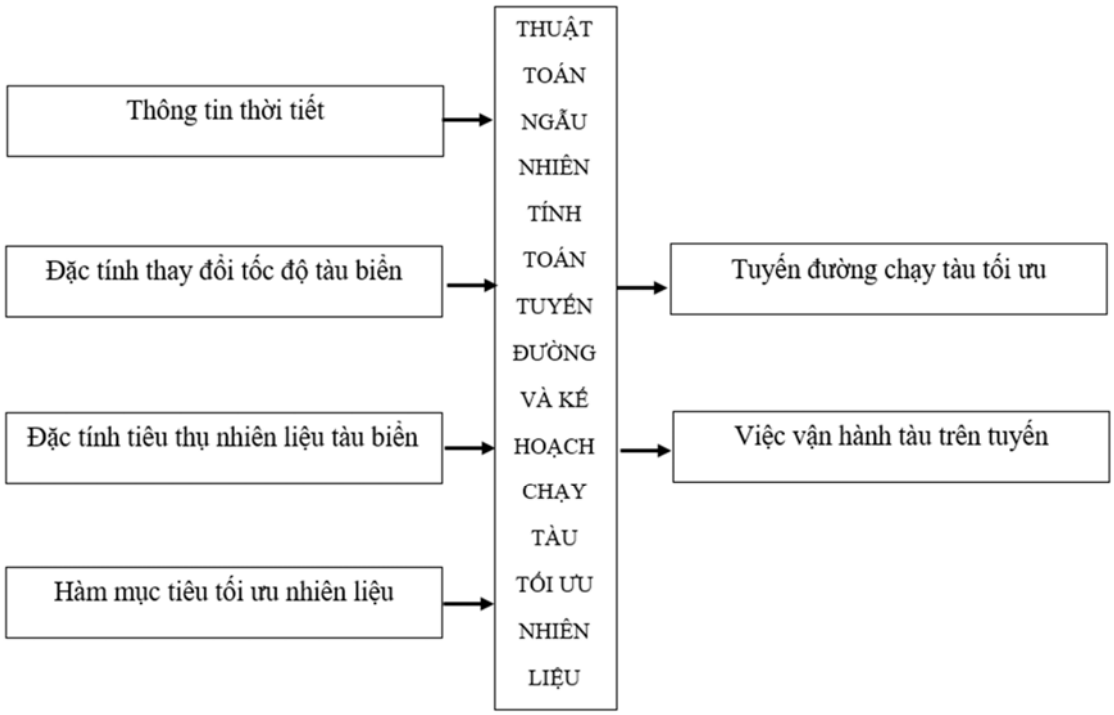
Chính vì vậy, Đề tài luận án tiến sĩ "***Nghiên cứu xây dựng thuật toán ngẫu nhiên tính toán tuyến đường và kế hoạch chạy tàu tối ưu trên cơ sở ảnh hưởng của các yếu tố thời tiết"*** nhằm tăng hiệu quả sử dụng năng lượng và giảm phát

thải khí nhà kính từ tàu biển mang tính thời sự, cấp thiết và thực tiễn rất cao, đáp ứng được các yêu cầu về quản lý Nhà nước chuyên ngành hàng hải.

2. Mục đích nghiên cứu của đề tài luận án

Mục đích nghiên cứu của đề tài luận án là nghiên cứu, xây dựng và ứng dụng thuật toán ngẫu nhiên để tính toán tuyến đường và kế hoạch chạy tàu tối ưu nhiên liệu dựa trên nguyên tắc just in time "*tàu đến cảng kịp lúc*" nhằm nâng cao hiệu quả sử dụng năng lượng, giảm phát thải khí nhà kính từ tàu biển góp phần bảo vệ môi trường biển một cách hiệu quả.

3. Nội dung nghiên cứu của đề tài luận án



Hình 1.1 Sơ đồ khối mô tả nội dung nghiên cứu của đề tài luận án

Nội dung nghiên cứu của đề tài luận án được mô tả trong hình 1.1, trong quá trình thực hiện đề tài luận án, NCS tiến hành:

- Tổng hợp thông tin thời tiết phục vụ tính toán tuyến đường và kế hoạch chạy tàu tối ưu, cụ thể:

+ Khai thác bản tin sóng toàn cầu, bản tin gió toàn cầu từ cơ sở dữ liệu của Viện nghiên cứu phát triển bền vững khí quyển nhân loại thuộc Đại học Kyoto, Nhật Bản, gọi tắt là Rish (Research Institute for Sustainable Humanoshere);

+ Khai thác dữ liệu dòng chảy của Dự án nghiên cứu, phân tích dòng chảy đại dương theo thời gian thực, gọi tắt là Oscar (Ocean Surface Current Analysis Real - Time) thuộc phòng thí nghiệm sức đẩy phản lực (Jet Propulsion Laboratory Physical Oceanography), Viện Công nghệ California (Viện quản lý các dự án của cơ quan hàng không vũ trụ Mỹ).

- Ứng dụng phương pháp bình phương nhỏ nhất xác định đặc tính thay đổi tốc độ tàu biển, đặc tính tiêu thụ nhiên liệu tàu biển trong từng điều kiện hành hải cụ thể (điều kiện sóng, gió, dòng chảy, chế độ máy (hay số vòng quay chân vịt) (rpm), mớn nước tàu (draft), hiệu số mớn nước tàu (trim)) phục vụ tính toán tuyến đường và kế hoạch chạy tàu tối ưu;

- Xây dựng hàm mục tiêu tối ưu nhiên liệu dựa trên nguyên tắc just in time "tàu đến cảng kịp lúc";

- Nghiên cứu, xây dựng thuật toán vi khuẩn cải tiến tính toán tuyến đường và kế hoạch chạy tàu tối ưu nhiên liệu dựa trên nguyên tắc just in time "tàu đến cảng kịp lúc".

4. Đối tượng và phạm vi nghiên cứu của đề tài luận án

4.1. Đối tượng nghiên cứu của đề tài luận án

Đối tượng nghiên cứu của đề tài luận án là:

- Nghiên cứu thông tin thời tiết (sóng, gió, dòng chảy);

- Nghiên cứu đặc tính thay đổi tốc độ tàu biển trong từng điều kiện hành hải cụ thể;

- Nghiên cứu đặc tính tiêu thụ nhiên liệu của tàu biển trong từng điều kiện hành hải cụ thể;

- Nghiên cứu hàm mục tiêu tối ưu nhiên liệu dựa trên nguyên tắc just in time "tàu đến cảng kịp lúc";

- Nghiên cứu thuật toán ngẫu nhiên.

4.2. Phạm vi nghiên cứu của đề tài luận án

Phạm vi nghiên cứu của đề tài:

- Về thông tin thời tiết, NCS tập trung nghiên cứu:

+ Bản tin sóng toàn cầu, bản tin gió toàn cầu của Rish [12, 16, 74] và;

+ Bản tin dòng chảy của Oscar [12, 16, 47, 48, 75];

- Về đặc tính tàu biển, NCS tập trung nghiên cứu:

+ Đặc tính thay đổi tốc độ tàu biển trong từng điều kiện hành hải cụ thể [17];

+ Đặc tính tiêu thụ nhiên liệu tàu biển trong từng điều kiện hành hải cụ thể [17];

+ Ứng dụng phương pháp bình phương nhỏ nhất xác định đặc tính thay đổi tốc độ tàu biển, đặc tính tiêu thụ nhiên liệu tàu biển trong từng điều kiện hành hải cụ thể [17].

- Về hàm mục tiêu:

Xây dựng hàm mục tiêu tối ưu nhiên liệu dựa trên nguyên tắc just in time “tàu đến cảng kịp lúc”.

- Về thuật toán ngẫu nhiên, NCS nghiên cứu, xây dựng thuật toán vi khuẩn cải tiến để tính toán tuyến đường chạy tàu tối ưu nhiên liệu dựa trên nguyên tắc just in time "tàu đến cảng kịp lúc" (Gọi tắt là tuyến đường tối ưu just in time - JIT).

5. Phương pháp nghiên cứu của đề tài luận án

- Phương pháp Chuyên gia: Xác định các yếu tố cần thiết, xây dựng hàm mục tiêu cho bài toán tính toán tuyến đường chạy tàu tối ưu nhiên liệu;

- Phương pháp tổng hợp: Tổng hợp các thông tin an toàn Hàng hải và thông tin thời tiết từ nhiều nguồn khác nhau;

- Phương pháp phân tích: Phân tích đánh giá độ chính xác của các bản tin tổng hợp được;

- Phương pháp mô phỏng: Để việc đánh giá đạt hiệu quả cao, NCS xây dựng các mô hình mô phỏng chạy thử trên máy tính.

6. Ý nghĩa khoa học và thực tiễn của đề tài luận án

Trong khai thác tàu, việc tiết kiệm nhiên liệu và an toàn là hai yếu tố luôn song hành với nhau. Để đạt được hiệu quả khai thác tối ưu, sự cân đối giữa tiết kiệm và an toàn là bài toán đặt ra cho Thuyền trưởng và các Sỹ quan Hàng hải về khía cạnh dẫn tàu giữa các cảng trên toàn thế giới.

Ngày nay, với sự hỗ trợ của khoa học công nghệ, thông tin truyền thông, thiết bị liên lạc, cũng như cơ sở dữ liệu phục vụ dẫn tàu luôn sẵn có và giúp ích rất nhiều cho Thuyền trưởng và các Sỹ quan trong việc hành hải, đồng thời hỗ trợ rất hữu hiệu cho công tác quản lý hoạt động tàu. Tuy nhiên, về phương pháp và công cụ hữu ích cụ thể là đưa ra tuyến đường và kế hoạch chạy tàu tối ưu còn chưa được nghiên cứu nhiều, đặc biệt là phục vụ cho đội ngũ Sỹ quan, thuyền viên Việt Nam.

Đề tài luận án "Nghiên cứu xây dựng thuật toán ngẫu nhiên tính toán tuyến đường và kế hoạch chạy tàu tối ưu trên cơ sở ảnh hưởng của các yếu tố thời tiết" đã đáp ứng được yêu cầu thực tiễn. Vấn đề nghiên cứu được đặt ra mang tính khoa học và có khả năng ứng dụng cao cho các tàu, đặc biệt là các tàu chạy tuyến quốc tế, quãng đường chạy tàu dài và đi qua nhiều khu vực có thời tiết biển và điều kiện hải dương thay đổi, diễn biến phức tạp.

6.1. Ý nghĩa khoa học

- Đề tài luận án là nguồn tài liệu tham khảo hữu ích cho tất cả các bạn độc giả quan tâm;

- Sản phẩm nghiên cứu của đề tài luận án giải quyết được một số hạn chế của các phương pháp tính toán tuyến đường tối ưu nhiên liệu khác hiện đang được ứng dụng.

6.2. Ý nghĩa thực tiễn của đề tài

Sản phẩm nghiên cứu của đề tài luận án là một giải pháp hữu hiệu góp phần nâng cao hiệu quả sử dụng năng lượng và giảm phát thải khí nhà kính từ tàu biển, đặc biệt trong bối cảnh các yêu cầu ngày càng khắt khe của IMO về việc giảm phát thải khí nhà kính từ tàu biển đã và đang được triển khai bắt buộc.

Ngoài ra, sản phẩm nghiên cứu của đề tài luận án khi được thẩm định và phê duyệt có thể sử dụng không những cho các đội tàu biển Việt Nam, cho các công ty quản lý, khai thác tàu, mà còn cho các Cảng vụ hàng hải trong quản lý giao thông hàng hải (VTS).

7. Đóng góp mới của đề tài luận án

Thực tiễn Hàng hải hiện nay cho thấy, vấn đề tối ưu hóa kế hoạch chạy tàu bằng cách ứng dụng hệ thống Hỗ trợ hàng hải (tính toán tuyến đường hàng hải khí tượng) đang ngày một trở nên phổ biến trên thế giới. Hiện nay, có rất nhiều hệ thống Hỗ trợ hàng hải được thuyền viên Việt Nam ứng dụng như: Hệ thống SOPS, hệ thống Met-Manager, Voyage Planner, …Tuy nhiên, điểm chung của thuyền viên Việt Nam khi ứng dụng các hệ thống Hỗ trợ hàng hải này là chưa thực sự làm chủ được ứng dụng hỗ trợ vì nhiều nguyên nhân như rào cản ngôn ngữ, chưa nắm chắc các thông tin, chưa hiểu đúng bản chất và thuật toán tính toán, cũng như công nghệ áp dụng.

Chính vì vậy khi các sản phẩm nghiên cứu của đề tài luận án được ứng dụng sẽ giúp cho đội ngũ thuyền viên Việt Nam tự tin, hiểu và làm chủ được úng dụng hỗ trợ hàng hải dễ dàng hơn với "sản phẩm made in Việt Nam", cụ thể như sau:

- Làm chủ được bản tin sóng toàn cầu, bản tin gió toàn cầu được mã hóa theo định dạng Grib 2 từ cơ sở dữ liệu của Viện nghiên cứu phát triển bền vững khí quyển nhân loại thuộc Đại học Kyoto, Nhật Bản (Rish – Research Institute for Sustainable Humanoshere);

- Làm chủ được dữ liệu dòng chảy toàn cầu được mã hóa theo định dạng netCDF (Format Network Common Data Form) từ cơ sở dã liệu của Dự án nghiên cứu, phân tích dòng chảy đại dương theo thời gian thực (Oscar – Ocean Surface Current Analysis Real Time) thuộc phòng thí nghiệm sức đẩy phản lực (Jet Propulsion Laboratory Physical Oceanography), Viện công nghệ California (Viện quản lý các dự án của cơ quan hàng không vũ trụ Mỹ);

Trên cơ sở đó tạo được các file dữ liệu thời tiết tổng hợp phục vụ tính toán tuyến đường và kế hoạch chạy tàu tối ưu nhiên liệu.

- Ứng dụng phương pháp bình phương nhỏ nhất xây dựng được bộ cơ sở dữ liệu đặc tính thay đổi tốc độ tàu biển, đặc tính tiêu thụ nhiên liệu tàu biển trong từng điều kiện hành hải cụ thể phục vụ tính toán tuyến đường và kế hoạch chạy tàu tối ưu;

- Xây dựng thuật toán vi khuẩn cải tiến tính toán tuyến đường và kế hoạch chạy tàu tối ưu nhiên liệu dựa trên nguyên tắc just in time "tàu đến cảng kịp lúc".

8. Kết cấu của đề tài luận án

Đề tài luận án được trình bày rõ ràng, mạch lạc và logic với kết cấu gồm:

Phần tóm tắt

Phần mở đầu:

1. Tính cấp thiết của đề tài luận án;

2. Mục đích nghiên cứu của đề tài luận án;

3. Nội dung nghiên cứu của đề tài luận án;

4. Đối tượng và phạm vi nghiên cứu của đề tài luận án;

5. Phương pháp nghiên cứu của đề tài luận án;

6. Ý nghĩa khoa học và thực tiễn của đề tài luận án;

7. Đóng góp mới của đề tài luận án;

8. Kết cấu của đề tài luận án.

Phần nội dung: Gồm 4 chương

Chương 1. Tổng quan về vấn đề nghiên cứu của đề tài luận án;

Chương 2. Tổng hợp thông tin thời tiết phục vụ tính toán tuyến đường và kế hoạch chạy tàu tối ưu;

Chương 3. Tổng hợp, phân tích đặc tính thay đổi tốc độ và đặc tính tiêu thụ nhiên liệu của tàu biển trong từng điều kiện hành hải cụ thể phục vụ tính toán tuyến đường và kế hoạch chạy tàu tối ưu;

Chương 4. Nghiên cứu, xây dựng thuật toán vi khuẩn cải tiến để tính toán tuyến đường và kế hoạch chạy tàu tối ưu nhiên liệu dựa trên nguyên tắc just in time "tàu đến cảng kịp lúc".

Phần kết luận:

1. Kết luận

2. Kiến nghị

Danh mục các công trình khoa học đã được công bố liên quan đến đề tài luận án;

Danh mục tài liệu tham khảo; và

Phụ lục

CHƯƠNG 1. TỔNG QUAN VỀ VẤN ĐỀ NGHIÊN CỨU CỦA ĐỀ TÀI LUẬN ÁN

1.1. Tình hình nghiên cứu liên quan đến đề tài luận án

Trong phần này, NCS tìm hiểu và tổng hợp một số tài liệu, công trình nghiên cứu khoa học trong và ngoài nước liên quan đến chủ đề của đề tài luận án đã được công bố gần đây.

1.1.1. Các giải pháp nâng cao hiệu quả sử dụng năng lượng và giảm phát thải khí nhà kính từ tàu biển [19]

NCS tìm hiểu và thống kê được hiện tại, có nhiều giải pháp đã và đang được áp dụng để nâng cao hiệu quả sử dụng năng lượng và giảm phát thải khí nhà kính từ tàu biển, mỗi giải pháp đều có ưu, nhược điểm riêng.

1.1.1.1. Nhóm giải pháp thiết kế tàu

- Mục đích tăng hiệu quả vận chuyển của tàu bằng việc tối ưu hóa thiết kế thân vỏ để giảm lực cản tàu, giảm trọng lượng tàu không, tăng tải trọng, …

- Thực tế thường sử dụng các giải pháp sau:

+ Tăng tải trọng: Tải trọng của tàu có thể được tăng lên nhờ thay đổi kết cấu hoặc sử dụng các vật liệu nhẹ hơn làm giảm trọng lượng tàu không. Đối với giải pháp này cần đặc biệt chú ý tới việc đảm bảo sức bền tàu;

+ Tối ưu hóa thiết kế thân tàu: Lực cản thân tàu chiếm tới 70% tiêu hao công suất máy do đó cải tiến thiết kế thân tàu đem lại hiệu quả rất lớn;

+ Tối ưu hóa hình dạng khí động học: Áp dụng các công nghệ thiết kế thích hợp cho phần cấu trúc thượng tầng, hoặc lắp đặt thêm các cấu trúc để thay đổi hướng dòng khí, lực cản, … từ đó tăng hiệu quả năng lượng, giảm phát thải khí nhà kính từ tàu;

+ Tối ưu hóa dòng theo: Tăng hiệu quả làm việc của chân vịt, giảm tiêu thụ nhiên liệu và các hiệu ứng không mong muốn như: ăn mòn, rung, tiếng ồn;

+ Nâng cao hiệu quả làm việc của chân vịt;

+ Năng lượng gió: Đầu tư nghiên cứu và lắp đặt các hệ thống động lực dùng năng lượng gió như buồm, diều hứng gió cho tàu;

+ Năng lượng mặt trời: Nghiên cứu đầu tư lắp đặt hệ thống năng lượng mặt trời (Pin mặt trời) cho tàu biển;

+ Năng lượng hạt nhân: Sử dụng năng lượng hạt nhân trên tàu biển sẽ loại trừ được việc phát thải khí nhà kính từ tàu biển, tuy nhiên cần đặc biệt chú ý tới vấn đề an toàn và xử lý chất thải hạt nhân.

- Ưu, nhược điểm của nhóm giải pháp thiết kế tàu:

+ Ưu điểm: Hiệu quả mang lại rất lớn;

+ Nhược điểm: Cần nhiều thời gian, chi phí rất cao và chỉ áp dụng được trong giai đoạn thiết kế tàu.

1.1.1.2. Nhóm giải pháp về công nghệ máy

- Các nhà sản xuất máy tàu biển đã đưa ra nhiều giải pháp thay đổi thiết kế và công nghệ máy nhằm mục đích tăng hiệu quả, giảm phát thải khí nhà kính từ tàu. Các giải pháp về công nghệ máy nhằm cải tiến động cơ hoạt động hiệu quả giảm tiêu thụ nhiên liệu và phát thải khí nhà kính từ tàu biển, gồm:

+ Điều chỉnh quá trình cháy;

+ Điều chỉnh quá trình phun nhiên liệu;

+ Điều chỉnh dòng khí nạp cho động cơ;

+ Phun nước áp lực cao;

+ Hồi lưu khí thải;

+ Hệ thống xử lý nhiệt khí thải;

+ Hệ thống phun ô-xy.

- Ưu, nhược điểm của nhóm giải pháp về công nghệ máy:

+ Ưu điểm: Hiệu quả mang lại rất lớn;

+ Nhược điểm: Việc cải tiến công nghệ máy mất thời gian dài, chi phí đầu tư nghiên cứu, thực hiện lớn và chỉ áp dụng được ở thời điểm chế tạo máy cho con tàu.

1.1.1.3. Nhóm giải pháp về khai thác tàu

- Trong lĩnh vực khai thác tàu chúng ta có thể giảm tốc độ tàu, tăng kích thước đội tàu vận tải để giảm lượng tiêu thụ nhiên liệu, phát thải khí nhà kính trên một đơn vị hàng hóa được vận chuyển.

- Hạn chế: Giảm tốc độ khai thác tàu đồng nghĩa với việc giảm khả năng cạnh tranh với các loại hình vận tải khác; Tăng kích thước đội tàu phụ thuộc rất lớn vào cơ sở hạ tầng của luồng lạch, cầu tàu, bến cảng, …

1.1.1.4. Nhóm giải pháp về nhiên liệu

- Hiện tại, thế giới đang có xu hướng tìm đến các dạng nhiên liệu sạch, thân thiện với môi trường như:

+ Nhiên liệu thấp lưu huỳnh;

+ Nhiên liệu Gas;

+ Nhiên liệu sinh học: Nhiên liệu lỏng (Bio – methanol, Bio – ethanol, Bio – butanol, v.v …);

+ Khí sinh học (Biogas);

+ Nhiên liệu sinh học rắn;

+ Nhiên liệu nhũ tương.

+ Nhiên liệu sạch, phát thải thấp như LNG, Hydrogen, Amonia,...

- Ưu, nhược điểm của nhóm giải pháp về nhiên liệu:

+ Ưu điểm: Rẻ hơn, sạch hơn, thân thiện với môi trường hơn;

+ Nhược điểm: Hạn chế về vấn đề cung ứng và an ninh, an toàn đối với một số nhiên liệu mới chưa được đánh giá toàn diện.

1.1.1.5. Nhóm giải pháp tối ưu hóa kế hoạch chạy tàu

Giải pháp được xem là hiệu quả, tiết kiệm chi phí được người Sỹ quan Hàng hải áp dụng phổ biến trong quá trình khai thác, vận hành con tàu là: Tối ưu hóa tuyến đường và kế hoạch chạy tàu, cụ thể:

- Phương pháp Hàng hải khí tượng (phương pháp Ischrone) [10, 18, 62, 63 64]:

Hàng hải khí tượng là việc xây dựng đường chạy tàu tối ưu căn cứ vào các dự báo thời tiết, tình trạng sóng cùng các đặc điểm của tàu khi đi từ 1 điểm tới 1 điểm cho trước.

Trong điều kiện thời tiết và tình trạng sóng nhất định, đường chạy tàu tối ưu là đường chạy tàu đảm bảo: An toàn tàu, sức khỏe thuyền viên, tiết kiệm nhiên liệu và thời gian hành trình được rút ngắn tối đa.

Ở một chế độ máy nhất định, tốc độ tàu thay đổi phụ thuộc vào điều kiện thời tiết thực tế (độ cao và hướng sóng, gió, hướng và tốc độ dòng chảy, …), khi đó đường đi ngắn nhất (Hàng hải cung vòng lớn) chưa chắc đã là đường chạy tàu tối ưu.

Thực tế, bằng cách lựa chọn đường đi thích hợp qua các điểm chuyển hướng, tàu có thể lợi dụng hoặc hạn chế ảnh hưởng của các yếu tố thời tiết, do đó có thể giảm thời gian hành trình, giảm lượng tiêu hao nhiên liệu, giảm phát thải khí nhà kính từ tàu biển và đảm bảo an toàn tàu.

+ Ưu điểm: Luôn tính đường đi tiết kiệm thời gian nhất tới bất kỳ 1 điểm mút nào (tức là tuyến hành trình tới điểm bất kỳ trên 1 Ischrone là tuyến tốn ít thời gian nhất để tới điểm đó).

+ Nhược điểm: Phương pháp này không thể giải quyết được các bài toán thực tế chẳng hạn như hạn chế tốc độ trong giai đoạn đầu để chờ các điều kiện thời tiết xấu đi qua. Hơn nữa hàm mục tiêu của phương pháp Ischrone luôn là "*thời gian hành trình nhỏ nhất*", do đó phương pháp này không linh hoạt khi cần tính toán đường chạy tàu tối ưu ứng với các hàm mục tiêu tối ưu khác.

- Tối ưu hóa tốc độ bằng kỹ thuật Just in Time:

Thực tế, giữa tốc độ tàu và mức tiêu hao nhiên liệu của tàu biển có mối liên hệ tương quan. Cụ thể, khi khai thác tàu ở tốc độ thấp thì lượng tiêu thụ nhiên liệu thấp hơn theo đó lượng khí nhà kính phát thải từ tàu biển giảm đi và ngược lại.

Do đó, trong quá trình khai thác tàu, dựa trên việc trao đổi thông tin đầy đủ, chính xác giữa các bên, người khai thác có thể biết chính xác thời điểm tàu cần có mặt tại cảng đến và khi đó, có thể tính toán chạy tàu với tốc độ phù hợp để

đảm bảo tiết kiệm nhiên liệu, giảm phát thải khí nhà kính nhờ chạy tàu với tốc độ thấp và tàu vẫn có mặt tại cảng đích đúng thời điểm – Đây chính là phương pháp Tối ưu hóa tốc độ tàu bằng kỹ thuật just in time "tàu đến cảng kịp lúc".

+ Ưu điểm: Tiết kiệm nhiên liệu tối đa cho tàu biển, giảm lượng khí nhà kính phát thải từ tàu xuống thấp nhất;

+ Nhược điểm: Phương pháp này yêu cầu thông tin liên tục và thông suốt giữa các bên trong quá trình khai thác tàu để tính toán khoảng thời gian dự trữ thích hợp, tránh tàu bị phạt do đến muộn.

- Kết hợp hàng hải khí tượng và tốc độ tối ưu:

Cách đơn giản nhất để thực hiện việc này là lặp lại việc tính toán hàng hải khí tượng cho tàu ở các chế độ máy khác nhau, sau đó lựa chọn chế độ máy mà ở đó thời gian tàu tới cảng đích thỏa mãn yêu cầu về thời gian chạy tàu, đồng thời mức tiêu thụ nhiên liệu trên hành trình là nhỏ nhất. Khi đó tuyến đường chạy tàu được lựa chọn là: Tuyến đường hàng hải khí tượng tương ứng với chế độ máy này.

+ Ưu điểm: Xây dựng được nhiều tuyến đường hàng hải khí tượng tương ứng với nhiều chế độ máy khác nhau;

+ Nhược điểm: Việc áp dụng vào thực tế không linh hoạt do bài toán thực tế luôn luôn thay đổi.

- Sử dụng dịch vụ tư vấn Hàng hải truyền thống, kết hợp với một số giải pháp kỹ thuật hiện có:

Sử dụng thông tin khí tượng dẫn đường của một số tổ chức cung cấp dịch vụ lớn trên thế giới hoặc ứng dụng mộ số hệ thống Hỗ trợ hàng hải tối ưu đang có hiện nay như: Hệ thống Char Co, Hệ thống tối ưu hóa hoạt động tàu (SPOS – Ship Performance Optimization System), AMI Seaware Routing, Sea Planner, phần mềm thời tiết Interactive weather của Clearpoint, ...

Hạn chế:

+ Chi phí cao;

+ Tiếp cận khó khăn, rào cản ngôn ngữ;

+ Hạn chế thông tin về phương pháp tính toán (thuyền viên không biết và làm chủ được);

+ Thiếu hướng dẫn khai thác do thiếu thông tin về đặc điểm khai thác tàu.

Ngoài ra, các kết quả đưa ra từ các dịch vụ tư vấn tuyến đường chạy tàu còn chưa thực sự đáng tin cậy và chuẩn xác do:

+ Chưa tính toán được đầy đủ, chi tiết đặc điểm điều động của tàu và ảnh hưởng của các yếu tố thời tiết đến tàu;

+ Hầu hết chưa tính toán tới vấn đề hiệu quả nhiên liệu khi tàu thay đổi chế độ máy;

+ Chưa tính toán tới nguy cơ có thể gặp phải, ví dụ như: nguy cơ đến muộn, gặp vùng thời tiết xấu, v.v … khi các yếu tố đầu vào thay đổi.

1.1.2. Một số phương pháp tính toán tối ưu được ứng dụng để tính toán tuyến đường chạy tàu

Mục tiêu tối ưu là giảm thiểu chi phí hoặc tối đa hóa hiệu quả. Thuật toán tối ưu là một quy trình được thực hiện lặp đi lặp lại việc so sánh các giải pháp khác nhau cho đến khi tìm thấy một giải pháp tối ưu hoặc thỏa đáng. Hiện nay, có rất nhiều thuật toán tối ưu được nghiên cứu và ứng dụng hiệu quả cho nhiều lĩnh vực khoa học, sản xuất và đời sống khác nhau.

Phương pháp tối ưu tính toán tuyến đường chạy tàu NCS đề cập đến ở đây chính là việc ứng dụng thuật toán tối ưu để tính toán tuyến đường chạy tàu thỏa mãn chi phí của tuyến (hay hàm mục tiêu gắn liền với tuyến).

Một số phương pháp tối ưu được ứng dụng để tính toán tuyến đường chạy tàu tối ưu:

1.1.2.1. Phương pháp quy hoạch động [78, 86]

Phương pháp quy hoạch động là phương pháp phổ biến để tìm tuyến đường ngắn nhất đi từ điểm đầu đến điểm cuối trên một mạng lưới. Mỗi đoạn giữa 2 nút được gắn với 1 chi phí nhất định (có thể là độ dài quãng đường hay thời gian di chuyển cần thiết). Tuyến đường tối ưu được lựa chọn là tuyến đi qua các nút, tới điểm cuối mà có tổng chi phí là nhỏ nhất.

1.1.2.2. Phương pháp đàn kiến (ACOA - Ant Colony Optimization Algorithm) [39, 50]

Phương pháp đàn kiến ứng dụng thuật toán tối ưu đàn kiến (Ant Colony Optimization Algorithm) mô phỏng quá trình tìm kiếm thức ăn của đàn kiến, theo đó, trong quá trình tìm kiếm thức ăn từ tổ, các cá thể kiến trải trên đường di chuyển một loại hóa chất đặc biệt (Pheromone) có thể được phát hiện và mờ dần theo thời gian. Pheromone do một cá thể phát ra có thể được nhận biết bởi các cá thể khác và các cá thể này có xu hướng đi theo đường của cá thể trước đã đi để tìm kiếm thức ăn. Trong quá trình di chuyển, các cá thể sau có thể đi chệch đường ban đầu đến nơi có thức ăn theo các đường khác nhau. Những đường ngắn hơn có thời gian di chuyển cần thiết ngắn, do đó lượng pheromone trải ra nhiều và dễ nhận hơn các đường trước đó. Cứ như vậy, sau một thời gian, toàn bộ đàn kiến sẽ tập trung trên một đường ngắn nhất từ tổ tới nơi có nguồn thức ăn.

1.1.2.3. Phương pháp vét cạn (EC - Exhausted Search) [80]

Phương pháp này ứng dụng thuật toán vét cạn và thường chỉ ứng dụng được với bài toán có kích thước nhỏ.

1.1.2.4. Phương pháp leo đồi (HCA - Hill Climbing Algorithm) [81]

Khả năng tìm kiếm được kết quả tối ưu toàn cục của thuật toán leo đồi là tương đối thấp.

1.1.2.5. Phương pháp di truyền (GA - Genetic Algorithm) [40]

Phương pháp GA ứng dụng thuật toán di truyền (Genetic Algorithm) để tính toán tuyến đường chạy tàu tối ưu.

Thuật toán di truyền là thuật toán tìm kiếm ngẫu nhiên dựa trên cơ chế chọn lọc tự nhiên. Quá trình tiến hóa tự nhiên là quá trình hoàn hảo nhất, hợp lý nhất và nó tự mang tính tối ưu.

Theo học thuyết Đắc Uyn, từ một quần thể (hay tập hợp các cá thể) ban đầu, trải qua quá trình biến đổi thích nghi với điều kiện sống tạo ra một lớp con cháu. Các cá thể tốt, thích nghi tốt sẽ được lựa chọn để lai tạo và đột biến, trong khi các cá thể kém hơn sẽ bị đào thải.

Đúc kết tư tưởng này của tự nhiên, thuật toán di truyền cũng duy trì một lớp lời giải (hay lớp quần thể) ban đầu, thông qua quá trình tiến hóa (lai tạo, đột biến) để hình thành một lớp mới với hy vọng lớp mới sẽ tốt hơn lớp cũ.

Quá trình tiến hóa diễn ra liên tục cho đến khi các hàm mục tiêu dần đạt được, khi đó lời giải của bài toán được xác định. Quá trình tiến hóa thế hệ sau bao giờ cũng tốt hơn, hoàn thiện hơn thế hệ trước.

1.1.2.6. Phương pháp vi khuẩn (BFOA - Bacterial Foraging Optimization Algorithm) [22, 25]

Phương pháp BFOA ứng dụng thuật toán tối ưu dựa trên phương pháp tìm kiếm thức ăn của bầy vi khuẩn (BFOA - Bacterial Foraging Optimization Algorithm) để tính toán tuyến đường chạy tàu tối ưu.

Thuật toán BFO (Bacterial Foraging Optimization Algorithm) được đề xuất lần đầu tiên bởi Passino vào năm 2002 đã thu hút được sự quan tâm của nhiều nhà nghiên cứu trong nhiều năm qua. BFOA - Thuật toán tối ưu dựa trên phương pháp tìm kiếm thức ăn của bầy vi khuẩn. Thuật toán là giải pháp khả thi, được áp dụng hiệu quả cho nhiều lĩnh vực khác nhau (như điều khiển tối ưu, trí tuệ nhân tạo, dự đoán điều hòa, …) bởi các ưu điểm mà thuật toán mang lại, cụ thể:

- Thuật toán BFO là thuật toán tối ưu dựa trên số đông các vi sinh vật đơn giản, các phần tử riêng biệt trong tập hợp có tính chất tự chủ và phân tán, không có sự điều khiển tập trung nên việc 1 hoặc 1 số phần tử kém hiệu quả hoặc thất bại không làm ảnh hưởng tới việc giải quyết vấn đề của cả tập hợp, các phần tử còn lại vẫn có khả năng tìm nghiệm tối ưu cho bài toán một cách độc lập. Nhờ vậy, thuật toán BFO hiệu quả và mạnh hơn so với các phương pháp số khác;

- Thuật toán BFO có thể được mở rộng một cách dễ dàng do: Việc hợp tác (kết bầy) của các cá thể là thông qua các liên lạc gián tiếp, nhờ sự mở rộng dễ dàng của thuật toán BFO, ta có thể tăng quy mô tập hợp vi khuẩn để giải quyết những vấn đề phức tạp hơn một cách nhanh chóng.

- Thuật toán BFO chủ yếu sử dụng các công thức toán học cơ bản do đó có thể áp dụng một cách đơn giản, nhanh chóng và hiệu quả trên máy tính;

- Thuật toán BFO khi được áp dụng không cần phải giả định về tính khả vi, hàm lồi cũng như các yêu cầu khác về mặt toán học nên thuật toán được ứng dụng để giải quyết nhiều vấn đề khác nhau về tối ưu.

1.2. Một số nghiên cứu về tính toán tuyến đường cho tàu biển

NCS, tổng hợp được một số công trình nghiên cứu khác nhau về việc tính toán tuyến đường cho tàu biển như sau:

- Nghiên cứu của Tsou, Ming-Cheng & Hsueh, Chao-Kuang (2010): "*The study of ship collision avoidance route planning by ant colony algorithm*", sử dụng thuật toán tối ưu đàn kiến để lập kế hoạch chạy tàu phòng ngừa va chạm tàu trên biển hỗ trợ sỹ quan trực ca ra quyết định [50];

- Nghiên cứu của LauraWalther, Anisa Rizvanolli, Mareike Wendebourg, Carlos Jahn (2016): *"Modeling and Optimization Algorithms in Ship Weather Routing",* sử dụng các phương pháp như thuật toán Dijkstra, quy hoạch động, phương pháp điều khiển tối ưu đến phương pháp đường đẳng thời gian để tối ưu hóa tuyến đường chạy tàu thời tiết [23];

- Nghiên cứu M.D. Nguyen et al (2013), *Multi-Scale Automatic Route Planning Algorithms for Sea-Going Vessel*, AMFUF 2013 [24];

- Nghiên cứu của Raphael Zaccone, Massimo Figari, Michele Martelli*I (*2018): *"An Optimization Tool For Ship Route Planning In Real Weather Scenarios",* các tác giả dựa trên các bản đồ dự báo thời tiết, sử dụng thuật toán tối ưu để lập tuyến đường chạy tàu tiêu thụ nhiên liệu tối thiểu [26];

- Nghiên cứu năm 2019 của A. P. Teixeira & C. Guedes Soares, "*AIS Based Shipping Routes Using the Dijkstra Algorithm"*, dựa vào dữ liệu tuyến đường chạy tàu trên Hệ thống tự động nhận dạng (AIS) đã sử dụng thuật toán Dijkstra để xây dựng tuyến đường chạy tàu an toàn [51];

- Nghiên cứu M.D. Nguyen et al, ICAIS 2012, sử dụng thuật toán vi khuẩn (BFOA – Bacterial Foraging Algorithm) nhằm mục đích tối ưu hóa hàm chi phí

(Cost Function) trong quá trình tránh va của tàu, được tính toán dựa trên thời gian chạy tàu, mức độ tuân thủ quy định về phòng ngừa đâm va tàu thuyền trên biển [52];

- Nghiên cứu Nguyen Minh Duc, Tamaru Hitoi (năm 2010) đã đưa ra nguyên lý cơ bản về việc xác định quỹ đạo tránh va cho tàu dựa trên các thông tin về mục tiêu và khu vực hành trình thu được, sử dụng phương pháp quy hoạch động dựa trên thuật toán Dijicka (Dynamic Programming) [54];

- Trong nghiên cứu Luận văn Tiến sĩ Nguyễn Minh Đức, tác giả sử dụng thuật toán tối ưu đàn kiến xác định tuyến đường tránh va cho tàu, với các thông tin về chuyển động của tàu khác thu nhận được qua các thiết bị theo dõi, giám sát chuyển động tàu (GPS, Radar, AIS) [57];

- Trong nghiên cứu Luận văn Tiến sĩ Phạm Ngọc Hà, tác giả xây dựng thuật toán BFO thích nghi để tính toán tuyến đường tìm kiếm cứu nạn tối ưu cho các phương tiện gặp nạn nhằm nâng cao năng lực tìm kiếm trong vùng biển từ Ninh Thuận - Kiên Giang [18].

NCS nhận thấy hướng tiếp cận theo phương pháp tối ưu được sử dụng trong các nghiên cứu trên được khẳng định là các phương pháp hiệu quả, có khả năng áp dụng cao trong các bài toán tối ưu phức tạp. Các phương pháp này hoàn toàn có thể nghiên cứu, sửa đổi để ứng dụng trong việc tính toán kế hoạch chạy tàu tối ưu nhiên liệu dựa trên nguyên tắc just in time "tàu đến cảng kịp lúc" mà NCS sẽ đề cập trong quá trình nghiên cứu đề tài luận án của mình.

1.3. Khái niệm về tuyến đường chạy tàu tối ưu và kế hoạch chạy tàu tối ưu

Công ước quốc tế về an toàn sinh mạng con người trên biển – SOLAS [68], quy định 34, Chương V về an toàn hàng hải: "Trước chuyến hành trình Thuyền trưởng phải đảm bảo rằng hành trình dự định đã được lập kế hoạch, sử dụng các hải đồ thích hợp và ấn phẩm hàng hải liên quan đến vùng biển đó có lưu ý đến các hướng dẫn và khuyến nghị của IMO đã ban hành". Bộ luật STCW [71] Mục A-III, phần 2 quy định về "kế hoạch hành trình" của tàu (kế hoạch chạy tàu cho toàn bộ hành trình) như sau.

"*Kế hoạch hành trình phải:*

. Tính đến bất kỳ các hệ thống tuyến hàng hải liên quan;

. Đảm bảo một khu vực biển đủ cho an toàn hành trình của tàu trong cả chuyến đi;

. Lượng định trước tất cả các chướng ngại hàng hải và điều kiện thời tiết không thuận lợi;

. Kế hoạch hành trình do thuyền phó hai chuẩn bị và thuyền trưởng ký duyệt trước khi hành trình bắt đầu."

Mục đích của tuyến đường và kế hoạch chạy tàu tối ưu chính là đảm bảo an toàn hàng hải, hiệu quả kinh tế (nâng cao hiệu quả sử dụng năng lượng) trên toàn bộ tuyến hành trình; góp phần bảo vệ môi trường, giảm phát thải khí nhà kính từ hoạt động khai thác tàu.

1.3.1. Khái niệm tuyến đường chạy tàu tối ưu (đường đi có lợi nhất) [7]

Tuyến đường chạy tàu tối ưu đảm bảo mang lại hiệu quả kinh tế cao nhất. Hiệu quả kinh tế có được khi hành trình an toàn và chi phí cho chuyến đi giảm. Tổng chi phí cho một chuyến đi (B) được tính theo công thức:

$$B = \frac{Q_c}{24} t_c + \frac{Q_d}{24} t_d \qquad (1.1)$$

Trong đó:

Q_d, Q_c: Chi phí cho một ngày đêm tàu đỗ và tàu chạy;

t_d, t_c: Thời gian tàu đỗ, tàu chạy.

Trong khai thác vận tải biển, các công ty khai thác vận tải biển luôn cố gắng rút ngắn thời gian cả chuyến đi: $T = t_c + t_d$ để giảm chi phí B của chuyến đi và quay vòng khai thác nhanh.

Chi phí cho một ngày chạy tàu thường lớn hơn một ngày tàu đỗ rất nhiều ($Q_c >> Q_d$), do đó lựa chọn và dẫn tàu theo đường đi tối ưu là một nghiệp vụ hàng hải quan trọng mang lại lợi ích kinh tế cao cho công ty.

Tuyến đường chạy tàu tối ưu là sự kết hợp của 03 yêu cầu sau:

- Đường chạy tàu ngắn nhất;

- Đường chạy tàu an toàn nhất;

- Đường chạy tàu hết ít thời gian nhất.

Thực tế, đường đi ngắn nhất chưa hẳn là đường đi an toàn nhất và đường đi hết ít thời gian nhất. Mặt khác, tính an toàn và kinh tế thường mâu thuẫn trong việc lựa chọn đường đi có lợi nhất, tùy theo điều kiện và hoàn cảnh cụ thể mới có thể đưa ra được phương án dẫn tàu tối ưu.

- Đường chạy tàu ngắn nhất:

Đường ngắn nhất giữa 2 điểm trên bề mặt trái đất là cung vòng lớn (Great circle) hay còn gọi là đường Ôc tô. Tuy nhiên, không phải lúc nào cũng dẫn tàu được theo cung vòng lớn vì nhiều lí do.

- Đường chạy tàu an toàn nhất là đường chạy tàu phải đảm bảo an toàn cho con người, tàu, hàng hóa và môi trường biển. Dựa trên tuyến đường ngắn nhất đã dự tính, nghiên cứu xem xét các yếu tố an toàn như độ sâu, dòng chảy, bão gió, áp thấp và các mối nguy hiểm khác, để lựa chọn hành trình chung và đường chạy tàu trên từng phân đoạn.

Phương án lựa chọn đường đi ngắn nhất và an toàn nhất dựa trên các tuyến hàng hải sau đây:

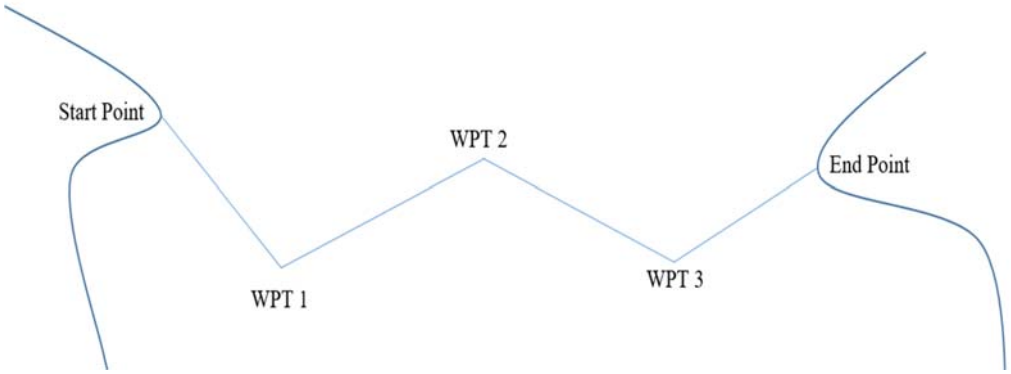
+ Tuyến đường đi hằng hướng (Rhumb line track);

+ Tuyến đường cung vòng lớn (Great circle track);

+ Tuyến hàng hải hỗn hợp (Composite track).

- Đường chạy tàu hết ít thời gian nhất: Trên cơ sở tuyến đường đi ngắn nhất và an toàn nhất đã dự định, xem xét các yếu tố khí tượng thủy văn để điều chỉnh rút ngắn thời gian chạy tàu. Thực tế, điều kiện khí tượng thủy văn là yếu tố khó định lượng, khó lựa chọn nhất nên cần có sự nghiên cứu kỹ lưỡng các dự báo, bản tin nhận được trong quá trình dẫn tàu và kết hợp với thông tin do dịch vụ khí tượng dẫn đường cung cấp.

Như vậy, tuyến đường chạy tàu tối ưu (hay đường đi có lợi nhất) là tuyến đường kết hợp được cả ba yếu tố trên để đảm bảo an toàn tối đa cho con người, tàu, hàng hóa và môi trường biển đồng thời mang lại hiệu quả kinh tế cao nhất cho chuyến hành trình.



Hình 1.2 Ví dụ về 1 tuyến đường chạy tàu tối ưu với điểm xuất phát, các điểm chuyển hướng và điểm đến.

Quá trình xây dựng (thiết lập) tuyến đường hàng hải tối ưu, người Sĩ quan hàng hải cần lưu tâm tới:

- Điều kiện khí tượng, thủy văn (sóng, gió, dòng chảy, tầm nhìn xa, …);
- Tình trạng tàu và các trang thiết bị của tàu;
- Các đặc tính và hạn chế của tàu;
- Đặc điểm của hàng được chở;
- Tình trạng sức khỏe của thuyền viên, hành khách, hàng hóa được chuyển chở;
- Hải đồ cần được chuẩn bị đầy đủ, đảm bảo tỉ lệ xích và được cập nhật phù hợp;
- Các thông báo hàng hải, các tài liệu khác liên quan như: Sailing Directions, List of Lights, List of Radio Aids to Navigation;
- Thêm vào đó là các thông tin hỗ trợ chuyến đi khác như hải đồ tuyến đường, hướng dẫn tuyến đường, tài liệu về thủy triều và dòng triều, ví dụ như: Lịch, Atlas;

Ngoài ra, như chúng ta đã biết năm 1948, IMCO (Intergovermental Maritime Consultative Organization – Tổ chức liên chính phủ giải quyết các vấn đề Hàng hải) ra đời, đây là tổ chức tiền thân của IMO, đến năm 1982, IMCO đổi tên thành IMO [77] như hiện nay.

Tiêu chí hoạt động của IMO ngay từ khi mới ra đời "Hàng hải an toàn trên biển sạch", đến năm 1995, tiêu chí hoạt động của IMO đổi thành "Hàng hải an toàn, an ninh và hiệu quả cao trên biển sạch". Như vậy, theo thời gian, cùng với sự phát triển không ngừng của khoa học kĩ thuật – công nghệ, tổ chức Hàng hải quốc tế IMO đã kịp thời điều chỉnh tiêu chí hoạt động của mình cho phù hợp (giữ nguyên tiêu chí an toàn và bảo vệ môi trường, thêm tiêu chí an ninh và hiệu quả cao). Chính vì vậy trên các tàu biển, nhiều hệ thống dịch vụ thông tin khí tượng và tuyến đường hiện đại đã được phát triển và sử dụng với mục đích: Đáp ứng đầy đủ các tiêu chí của IMO [77].

Nhiều thiết bị, hệ thống hỗ trợ đã trở thành hạng mục bắt buộc phải lắp đặt trên tàu biển theo qui định của IMO, được trính dẫn trong SOLAS [68], chẳng hạn như: ECDIS [6, 32], INMARSAT C [6, 79], …

Do đó, trong quá trình xây dựng (thiết lập) tuyến đường hàng hải tối ưu, người Sĩ quan hàng hải có thể tham khảo thêm thông tin từ các trang thiết bị buồng lái hiện đại, các yêu cầu đối với trực ca hàng hải theo Công ước STCW 78/95 sửa đổi Manila 2010, sử dụng các hệ thống hỗ trợ dịch vụ hàng hải, ví dụ như: ECDIS [6, 32], INMARSAT [6, 79], Hệ thống tối ưu hóa hoạt động tàu (SPOS – Ship Performance Optimization System), hệ thống Chart Co, hệ thống Voyage Planner, VTS, Ship Report System, Hàng hải khí tượng, hoa tiêu, … Hơn nữa, người sĩ quan hàng hải cũng xem xét đến yếu tố vùng mạn khô quốc tế, khu vực theo mùa; thời gian từng khu vực theo mùa, là những quy định về chiều cao mạn khô tối thiểu mà tàu có thể chuyên chở hàng hóa khi đi qua các vùng biển, khu vực nào những thời gian khác nhau, bên cạnh đó tham khảo thông tin trong "Bản đồ chiều cao mạn khô (Loadline chart) theo Công ước LOADLINE 1966 [70].

Dựa vào các thông tin đầy đủ thu thập được, người Sĩ quan hàng hải thiết lập tuyến đường hàng hải tối ưu từ cầu tới cầu, bao gồm cả các khu vực có hoa tiêu hỗ trợ (tuyến đường sau đó có sự kiểm tra, phê duyệt của Thuyền trưởng).

Trên cơ sở tuyến đường hàng hải tối ưu vừa thiết lập, người Sĩ quan hàng hải sẽ có đầy đủ các thông số cần thiết để dẫn tàu an toàn như: Hướng đi, các khu vực nguy hiểm, các khu vực có dịch vụ VTS, báo cáo tàu hoặc các khu vực khác cần quan tâm, chân hoa tiêu (độ sâu dưới đáy tàu) UKC – Under Keel Clearance, đặc điểm khu vực hàng hải, tốc độ an toàn của tàu trên từng đoạn tuyến, điểm chuyển hướng và quĩ đạo của tàu khi qua các điểm chuyển hướng dưới tác động của các yếu tố thời tiết, khí tượng thủy văn (sóng, gió, dòng chảy, thủy triều), độ sâu khu vực chạy tàu, khu vực nông cạn, …

1.3.2. Khái niệm kế hoạch chạy tàu tối ưu [7]

Để nâng cao hơn nữa hiệu quả sử dụng năng lượng trên tàu biển và giảm thiểu đến mức thấp nhất lượng khí nhà kính phát thải từ tàu biển, người Sĩ quan hàng hải cần có kế hoạch chạy tàu tối ưu trên từng đoạn của tuyến đường tối ưu đã được thiết lập hay nói cách khác người Sĩ quan Hàng hải cần phải có kế hoạch vận hành tàu tối ưu (phương án dẫn tàu tối ưu) trên từng đoạn tuyến đã được xây dựng.

Phương án dẫn tàu tối ưu (kế hoạch vận hành tàu tối ưu) ở đây chính là việc sử dụng chế độ máy, tốc độ tàu, các đặc tính điều động, đặc tính quay trở, … của tàu một cách hợp lý trong suốt quá trình dẫn tàu theo tuyến đường tối ưu dự kiến đã được xây dựng trước.

Đồng thời khi có các vấn đề phát sinh, chẳng hạn như thay đổi bất thường của thời tiết thì kế hoạch vận hành tàu tối ưu ở đây chính là sự thay đổi linh hoạt phù hợp với hoàn cảnh thực tế để đảm bảo an toàn tàu hoặc để nâng cao hiệu quả năng lượng của tàu góp phần giảm thiểu khí nhà kính, chống biến đổi khí hậu.

Ứng với điều kiện thời tiết thay đổi mới cập nhật, người Sĩ quan hàng hải có thể tính toán lại tuyến đường hàng hải từ vị trí hiện tại của tàu để có thể tiết kiệm nhiên liệu hoặc thời gian hành trình hiệu quả nhất.

Tốc độ tàu hoặc tốc độ an toàn của tàu trên từng đoạn tuyến cũng cần được người Sĩ quan hàng hải lưu tâm, thay đổi linh hoạt, hợp lí tùy thuộc vào đặc tính tàu, đặc điểm khu vực hàng hải, thời điểm tàu hành trình qua đoạn đó (ngày hay đêm), độ sâu, khu vực nông cạn, …

Như vậy, việc vận hành tối ưu luôn luôn đòi hỏi người Sĩ quan hàng hải phải có phương án dự phòng, sẵn sàng thay đổi một cách linh hoạt và hợp lí nhất.

Thực tế, giữa tốc độ tàu và mức tiêu hao nhiên liệu của tàu biển có mối liên hệ tương quan, cụ thể: Khi khai thác tàu ở tốc độ thấp thì lượng tiêu thụ nhiên liệu thấp hơn, theo đó lượng khí nhà kính phát thải từ tàu biển giảm đi và ngược lại khi khai thác tàu ở tốc độ cao thì lượng tiêu hao nhiên liệu sẽ tăng lên dẫn đến lượng khí nhà kính phát thải từ hoạt động khai thác tàu biển cũng tăng lên.

Do đó, trong quá trình khai thác tàu, dựa trên việc trao đổi thông tin đầy đủ, chính xác giữa các bên, người khai thác có thể biết chính xác thời điểm tàu cần có mặt tại cảng đến và khi đó, có thể tính toán chạy tàu với tốc độ phù hợp để đảm bảo tiết kiệm nhiên liệu, giảm phát thải khí nhà kính nhờ chạy tàu với tốc độ thấp và tàu vẫn có mặt tại cảng đích đúng thời hạn – Đây chính là phương pháp Tối ưu hóa tốc độ tàu hay tối ưu hóa nhiên liệu tàu biển bằng kỹ thuật (nguyên tắc) just in time “tàu đến cảng kịp lúc”.

Sử dụng phương pháp này sẽ tiết kiệm được nhiên liệu tối đa cho tàu biển, giảm lượng khí nhà kính phát thải từ tàu xuống thấp nhất. Trong chương trình tăng hiệu quả năng lượng, giảm phát thải khí nhà kính từ tàu biển: Việc giảm tốc độ tàu xuống thấp đang được xem xét một cách nghiêm túc nhờ vào các ưu điểm mà nó mang lại.

Phương pháp tuyến đường chạy tàu tối ưu nhiên liệu just in time yêu cầu thông tin liên tục và thông suốt giữa các bên trong quá trình khai thác tàu để tính toán khoảng thời gian dự trữ thích hợp, tránh tàu bị phạt do đến muộn.

1.4. Các yếu tố ảnh hưởng tới việc tính toán tuyến đường và kế hoạch chạy tàu tối ưu

Thực tế Hàng hải cho thấy có rất nhiều yếu tố ảnh hưởng tới việc tính toán tuyến đường và kế hoạch chạy tàu tối ưu, tuy nhiên trong khuôn khổ nghiên cứu của đề tài luận án, NCS xem xét ảnh hưởng của các yếu tố sau:

1.4.1. Các yếu tố thời tiết, khí tượng thủy văn

Các yếu tố thời tiết, khí tượng thủy văn ảnh hưởng trực tiếp tới hoạt động (tốc độ) tàu. Trong phần này, NCS tập trung nghiên cứu ảnh hưởng của 3 yếu tố: Sóng, gió và dòng chảy, cụ thể:

- Sóng biển [8, 27, 35, 36]: Sự tổn thất tốc độ tàu chủ yếu phụ thuộc vào độ cao sóng. Sóng gây ra các dao động (lắc ngang, bổ dọc, …) làm ảnh hưởng đến lực đẩy chân vịt và làm tăng lực cản khi phải sử dụng bánh lái liên tục để duy trình hướng đi của tàu.

Khi tàu chạy với vận tốc nhỏ, kháng lực sóng không lớn lắm, chiếm khoảng 20-30% kháng lực, nhưng khi tốc độ tàu lớn, chiếm tới 60-70% toàn bộ kháng lực.

Thực tế, mối liên hệ giữa độ cao sóng và tốc độ tàu: Khi tàu chạy ngược sóng, tốc độ tàu giảm và tăng lên khi tàu đi xuôi sóng với điều kiện độ cao sóng phải ở một giới hạn nhất định (không lớn lắm); trường hợp sóng to, tốc độ tàu sẽ giảm bất kể là tàu đang đi ngược hay xuôi sóng. Do đó, khi biển động người sĩ quan hàng hải liên tục sử dụng bánh lái để giữ ổn định cho tàu và giảm tốc độ để tránh cho tàu bị rung lắc ảnh hưởng đến kết cấu, an toàn tàu và sức khỏe thuyền viên.

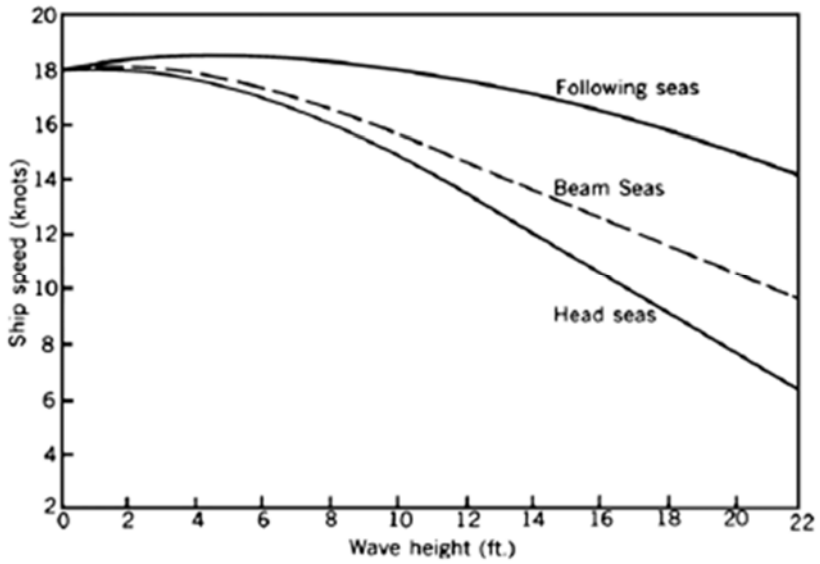
- Gió [8, 35]: Trên biển, gió là thành phần chủ yếu tạo ra sóng bề mặt, còn các thành phần khác (sóng triều, sóng ngắn, sóng do động đất,…) thì không thường xuyên và ảnh hưởng của chúng cũng không đáng kể.

Mối liên hệ giữa gió và tốc độ tàu: Thông thường khi gió nhẹ, tốc độ tàu giảm khi tàu đi ngược gió và tăng lên khi tàu đi xuôi gió do ảnh hưởng của sóng gây ra bởi gió này. Tuy nhiên, lực cản của gió đối với chuyển động tàu phụ thuộc

vào tốc độ gió, tốc độ tàu, góc hợp bởi hướng đi của tàu và hướng gió, hình dáng thân tàu và thượng tầng kiến trúc tàu, … và cần được xác định cho từng tàu cụ thể.

Thực tế, ảnh hưởng của sóng (sóng gió và sóng lừng) đối với các tàu buôn lớn hơn rất nhiều so với ảnh hưởng của gió, tuy nhiên, ta không thể tách rời ảnh hưởng của từng yếu tố riêng biệt. Hơn nữa, gió cũng chính là nguyên nhân gây ra sóng và tốc độ, hướng gió có liên quan mật thiết với độ cao, hướng sóng. Vì vậy, trong nhiều trường hợp, ảnh hưởng của gió được bao hàm luôn trong ảnh hưởng của sóng tới tốc độ tàu.

Hình vẽ mô tả kết quả thực nghiệm ảnh hưởng của sóng đối với 1 tàu có tốc độ khai thác là 18 kts khi tàu đi xuôi sóng, ngang sóng và ngược sóng [10].



Hình 1.3 Hình vẽ mô tả kết quả thực nghiệm ảnh hưởng của sóng tới tốc độ tàu trong các trường hợp tàu đi xuôi sóng, ngang sóng và ngược sóng.

- Dòng chảy (hải lưu) bề mặt [8, 35]:

Trên các đại dương, luôn luôn tồn tại những dòng chảy tương đối ổn định, các dòng chảy này gọi là hải lưu. Hải lưu được phân thành: Hải lưu gió và hải lưu Gradient (nếu xét theo nguyên nhân sinh ra dòng chảy) hoặc được phân thành: Hải lưu nóng và hải lưu lạnh (nếu xét theo đặc điểm nhiệt độ môi trường nước).

Hải lưu làm giảm tốc độ khi tàu chạy ngược dòng và ngược lại, tốc độ tàu sẽ tăng lên khi tàu xuôi dòng.

Ngoài ra, trong quá trình dẫn tàu hành hải theo tuyến đường tối ưu được xây dựng trước đó, người sĩ quan hàng hải cũng cần đặc biệt lưu tâm đến ảnh hưởng của các yếu tố khí tượng thủy văn khác xuất hiện trên tuyến, đặc biệt như bão, thủy triều (khi tàu chạy trong luồng, khu vực cảng biển), …

Thực tế, các thông tin thời tiết, các yếu tố khí tượng thủy văn tác động tới tuyến đường chạy tàu tối ưu (ảnh hưởng tới hoạt động tàu), người sĩ quan hàng hải có thể thu thập được từ rất nhiều nguồn khác nhau. Sau đây là một số nguồn phổ biến:

- Từ các thiết bị (hay hệ thống) có chức năng thu nhận thông tin thời tiết được trang bị phổ biến trên tàu như: Máy thu Navtex [8, 33, 34], máy thu thời tiết Facsimile [8];

- Từ các dịch vụ chuyên dụng như: Dịch vụ gọi nhóm tăng cường (EGC - Enhanced Group Call) của Inmarsat C [6, 79] hay sử dụng dịch vụ của một số hệ thống như: Hệ thống tối ưu hóa hoạt động tàu (SPOS – Ship Performance Optimization System), hệ thống Chart Co, hệ thống Voyage Planner, …

- Tức các nguồn thời tiết dạng số: Hiện nay, dữ liệu thời tiết của các cơ quan, tổ chức khí tượng lớn trên thế giới đều được mã hóa bằng định dạng Grib File (phổ biến hiện nay là định dạng Grib2).

Trong quá trình nghiên cứu thực hiện đề tài luận án, NCS cũng đã nghiên cứu xây dựng và cung cấp cho người sĩ quan hàng hải một bộ công cụ (hay phần mềm) hữu hiệu nhằm thu thập và xử lí thông tin thời tiết, cụ thể là bản tin sóng toàn cầu, bản tin gió toàn cầu của Rish và dòng chảy của Oscar phục vụ mục đích tính toán tuyến đường và kế hoạch chạy tàu tối ưu (được trình bày cụ thể ở Chương 2 của đề tài luận án).

1.4.2. Các đặc tính của tàu

Đặc tính tàu có ảnh hưởng quan trọng tới việc lựa chọn và áp dụng tuyến đường hàng hải tối ưu, liên quan đến vấn đề nghiên cứu, NCS xem xét ảnh hưởng

của đặc tính chuyển động tàu biển (đặc tính thay đổi tốc độ tàu biển) dưới tác động của các yếu tố thời tiết sóng, gió, dòng chảy theo rpm, draft, trim và đặc tính tiêu thụ nhiên liệu tàu biển tới việc lựa chọn và áp dụng tuyến đường hàng hải tối ưu.

Ứng dụng thuật toán bình phương nhỏ nhất (hay phương pháp bình phương nhỏ nhất), NCS xác định được đặc tính thay đổi tốc độ và đặc tính tiêu thụ nhiên liệu tối ưu trong từng điều kiện hành hải cụ thể (xác định tốc độ tối ưu, mức tiêu thụ tiêu thụ nhiên liệu tối ưu ứng với từng điều kiện sóng, gió, dòng chảy với số vòng quay chân vịt (rpm), mớn nước tàu (draft), hiệu số mớn nước tàu (trim) cụ thể). Kết quả nghiên cứu này có ý nghĩa vô cùng quan trọng trong việc lựa chọn và áp dụng tuyến đường hàng hải tối ưu (*được trình bày cụ thể ở Chương 3 của đề tài luận án*)

1.4.3. Một số yếu tố quan trọng khác

Ngoài việc xem xét ảnh hưởng của các yếu tố thời tiết, khí tượng thủy văn và đặc tính tàu (gồm đặc tính chuyển động tàu biển (đặc tính thay đổi tốc độ tàu biển), đặc tính tiêu thụ nhiên liệu tàu biển), NCS cũng xem xét ảnh hưởng của một số yếu tố quan trọng khác tới việc tính toán, lựa chọn tuyến đường hàng hải tối ưu có xem xét các hướng dẫn trong tài liệu "Ocean Passages for the World (Vol 1), NP136 Chương 1 [73] và Nghị quyết A.893(21)" của IMO [72], cụ thể như sau:

- Khối lượng hàng cần vận chuyển (mớn nước vận chuyển);
- Dự trữ nhiên liệu, nước ngọt cần thiết cho toàn tuyến;
- Độ sâu luồng lạch trên toàn tuyến;
- Mớn nước tối đa theo mùa, theo khu vực địa lý;
- Tính toán mớn nước tối ưu cho tàu (hiệu số mớn nước tối ưu), có tính toán tới sự thay đổi mớn nước do sự thay đổi lượng dự trữ nhiên liệu, nước ngọt trên toàn tuyến;
- Việc duy trì thông tin liên lạc thông suốt giữa các bên, cập nhật thời hạn yêu cầu tàu đến điểm đích để tính toán thời gian dự trữ, thời gian chờ hợp lý tránh

tình trạng tàu bị phạt do đến muộn (điểm đích ở đây có thể là cảng đến, điểm chờ qua kênh, chờ vào cảng, …).

1.5. Kết luận chương 1

Kết thúc chương 1, NCS đạt được một số kết quả nghiên cứu cụ thể như sau:

Trước tiên, NCS đã tìm hiểu và tổng hợp được một số tài liệu, công trình nghiên cứu khoa học trong và ngoài nước liên quan đến chủ đề của đề tài luận án đã được công bố.

Tiếp theo, NCS tìm hiểu khái niệm tuyến đường chạy tàu tối ưu (hay đường đi có lợi nhất) và khái niệm kế hoạch chạy tàu tối ưu (hay phương án vận hành tàu tối ưu). Đặc biệt, NCS tìm hiểu khái niệm tuyến đường chạy tàu tối ưu nhiên liệu dựa trên nguyên tắc just in time "tàu đến cảng kịp lúc" (gọi tắt là tuyến đường chạy tàu tối ưu JIT).

Cuối cùng, NCS nêu các yếu tố ảnh hưởng tới việc tính toán tuyến đường và kế hoạch chạy tàu tối ưu, gồm: Ảnh hưởng của các yếu tố thời tiết, khí tượng thủy văn; Ảnh hưởng của đặc tính thay đổi tốc độ và đặc tính tiêu thụ nhiên liệu của tàu biển trong từng điều kiện hành hải cụ thể (xét ảnh hưởng của sóng, gió, dòng chảy tương ứng với chế độ máy_ rpm, mớn nước tàu _ draft, hiệu số mớn nước tàu _ trim nhất định). Ngoài ra, NCS cũng nêu ảnh hưởng của một số yếu tố quan trọng khác tới việc tính toán tuyến đường và kế hoạch chạy tàu tối ưu.

CHƯƠNG 2: TỔNG HỢP THÔNG TIN THỜI TIẾT PHỤC VỤ TÍNH TOÁN TUYẾN ĐƯỜNG VÀ KẾ HOẠCH CHẠY TÀU TỐI ƯU

2.1. Việc thu thập thông tin thời tiết ở trên tàu hiện nay

Trong quá trình hành hải, người Sỹ quan Hàng hải có thể tiếp cận thông tin thời tiết từ nhiều nguồn khác nhau. Sau đây là một số nguồn thông tin thời tiết phổ biến mà người Sỹ quan Hàng hải thường xuyên được tiếp cận.

2.1.1. Một số nguồn thông tin thời tiết tiếp cận được trên tàu

2.1.1.1. Navtex [8, 13, 34]

Máy thu Navtex trang bị trên tàu biển có thể thu được 14 loại bức điện (gọi là bức điện Navtex). Các bức điện Navtex được ký hiệu dưới dạng chữ cái, cụ thể:

A* - Thông báo hàng hải (Navigational Warning);

B* - Thông báo về bão (Meteorological Warning);

C - Báo cáo về băng (Ice Report);

D* - Thông tin về tìm kiếm cứu nạn và cảnh báo cướp biển tấn công (Search and rescue information and pirate attack warning);

E - Dự báo thời tiết (Meteorological Forecast);

F - Bức điện về dịch vụ hoa tiêu (Pilot Message);

G - Bức điện AIS (AIS message);

H - Bức điện LORAN (Loran-C message);

I - Bức điện hiện nay không sử dụng (Reserved presently not used);

J - Bức điện về hàng hải vệ tinh (SATNAV Message);

K - Các bức điện liên quan đến các hệ thống trợ giúp hàng hải điện tử khác (Other Electronic Navigational Aid System Message);

L* - Cảnh báo hàng hải bổ sung (Navigational Warning additional to A);

V, W, X, Y - Các dịch vụ đặc biệt (Special services-allocation by NAVTEX panel);

Z - Không có điện (No message on hand).

Trong đó 4 loại bức điện: A^*, B^*, D^*, L^* là các bức điện ưu tiên mặc định thu tức là không sử dụng được chế độ loại trừ bức điện loại này trong máy.

Như vậy, người Sỹ quan Hàng hải có thể tiếp cận được thông tin thời tiết thông qua một số bức điện về thời tiết mà máy thu Navtex thu nhận được từ trạm bờ, ví dụ: Bức điện loại B^* - Thông báo về bão (Meteorological Warning), bức điện loại C - Báo cáo về băng (Ice Report), bức điện loại E - Dự báo thời tiết (Meteorological Forecast).

2.1.1.2. Máy thu thời tiết Facsimile [8]

Ngày nay, máy thu thời tiết Facsimile không phải là hạng mục bắt buộc phải trang bị trên tàu biển (theo qui định của SOLAS) đặc biệt là khi có sự xuất hiện của Navtex và dịch vụ EGC của Inmarsat C, tuy nhiên trên các tàu biển máy thu thời tiết Facsimile vẫn rất phổ biến do các ưu điểm về thông tin thời tiết mà nó mang lại cho người Sỹ quan hàng hải, cụ thể:

- Các thông tin được hiển thị dưới cả dạng chữ và hình ảnh giúp chúng ta có cái nhìn toàn cảnh về tình hình thời tiết và tình trạng mặt biển trên một khu vực rộng lớn;

- Các yếu tố ảnh hưởng trực tiếp đến an toàn hàng hải như hướng và độ cao của sóng, sóng lừng, gió bão và các hiện tượng thời tiết bất lợi như sương mù có thể được cảnh báo sớm nhờ việc khoanh vùng, đánh dấu các khu vực và cũng có thể kèm theo các ghi chú của trạm phát.

Các bản đồ thời tiết hay vùng băng, hoặc các thông tin hữu ích khác đối với người đi biển được các trạm bờ tổng hợp và định kỳ gửi đi vào những khung thời gian nhất định và có thể chia làm 2 loại chính là:

- Bản đồ thời tiết hiện tại hay còn gọi là bản đồ phân tích (Analysis): Phân tích và hiển thị tình hình hiện tại dựa trên các dữ liệu thu được từ các trạm quan trắc tại cùng một thời điểm;

- Bản đồ thời tiết tương lai (Dự báo-Forecast): Dự báo tình hình trong thời gian tới dựa trên các dữ liệu hiện tại và các qui luật vận hành, tương tác giữa các yếu tố khí tượng thủy văn.

2.1.1.3. Dịch vụ chuyên dụng

- Dịch vụ gọi nhóm tăng cường (EGC-Enhanced Group Call) [6, 79]

Theo quy định của IMO (được trích dẫn trong Solas), việc lắp đặt thiết bị INMARSAT-C trên tàu biển là bắt buộc (Thiết bị Inmarsat – C đáp ứng được 6/9 chức năng yêu cầu của hệ thống GMDSS).

Thiết bị Inmarsat – C có khả năng nhận được các bức điện quảng bá (gửi cho nhiều người nhận) gọi là EGC. EGC cho phép hai loại hình quảng bá được truyền phát đó là SafetyNET (thực hiện việc truyền phát thông tin an toàn hàng hải trong đó có thông tin thời tiết) và FleetNet (cho phép các thông tin thương mại được gửi đi tới một nhóm người sử dụng cụ thể).

- Ngoài ra, người Sỹ quan hàng hải có thể tiếp cận thông tin thời tiết thông qua việc sử dụng dịch vụ của một số hệ thống như: Hệ thống tối ưu hóa hoạt động tàu (SPOS – Ship Performance Optimization System), hệ thống Chart Co, hệ thống Voyage Planner, …

2.1.2. Các nguồn thời tiết dạng số

- Kết quả quan trắc thời tiết của Trung tâm dự báo khí tượng thủy văn Quốc gia [83] (National Centre for Hydrometeorological Forecasting – NCHMF)

https://www.nchmf.gov.vn

- Các công ty chuyên cung cấp dịch vụ thời tiết nước ngoài với độ chính xác rất cao: Công ty Fugro GEOS [84] (https://www.fugroweather.com), công ty Offshore Weather Services [85] (https://www.offshoreweather.com);

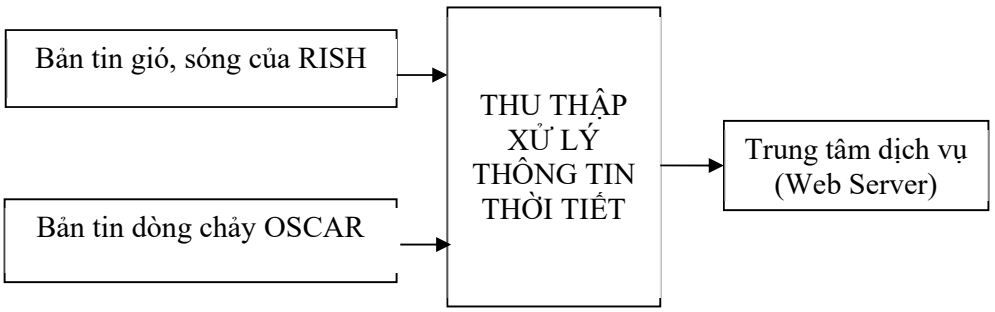
- Dữ liệu thời tiết từ các cơ quan khí tượng thủy văn lớn trên thế giới: Trung tâm dự báo hạn vừa của Châu Âu, cơ quan khí tượng của Nhật, cơ quan khí tượng của Pháp, cơ quan khí tượng của Mỹ.

Thực tế hiện nay, dữ liệu thời tiết của các cơ quan, tổ chức khí tượng lớn trên thế giới đều được mã hóa bằng định dạng Grib File (phổ biến hiện nay là định dạng Grib2).

2.2. Thông tin thời tiết phục vụ tính toán kế hoạch chạy tàu tối ưu được nghiên cứu trong đề tài luận án

Thực tiễn hàng hải cho thấy người Sỹ quan Hàng hải có thể tiếp cận thông tin thời tiết một cách dễ dàng từ rất nhiều nguồn khác nhau, tuy nhiên với mục tiêu nghiên cứu khoa học, NCS sử dụng thông tin thời tiết gồm: Dữ liệu sóng, gió của Rish và dữ liệu dòng chảy của Oscar để tính toán tuyến đường và kế hoạch chạy tàu tối ưu.

Việc thu thập thông tin thời tiết phục vụ tính toán tuyến đường và kế hoạch chạy tàu tối ưu được thể hiện thông qua sơ đồ dưới đây:



Hình 2.1 Sơ đồ nguyên lý thu thập thông tin thời tiết phục vụ tính toán tuyến đường và kế hoạch chạy tàu tối ưu được thực hiện trong đề tài luận án

Thông tin thời tiết phục vụ tính toán tuyến đường và kế hoạch chạy tàu tối ưu được nghiên cứu trong đề tài gồm: Bản tin gió toàn cầu, bản tin sóng toàn cầu sóng của Rish và bản tin dòng chảy Oscar.

2.2.1. Bản tin sóng toàn cầu, bản tin gió toàn cầu của Rish [16, 74]

Viện nghiên cứu phát triển bền vững khí quyển nhân loại thuộc Đại học Kyoto, Nhật Bản Rish (Research Institute for Sustainable Humanoshere)

https://www.database.rish.kyoto-u.ac.jp/arch/jmadata/data/gpv/original

Rish xây dựng, cập nhật và duy trì một hệ thống cơ sở dữ liệu về dự báo và phân tích thời tiết bao gồm nhiều loại dự báo khác nhau được lưu trữ dưới dạng định dạng Grib 2 nhằm mục đích phục vụ nghiên cứu khoa học. Theo qui định của Rish cùng các cơ quan hợp tác (ví dụ: Japan Meteorology Agency), các dữ

liệu thời tiết được cung cấp miễn phí cho hợp tác nghiên cứu khoa học, trường hợp dùng cho mục đích thương mại phải có thỏa thuận riêng và mất phí.

Dữ liệu của Rish được lưu theo định dạng Grib file (cụ thể là Grib 2).

2.2.2. Dữ liệu dòng chảy Oscar [16, 47, 75]

Dự án nghiên cứu, phân tích dòng chảy đại dương theo thời gian thực Oscar (Ocean Surface Current Analysis Real – time)

Dữ liệu dòng chảy Oscar có định dạng netCDF (Format Network Common Data Form) chứa dữ liệu tốc độ dòng chảy theo vĩ tuyến (u) và kinh tuyến (v). Ví dụ bảng dữ liệu dòng chảy Oscar:

Lat	Long	u	v	Speed	Direction
5.0	102.0	-0.40	-0.50	0.64	38.7
5.0	102.5	-0.40	-0.50	0.64	38.7
5.0	103.0	-0.40	-0.50	0.64	38.7
5.0	103.5	-0.41	-0.49	0.64	39.9
5.0	104.0	-0.39	-0.49	0.63	38.5
5.0	104.5	-0.40	-0.49	0.63	39.2
5.0	105.0	-0.38	-0.48	0.61	38.4
5.0	105.5	-0.39	-0.48	0.62	39.1
5.0	106.0	-0.37	-0.48	0.61	37.6
5.0	106.5	-0.38	-0.47	0.60	39.0
5.0	107.0	-0.36	-0.47	0.59	37.5
5.0	107.5	-0.37	-0.47	0.60	38.2
5.0	108.0	-0.35	-0.46	0.58	37.3
5.0	108.5	-0.36	-0.46	0.58	38.0
5.0	109.0	-0.34	-0.46	0.57	36.5
5.0	109.5	-0.35	-0.45	0.57	37.9
5.0	110.0	-0.33	-0.45	0.56	36.3

Hình 2.2 Dữ liệu dòng chảy OSCAR

Dữ liệu dòng chảy OSCAR được cung cấp miễn phí với mục đích nghiên cứu khoa học (nếu sử dụng với mục đích khác thì phải được cấp phép và mất phí) bởi phòng thí nghiệm sức đẩy phản lực (Jet Propulsion Laboratory Physical Oceanography) thuộc Viện công nghệ California (Viện quản lý các dự án của cơ quan hàng không vũ trụ Mỹ).

2.3. Khai thác thông tin thời tiết dạng Grib file phục vụ tính toán tuyến đường và kế hoạch chạy tàu tối ưu

2.3.1. Thông tin thời tiết có định dạng Grib file [16, 74]

Dữ liệu của Rish được lưu theo định dạng Grib file (cụ thể là Grib 2).

GRIB là chữ viết tắt tiếng anh của Gridded Binary hay General Regularly distributed Information in Binary Form: Định dạng số liệu chính xác thường được sử dụng trong khí tượng để lưu trữ dữ liệu thời tiết quá khứ và dữ liệu dự báo thời tiết (Grib được tổ chức khí tượng thế giới WMO – World Meteorology Organization tiêu chuẩn hóa theo quy định về mã hóa số 306 (WMO Manual on Code No 306)).

GRIB có 3 phiên bản, trong đó:

- Grib 0, Grib 1: Đây là 2 phiên bản đầu, hiện tại không còn được sử dụng nữa;

- Grib 2: Được sử dụng rất phổ biến hiện nay nhờ các ưu điểm vượt trội, cụ thể như sau:

+ Tự định nghĩa (Self Description);

+ Linh hoạt;

+ Dễ mở rộng;

+ Có thể nén dữ liệu;

+ Có thể được bảo trì và sửa đổi dễ dạng trong trường hợp cần có các trường thông tin mới.

Trên thế giới, hầu hết các cơ quan dự báo khí tượng đều sử dụng định dạng Grib 2 để mã hóa dữ liệu thời tiết, do đó giải mã được định dạng này sẽ giúp NCS tiếp cận được các thông tin thời tiết đầy đủ, làm cơ sở để tính toán phương án chạy tàu tối ưu.

NCS sử dụng dữ liệu thời tiết (sóng, gió) lấy từ RISH, gồm:

- Bản tin dự báo gió toàn cầu (GSM – Global Surface Model)

File bản tin dự báo gió áp dụng toàn cầu có dạng:

Z_C_RJTD_yyyymmddhh0000_GSM_GPV_Rgl_FDtttt_grib2.bin

Trong đó:

+ Dlat = 0.5 deg: Khoảng cách giữa các điểm nút theo vĩ tuyến

+ Dlon = 0.5 deg: Khoảng cách giữa các điểm nút theo kinh tuyến;

+ yearyearyearyearmonthmonthdaydayhourhour0000: Thời điểm bắt đầu (UTC), hh = 00, 06, 12, 18 (Trong một ngày có 4 thời điểm quan trắc tướng ứng với hh = 00, 06, 12 và 18);

+ tttt ...: thời hạn dự báo (Tại mỗi thời điểm quan trắc trong ngày sẽ dự báo cho 134 giờ tiếp theo).

Nội dung bản tin dự báo gió của Rish gồm các thông tin: Vĩ độ, kinh độ, tốc độ gió theo hướng W-E (m/s), tốc độ gió theo hướng N-S (m/s), tốc độ gió (m/s) và hướng gió.

Ví dụ về bản tin dự báo gió toàn cầu ngày 12/12/2020 tại thời điểm quan trắc hh = 00, thời hạn dự báo là 132 giờ (tttt = 0512).

Hình 2.3 Ví dụ bản tin dự báo gió toàn cầu ngày 12/12/2020 (tại thời điểm quan trắc hh=00)

- Bản tin dự báo sóng toàn cầu (GWM – Global Wave Model)

File bản tin dự báo sóng áp dụng toàn cầu có dạng:

Z__C_RJTD_yyyymmddhh0000_GWM_GPV_Rgl_Gll0p5deg_FDtttt-tttt_grib2.bin

Trong đó:

+ yyyymmddhh0000: Thời điểm bắt đầu dự báo (Thời điểm quan trắc hh = 00, 06, 12, 18);

+ FDtttt-tttt: Thời hạn dự báo (tttt-tttt = 0000-0312 và 0318-0512).

Ví dụ về bản tin dự báo sóng toàn cầu ngày 12/12/2020

Z__C_RJTD_20201212000000_GWM_GPV_Rgl_Gll0p5deg_FD0000-0312_grib2.bin
Z__C_RJTD_20201212000000_GWM_GPV_Rgl_Gll0p5deg_FD0318-0512_grib2.bin

Hình 2.4 Ví dụ bản tin dự báo sóng toàn cầu ngày 12/12/2020 (tại thời điểm quan trắc hh=00)

2.3.2. Xây dựng phần mềm trích xuất dữ liệu thời tiết (sóng, gió) định dạng Grib 2 của Rish

Toàn bộ dữ liệu thời tiết (sóng, gió) của Rish đều được mã hóa bởi định dạng Grib 2. Do dó, để sử dụng được các dữ liệu thời tiết này, NCS sử dụng ngôn ngữ lập trình Visual Basic 2010 (VB 2010) viết chương trình giải mã file có định dạng Grib 2.

Hình 2.5 Giao diện chương trình quản lý cơ sở dữ liệu

Đăng nhập hệ thống

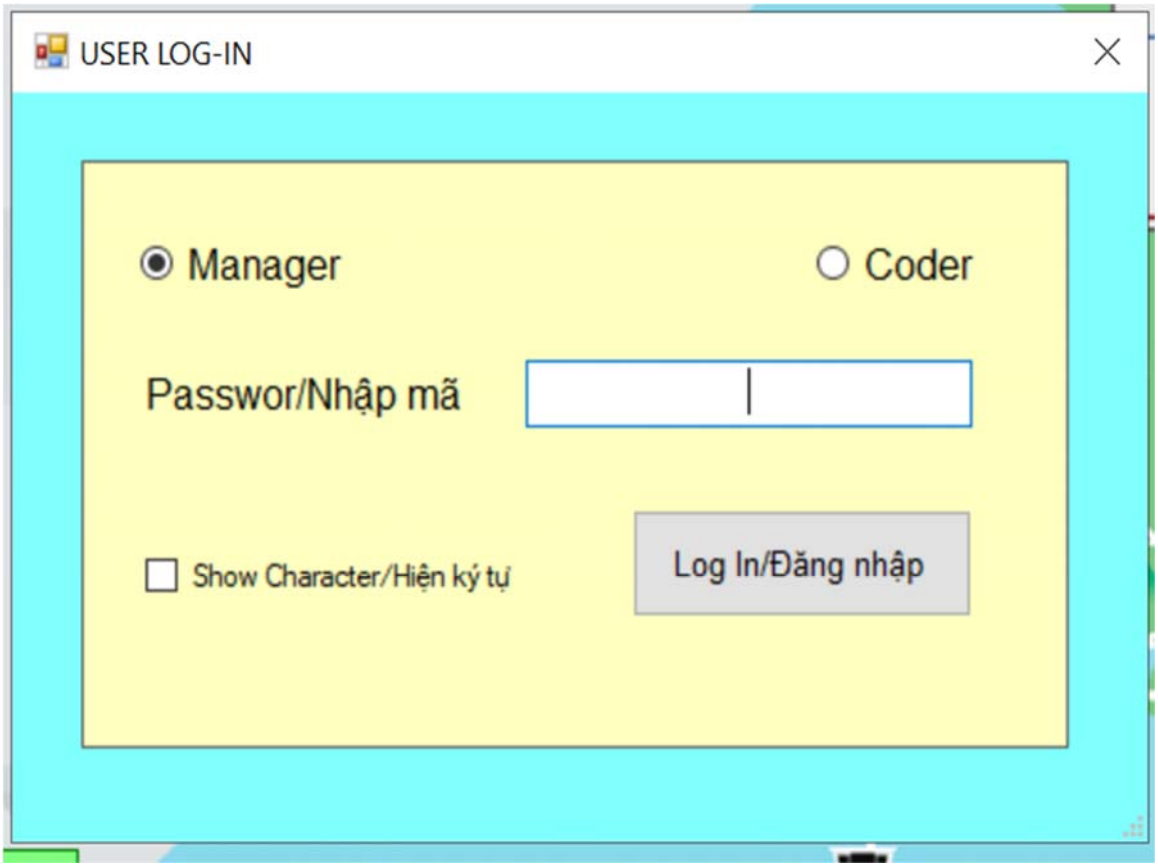
Module quản lý công ty và đội tàu

Module Upload dữ liệu thời tiết

Module quản lý thông tin thời tiết

Đăng xuất khỏi hệ thống

Đăng nhập vào hệ thống: Nhập mã và Log in vào hệ thống



Hình 2.6 Giao diện đăng nhập vào hệ thống

2.3.3. Quy trình thu thập thông tin thời tiết (sóng, gió) từ cơ sở dữ liệu của Rish

NCS tiến hành thu thập và xử lí bản tin gió, sóng của Rish, chuyển dữ liệu tới trung tâm dịch vụ (trang bị Web – Server) phục vụ tính toán tuyến đường và kế hoạch chạy tàu tối ưu.

*** Bước 1. Thu thập dữ liệu thời tiết (sóng, gió) của Rish**

Để tiếp cận dữ liệu thời tiết phục vụ mục đích nghiên cứu khoa học của Rish, NCS truy cập địa chỉ website:

https://www.database.rish.kyoto-u.ac.jp/arch/jmadata/data/gpv/original

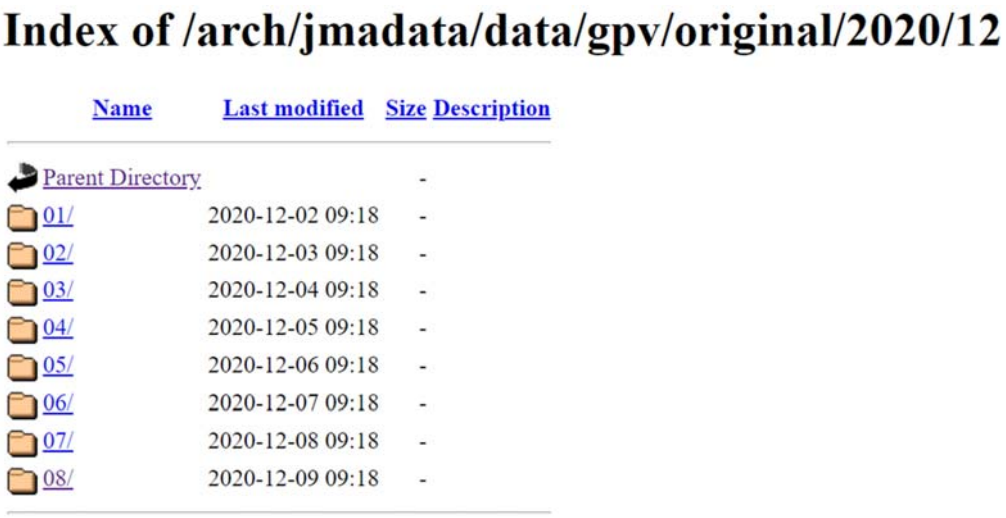
Hình 2.7 Dữ liệu thời tiết được lưu trữ theo từng năm

(Ví dụ hình 2.7 là thư mục lưu trữ dữ liệu thời tiết năm 2004, 2005)

Dữ liệu thời tiết của 1 năm được lưu trữ theo từng tháng của năm đó, ví dụ:

Hình 2.8 Dữ liệu thời tiết từ 01/2020 đến 12/2020

Dữ liệu thời tiết của 1 tháng được lưu trữ theo từng ngày của tháng đó, ví dụ:



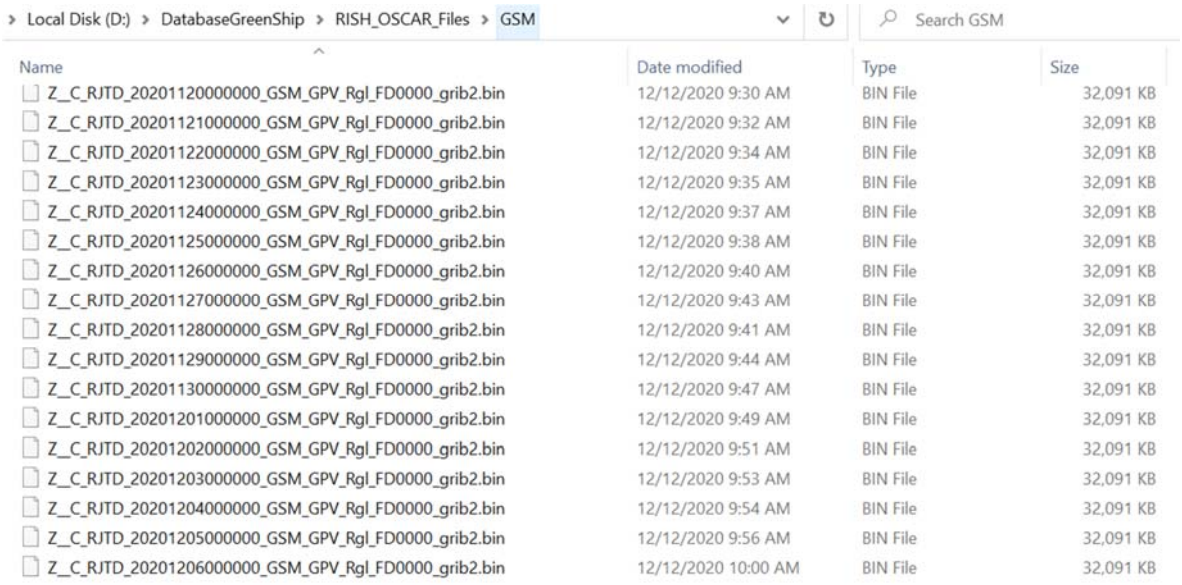
Hình 2.9 Dữ liệu thời tiết từ ngày 01/12/2020 đến 08/12/2020

Dữ liệu thời tiết của một ngày được cập nhật tới thời điểm hiện tại và được mã hóa bằng định dạng Grib 2, cụ thể:

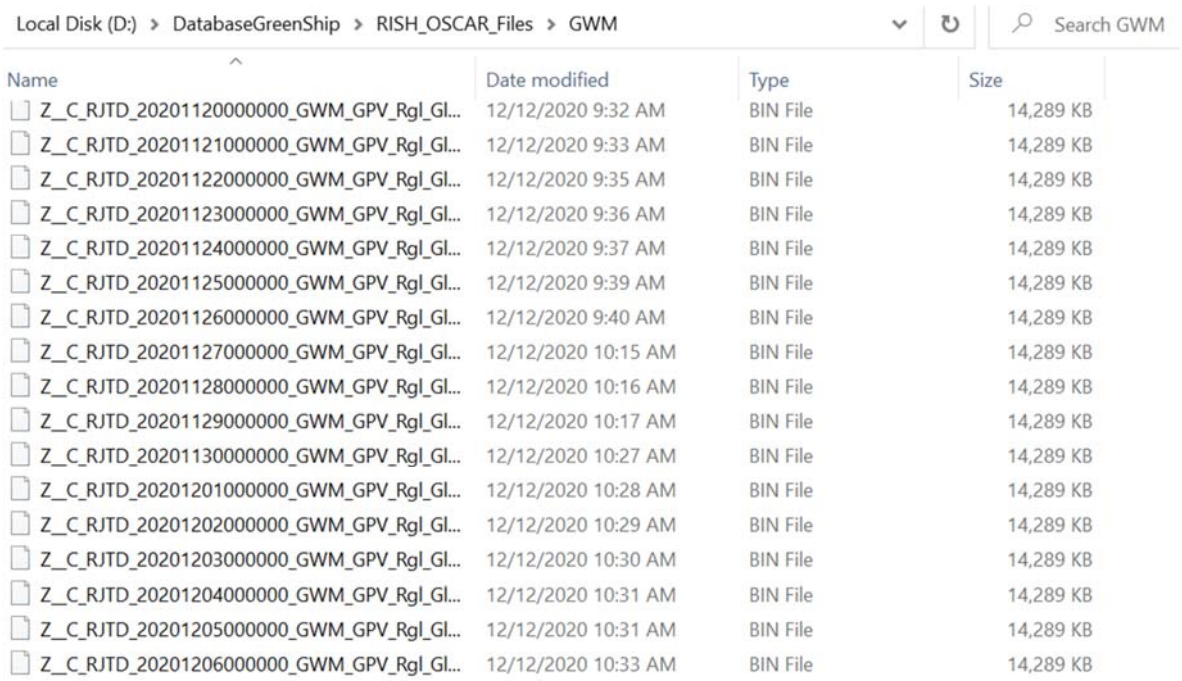
Hình 2.10 Dữ liệu thời tiết ngày 08/12/2020 được mã hóa bằng định dạng Grib 2

NCS Dowload các bản tin dự báo gió và sóng. Ví dụ: NCS dowload các file dự báo gió và sóng của Rish từ ngày 16/11/2020 đến ngày 06/12/2020.

Các file dự báo sóng, gió toàn cầu của Rish sau khi dowload được lưu trữ trong các thu mục GSM (Global Surface Model) và GWM (Global Wave Model).



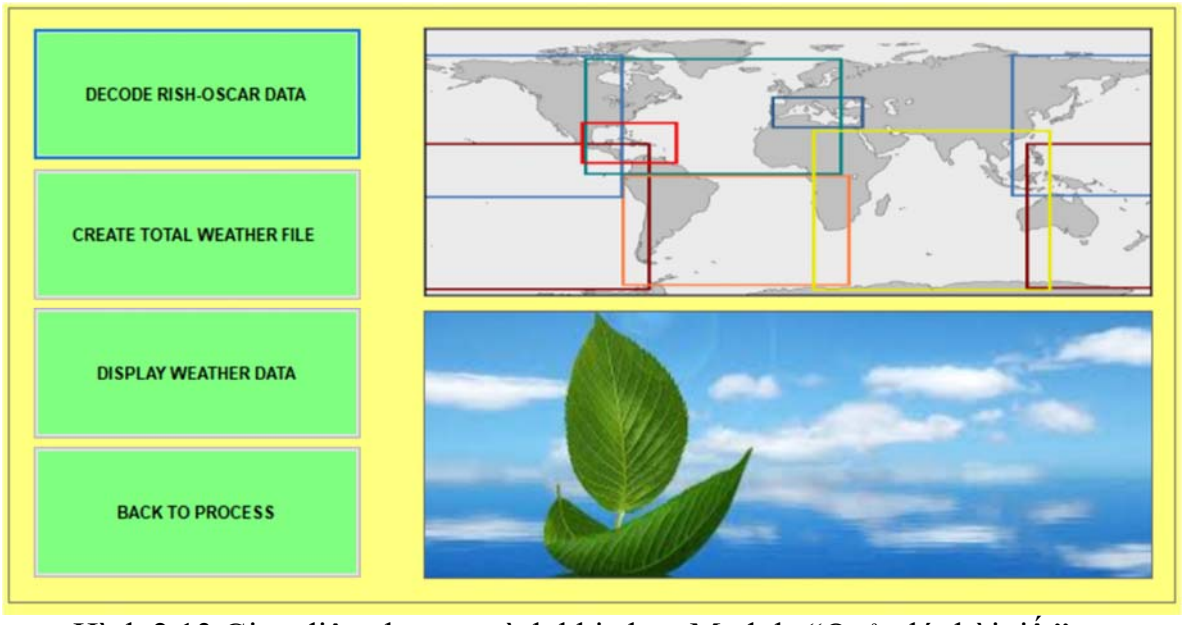
Hình 2.11 Dữ liệu dự báo gió toàn cầu của Rish từ ngày 16/11/2020 đến 06/12/2020



Hình 2.12 Dữ liệu dự báo sóng toàn cầu của Rish từ ngày 16/11/2020 đến 06/12/2020

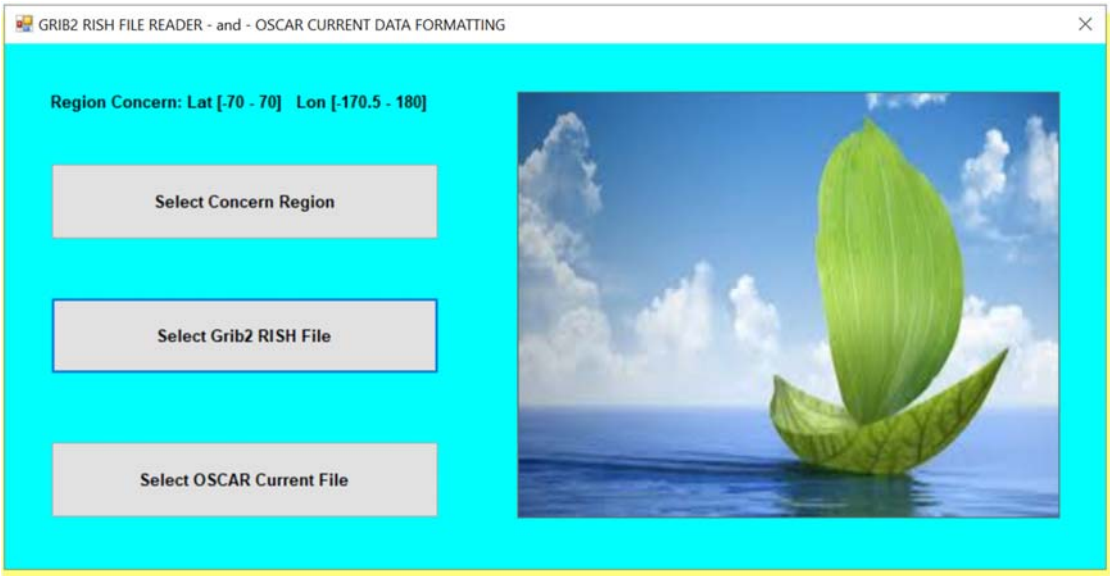
*** Bước 2. Giải mã dữ liệu thời tiết (sóng, gió) của Rish**

- Đăng nhập vào hệ thống: Nhập mã và Log in vào hệ thống
- Chọn Module "Quản lý thời tiết" 



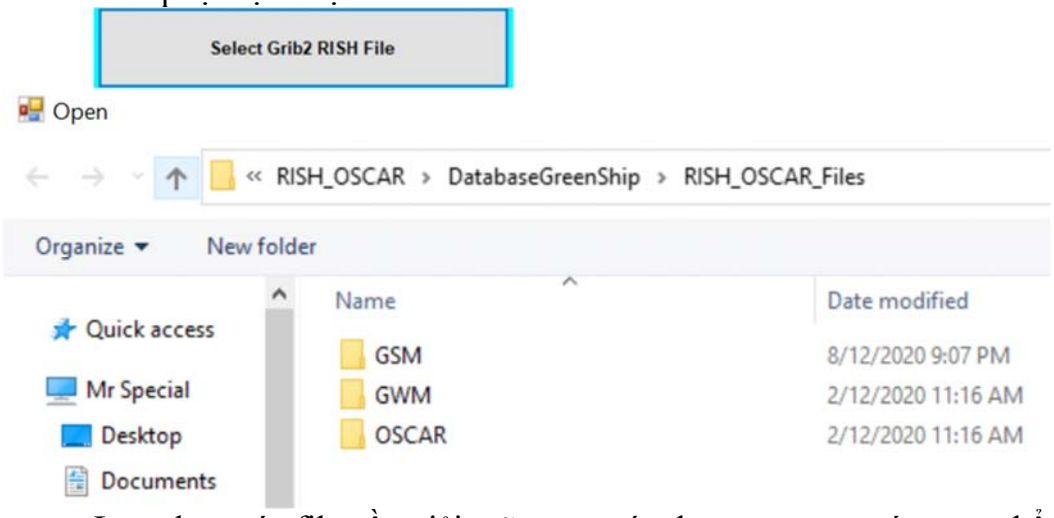
Hình 2.13 Giao diện chương trình khi chọn Module "Quản lý thời tiết"

- Chọn "DECODE RISH-OSCAR DATA" 

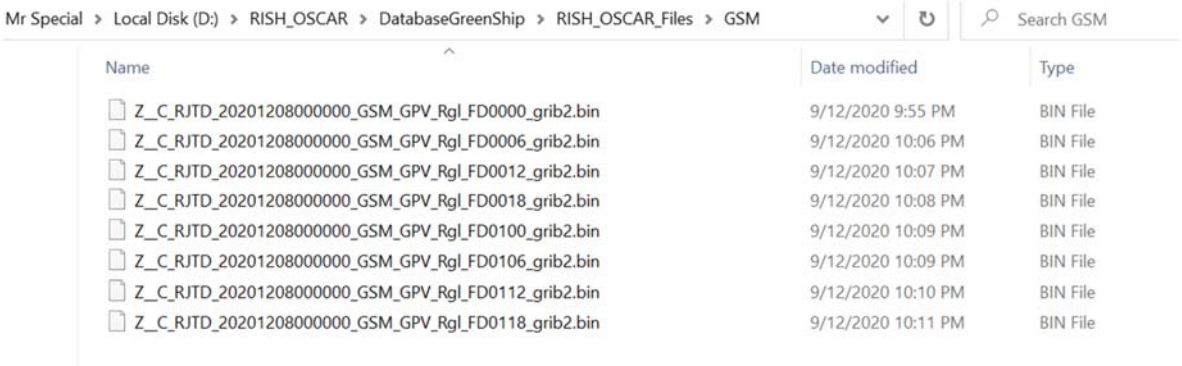


Hình 2.14 Giao diện hệ thống sau khi chọn "DECODE RISH-OSCAR DATA"

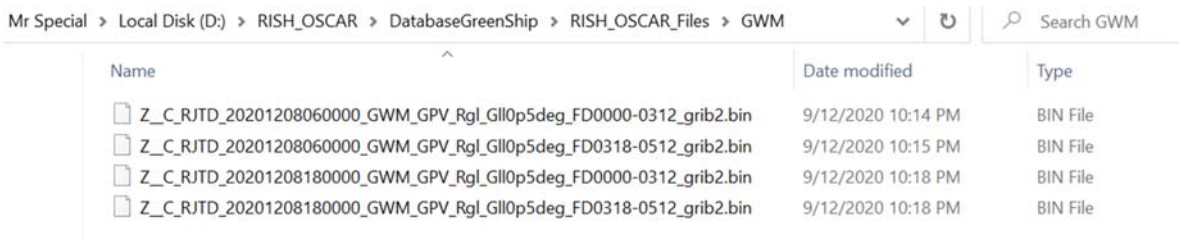
- Tiếp tục lựa chọn “Select Grib2 RISH File”



Lựa chọn các file cần giải mã trong các thư mục tương ứng, cụ thể: GSM (Global Surface Model) – Thư mục chứa các bản tin dự báo gió được lưu trữ dưới dạng Grib2; GWM (Global Wave Model) – Thư mục chứa các bản tin dự báo sóng được lưu trữ dưới dạng Grib2.

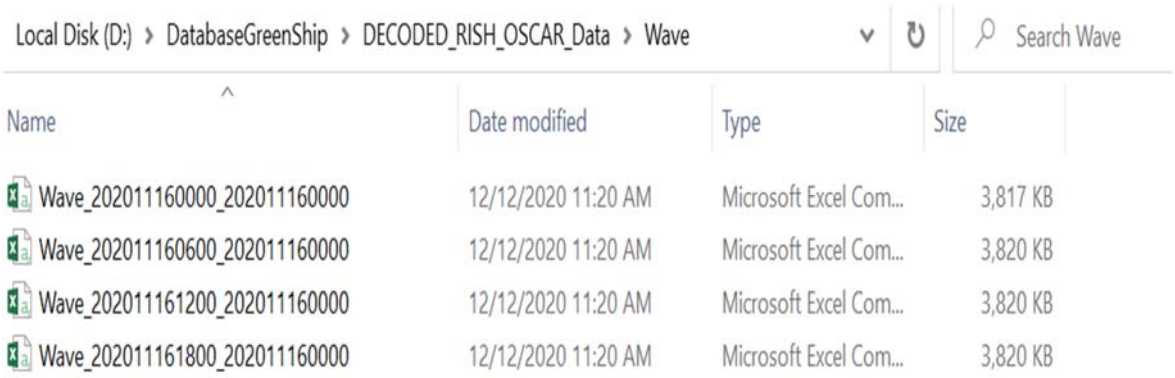


Hình 2.15 Thư mục GSM

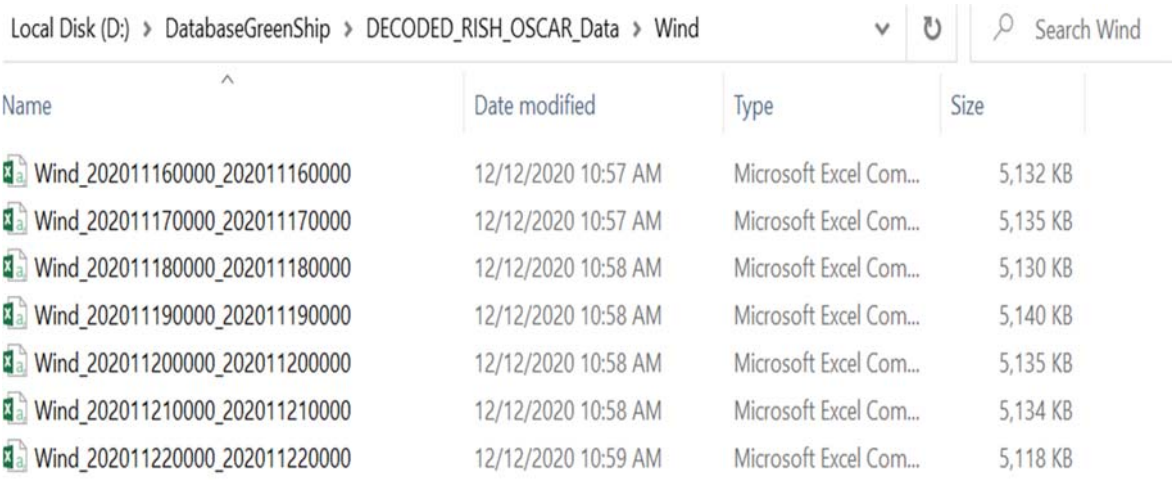


Hình 2.16 Thư mục GWM

- Kết quả thu được: Dữ liệu thời tiết sóng, gió của Rish sau khi giải mã sẽ được lưu trữ trong thư mục DECODED RISH_OSCAR_Data Wave/ Wind tương ứng.

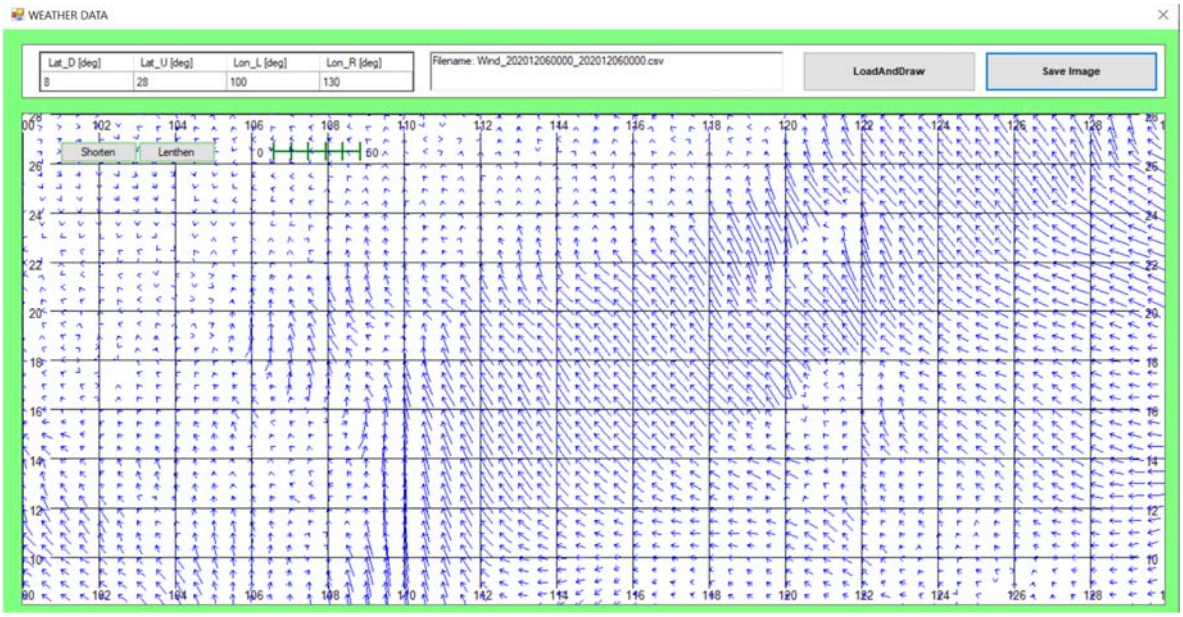


Hình 2.17 Kết quả giải mã bản tin dự báo sóng của Rish ngày 16/11/2020 (tại các thời điểm quan trắc 00, 06, 12 và 18 giờ)

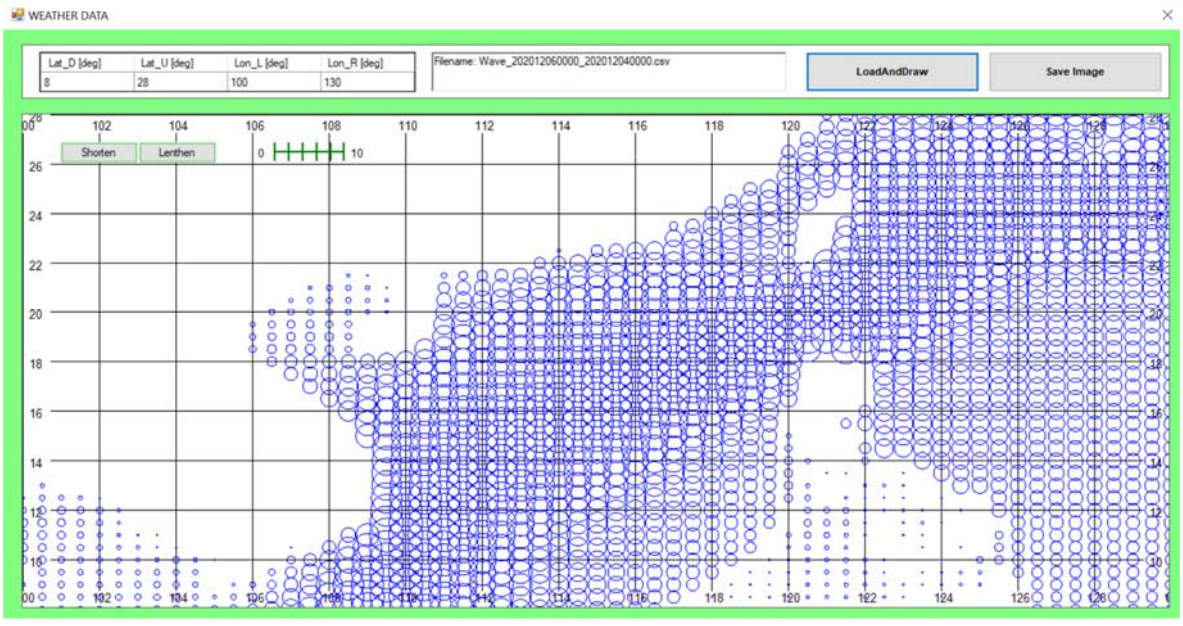


Hình 2.18 Kết quả giải mã bản tin dự báo gió của Rish từ ngày 16/11/2020 đến 22/11/2020

- Chọn Display Weather Data để hiển thị hình ảnh dữ liệu sóng, gió vừa giải mã 



Hình 2.19 Hiển thị hình ảnh gió toàn cầu ngày 06/12/2020



Hình 2.20 Hiển thị hình ánh sóng toàn cầu ngày 06/12/2020

NCS sử dụng dữ liệu sóng, gió của Rish sau khi giải mã kết hợp với dữ liệu dòng chảy Oscar (việc thu thập xử lí dữ liệu dòng chảy Oscar sẽ được trình bày

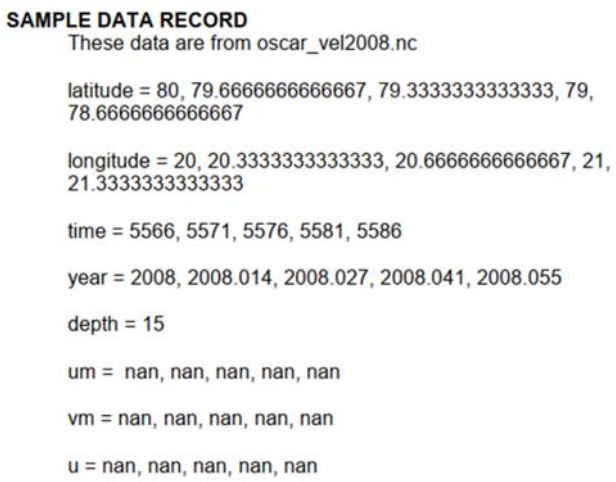
trong phần sau) để tổng hợp file dữ liệu thời tiết tổng hợp phục vụ tính toán tuyến đường và kế hoạch chạy tàu tối ưu.

2.4. Khai thác thông tin dòng chảy từ cơ sở dữ liệu dòng chảy Oscar phục vụ tính toán tuyến đường và kế hoạch chạy tàu tối ưu

2.4.1. Tổng quan về cơ sở dữ liệu dòng chảy OSCAR [16, 47, 75]

Oscar là chữ viết tắt tiếng Anh của Ocean Surface Analysis Real – Time, Dự án nghiên cứu, phân tích dòng chảy đại dương theo thời gian thực.

Tương tự Rish, Dự án nghiên cứu, phân tích dòng chảy theo thời gian thực Oscar xây dựng, cập nhật và duy trì một hệ thống cơ sở dữ liệu về dự báo và phân tích dòng chảy toàn cầu được lưu trữ dưới dạng định dạng netCDF (Format Network Common Data Form) nhằm phục vụ mục đích nghiên cứu khoa học (Nếu sử dụng cho mục đích khác thì phải trả phí).


SAMPLE DATA RECORD
These data are from oscar_vel2008.nc

latitude = 80, 79.6666666666667, 79.3333333333333, 79, 78.6666666666667

longitude = 20, 20.3333333333333, 20.6666666666667, 21, 21.3333333333333

time = 5566, 5571, 5576, 5581, 5586

year = 2008, 2008.014, 2008.027, 2008.041, 2008.055

depth = 15

um = nan, nan, nan, nan, nan

vm = nan, nan, nan, nan, nan

u = nan, nan, nan, nan, nan

Hình 2.21 Ví dụ dữ liệu dòng chảy Oscar

Dữ liệu dòng chảy Oscar gồm thông số tốc độ dòng chảy theo vĩ tuyến (u) và theo kinh tuyến (v).

Để tiếp cận dữ liệu dòng chảy toàn cầu của Oscar phục vụ mục đích Nghiên cứu khoa học, NCS truy cập địa chỉ website:

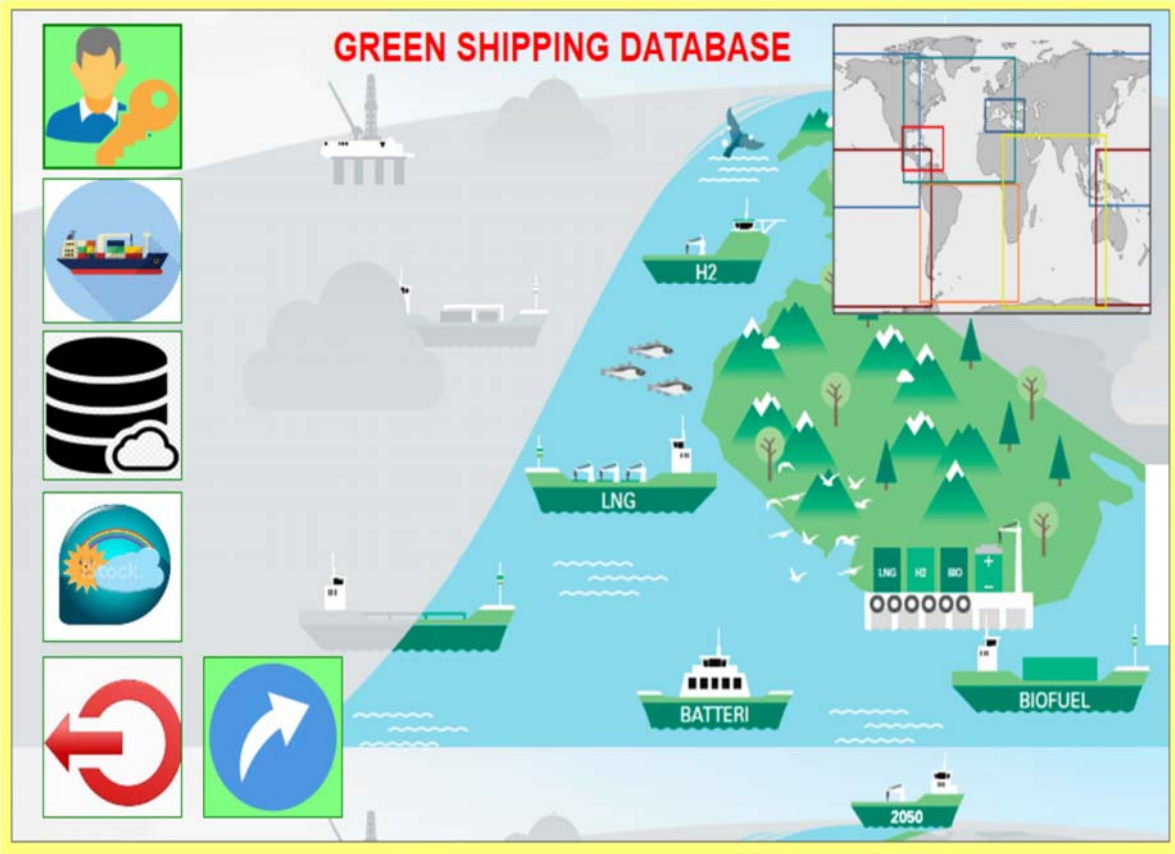
https://podaac-opendap.jpl.nasa.gov/opendap/hyrax/allData/oscar/preview/L4/oscar_third_deg/

Hình 2.22 Địa chỉ truy cập để lấy dữ liệu dòng chảy Oscar

2.4.2. Xây dựng phần mềm khai thác thông tin từ cơ sở dữ liệu dòng chảy OSCAR [16]

2.4.2.1. Phần mềm khai thác thông tin từ cơ sở dữ liệu dòng chảy OSCAR

Dữ liệu dòng chảy toàn cầu của Oscar được mã hóa bởi định dạng netCDF, để khai thác thông tin từ cơ sở dữ liệu dòng chảy Oscar (tương tự dữ liệu sóng, gió của Rish), NCS sử dụng ngôn ngữ lập trình Visual Basic 2010 (VB 2010) xây dựng phần mềm khai thác dữ liệu này phục vụ mục đích NCKH (tính toán kế hoạch chạy tàu tối ưu).



Hình 2.23 Giao diện chương trình quản lý cơ sở dữ liệu thời tiết

Đăng nhập hệ thống

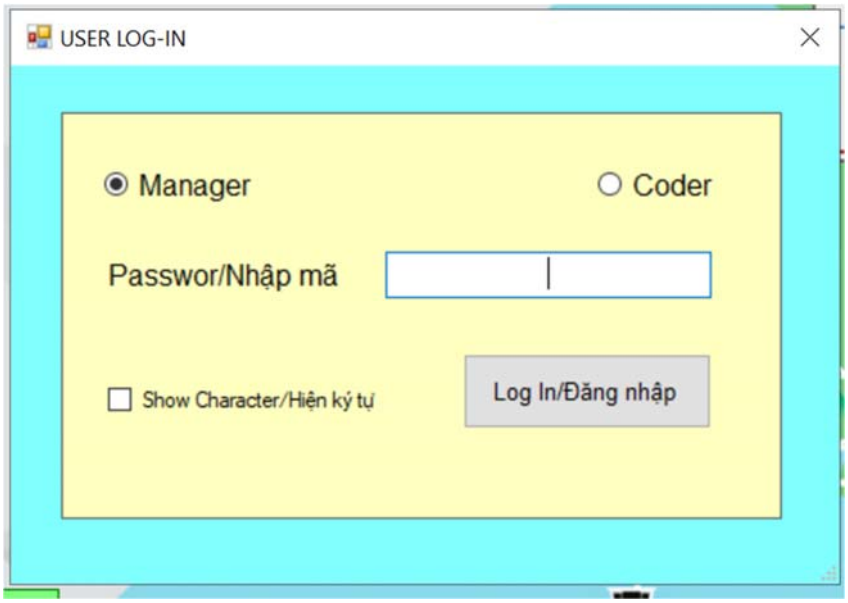
Module quản lý công ty và đội tàu

Module Upload dữ liệu thời tiết

Module quản lý thông tin thời tiết

Đăng xuất khỏi hệ thống

Đăng nhập vào hệ thống: Nhập mã và Log in vào hệ thống



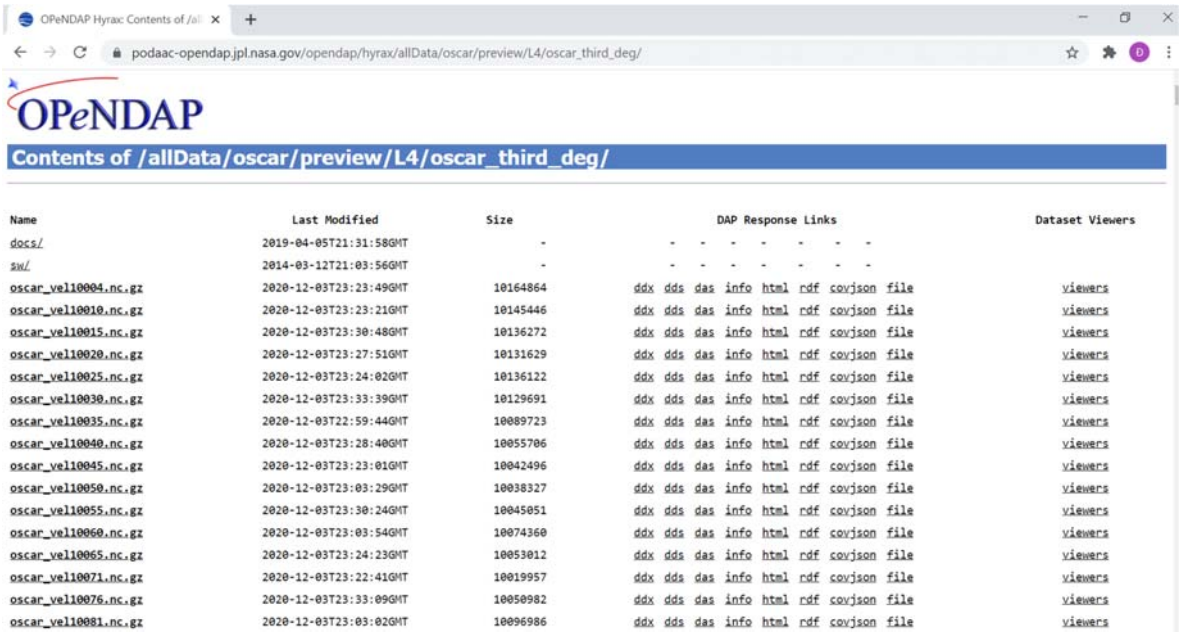
Hình 2.24 Giao diện đăng nhập vào hệ thống

2.4.2.2. Quy trình khai thác thông tin từ cơ sở dữ liệu dòng chảy OSCAR

Bước 1: Thu thập dữ liệu dòng chảy OSCAR

- Truy cập địa chỉ website:

https://podaac-opendap.jpl.nasa.gov/opendap/hyrax/allData/oscar/preview/L4/oscar_third_deg/



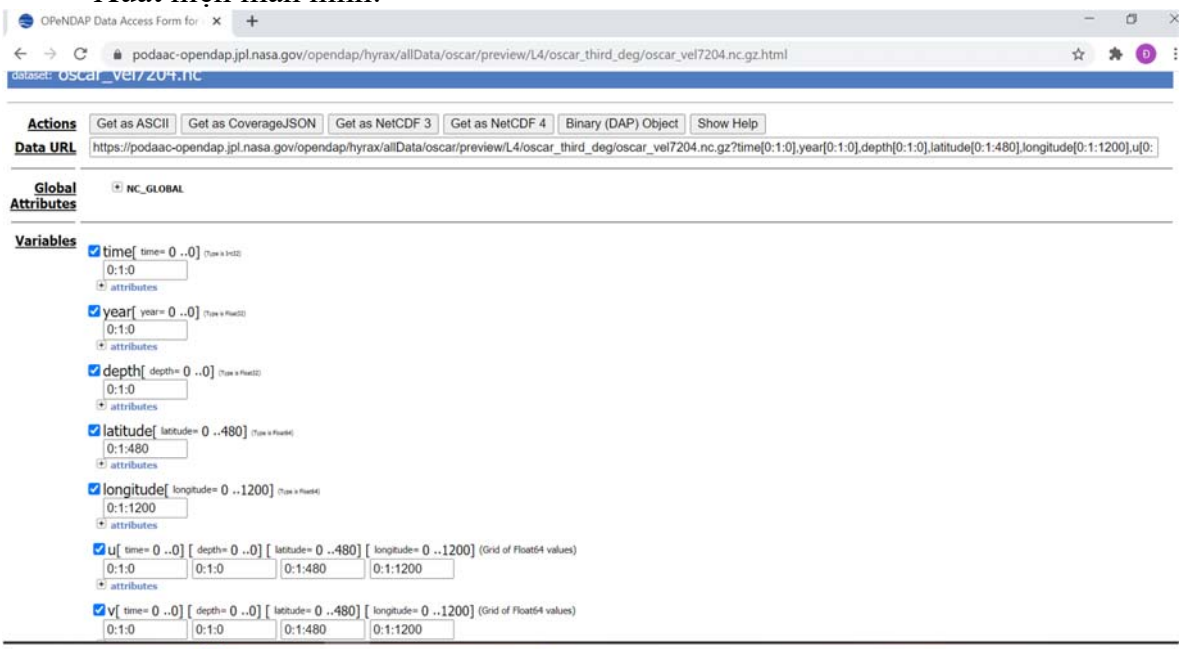
Hình 2.25 Dữ liệu dòng chảy OSCAR

Bước 2: Giải mã dữ liệu dòng chảy OSCAR

- Nhấn lựa chọn 1 file
Ví dụ:

oscar_vel7204.nc.gz 2018-12-05T02:26:32GMT 9854498 ddx dds das info html rdf covjson file viewers

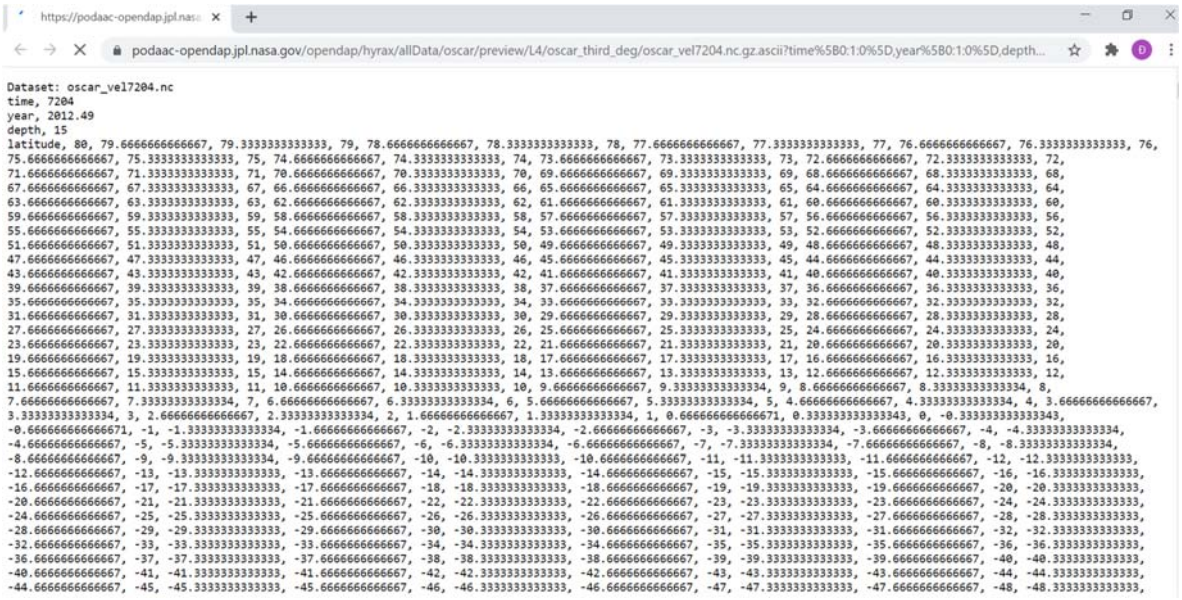
Xuất hiện màn hình:



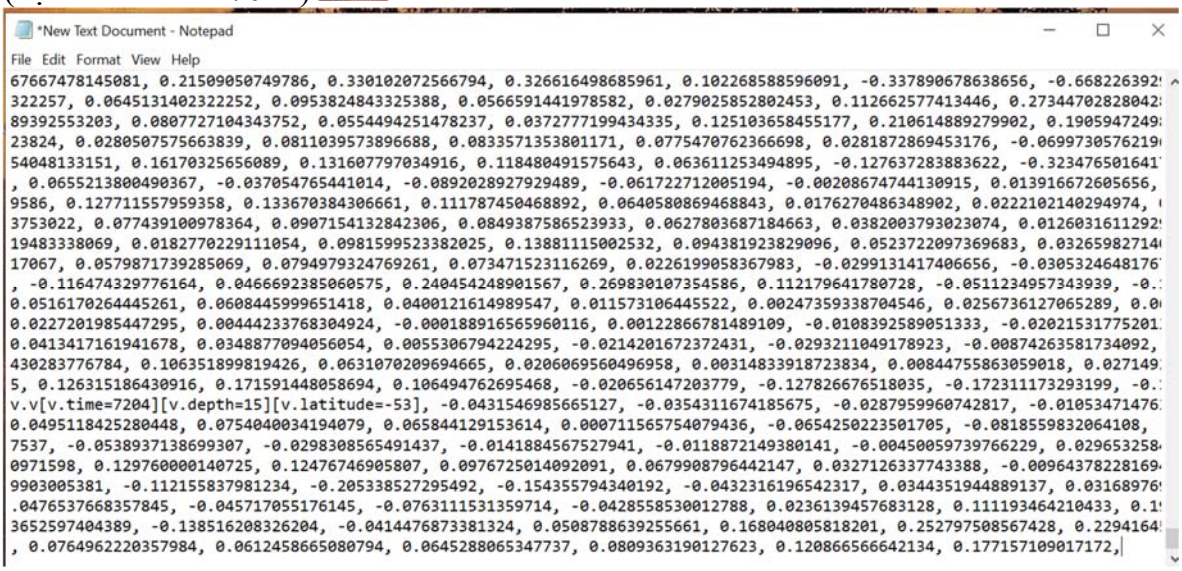
- Tích ☑ vào time, year, depth, latitude, longitude, u, v và chọn

Get as ASCII

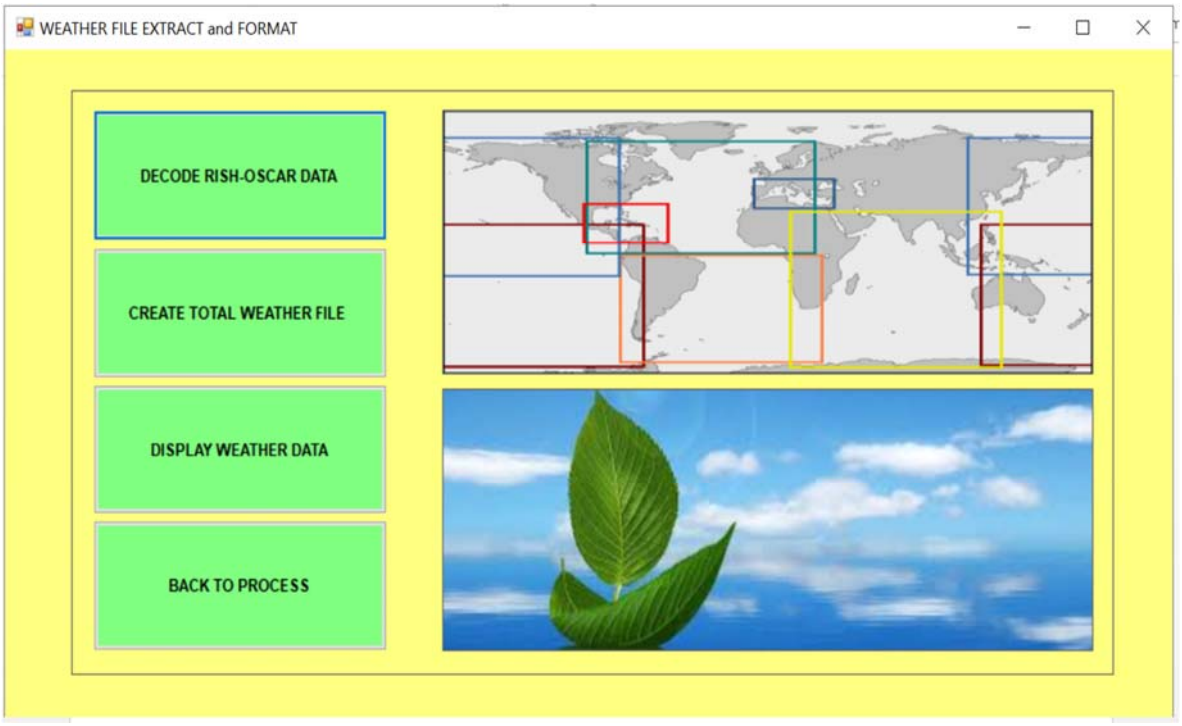
Xuất hiện màn hình:



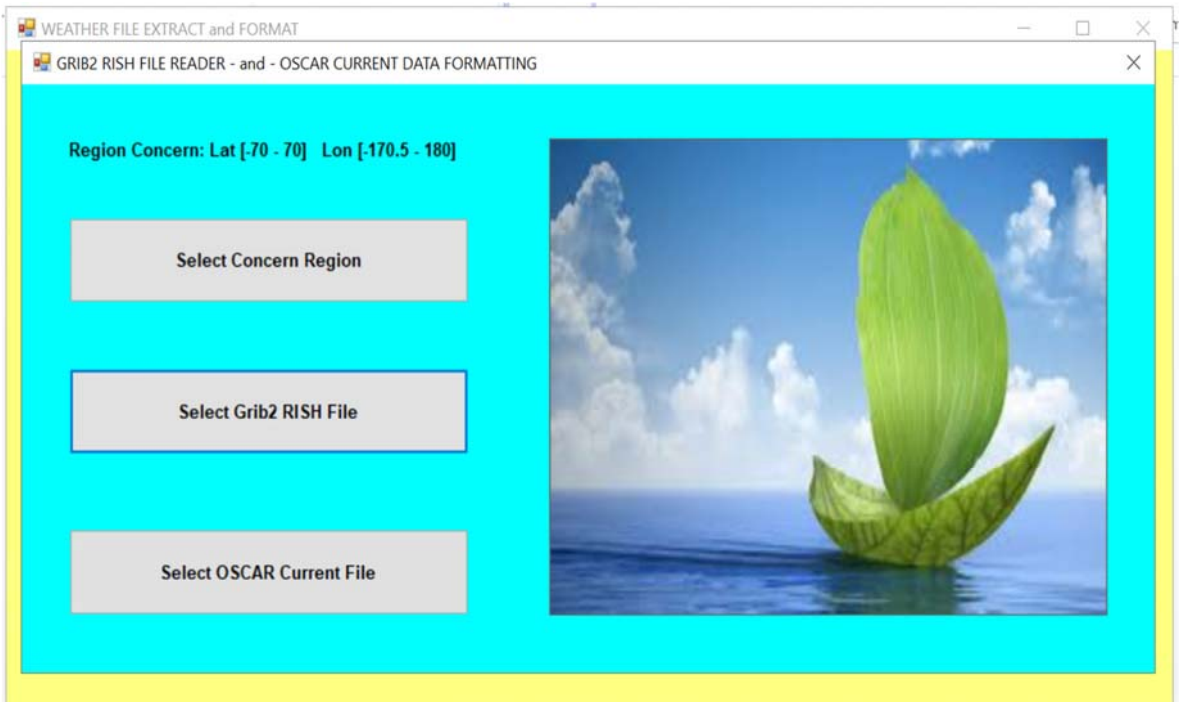
- Copy toàn bộ dữ liệu ở trang màn hình này và paste vào Text Document (đặt tên: oscar 17024)



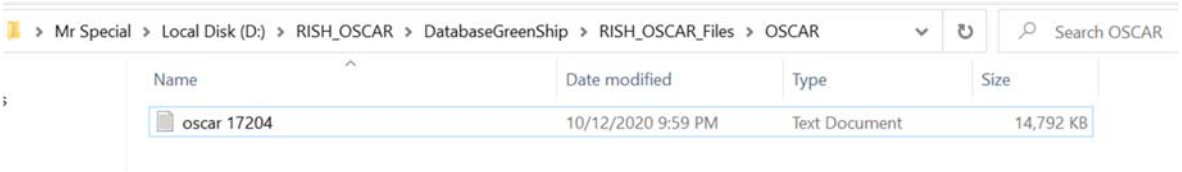
- Giải mã dữ liệu dòng chảy OSCAR “DECODE RISH_OSCAR DATA”



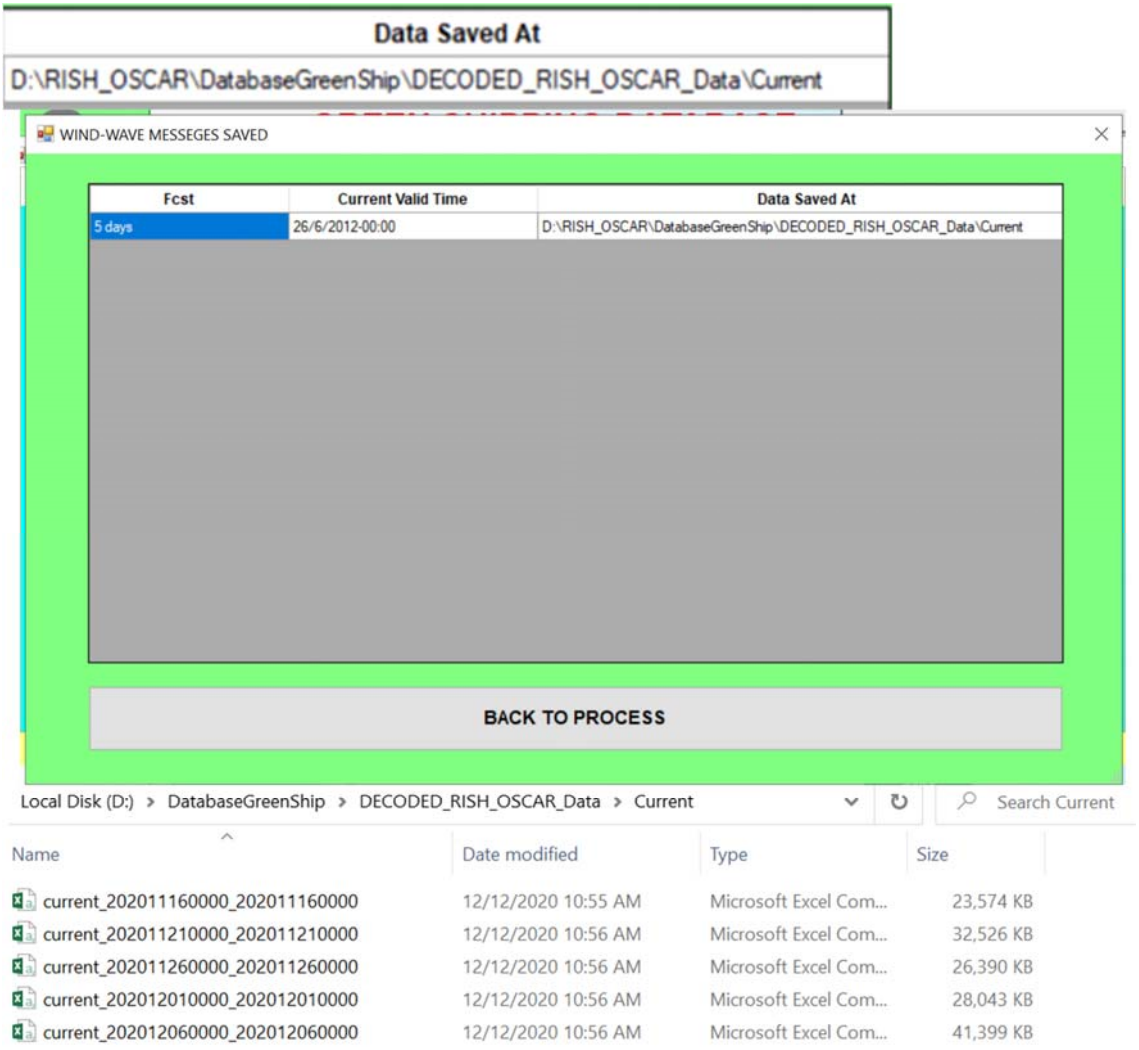
- Lựa chọn “Select Oscar Current File”



- Lựa chọn file: oscar 17024



- Kết quả dữ liệu giải mã được lưu trữ trong thư mục



Hình 2.26 Kết quả giải mã dữ liệu dòng chảy OSCAR từ ngày 16/11/2020 đến ngày 06/12/2020

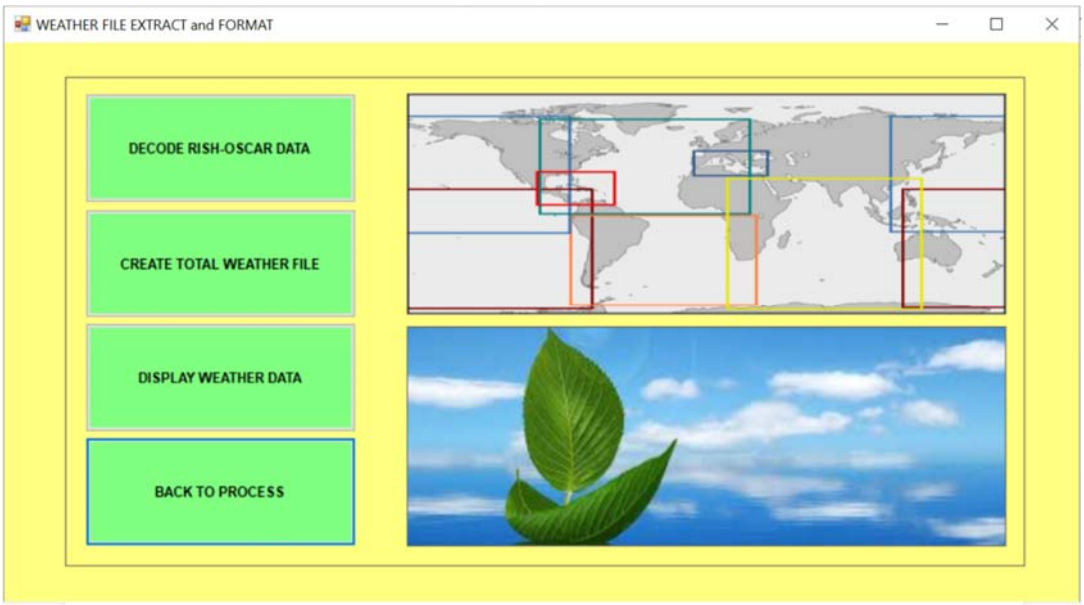
- Hiển thị dữ liệu vừa giải mã

Hình 2.27 Ví dụ hiển thị dữ liệu dòng chảy Oscar ngày 26/06/2020

2.5. Tạo và Upload file dữ liệu thời tiết tổng hợp phục vụ tính toán tuyến đường và kế hoạch chạy tàu tối ưu.

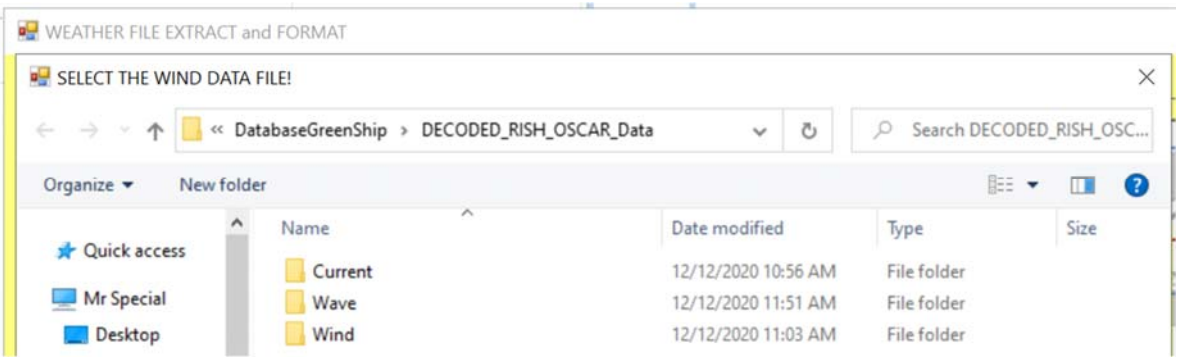
2.5.1. Tạo file dữ liệu thời tiết tổng hợp

- Lựa chọn CREATE TOTAL WEATHER FILE



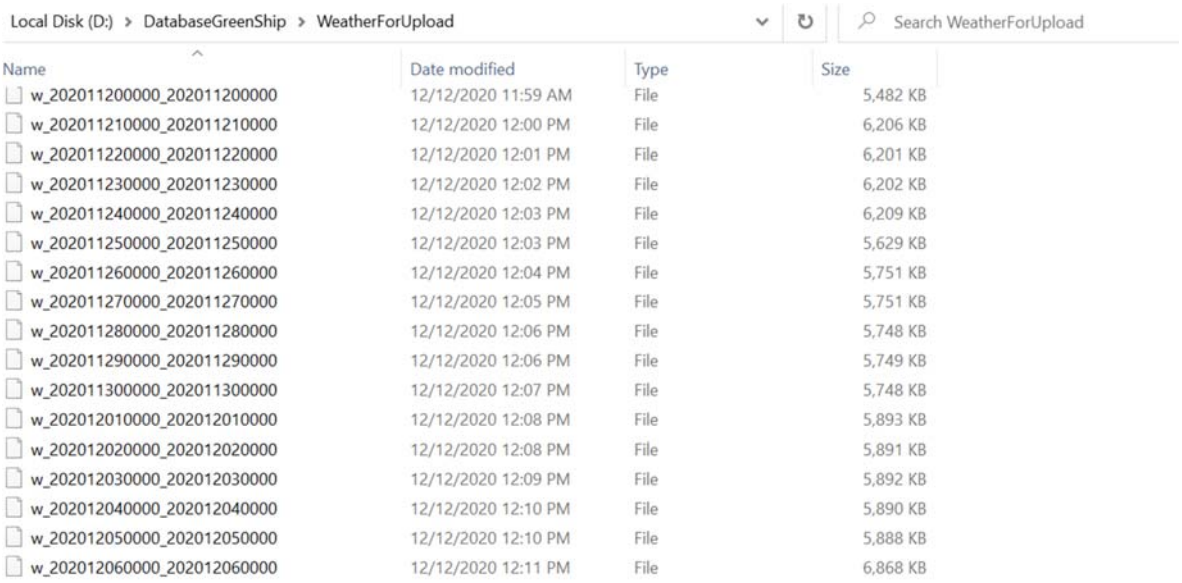
Lần lượt chọn các file dữ liệu gió, sóng và dòng chảy đã được giải mã trong các thư mục tương ứng: Wind – gió, Wave – sóng và Current – Dòng chảy. Lưu

ý lựa chọn các file dữ liệu gió, sóng, dòng chảy trong cùng ngày hoặc khoảng thời gian gần nhau.



Hình 2.28 Lựa chọn dữ liệu gió, sóng, dòng chảy đã được giải mã để tạo file thời tiết tổng hợp

- Các file thời tiết tổng hợp tạo ra sẽ được lưu trữ phục vụ cho bước Upload dữ liệu thời tiết tiếp theo



Hình 2.29 File thời tiết tổng hợp của từng ngày từ 20/11/2020 đến 06/12/2020

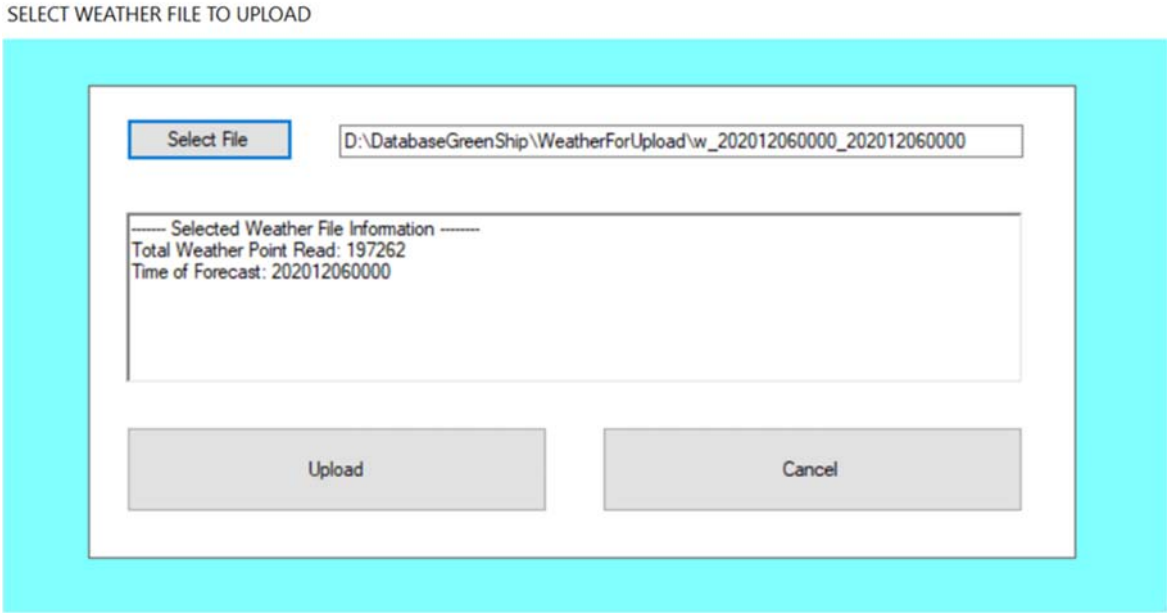
2.5.2. Upload file dữ liệu thời tiết tổng hợp

- Chọn Module 

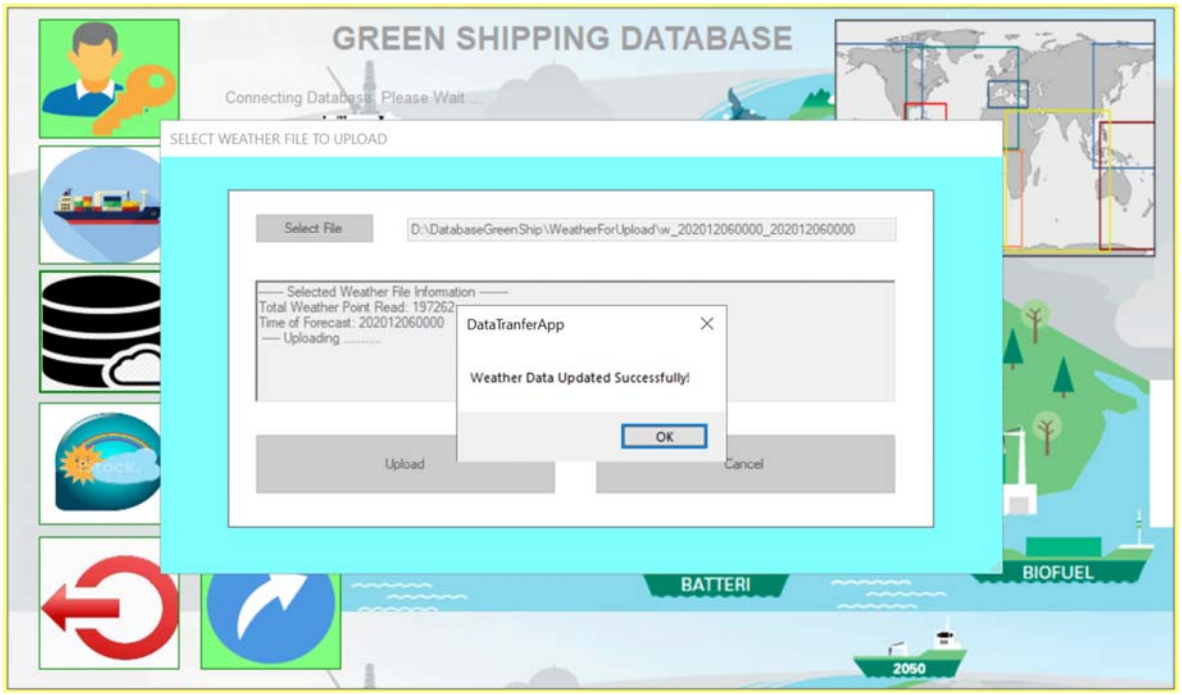
Hình 2.30 Giao diện chương trình khi lựa chọn chức năng Upload Weather Data

- Lựa chọn file để upload

Lần lượt lựa chọn các file thời tiết tổng hợp đã được tạo Hình 2.19 để Upload. Quá trình upload dữ liệu thời tiết sẽ mất thời gian và chương trình sẽ thông báo khi upload dữ liệu thời tiết thành công.



Hình 2.31 Lựa chọn file thời tiết tổng hợp ngày 06/12/2020 để upload



Hình 2.32 Dữ liệu thời tiết tổng hợp ngày 06/12/2020 được upload thành công

Thực hiện quy trình thu thập và xử lý dữ liệu thời tiết sóng, gió của Rish; dòng chảy của Oscar, ta sẽ có được nguồn dữ liệu thời tiết phục vụ tính toán kế hoạch chạy tàu tối ưu.

2.6. Kết luận chương 2

Mục tiêu nghiên cứu trọng tâm mà NCS đặt ra trong chương 2 là tổng hợp được thông tin thời tiết phục vụ tính toán tuyến đường và kế hoạch chạy tàu tối ưu.

Kết thúc chương 2, NCS đạt được các kết quả nghiên cứu quan trọng:

- Làm chủ được bản tin sóng toàn cầu, bản tin gió toàn cầu được mã hóa theo định dạng Grib 2 từ cơ sở dữ liệu của Viện nghiên cứu phát triển bền vững khí quyển nhân loại thuộc Đại học Kyoto, Nhật Bản (Rish – Research Institute for Sustainable Humanoshere);

- Làm chủ được dữ liệu dòng chảy toàn cầu được mã hóa theo định dạng netCDF (Format Network Common Data Form) từ cơ sở dã liệu của Dự án nghiên cứu, phân tích dòng chảy đại dương theo thời gian thực (Oscar – Ocean Surface Current Analysis Real Time) thuộc phòng thí nghiệm sức đẩy phản lực (Jet Propulsion Laboratory Physical Oceanography), Viện công nghệ California (Viện quản lý các dự án của cơ quan hàng không vũ trụ Mỹ);

Trên cơ sở đó tạo được các file dữ liệu thời tiết tổng hợp phục vụ tính toán tuyến đường và kế hoạch chạy tàu tối ưu nhiên liệu.

CHƯƠNG 3: TỔNG HỢP, PHÂN TÍCH ĐẶC TÍNH THAY ĐỔI TỐC ĐỘ VÀ ĐẶC TÍNH TIÊU THỤ NHIÊN LIỆU CỦA TÀU BIỂN TRONG TỪNG ĐIỀU KIỆN HÀNH HẢI CỤ THỂ BẰNG PHƯƠNG PHÁP BÌNH PHƯƠNG NHỎ NHẤT PHỤC VỤ TÍNH TOÁN TUYẾN ĐƯỜNG VÀ KẾ HOẠCH CHẠY TÀU TỐI ƯU

Trong chương này, NCS nghiên cứu đặc tính thay đổi tốc độ và đặc tính tiêu thụ nhiên liệu của tàu biển trong từng điều kiện hành hải cụ thể.

NCS xem xét điều kiện hành hải cụ thể ở đây chính là điều kiện sóng, gió, dòng chảy tương ứng với rpm, draft, trim nhất định ảnh hưởng tới tốc độ và mức tiêu hao nhiên liệu của tàu biển, trong đó:

- Với yếu tố sóng gồm: Hướng sóng và độ cao sóng (m);
- Với yếu tố gió gồm: Tốc độ gió (Knots) và hướng gió; và
- Với yếu tố dòng chảy: Hướng dòng;
- RPM _ Revolutions Per Minute (Vòng/phút) chính là số vòng quay chân vịt (hay chế độ máy);
- Draft: Mớn nước tàu (m);
- Trim: Hiệu số mớn nước của tàu (m).

Đồng thời, NCS ứng dụng phương pháp bình phương nhỏ nhất để xác định đặc tính thay đổi tốc độ và đặc tính tiêu thụ nhiên liệu tàu biển trong từng điều kiện hành hải cụ thể.

Ngoài ra, NCS xây dựng phần mềm tổng hợp, phân tích đặc tính thay đổi tốc độ và đặc tính tiêu thụ nhiên liệu tàu biển trong từng điều kiện hành hải cụ thể phục vụ tính toán tuyến đường và kế hoạch chạy tàu tối ưu.

3.1. Đặc tính thay đổi tốc độ tàu biển trong từng điều kiện hành hải cụ thể [17]

Đặc tính thay đổi tốc độ tàu biển trong từng điều kiện hành hải cụ thể thể nói lên khả năng "thích ứng" của tàu biển khi có ảnh hưởng của các yếu tố thời tiết ứng với rpm, draft và trim cụ thể của tàu biển.

Bằng cách liệt kê, ghi lại kết quả (ghi chép dữ liệu đồng nhất dựa trên các thông tin được lấy từ các Nhật ký tàu như nhật ký buồng lái, nhật ký boong _ máy, nhật ký buồng máy, … và các bản ghi chính thức khác trên tàu), NCS xây dựng bộ cơ sở dữ liệu đặc tính thay đổi tốc độ tàu biển trong từng điều kiện hành hải cụ thể.

Dữ liệu ghi lại được lập thành bảng có dạng như sau:

Bảng 3.1 Mẫu bảng ghi lại dữ liệu đặc tính thay đổi tốc độ tàu biển trong từng điều kiện hành hải cụ thể

No/STT	Date/Ngày	Hour/Giờ	Wind/Gió		Wave/Sóng		Current	Trim	Draft	RPM	SPD (kts)
			(deg/kts)		*(deg/m)*		*(deg)*	*(m)*	*(m)*	*(V/P)*	*Tốc độ*
1											
2											
3											

Trên thực tế, NCS bỏ qua ảnh hưởng của sóng và dòng chảy, chỉ xét ảnh hưởng của yếu tố gió (với hướng gió và tốc độ gió thay đổi), ví dụ: gió ngược, gió vát (Hướng gió tạo với hướng tàu 1 góc 15^0, 30^0, 45^0, 60^0), gió ngang và gió xuôi.

Với cách làm như trên, NCS hướng tới mục tiêu áp dụng được cho bất kỳ một con tàu của một công ty nào.

3.2. Đặc tính tiêu thụ nhiên liệu của tàu biển trong từng điều kiện hành hải cụ thể [7]

Mức tiêu thụ nhiên liệu FC (Fuel Consumption) được xác định bằng lượng nhiên liệu máy chính và các máy phụ bao gồm cả nồi hơi và các lò đốt rác trên biển và trong cảng cho 1 chuyến đi hoặc một khoảng thời gian xem xét (ví dụ một ngày).

Mức tiêu thụ nhiên liệu tàu biển là một đại lượng phụ thuộc vào các yếu tố hướng gió, tốc độ gió, hướng sóng, độ cao sóng, trim, draft và rpm.

Để xác định được mức tiêu thụ nhiên liệu tàu biển trong một khoảng thời gian xem xét (ví dụ trong một chuyến đi) thì phương pháp tối ưu truyền thống thường xuyên được sử dụng trên tàu biển là phương pháp ghi dữ liệu đồng nhất.

Phương pháp thu thập dữ liệu: Theo NCS đánh giá phương pháp tối ưu nhất được sử dụng ở đây chính là sử dụng bảng biểu, liệt kê, ghi chép dữ liệu đồng

nhất dựa trên các thông tin được lấy từ các Nhật ký tàu như nhật ký buồng lái, nhật ký boong _ máy, nhật ký buồng máy, nhật ký nhận dầu, … và các bản ghi chính thức khác trên tàu.

Dữ liệu ghi lại được lập thành bảng có dạng như sau:

Bảng 3.2 Mẫu bảng ghi lại dữ liệu đặc tính tiêu thụ nhiên liệu của tàu biển trong từng điều kiện hành hải cụ thể

No	Date/ Ngày … Giờ..	Tình trạng hoạt động	Fuel [t] (HFO)	Fuel (t) (Diesel/ Gas Oil)	Khối lượng hàng chuyển chở	Khoảng cách hành trình
1						
2						
3						
4						
5						
…						

Với cách làm như trên, NCS hướng tới mục tiêu áp dụng được cho bất kỳ một con tàu nào.

3.3. Xác định đặc tính thay đổi tốc độ và tiêu thụ nhiên liệu của tàu biển trong từng điều kiện hành hải cụ thể bằng phương pháp bình phương nhỏ nhất

3.3.1. Phương pháp bình phương nhỏ nhất [9, 14, 37, 38]

Phương pháp bình phương nhỏ nhất là phương pháp được sử dụng rộng rãi nhất để xác định các thông số của một mô hình hay các giá trị chưa biết khác dựa trên một tập hợp các kết quả quan trắc hoặc đo đạc.

Mục tiêu của phương pháp này là tính một bộ giá trị gần đúng cho các tham số (hệ số) của một hàm sao cho giá trị hàm phù hợp với các giá trị quan trắc nhất.

Giả sử có hàm f(x) với bộ các hệ số θ (cần xác định), hay có thể viết:

$y = f(\theta, x)$ với $\theta = [\theta_1, \theta_2, \dots., \theta_m]$ (3.1)

Bằng các quan trắc, ta xác định được một tập hợp các giá trị đầu ra của hàm ứng với các giá trị đầu vào tương ứng là:

$(y_1, x_1), (y_2, x_2), (y_3, x_3), \dots, (y_n, x_n)$ với $n \gg m$ (3.2)

Đây là bài toán với số phương trình nhiều hơn số ẩn. Tuy nhiên, trong các phép đo luôn tồn tại sai số nên không thể xác định được bộ hệ số θ duy nhất.

Ứng với bộ hệ số (tham số) θ, ta có:

$$\begin{cases} y_1 - f(\theta_1, x_1) = r_1 \\ y_2 - f(\theta_2, x_2) = r_2 \\ \cdots \\ y_n - f(\theta_n, x_n) = r_n \end{cases} \quad (3.3)$$

Với $R = (r_1, r_2, r_3, \cdots, r_n)$ là các sai số dư hay là các sai lệch của hàm $f(\theta, x)$ so với giá trị quan sát.

Ta đặt: $S(\theta) = \sum_{i=1}^{n} r_i^2$ (3.4)

(4) là tổng bình phương các sai lệch. Nhiệm vụ đặt ra trong bài toán bình phương nhỏ nhất là xác định vec-tơ θ^* sao cho tổng này nhỏ nhất.

$S(\theta) = minS(\theta)$ hay $\theta^* = argmin_\theta S(\theta^*)$ (3.5)

Minh họa ứng dụng của một bài toán bình phương nhỏ nhất trong việc xác định vị trí tàu bằng cách đo phương vị tới các mục tiêu bờ: Khi thực hiện đo phương vị tới nhiều hơn 2 mục tiêu và thao tác trên hải đồ, ta được các đường thẳng (đường phương vị) trên hải đồ. Tuy nhiên, khi có sai số tác động, các đường vị trí không đồng quy.

Khi đó, vị trí tin cậy nhất của tàu là vị trí phù hợp nhất (best fit) với các đường vị trí, hay là vị trí ứng với tổng bình phương khoảng cách còn lại đến các đường vị trí là nhỏ nhất. Bài toán bình phương nhỏ nhất giúp ta tính được vị trí này.

3.3.2. Ứng dụng phương pháp bình phương nhỏ nhất xác định đặc tính tốc độ và đặc tính tiêu thụ nhiên liệu của tàu biển trong từng điều kiện hành hải cụ thể [17]

3.3.2.1. Ứng dụng phương pháp bình phương nhỏ nhất xác định đặc tính tốc độ tàu biển trong từng điều kiện hành hải cụ thể

Bằng phương pháp ghi dữ liệu thống nhất, NCS ghi lại các kết quả thể hiện mối liên hệ tốc độ tàu (V) và các yếu tố hướng sóng, độ cao sóng, hướng gió, tốc độ gió, draft, trim và rpm theo bảng sau:

Bảng 3.3 Mẫu bảng ghi lại tốc độ tàu trong từng điều kiện hành hải cụ thể

STT	Ngày	Giờ	Gió		Sóng		Trim	Draft	RPM	Tốc độ tàu V
			Deg	Kts	Deg	m	m	m	Vòng/phút	
1	Dec 1st	12.00	Wi_{d1}	Wi_{s1}	W_{d1}	W_{h1}	T_1	D_1	Rpm_1	V_1
2	Dec2 2d	06.00	Wi_{d2}	Wi_{s2}	W_{d2}	W_{h2}	T_2	D_2	Rpm_2	V_2
…	…	…	…	…	…	…	…	…	…	…
n	…	…	Wi_{dn}	Wi_{sn}	W_{dn}	W_{hn}	T_n	D_n	Rpm_n	V_n

Từ bảng trên, ta thấy mối liên hệ hàm số giữa tốc độ tàu (V) và các yếu tố hướng sóng, độ cao sóng, hướng gió, tốc độ gió, draft, trim và rpm được thể hiện như sau:

$$\begin{cases} V_1 = a_1D_1 + a_1T_1 + a_1rpm_1 + a_1w_{id1} + a_1w_{is1} + a_1w_{d1} + a_1w_{h1} \\ V_2 = a_2D_2 + a_2T_2 + a_2rpm_2 + a_2w_{id2} + a_2w_{is2} + a_2w_{d2} + a_2w_{h2} \\ \cdots \\ V_n = a_nD_n + a_nT_n + a_nrpm_n + a_nw_{idn} + a_nw_{isn} + a_nw_{dn} + a_nw_{hn} \end{cases} \quad (3.6)$$

Trong đó:

D_i: Mớn nước tàu tại thời điểm quan trắc thứ i;

T_i: Độ chúi của tàu tại thời điểm quan trắc thứ i;

rpm_i: Số vòng quay chân vịt tại thời điểm quan trắc thứ i;

W_{idi}: Hướng gió tại tại thời điểm quan trắc thứ i;

W_{isi}: Tốc độ gió tại thời điểm quan trắc thư i;

W_{di}: Hướng sóng tại thời điểm quan trắc thứ i;

W_{hi}: Độ cao sóng tại thời điểm quan trắc thư i;

a_i: Hệ số tại thời điểm quan trắc thứ i (i = 1 ÷ n).

Thực tế, tại mỗi thời điểm quan trắc thứ i luôn tồn tại một lượng sai số d_i, tức là:

$$\begin{cases} V_1^* = a_1D_1 + a_1T_1 + a_1rpm_1 + a_1w_{id1} + a_1w_{is1} + a_1w_{d1} + a_1w_{h1} + d_1 \\ V_2^* = a_2D_2 + a_2T_2 + a_2rpm_2 + a_2w_{id2} + a_2w_{is2} + a_2w_{d2} + a_2w_{h2} + d_2 \\ \cdots \\ V_n^* = a_nD_n + a_nT_n + a_nrpm_n + a_nw_{idn} + a_nw_{isn} + a_nw_{dn} + a_nw_{hn} + d_n \end{cases} \quad (3.7)$$

Xét hàm số:

$$f = (V_1 - V_1^*)^2 + (V_2 - V_2^*)^2 + \cdots + (V_n - V_n^*)^2 \quad (3.8)$$

Mục tiêu của thuật toán bình phương nhỏ nhất nhằm xác định các hệ số a_i sao cho tổng bình phương của các sai số nói trên là bé nhất, tức là:

$$\begin{cases} f'(a_1) = 0 \\ f'(a_2) = 0 \\ \cdots \\ f'(a_n) = 0 \end{cases} \quad (3.9)$$

Giải hệ phương trình (3.9) ta tìm được bộ hệ số a_i (i = 1÷n), trong đó n là số lần quan trắc thực tế.

3.3.2.2. Ứng dụng phương pháp bình phương nhỏ nhất xác định đặc tính tiêu thụ nhiên liệu tàu biển trong từng điều kiện hành hải cụ thể

Thông qua phương pháp ghi dữ liệu đồng nhất, NCS ghi lại kết quả thể hiện mối liên hệ giữa mức tiêu thụ nhiên liệu (FC) và các yếu tố hướng sóng, độ cao sóng, hướng gió, tốc độ gió, draft, trim và rpm theo bảng sau:

Bảng 3.4 Mẫu bảng ghi lại kết mức tiêu thụ nhiên liệu của tàu biển trong từng điều kiện hành hải cụ thể

STT	Ngày	Giờ	Gió		Sóng		Trim	Draft	RPM	
			Deg	Kts	Deg	m	m	m	Vòng/phút	FC
1	Dec 1st	12.00	Wi_{d1}	Wi_{s1}	W_{d1}	W_{h1}	T_1	D_1	Rpm_1	FC_1
2	Dec2^{2d}	06.00	Wi_{d2}	Wi_{s2}	W_{d2}	W_{h2}	T_2	D_2	Rpm_2	FC_2
…	…	…	…	…	…	…	…	…	…	…
n	…	…	Wi_{dn}	Wi_{sn}	W_{dn}	W_{hh}	T_n	D_n	Rpm_n	FC_n

Từ bảng trên, ta thấy mối liên hệ hàm số giữa mức tiêu thụ nhiên liệu của tàu (FC) và các yếu tố hướng sóng, độ cao sóng, hướng gió, tốc độ gió, draft, trim và rpm được thể hiện như sau:

$$\begin{cases} FC_1 = a_1D_1 + a_1T_1 + a_1rpm_1 + a_1w_{id1} + a_1w_{is1} + a_1w_{d1} + a_1w_{h1} \\ FC_2 = a_2D_2 + a_2T_2 + a_2rpm_2 + a_2w_{id2} + a_2w_{is2} + a_2w_{d2} + a_2w_{h2} \\ \cdots \\ FC_n = a_nD_n + a_nT_n + a_nrpm_n + a_nw_{idn} + a_nw_{isn} + a_nw_{dn} + a_nw_{hn} \end{cases} \quad (3.10)$$

Trong đó:

D_i: Mớn nước tàu tại thời điểm quan trắc thứ i;

T_1: Độ chúi của tàu tại thời điểm quan trắc thứ i;

rpm_i: Số vòng quay chân vịt tại thời điểm quan trắc thứ i;

W_{idi}: Hướng gió tại tại thời điểm quan trắc thứ i;

W_{isi}: Tốc độ gió tại thời điểm quan trắc thư i;

$W_{di:}$ Hướng sóng tại thời điểm quan trắc thứ i;

W_{hi}: Độ cao sóng tại thời điểm quan trắc thư i;

a_i: Hệ số tại thời điểm quan trắc thứ i (i = 1 ÷ n).

Thực tế, tại mỗi thời điểm quan trắc thứ i luôn tồn tại một lượng sai số r_i, tức là:

$$\begin{cases} FC_1^* = a_1D_1 + a_1T_1 + a_1rpm_1 + a_1w_{id1} + a_1w_{is1} + a_1w_{d1} + a_1w_{h1} + r_1 \\ FC_2^* = a_2D_2 + a_2T_2 + a_2rpm_2 + a_2w_{id2} + a_2w_{is2} + a_2w_{d2} + a_2w_{h2} + r_2 \\ \cdots \\ FC_n^* = a_nD_n + a_nT_n + a_nrpm_n + a_nw_{idn} + a_nw_{isn} + a_nw_{dn} + a_nw_{hn} + r_n \end{cases} \quad (3.11)$$

Xét hàm số:

$$f = (FC_1 - FC_1^*)^2 + (FC_2 - FC_2^*)^2 + \cdots + (FC_n - FC_n^*)^2 \quad (3.12)$$

Mục tiêu của thuật toán bình phương nhỏ nhất nhằm xác định các hệ số a_i sao cho tổng bình phương của các sai số nói trên là bé nhất, tức là:

$$\begin{cases} f'(a_1) = 0 \\ f'(a_2) = 0 \\ \cdots \\ f'(a_n) = 0 \end{cases} \quad (3.13)$$

Giải hệ phương trình (3.13) ta tìm được bộ hệ số a_i (i = 1÷n), trong đó n là số lần quan trắc thực tế.

3.4. Phần mềm và mô hình tổng hợp, phân tích đặc tính thay đổi tốc độ và đặc tính tiêu thụ nhiên liệu của tàu biển trong từng điều kiện hành hải cụ thể phục vụ tính toán tuyến đường và kế hoạch chạy tàu tối ưu [17]

3.4.1. Tổng quan về phần mềm

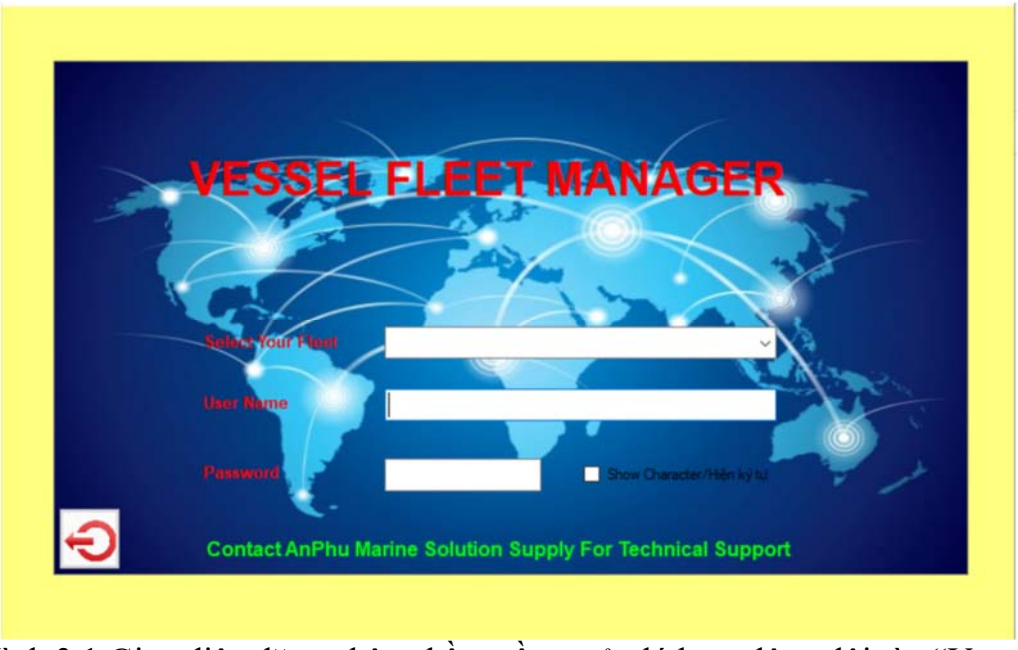
Trong quá trình hành hải, việc theo dõi, đánh giá đặc tính thay đổi tốc độ và đặc tính tiêu thụ nhiên liệu của tàu biển trong mỗi điều kiện hành hải nhất định sẽ giúp người Sỹ quan hàng hải đưa ra những phương án tối ưu cho hoạt động và an toàn của tàu trong suốt thời gian hành trình.

Ứng dụng phương pháp bình phương nhỏ nhất, kết hợp với ngôn ngữ lập trình Visual Basic 2010, NCS tiến hành xây dựng phần mềm Quản lý hoạt động đội tàu "Vessel Fleet Manager" với các chức năng chủ yếu sau:

- Tổng hợp, phân tích đặc tính thay đổi tốc độ tàu biển trong từng điều kiện hành hải cụ thể;

- Tổng hợp, phân tích đặc tính tiêu thụ nhiên liệu tàu biển trong từng điều kiện hành hải cụ thể.

Phần mềm có giao diện chính như sau:



Hình 3.1 Giao diện đăng nhập phần mềm quản lý hoạt động đội tàu "Vessel Fleet Manager"

* Đăng nhập phần mềm quản lý hoạt động đội tàu "Vessel Fleet Manager"

- Lựa chọn: Đội tàu (Fleet), Username và Password để đăng nhập

Hình 3.2 Giao diện phần mềm khi đăng nhập

Hình 3.3 Giao diện phần mềm sau khi đăng nhập

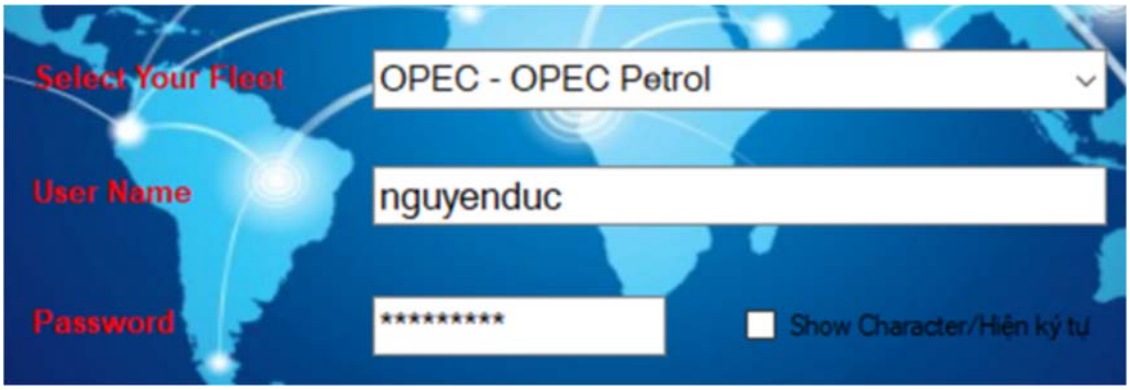
- Giả sử lựa chọn: Tàu (MV Gas Nirvana – (IMO-9140607))

Hình 3.4 Giao diện phần mềm sau khi đã lựa chọn tàu MV Gas Nirvana (IMO - 9140607)

3.4.2. Quy trình quản lý thông tin về đặc tính thay đổi tốc độ tàu biển trong từng điều kiện hành hải cụ thể

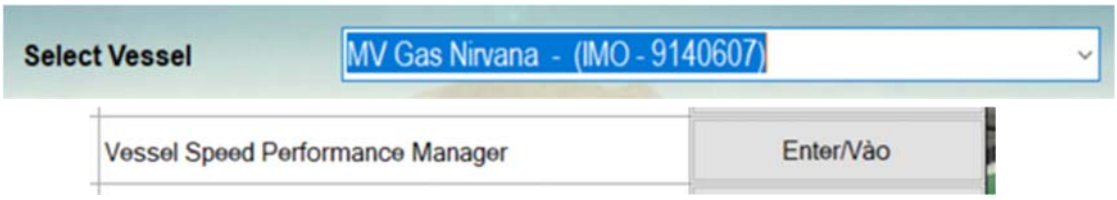
Bước 1: Đăng nhập phần mềm quản lý đội tàu "Vessel Fleet Manager"

- Lựa chọn: Đội tàu (Fleet), Username và Password để đăng nhập, ví dụ:



Hình 3.5 Đăng nhập phần mềm quản lý hoạt động đội tàu

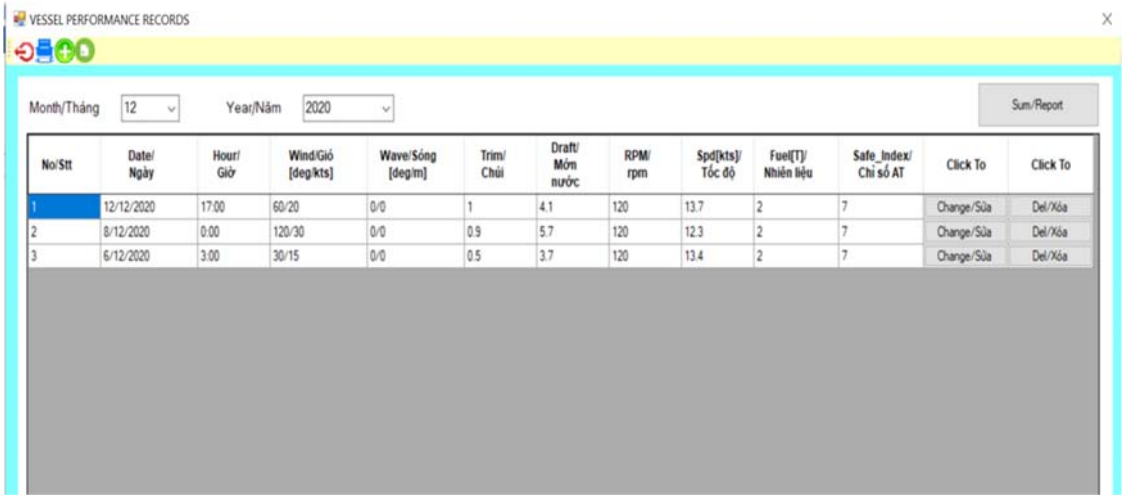
- Ví dụ lựa chọn: Tàu (MV Gas Nirvana – (IMO-9140607)) và "Vessel Speed Performance Manager" để nhập liệu



Hình 3.6 Lựa chọn tàu MV Gas Nirvana – (IMO - 91400607) và Vessel Speed Performance Manager

Bước 2: Nhập dữ liệu

NCS tiến hành nhập dữ liệu để ghi lại đặc tính thay đổi tốc độ của tàu biển trong từng điều kiện hành hải cụ thể.



Hình 3.7 Giao diện phần mềm sau khi lựa chọn tàu (MV Gas Nirvana – (IMO-9140607)) và "Vessel Speed Performance Manager" để nhập liệu

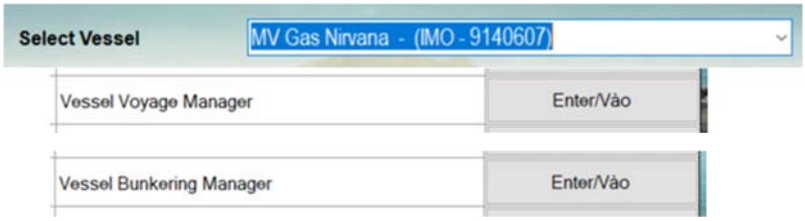
3.4.3. Quy trình quản lý thông tin về đặc tính tiêu thụ nhiên liệu của tàu biển trong từng điều kiện hành hải cụ thể

Tương tự mục 3.4.2, NCS lần lượt tiến hành các bước sau:

Bước 1: Đăng nhập phần mềm quản lý đội tàu "Vessel Fleet Manager"

- Lựa chọn: Đội tàu (Fleet), Username và Password để đăng nhập;

- Ví dụ lựa chọn: Tàu (MV Gas Nirvana – (IMO-9140607)), "Vessel Voyage Manager" và "Vessel Bunkering Manager" để nhập liệu

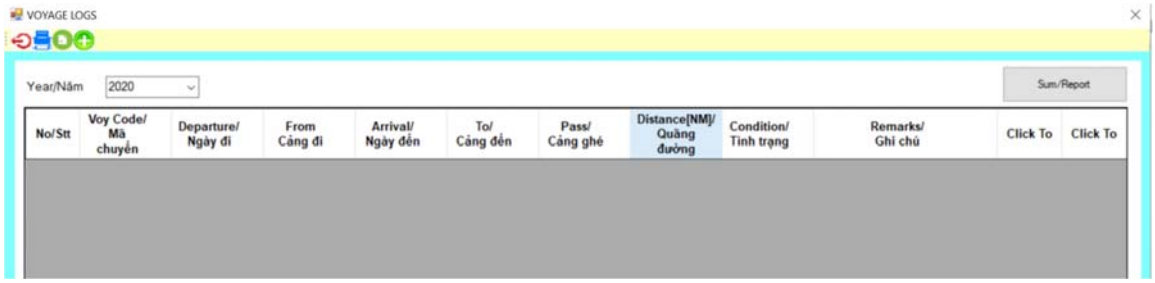


Hình 3.8 Lựa chọn tàu MV Gas Nirvana – (IMO - 91400607), "Vessel Voyage Manager" và "Vessel Bunkering Manager"

Bước 2: Nhập dữ liệu

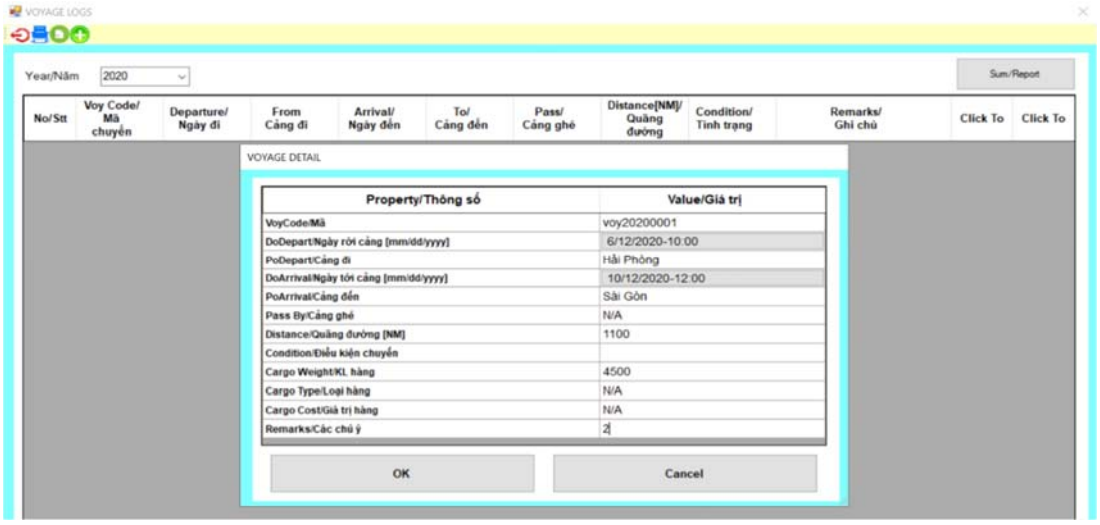
*** Nhập dữ liệu quản lý chuyến hành trình**

- Sau khi lựa chọn Vessel Voyage Manager Enter/Vào giao diện phần mềm có dạng như sau:

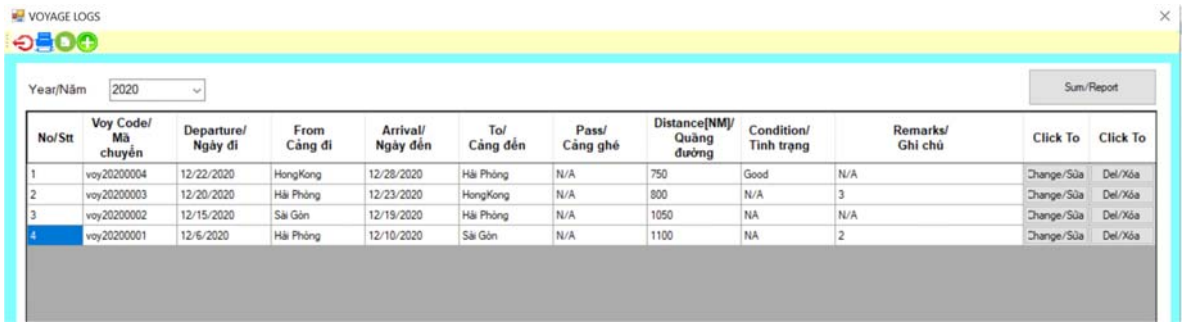


Hình 3.9 Giao diện phần mềm sau khi lựa chọn "Vessel Voyage Manager"

- Nhập dữ liệu các chuyến hành trình theo thực tế, ví dụ:



Hình 3.10 Giao diện phần mềm khi nhập dữ liệu chuyến hành trình



Hình 3.11 Dữ liệu các chuyến hành trình sau khi nhập được lưu lại

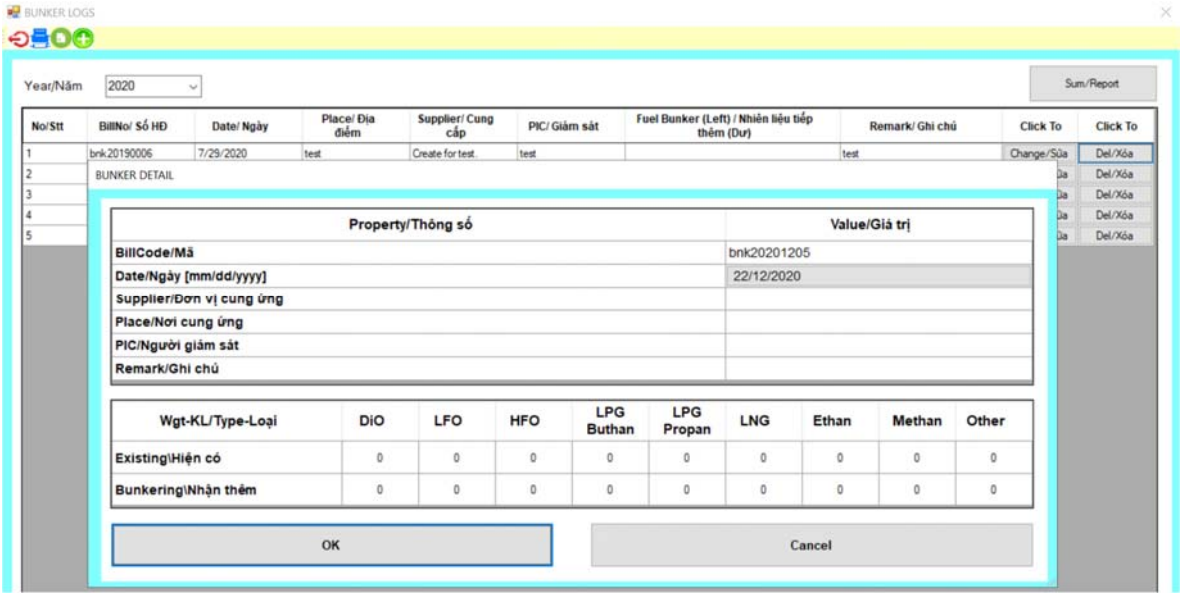
Lưu ý: Quá trình nhập dữ liệu nếu bị sai/ nhầm lẫn có thể sửa lại/ xóa bỏ dữ liệu (nếu cần).

Bằng cách làm này, chúng ta sẽ quản lý được dữ liệu các chuyến hành trình.

*** Nhập dữ liệu quản lý nhiên liệu cho mỗi chuyến hành trình**

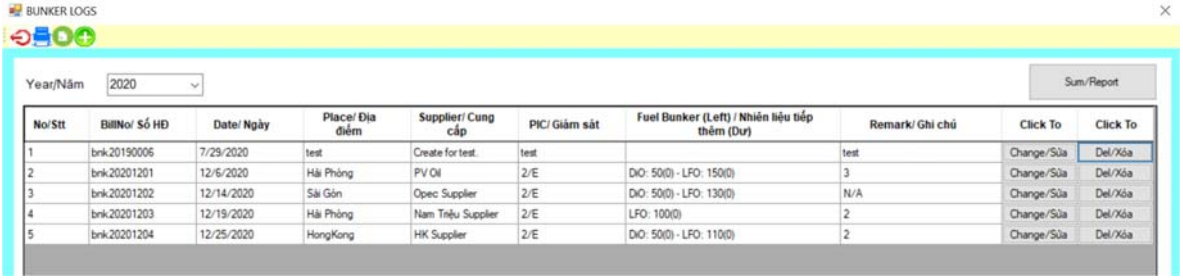
- Sau khi lựa chọn 

giao diện phần mềm có dạng như sau:



Hình 3.12 Giao diện phần mềm khi nhập dữ liệu quản lý nhiên liệu cho mỗi chuyến hành trình

- Nhập dữ liệu quản lý nhiên liệu tàu của mỗi chuyến hành trình theo thực tế, ví dụ như sau:

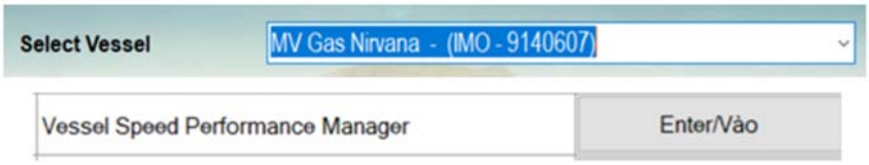


Hình 3.13 Ví dụ dữ liệu quản lý nhiên liệu mỗi chuyến hành trình được lưu lại theo thực tế

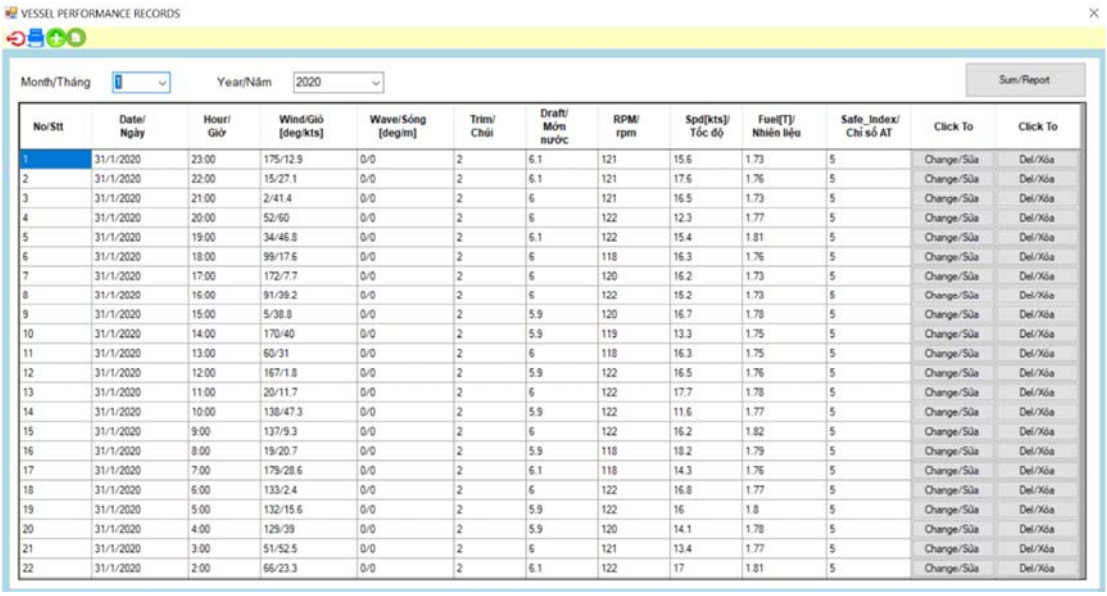
Lưu ý: Quá trình nhập dữ liệu nếu bị sai/ nhầm lẫn có thể sửa lại/ xóa bỏ dữ liệu (nếu cần).

3.4.4. Một số kết quả tổng hợp, phân tích [17]

NCS lựa chọn tàu MV Gas Nirvana – (IMO No. 9140607)) thuộc đội tàu công ty Opec để theo dõi, phân tích đặc tính thay đổi tốc độ và tiêu thụ nhiên liêu trong từng trường hợp cụ thể.



Hình 3.14 Lựa chọn MV Gas Nirvana để ghi nhận dữ liệu



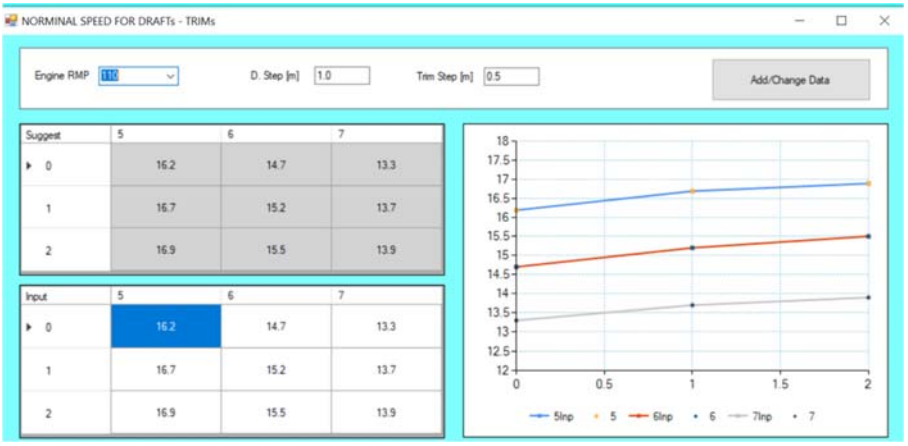
No/Stt	Date/ Ngày	Hour/ Giờ	Wind/Gió [deg/kts]	Wave/Sóng [deg/m]	Trim/ Chúi	Draft/ Mớn nước	RPM/ rpm	Spd[kts]/ Tốc độ	Fuel[T]/ Nhiên liệu	Safe_Index/ Chỉ số AT	Click To	Click To
1	31/1/2020	23:00	175/12.9	0/0	2	6.1	121	15.6	1.73	5	Change/Sửa	Del./Xóa
2	31/1/2020	22:00	15/27.1	0/0	2	6.1	121	17.6	1.76	5	Change/Sửa	Del./Xóa
3	31/1/2020	21:00	2/41.4	0/0	2	6	121	16.5	1.73	5	Change/Sửa	Del./Xóa
4	31/1/2020	20:00	52/60	0/0	2	6	122	12.3	1.77	5	Change/Sửa	Del./Xóa
5	31/1/2020	19:00	34/46.8	0/0	2	6.1	122	15.4	1.81	5	Change/Sửa	Del./Xóa
6	31/1/2020	18:00	99/17.6	0/0	2	6	118	16.3	1.76	5	Change/Sửa	Del./Xóa
7	31/1/2020	17:00	172/7.7	0/0	2	6	120	16.2	1.73	5	Change/Sửa	Del./Xóa
8	31/1/2020	16:00	91/39.2	0/0	2	6	122	15.2	1.73	5	Change/Sửa	Del./Xóa
9	31/1/2020	15:00	5/38.8	0/0	2	5.9	120	16.7	1.78	5	Change/Sửa	Del./Xóa
10	31/1/2020	14:00	170/40	0/0	2	5.9	119	13.3	1.75	5	Change/Sửa	Del./Xóa
11	31/1/2020	13:00	60/31	0/0	2	6	118	16.3	1.75	5	Change/Sửa	Del./Xóa
12	31/1/2020	12:00	167/1.8	0/0	2	5.9	122	16.5	1.76	5	Change/Sửa	Del./Xóa
13	31/1/2020	11:00	20/11.7	0/0	2	6	122	17.7	1.78	5	Change/Sửa	Del./Xóa
14	31/1/2020	10:00	138/47.3	0/0	2	5.9	122	11.6	1.77	5	Change/Sửa	Del./Xóa
15	31/1/2020	9:00	137/9.3	0/0	2	6	122	16.2	1.82	5	Change/Sửa	Del./Xóa
16	31/1/2020	8:00	19/20.7	0/0	2	5.9	118	18.2	1.79	5	Change/Sửa	Del./Xóa
17	31/1/2020	7:00	179/28.6	0/0	2	6.1	118	14.3	1.76	5	Change/Sửa	Del./Xóa
18	31/1/2020	6:00	133/2.4	0/0	2	6	122	16.8	1.77	5	Change/Sửa	Del./Xóa
19	31/1/2020	5:00	132/15.6	0/0	2	5.9	122	16	1.8	5	Change/Sửa	Del./Xóa
20	31/1/2020	4:00	129/39	0/0	2	5.9	120	14.1	1.78	5	Change/Sửa	Del./Xóa
21	31/1/2020	3:00	51/52.5	0/0	2	6	121	13.4	1.77	5	Change/Sửa	Del./Xóa
22	31/1/2020	2:00	66/23.3	0/0	2	6.1	122	17	1.81	5	Change/Sửa	Del./Xóa

Hình 3.15 Giao diện phần mềm khi nhập dữ liệu của M.V Gas Nirvana

Sau khi nhập dữ liệu xong, NCS tiến hành phân tích trong một số điều kiện hàng hải nhất định, ví dụ:

* Phân tích đặc tính thay đổi tốc độ tàu biển trong điều khiển không có ảnh hưởng của sóng, gió (bỏ qua ảnh hưởng của dòng chảy)

Tiến hành phân tích đặc tính thay đổi tốc độ MV Gas Nirvana trong điều kiện không có ảnh hưởng của sóng, gió (biển lặng sóng và gió hoặc sóng, gió cấp 1, cấp 2) với giá trị vòng quay chân vịt 110 rpm, độ sai lệch mớn nước là 1.0 m và sai lệch hiệu số mớn nước 0.5 m, cho kết quả theo như Hình 3.16.

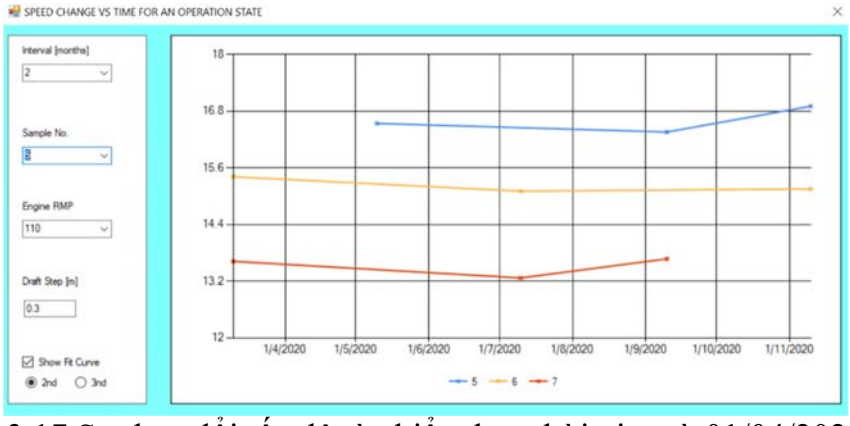


Hình 3.16. Đặc tính thay đổi tốc độ tàu tương ứng với giá trị mớn nước 5m, 6m, và 7m.

Theo Hình 3.16, từ giá trị gợi ý của phần mềm, Sỹ quan hàng hải có thể tham khảo và thu thập giá trị tốc độ tàu, hiển thị đặc tính thay đổi chuyển động của tàu thông qua đồ thị tương ứng với giá trị mớn nước lần lượt là 5m, 6m, và 7m.

* Phân tích đặc tính thay đổi tốc độ tàu biển theo thời gian

Với khoảng thời gian 2 tháng, giá trị vòng quay chân vịt 110 rpm, độ sai lệch mớn nước là 0.3 m và sai lệch hiệu số mớn nước 0.5 m, đặc tính tốc độ của tàu MV Gas Nirvana thay đổi theo thời gian tương ứng với giá trị mớn nước 5m, 6m, và 7m như Hình 3.17.

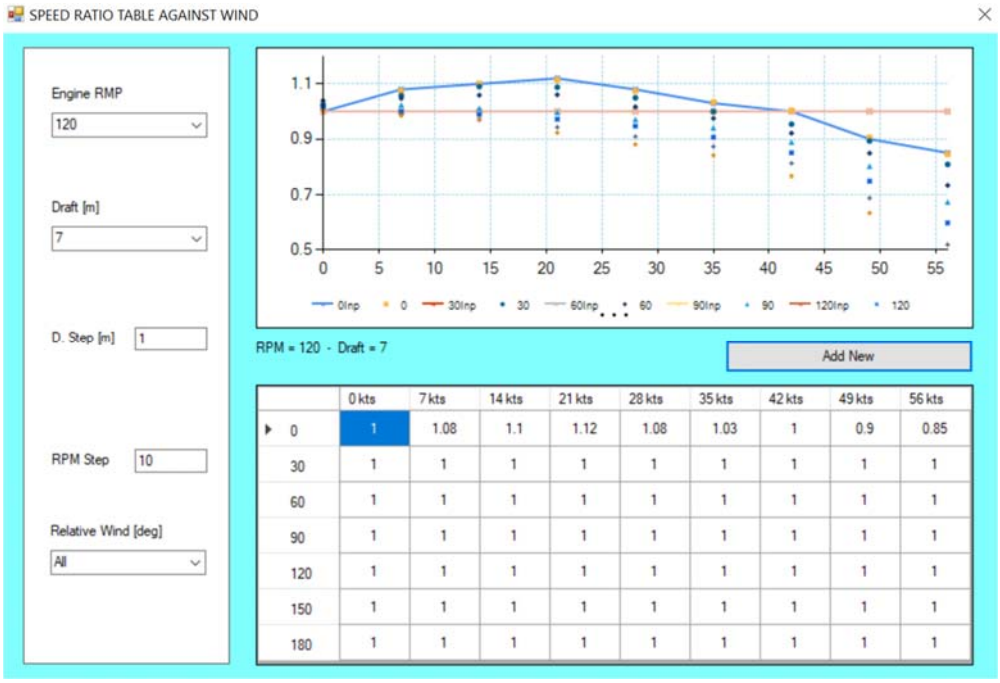


Hình 3.17 Sự thay đổi tốc độ tàu biển theo thời gian từ 01/04/2020 đến 01/11/2020.

* Phân tích đặc tính thay đổi tốc độ tàu biển khi có ảnh hưởng của gió và sóng (nếu có)

Trên thực tế, việc quan trắc và ghi lại các giá trị sóng là nhiệm vụ hết sức khó khăn đối với người Sỹ quan hàng hải, do đó trong trường hợp này NCS chỉ phân tích sự thay đổi tốc độ tàu khi có ảnh hưởng của gió và bỏ qua yếu tố sóng.

Khi đó, sự thay đổi tốc độ tàu khi có ảnh hưởng của gió được tính toán từ phần mềm sẽ được hiển thị theo đồ thị Hình 3.18.



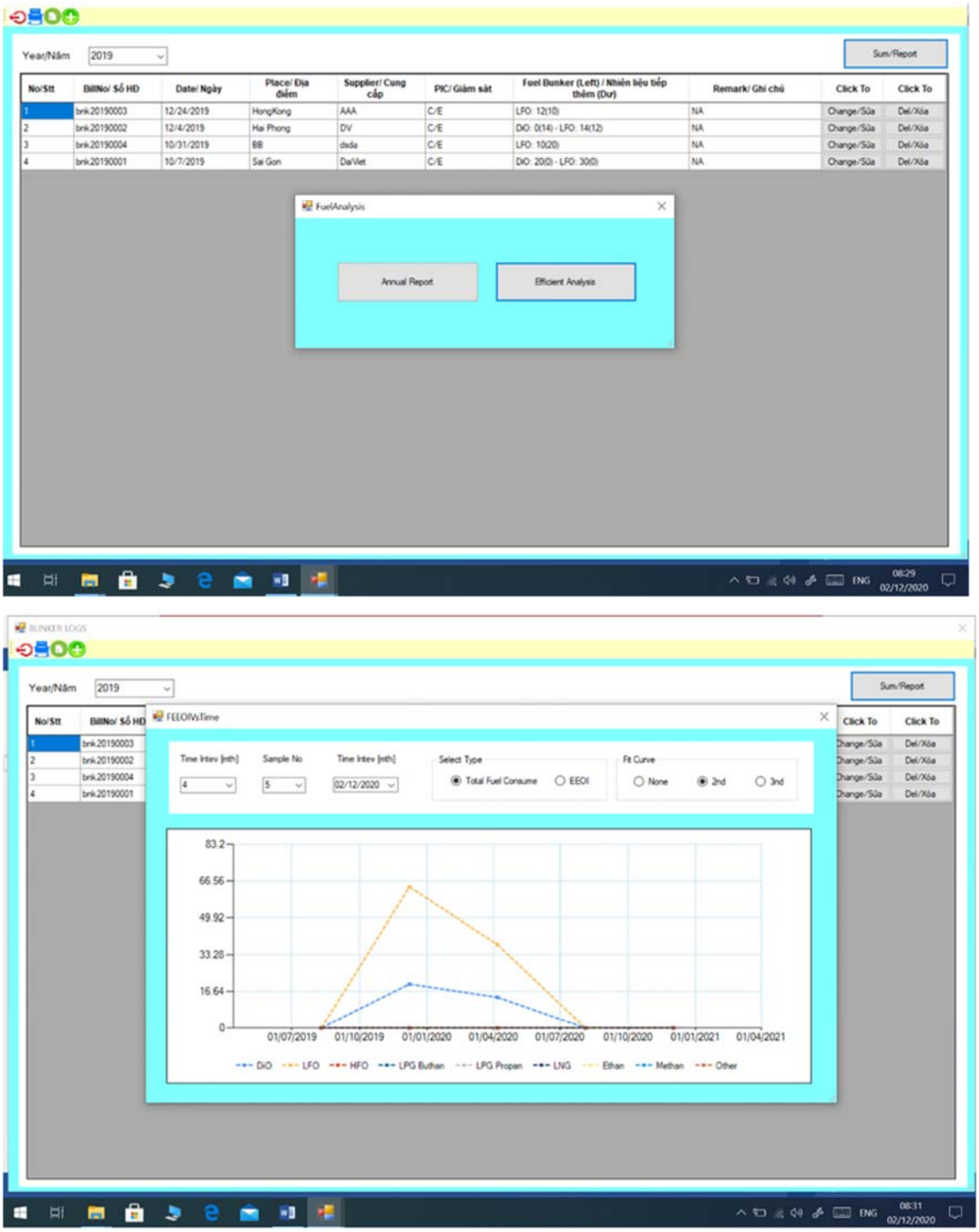
	0 kts	7 kts	14 kts	21 kts	28 kts	35 kts	42 kts	49 kts	56 kts
0	1	1.08	1.1	1.12	1.08	1.03	1	0.9	0.85
30	1	1	1	1	1	1	1	1	1
60	1	1	1	1	1	1	1	1	1
90	1	1	1	1	1	1	1	1	1
120	1	1	1	1	1	1	1	1	1
150	1	1	1	1	1	1	1	1	1
180	1	1	1	1	1	1	1	1	1

Hình 3.18 Sự thay đổi tốc độ tàu biển khi hướng gió tương đối thay đổi từ 0^0 đến 180^0

* Phân tích hiệu quả sử dụng năng lượng

Ứng dụng phần mềm phân tích mức sử dụng nhiên liệu hiệu quả, giảm thiểu lượng phát thải khí Co_2 được trích xuất dưới dạng biểu đồ.

NCS lựa chọn tab Sum/Report, sau đó chọn tab Efficient Analysis



Hình 3.19 Giao diện phân tích hiệu quả sử dụng năng lượng

3.5. Kết luận chương 3

Kết thúc chương 3, NCS đã đạt được các kết quả nghiên cứu chính sau:

- NCS khái quát được đặc tính thay đổi tốc độ và đặc tính tiêu thụ nhiên liệu của tàu biển trong từng điều kiện hành hải cụ thể;

- Ứng dụng phương pháp bình phương nhỏ nhất NCS xác định được đặc tính thay đổi tốc độ và đặc tính tiêu thụ nhiên liệu tàu biển trong từng điều kiện hành hải cụ thể;

- Đồng thời, NCS xây dựng phần mềm tổng hợp, phân tích đặc tính thay đổi tốc độ và đặc tính tiêu thụ nhiên liệu tàu biển trong từng điều kiện hành hải cụ thể phục vụ tính toán tuyến đường và kế hoạch chạy tàu tối ưu.

CHƯƠNG 4: NGHIÊN CỨU, XÂY DỰNG THUẬT TOÁN VI KHUẨN CẢI TIẾN ĐỂ TÍNH TOÁN TUYẾN ĐƯỜNG VÀ KẾ HOẠCH CHẠY TÀU TỐI ƯU NHIÊN LIỆU DỰA TRÊN NGUYÊN TẮC JUST IN TIME "TÀU ĐẾN CẢNG KỊP LÚC"

4.1. Tổng quan về thuật toán vi khuẩn (BFOA _ Bacterial Foraging Optimization Algorithm)

4.1.1. Khái niệm

Thuật toán vi khuẩn (BFOA _ Bacterial Foraging Optimization Algorithm) được đề xuất lần đầu tiên bởi Passino [22, 25] vào năm 2002 đã thu hút được sự quan tâm của nhiều nhà nghiên cứu trong nhiều năm qua.

Thuật toán vi khuẩn là thuật toán tối ưu dựa trên phương pháp tìm kiếm thức ăn của bầy vi khuẩn. Thuật toán có rất nhiều ưu điểm và đã thu hút được sự quan tâm của nhiều nhà nghiên cứu trong nhiều lĩnh vực. Hiện nay, thuật toán vi khuẩn luôn là giải pháp khả thi, được áp dụng hiệu quả cho nhiều lĩnh vực khác nhau như điều khiển tối ưu, trí tuệ nhân tạo, dự đoán điều hòa, …

NCS nêu bật một số ưu điểm của thuật toán vi khuẩn, cụ thể:

- BFOA là thuật toán tối ưu dựa trên số đông các vi sinh vật đơn giản, các phần tử riêng biệt trong tập hợp có tính chất tự chủ và phân tán, không có sự điều khiển tập trung nên việc 1 hoặc 1 số phần tử kém hiệu quả hoặc thất bại không làm ảnh hưởng tới việc giải quyết vấn đề của cả tập hợp, các phần tử còn lại vẫn có khả năng tìm nghiệm tối ưu cho bài toán một cách độc lập. Nhờ vậy, thuật toán BFO hiệu quả và mạnh hơn so với các phương pháp số khác;

- BFOA có thể được mở rộng một cách dễ dàng do: Việc hợp tác (kết bầy) của các cá thể là thông qua các liên lạc gián tiếp, nhờ sự mở rộng dễ dàng của BFOA, ta có thể tăng quy mô tập hợp vi khuẩn để giải quyết những vấn đề phức tạp hơn một cách nhanh chóng;

- BFOA chủ yếu sử dụng các công thức toán học cơ bản do đó có thể áp dụng một cách đơn giản, nhanh chóng và hiệu quả trên máy tính;

- BFOA khi được áp dụng không cần phải giả định về tính khả vi, hàm lồi cũng như các yêu cầu khác về mặt toán học nên thuật toán được ứng dụng để giải quyết nhiều vấn đề khác nhau về tối ưu.

4.1.2. Nguyên lý chung của thuật toán vi khuẩn (BFOA) [22, 25]

Giả sử, ta cần tìm cực tiểu của 1 hàm $Q(S)$ với miền xác định cho trước (được gọi là miền giải pháp, miền nghiệm số hay miền tìm kiếm). Ngoài ra, 1 nghiệm S phải thỏa mãn một số điều kiện ràng buộc nhất định.

Thực tế, bài toán này hết sức phức tạp và không thể (hoặc rất khó có thể) giải được bằng các phương pháp giải tích thông thường, hoặc thậm chí là các phương pháp số vì các khó khăn khi cần tính giá trị đạo hàm (hay Gradient) $\Delta Q(S)$. Vì vậy, các bài toán này được gọi là bài toán tối ưu không sử dụng gradient. Đối với các bài toán này, thay vì xác định giá trị tối ưu chính xác, bài toán có thể được coi như đã giải quyết xong nếu ta tìm được nghiệm xấp xỉ với giá trị tối ưu.

Dựa trên mô phỏng quá trình phát triển của tập hợp vi khuẩn, BFOA là một công cụ tối ưu hóa bằng cách lặp đi lặp lại việc tìm các phương án (hay một nghiệm khả thi) tốt hơn từ một phương án ban đầu, trong đó, các phương án mới được tạo ra một cách ngẫu nhiên. Mỗi phương án được thể hiện bằng vị trí của một cá thể vi khuẩn trong một không gian đặc trưng cho không gian nghiệm. Trong phần này, vị trí của một cá thể được dùng đồng nghĩa với một phương án hay một giải pháp cho bài toán tối ưu.

Áp dụng BFOA, bài toán có thể được giải quyết nhờ việc lặp đi lặp lại mô phỏng về sự phát triển của tập hợp vi khuẩn được thể hiện qua các giai đoạn trong thời gian sống của mỗi cá thể như sau:

- Mỗi cá thể vi khẩn tự thích nghi với môi trường và phát triển;

- Sự phù hợp của cá thể vi khuẩn với môi trường (hay gọi đơn giản là sức khỏe) được xác định trong mỗi thế hệ. Các cá thể vi khuẩn khỏe nhất tồn tại được và sinh sôi, các cá thể yếu chết đi và không còn tồn tại trong tập hợp;

- Các cá thể vi khuẩn tồn tại được trong tập hợp có xu hướng kết bầy.

Như vậy, sau mỗi vòng lặp như trên, các cá thể vi khuẩn sẽ khỏe hơn các cá thể trước đó, tức là vi khuẩn đang tiến gần hơn tới nghiệm tối ưu của bài toán. Nếu được thiết kế phù hợp, thuật toán sẽ cho phép tìm được giá trị gần đúng của phương án tối ưu sau một số hữu hạn các vòng lặp như vừa nêu.

NCS nêu ra một số thuật ngữ được sử dụng để minh họa cho BFOA:

$$S = [P(1), P(2), \cdots, P(i), \cdots, P(N)] \quad (4.1)$$

(4.1) là 1 vector thể hiện 1 phương án hay chính là vị trí của 1 cá thể vi khuẩn trong không gian tìm kiếm.

Trong đó:

i: tọa độ thứ i;

N: số chiều của không gian tìm kiếm;

$$V = [V(1), V(2), \cdots, V(i), \cdots V(N)] \quad (4.2)$$

(4.2) là 1 vector thể hiện hướng tìm kiếm, với $V(i) = 0$ hoặc $V(i) = 1$

D: Khoảng cách dịch chuyển (hay còn gọi là bước tìm);

Q: Hàm mục tiêu (hay là chỉ số sức khỏe của cá thể vi khuẩn);

dQ: giá trị bổ xung cho hàm mục tiêu thể hiện mối liên hệ giữa các cá thể vi khuẩn.

4.1.3. Phân loại thuật toán vi khuẩn

Thuật toán vi khuẩn được chia thành thuật toán vi khuẩn cổ điển và thuật toán vi khuẩn cải tiến:

4.1.3.1. Thuật toán vi khuẩn cổ điển

Thuật toán vi khuẩn cổ điển được thực hiện qua 3 bước như sau:

*** Bước 1: Khởi tạo**

Khởi tạo tập hợp vi khuẩn, trong đó mỗi cá thể vị khuẩn được đặt ở một vị trí ngẫu nhiên trong không gian tìm kiếm.

*** Bước 2: Tiến hóa**

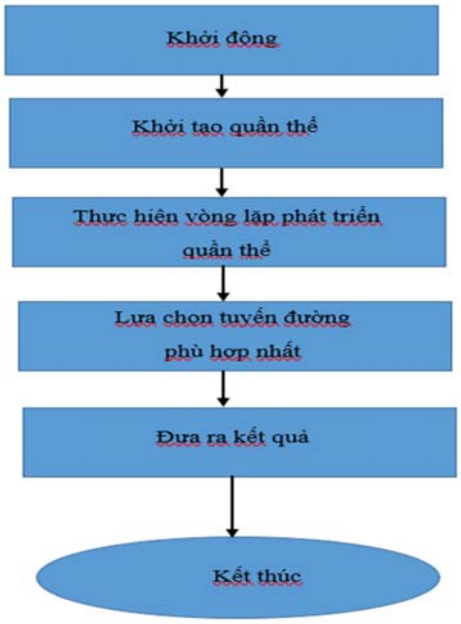
Lặp đi lặp lại quá trình tiến hóa của tập hợp vi khuẩn qua 3 bước nhỏ:

- Tìm kiếm, phát triển và kết bầy (chemotaxis and swarming);
- Sinh sản;
- Loại trừ và phân tán.

Quá trình lặp trong bước này cho phép tập hợp vi khuẩn tập trung dần về vùng thuận lợi nhất trong không gian tìm kiếm (vùng có hàm mục tiêu lớn nhất). Tiếp đến, thông qua việc kết bầy, việc tìm kiếm cục bộ được phân bổ thích hợp. Nhờ quá trình sinh sản, quá trình tìm kiếm được tập trung nhiều hơn vào các khu vực cụ thể. Cuối cùng, việc phân tán sẽ giúp cho quá trình tìm kiếm tránh được các giá trị cục bộ.

*** Bước 3: Kết thúc**

Trả về phương án ứng với cá thể vi khuẩn có sức khỏe tốt nhất.



Hình 4.1 Sơ đồ khối mô tả các bước của thuật toán vi khuẩn cổ điển

4.1.3.2. Thuật toán vi khuẩn cải tiến

Bằng nhiều nghiên cứu và thử nghiệm với nhiều hàm đa mô hình có độ phức tạp khác nhau, các nhà khoa học đã chỉ ra nhược điểm của thuật toán vi khuẩn cổ điển chính là: Khả năng tìm kiếm giá trị tối ưu giảm mạnh khi số chiều của không gian tìm kiếm hay nói cách khác độ phức tạp của bài toán tăng lên.

Nhằm mục đích nâng cao hiệu quả tìm kiếm, thuật toán vi khuẩn cải tiến đã được rất nhiều nhà khoa học nghiên cứu, cải tiến và cho ra đời, cụ thể:

* M. Tripathy, S. Mishra, et al [41] đề xuất thuật toán BFO trong đó áp dụng 2 cải tiến:

- Cải tiến thứ nhất: Giá trị tối ưu ở mỗi vị trí được sử dụng thay vì giá trị trung bình của tất cả các bước;

- Cải tiến thứ hai: Khoảng cách từ mỗi vi khuẩn tới vi khuẩn ứng với nghiệm số tốt nhất đã tìm được được sử dụng như một yếu tố để thay đổi độ dài bước dịch chuyển (bước trượt), nhờ vậy tốc độ hội tụ được tăng lên.

* S. Mishra [42] đề xuất một phương pháp mờ để xác định độ dài bước di chuyển tối ưu cho thuật toán vi khuẩn tại mỗi thời điểm.

* C. Ying et al [20] đề xuất phương án trong đó, việc kết bầy được thực hiện tương tự như việc kết bầy trong thuật toán PSO (Particle Swarm Optimization Algorithm), theo đó, vị trí của mỗi cá thể vi khuẩn sau mỗi bước di chuyển được xác định theo công thức:

$$if Q(S^b(j) < Q\left(S^i(j+1)\right) then$$

$$S^i_{new}(j+1) = S^i_{old}(j+1) + C_{cc}x(S^b(j) - S^i(j)) \quad (4.3)$$

Where b is the best bacterium in previous chemotaxic step, C_{cc} is an attraction factor.

Tiếp theo, bước di chuyển được giảm dần sau mỗi vòng lặp, thuật toán này được gọi là thuật toán FSBA (Fast Bacteria Swarming Algorithm).

* H. Chen, Y. Zhu and K. Hu [21] phát triển thuật toán BFO thích nghi (Adaptive Bacteria Foraging Optimization Algorithms - ABFA) trong đó, độ dài bước di chuyển được điều chỉnh theo độ chính xác yêu cầu trong mỗi giai đoạn tìm kiếm, được gọi là giai đoạn khám phá (exploration) và giai đoạn khai thác (exploitation).

Trong giai đoạn "khám phá", bước di chuyển dài được sử dụng để lướt tới các vùng chưa được kiểm tra đồng thời tăng tốc độ tiếp cận giá trị tối ưu; Tiếp đó, để "khai thác" bước dịch chuyển nhỏ được sử dụng để tìm kiếm trong vùng có nhiều khả năng xuất hiện giá trị tối ưu.

Ngoài ra còn nhiều cải tiến cho thuật toán vi khuẩn khác, hầu hết các sửa đổi đều mang tính chuyên biệt cho từng ứng dụng cụ thể, đồng thời không có quy tắc chung cho việc lựa chọn các thông số thiết kế thuật toán.

Đặc biệt, trong nước thông qua việc thực hiện đề tài nghiên cứu khoa học cấp trường năm 2013 [10], TS. Nguyễn Minh Đức đã cải tiến thuật toán vi khuẩn và ứng dụng để xác định tuyến đường hàng hải khí tượng tối ưu với mục tiêu của việc lập tuyến (hay hàm mục tiêu) là thời gian hành trình ngắn nhất (với điều kiện an toàn của tàu, hàng hóa và sức khỏe thuyền viên luôn được đảm bảo). Khi đó, chi phí gắn với tuyến hành trình chính là thời gian hành trình được tính là:

$$Q(S) = T_1 + T_2 + \cdots + T_n \quad (4.4)$$

Trong đó: T_i là thời gian cần thiết để hoàn thành đoạn thứ i của tuyến

Bài toán tối ưu cần giải sẽ là: Xác định bộ các điểm nút (hay điểm chuyển hướng Waypoint) trên mỗi đường để tổng thời gian hành trình của tàu là ngắn nhất hay xác định vector S^* sao cho:

$$Q(S^*) = MinQ(S) \quad (4.5)$$

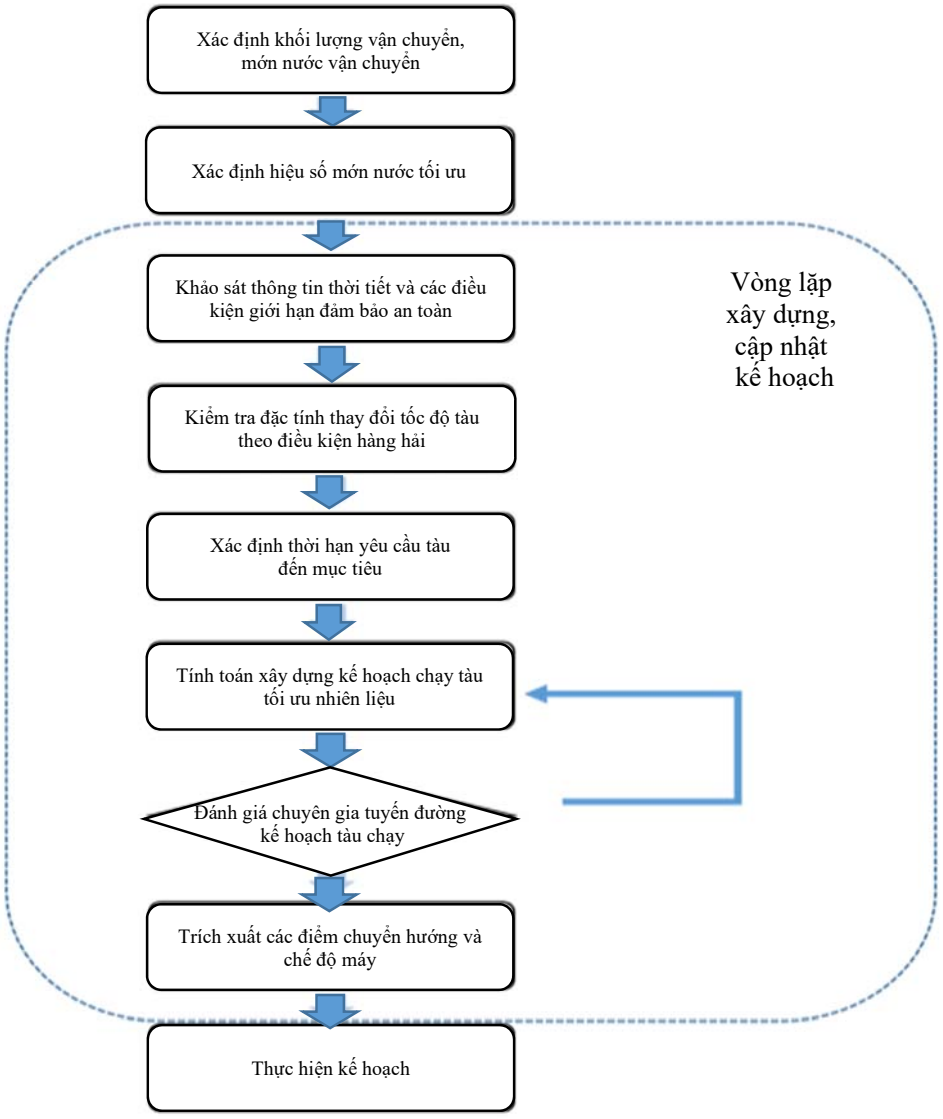
Như vậy kể từ khi được đề xuất lần đầu tiên vào năm 2002 thuật toán vi khuẩn với nhiều ưu điểm vượt trội đã thu hút được sự quan tâm của nhiều nhà nghiên cứu. Theo thời gian, để tăng hiệu quả tìm kiếm và ứng dụng hiệu quả hơn nữa cho các lĩnh vực, các nhà khoa học không ngừng nghiên cứu, cải tiến để thuật toán mang tính chuyên biệt.

Trong chương này, NCS nghiên cứu, đưa ra cải tiến riêng cho thuật toán vi khuẩn nhằm tính toán tuyến đường và kế hoạch chạy tàu tối ưu nhiên liệu dựa nguyên tắc Just in Time "Tàu đến cảng kịp lúc".

4.2. Sơ đồ tổng quát hướng dẫn việc tính toán và áp dụng tuyến đường chạy tàu tối ưu nhiên liệu dựa trên nguyên tắc just in time "tàu đến cảng kịp lúc"

Áp dụng tuyến đường chạy tàu tối ưu nhiên liệu dựa trên nguyên tắc just in time "tàu đến cảng kịp lúc" bao gồm việc tính toán tuyến đường chạy tàu tối ưu và việc vận hành tàu một cách hợp lý trên từng đoạn tuyến đã được tính toán, kết hợp duy trì thông tin liên lạc thông suốt giữa các bên để đảm bảo "tàu đến cảng kịp lúc".

NCS đề xuất sơ đồ tổng quát hướng dẫn việc tính toán và áp dụng tuyến đường chạy tàu tối ưu nhiên liệu dựa trên nguyên tắc just in time "tàu đến cảng kịp lúc" như hình vẽ sau:



Hình 4.2 Sơ đồ hướng dẫn việc tính toán và áp dụng tuyến đường chạy tàu tối ưu nhiên liệu dựa trên nguyên tắc just in time "tàu đến cảng kịp lúc"

Như vậy, việc tính toán và áp dụng tuyến đường chạy tàu tối ưu just in time cần phải:

- Căn cứ vào các điều kiện của chuyến hành trình thực tế, điều kiện thực tế luồng lạch của cảng đi và cảng đến, điều kiện thời tiết, đặc điểm vận hành, giới hạn an toàn tàu;

- Tính tới thời gian yêu cầu tàu đến, cũng như sự thay đổi bất ngờ của các điều kiện thời tiết, đặc điểm hành trình dự tính;

- Trước khi bắt đầu hành trình, tuyến đường và tốc độ tàu được tính toán cho mớn nước, trạng thái xếp hàng và dự báo thời tiết gần nhất. Trong quá trình chạy tàu, khi điều kiện thời tiết thay đổi và các bản tin thời tiết được cập nhật, tuyến đường chạy tàu và kế hoạch chạy tàu có thể được tính toán lại cho phù hợp để áp dụng;

- Việc lựa chọn các điều kiện giới hạn an toàn tàu (ví dụ thời tiết, vĩ độ giới hạn, …) là trách nhiệm của Thuyền trưởng hoặc các Sỹ quan được phân công trên cơ sở kinh nghiệp và hiểu biết về con tàu của mình.

Các công việc cụ thể cần làm bao gồm:

* Tính toán lượng hàng vận chuyển:

- Lượng hàng cần vận chuyển;

- Độ sâu luồng lạch trên toàn tuyến;

- Lượng dự trữ nhiên liệu, nước ngọt cần thiết cho toàn tuyến;

- Mớn nước tối đa theo mùa, theo khu vùng địa lý.

* Tính toán mớn nước, chênh lệch hiệu số mớn nước tối ưu:

- Sự thay đổi mớn nước do thay đổi lượng dự trữ nhiên liệu, nước ngọt trên toàn tuyến;

- Mớn nước tối ưu tham khảo từ đồ thị thay đổi tốc độ tàu theo hiệu số mớn nước [5];

- Sự cần thiết duy trì tính năng điều động tàu.

* Khảo sát thông tin thời tiết trên toàn bộ tuyến hành trình:

- Các yếu tố thời tiết nguy hiểm theo dự báo;

- Cập nhật thông tin thời tiết dạng số toàn tuyến trên hệ thống;

- Xác định các điều kiện thời tiết tới hạn cho tàu.

* Rà soát đặc tính thay đổi tốc độ tàu biển và đặc tính tiêu thụ nhiên liệu tàu biển trong từng điều kiện hành hải cụ thể

* Xác định thời hạn mục tiêu tàu cần đến điểm đích:

- Điểm đến xác định có thể là cảng đích, hoặc điểm chờ qua kênh, chờ vào cảng;

- Thời hạn mục tiêu cần có dự trữ phù hợp phòng trường hợp có thay đổi bất thường trong hành trình.

* Tính toán tuyến đường chạy tàu tối ưu just in time:

- Chọn các chế độ vòng tua máy;

- Nhập các hạn chế thời tiết;

- Nhập các vĩ độ, kính độ giới hạn;

- Nhập thời gian yêu cầu tàu đến;

- Nhập các đường hạn chế;

- Chạy phần mềm tính toán tuyến đường;

- Kiểm tra tuyến đường tính toán, tính lại để nâng cao chất lượng (nếu cần);

- Kiểm tra để thay đổi các điều kiện giới hạn nếu thời gian tàu đến không đáp ứng yêu cầu.

* Trích xuất các điểm chuyển hướng (WPTs) và chế độ máy:

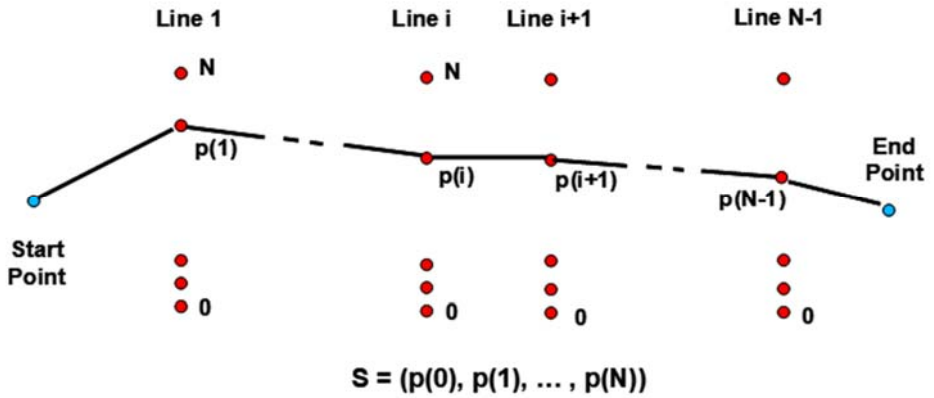
- Lấy danh sách các điểm chuyển hướng;

- Lựa chọn chế độ máy (rpm) trên từng đoạn tuyến.

4.3. Nghiên cứu, xây dựng thuật toán vi khuẩn cải tiến tính toán tuyến đường và kế hoạch chạy tàu tối ưu nhiên liệu dựa trên nguyên tắc just in time "tàu đến cảng kịp lúc"

Để đạt được mục tiêu nghiên cứu của đề tài luận án là nâng cao hơn nữa hiệu quả sử dụng năng lượng trên tàu biển và giảm thiểu đến mức thấp nhất lượng khí nhà kính phát thải từ tàu biển, NCS nghiên cứu, xây dựng thuật toán vi khuẩn cải tiến, ứng dụng tính toán tuyến đường và kế hoạch chạy tàu tối ưu nhiên liệu dựa trên nguyên tắc just in time "tàu đến cảng kịp lúc".

4.3.1. Khái niệm về không gian tìm kiếm (hay mạng các nút), đường nút và tuyến Hàng hải



Hình 4.3 Hình vẽ mô phỏng không gian tìm kiếm (mạng các nút), đường nút và tuyến Hàng hải

Điểm nút hay hay điểm chuyển hướng (điểm Waypoint - WPT) thường được ký hiệu bởi 3 chữ số, trong đó 2 chữ số đầu thể hiện nhóm điểm và chữ số thứ 3 thể hiện số thứ tự của điểm trong nhóm.

Điểm chuyển hướng (Waypoint) còn có thể được đặt tên bằng các biểu tượng và các ký tự chữ cái đi kèm. Ví dụ: 311, 312, 313, 314, … tức là nhóm điểm 31, các điểm trong nhóm lần lượt là 1, 2, 3 và 4.

Thông thường các điểm chuyển hướng (WPTs) trong cùng một tuyến Hàng hải được đặt tên theo nhóm để tiện theo dõi và quản lý. Một tuyến đường Hàng hải bao gồm nhiều điểm chuyển hướng, tối thiểu là từ 2 điểm chuyển hướng trở lên. Thực tế Hàng hải cho thấy số điểm chuyển hướng (Waypoint) hay điểm nút và giãn cách giữa các điểm nút được lựa chọn tùy thuộc vào điều kiện hàng hải thực tế miễn sao an toàn tàu, hàng hóa, sức khỏe thuyền viên được đảm bảo và tiết kiệm chi phí nhất, tối ưu hóa nhiên liệu nhất.

Để tìm đường đi tối ưu cho tàu, trước tiên, ta cần xác lập một không gian tìm kiếm chính là một mạng các điểm chuyển hướng (WPTs) hay điểm nút như hình 4.3.

Theo hình 4.3, tuyến đường Hàng hải bắt đầu từ điểm xuất phát (Start Point) đến điểm kết thúc (End Point) được chia thành N đoạn liên tiếp. Mỗi đoạn của

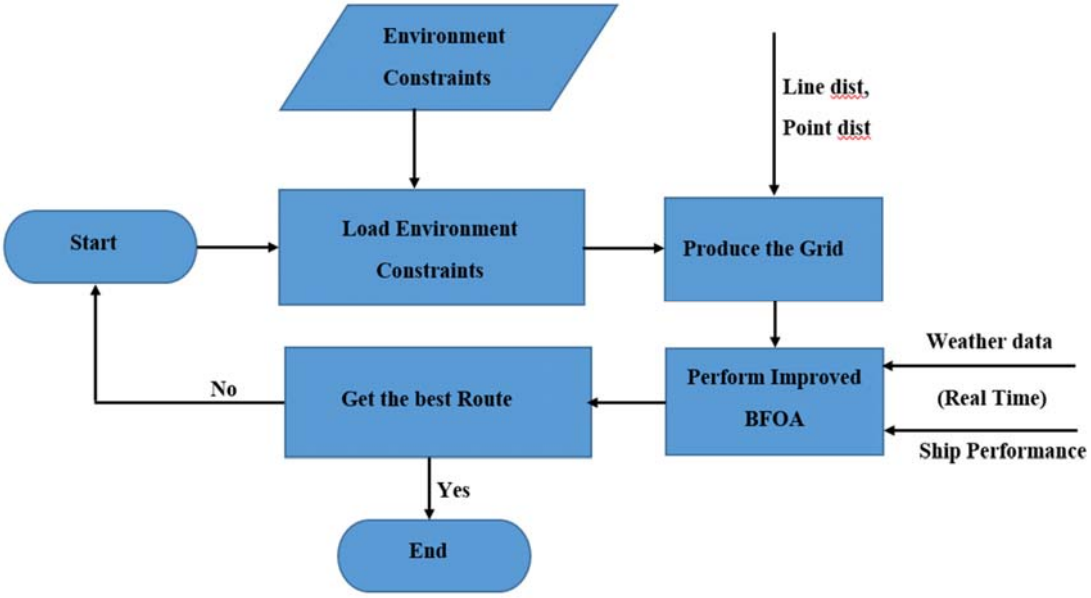
tuyến là đường nối từ một điểm nút (hay 1 Waypoint) trên 1 đường ($line_i$) tới 1 điểm nút khác (hay 1 Waypoint) khác trên đường tiếp theo ($line_{i+1}$).

Như vậy, một tuyến đường Hàng hải từ điểm xuất phát đến điểm kết thúc là một tuyến từ điểm đầu, đi qua các điểm nút trung gian và tiến tới điểm cuối. Một đoạn của tuyến chỉ được thực hiện được khi đảm bảo các giới hạn an toàn tàu như: không đi vào khu vực nông cạn, chật hẹp hoặc cắt đường bờ; Thời tiết trên tuyến nằm trong điều kiện giới hạn an toàn tàu.

Mạng lưới các điểm nút theo như Hình 4.3 chính là không gian tìm kiếm.

4.3.2. Sơ đồ khối nguyên lý tính toán tuyến đường chạy tàu tối ưu nhiên liệu dựa trên nguyên tắc just in time "tàu đến cảng kịp lúc" ứng dụng thuật toán vi khuẩn cải tiến

Sơ đồ khối trong hình vẽ sau đây mô tả nguyên lý tính toán tuyến đường chạy tàu tối ưu nhiên liệu dựa trên nguyên tắc just in time "tàu đến cảng kịp lúc" bằng cách ứng dụng thuật toán vi khuẩn cải tiến.



Hình 4.4 Sơ đồ nguyên lý tính toán tuyến đường chạy tàu tối ưu nhiên liệu dựa trên nguyên tắc just in time "tàu đến cảng kịp lúc" ứng dụng BFOA cải tiến

Theo sơ đồ hình 4.4, trước tiên căn cứ vào vị trí điểm đầu và điểm cuối được nhập, các hạn chế đối với tuyến đường sẽ được đưa vào chương trình, các

hạn chế này bao gồm: Đường bờ biển, các đường đẳng sâu giới hạn các độ sâu phù hợp cho hành trình tàu.

Tiếp đến, mạng lưới các nút (hay không gian tìm kiếm) được xây dựng cho vùng biển từ điểm xuất phát tới điểm đích. Thuật toán vi khuẩn cải tiến sẽ hoạt động trên mạng lưới (hay không gian tìm kiếm) này để tìm tuyến đường chạy tàu tối ưu nhiên liệu just in time, với các số liệu đầu vào là:

- Điều kiện thời tiết tại vị trí cụ thể vào thời điểm cụ thể (điều kiện sóng, gió, dòng chảy được cập nhật theo thời gian thực);

- Đặc tính chuyển động tàu biển (đặc tính thay đổi tốc độ tàu biển) trong điều kiện hành hải cụ thể;

- Đặc tính tiêu thụ nhiên liệu tàu biển trong điều kiện hành hải cụ thể.

Từ các tuyến tốt nhất xác định được (tương ứng với vị trí của các vi khuẩn khỏe nhất), tuyến đường gần với tuyến đường tối ưu sẽ được trả về.

4.3.3. Hàm mục tiêu của tuyến đường chạy tàu tối ưu nhiên liệu dựa trên nguyên tắc just in time "tàu đến cảng kịp lúc"

Mỗi tuyến đường chạy tàu luôn được gắn với một chi phí nhất định gọi là hàm mục tiêu của tuyến đường.

Ví dụ, đối với bài toán hàng hải khí tượng thông thường, yếu tố được quan tâm chính là thời gian hành trình, khi đó hàm mục tiêu của tuyến đường hàng hải khí tượng thông thường chính là thời gian hành trình ngắn nhất (với điều kiện tàu, hàng hóa và sức khỏe thuyền viên được đảm bảo).

Khi yếu tố được quan tâm thay đổi thì hàm mục tiêu của tuyến đường chạy tàu thay đổi, ví dụ: Khi lượng nhiên liệu tiêu thụ được quan tâm thì hàm mục tiêu tuyến đường chính là việc tối ưu hóa nhiên liệu.

Để việc mô phỏng đơn giản và cụ thể, trong phạm vị nghiên cứu của đề tài luận án, NCS lựa chọn hàm mục tiêu của tuyến đường chạy tàu chính là tối ưu nhiên liệu dựa trên nguyên tắc just in time "tàu đến cảng kịp lúc".

Khi đó, hàm mục tiêu của tuyến Q(S) được tính là:

$$Q(S) = T_1 + T_2 + \cdots + T_i + \cdots + T_n \quad (4.6)$$

Trong đó: T_i là lượng nhiên liệu tiêu thụ cần thiết để hoàn thành đoạn thứ i của tuyến.

Bài toán tối ưu cần giải sẽ là: Xác định bộ các nút (hay các WPTs) trên mỗi đường để tổng lượng nhiên liệu tiêu thụ trên tuyến của tàu là nhỏ nhất. Hay xác định vector S* sao cho:

$$Q(S^*) = Min\ Q(S) \quad (4.7)$$

Hàm mục tiêu của tuyến đường chạy tàu tối ưu nhiên liệu dựa trên nguyên tắc just in time được mô tả như sau:

"Public Sub CalculateQualityWithFuel(ByRef TimeMustReach As Double, ByRef targetFuelConsumeTons As Double)

Dim retn As Double = 0

Dim tmpTime As Double = 0

Dim tmpFuel As Double = 0

For I As Integer = 0 To travTime.Length - 1

tmpTime += travTime(I)

*tmpFuel += travTime(I) * fuelTpH(I)*

Next

If tmpTime >= TimeMustReach Then

Quality = tmpTime / TimeMustReach + 9999.9

*ElseIf tmpTime <= 0.85 * TimeMustReach Then*

*Quality = 0.85 + 3 * tmpFuel / targetFuelConsumeTons*

Else

*Quality = tmpTime / TimeMustReach + 3 * tmpFuel / targetFuelConsumeTons*

End If

End Sub"

4.3.4. Thuật toán vi khuẩn tính toán tuyến đường chạy tàu tối ưu nhiên liệu dựa tên nguyên tắc just in time "tàu đến cảng kịp lúc"

Thuật toán vi khuẩn được thực hiện nhờ việc lặp lại các quá trình tìm kiếm cục bộ, kết bầy, sinh sản và triệt tiêu. Trong phần này, NCS giới thiệu một số công thức cơ bản trong thuật toán vi khuẩn để tính toán tuyến đường chạy tàu tối ưu just in time.

4.3.4.1. Khởi tạo tập hợp vi khuẩn (Bacteria Position Initialization)

Mục đích của quá trình này là khởi tạo vị trí ban đầu cho các vi khuẩn trong tập hợp một cách ngẫu nhiên. Mỗi vị trí của một cá thể vi khuẩn S (tương ứng với một nghiệm khả thi trong bài toán tối ưu) là một bộ các điểm nút trên các đường của mạng lưới. Vậy, việc khởi tạo vị trí ban đầu của vi khuẩn chính là lựa chọn một cách ngẫu nhiên các điểm nút với điều kiện trên suốt tuyến hành trình, an toàn của tàu, hàng hóa và sức khỏe thuyền viên được đảm bảo.

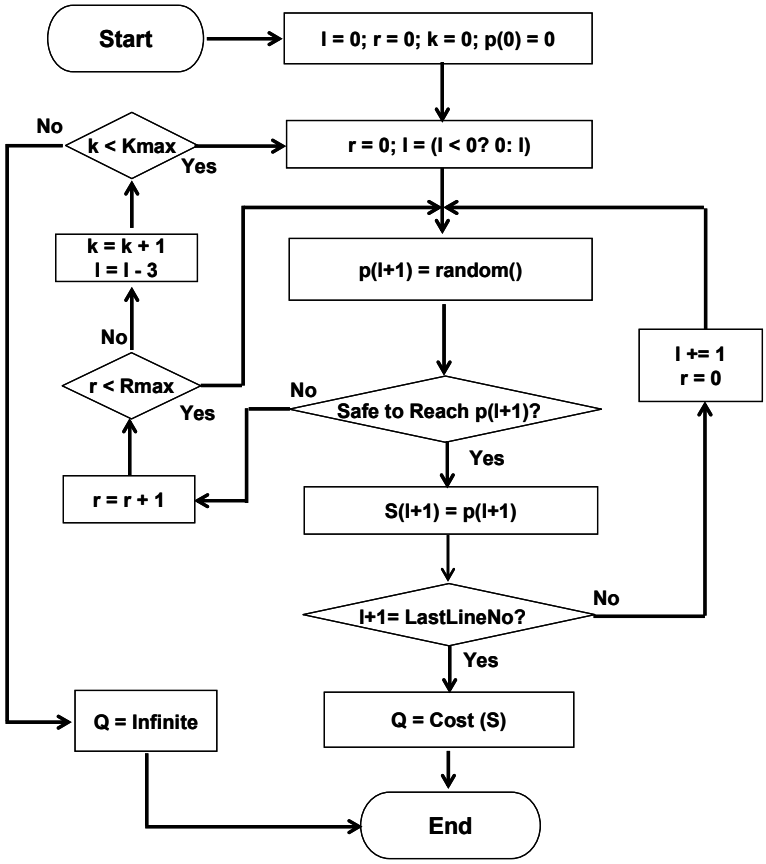
Việc này được thực hiện như minh họa ở sơ đồ trong hình 4.5, trong đó sử dụng 2 thông số thiết kế là:

- Kmax: Tổng số lần thử tối đa để tìm nút tiếp theo;
- Rmax: Số lần thử tối đa để tìm nút tiếp theo từ 1 nút trước đó.

Biến l được sử dụng để đếm số thứ tự của đường mà tàu đã chạy tới và vì vậy, ban đầu, giá trị biến được đặt là 0.

Từ 1 điểm p(l) trên đường thứ l, điểm p(l+1) trên đường thứ (l+1) được lựa chọn một cách ngẫu nhiên và an toàn tàu, hàng hóa được kiểm tra cho hành trình từ điểm p(l) đến điểm p(l+1). Nếu an toàn tàu, hàng hóa được đảm bảo, điểm p(l+1) sẽ được lựa chọn và chương trình sẽ tiếp tục tìm điểm nút tiếp theo từ nút này. Ngược lại, nếu từ p(l) tàu không thể tới p(l+1) một cách an toàn, một nút khác sẽ được chọn thay thế và biến đếm r được tăng lên.

Nếu không thể tìm được điểm nút p(l+1) đảm bảo an toàn sau Rmax lần thử, quy trình khởi tạo sẽ được tiến hành lại từ nút phía trước nút p(l) một số bước.



Hình 4.5 Khởi tạo một tuyến (Route Initialization)

Nếu có thể tìm được một tuyến đường cho tàu tới đích an toàn, tuyến này sẽ được gán cho một cá thể vi khuẩn. Tiếp đó, chi phí của tuyến (hay chỉ số sức khỏe của cá thể vi khuẩn) sẽ được tính toán và sử dụng cho các bước tiếp theo.

4.3.4.2. Di chuyển Chemotaxis

Xuất phát từ một vị trí ứng với một tuyến cho trước, vi khuẩn sẽ tìm các vị trí xung quanh (hay thực hiện các sửa đổi đối với tuyến) sao cho chi phí của tuyến (tức là hàm mục tiêu) giảm đi. Việc này được thực hiện qua các di chuyển của các thể vi khuẩn được gọi là di chuyển Chemotaxis. Hành động này được minh họa theo sơ đồ trong hình 4.6.

Di chuyển Chemotaxis của vi khuẩn là tổng hợp kết quả của 2 hành động: Chuyển động xoay (tumble) và bơi (swim).

Một chuyển động xoay (tumble) của vi khuẩn được thể hiện bằng 1 vector V, thể hiện hướng tìm kiếm của cá thể vi khuẩn. Nói cách khác, đó chính là hướng

thay đổi của một vài phần tử hay một đoạn của tuyến. Vector V được định nghĩa như sau:

$$V = [V(1), V(2), \cdots, V(i), \cdots, V(N)] \quad (4.8)$$

Where

$V(i) = 0$ hoặc $V(i) = \pm 1$

N: Number of grid lines

Vì không gian tìm kiếm của bài toán có số chiều cao (N lớn) nên vector V có thể có rất nhiều giá trị. Trong đề tài này, dựa trên cơ sở xem xét sự thay đổi của tuyến mong muốn, vector V được giới hạn trong tập hợp các vector sau:

$$\begin{aligned}
&V_1 = [0, \cdots, 0, 1, 0, \cdots, 0]\ i.e.\ V(i) = 1\ if\ i = k; V(i) = 0\ otherwise \\
&V_2 = [0, \cdots, 0, 1, 1, 0, \cdots, 0]\ i.e.\ V(i) = 1\ if\ k \le i \le k+1; V(i) = 0\ otherwise \\
&V_3 = [0, \cdots, 0, 1, 1, 1, 0, \cdots, 0]\ i.e.\ V(i) = 1\ if\ k \le i \le k+2; V(i) = 0\ otherwise \\
&V_4 = [0, \cdots, 0, -1, 0, \cdots, 0]\ i.e.\ V(i) = -1\ if\ i = k; V(i) = 0\ otherwise \\
&V_5 = [0, \cdots, 0, -1, -1, 0, \cdots, 0]\ i.e.\ V(i) = -1\ if\ k \le i \le k+1; V(i) = 0\ otherwise \\
&V_6 = [0, \cdots, 0, -1, -1, -1, 0, \cdots, 0]\ i.e.\ V(i) = -1\ if\ k \le i \le k+2; V(i) = 0\ otherwise \\
&V_7 = [0, \cdots, 0, 1, 0, \cdots, 0, -1, 0 \cdots, 0]\ i.e.\ V(i) = 1\ if\ i = k_1; V(i) = -1; if\ i = k_2; V(i) = 0\ otherwise \\
&where\ k, k_1, k_2\ are\ random\ value\ produced\ at\ each\ tumble
\end{aligned} \quad (4.9)$$

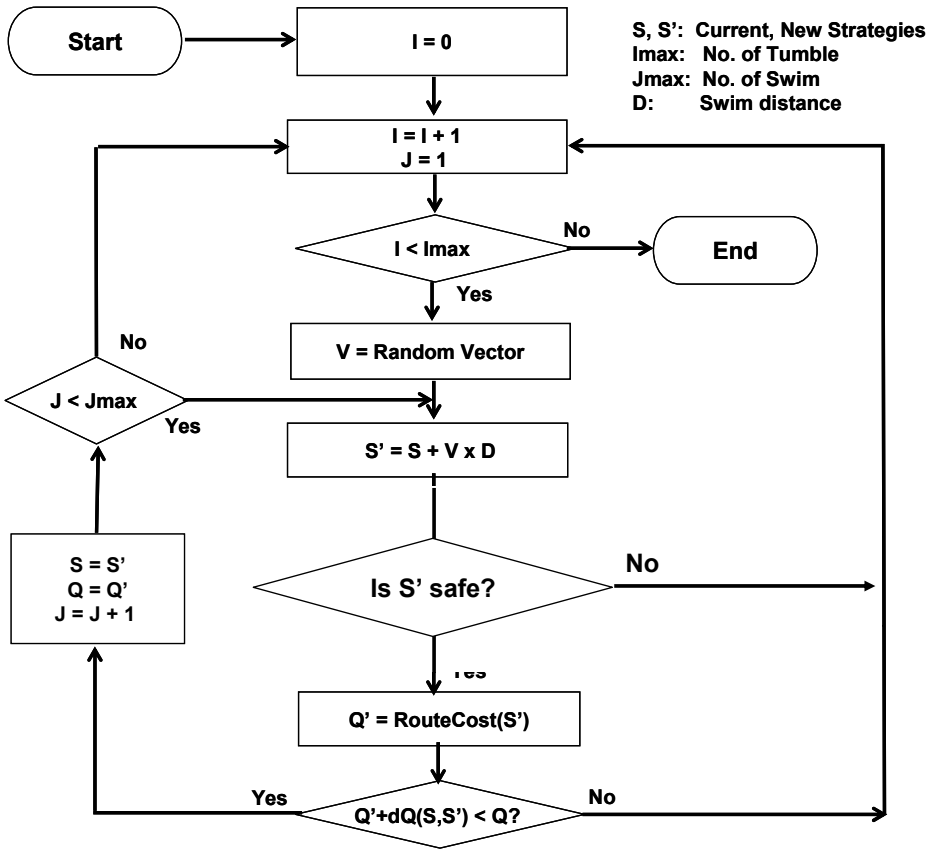
Việc chọn hướng tìm kiếm (tumble) đơn giản chỉ là việc chọn ngẫu nhiên theo xác suất một trong 7 vector trên. Để tăng hiệu quả tìm kiếm, đề tài lựa chọn các giá trị xác suất sao cho các vector V_1, V_4 xuất hiện nhiều hơn so với các vector V_2 and V_5. Đồng thời, các vector V_2 and V_5 xuất hiện thường xuyên hơn các vector còn lại.

Sau mỗi chuyển động xoay (tumble), cá thể vi khuẩn thực hiện một hoặc vài chuyển động bơi (swim). Số chuyển động bơi được thực hiện tùy thuộc vào mức độ thành công trên hướng đã chọn (nếu giá trị chi phí giảm đi sau mỗi chuyển động bơi thì vi khuẩn thực hiện chuyển động bơi theo hướng này số lần lớn hơn). Theo đó, vị trí mới của vi khuẩn sau mỗi chuyển động được xác định theo công thức:

$$S' = S + V \cdot D \quad (4.10)$$

Vị trí mới S’ (hay là tuyến mới) được kiểm tra để đảm bảo rằng trên tuyến này tàu an toàn. Tiếp đó, chi phí trên tuyến mới được so sánh với tuyến đang được lưu trữ. Nếu chi phí thấp hơn, tức là tuyến mới tốt hơn tuyến cũ, cá thể vi khuẩn sẽ được chuyển vị trí sang vị trí mới.

Chú ý rằng ở đây, giá trị chi phí bổ sung dQ được đưa thêm vào công thức, thể hiện sự liên lạc giữa các vi khuẩn ở gần nhau. Ảnh hưởng của dQ sẽ được nêu trong phần tiếp theo.



Hình 4.6 Di chuyển Chemotaxis của vi khuẩn để tối ưu từng bước cho tuyến

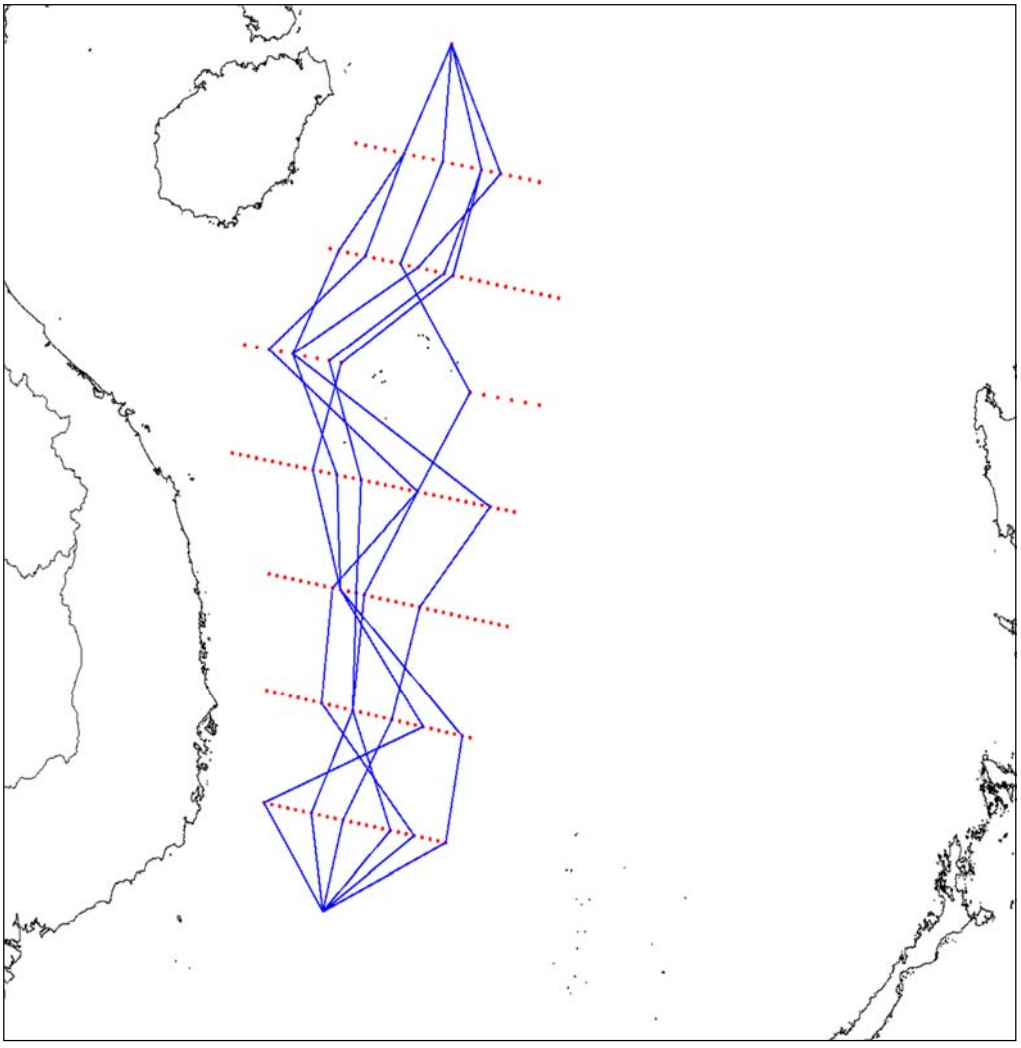
Vi khuẩn sẽ thực hiện di chuyển chemotaxis này lặp đi lặp lại một số lần nhất định. Tuyến đường hàng hải tương ứng với vị trí vi khuẩn, theo đó, cũng tốt dần lên.

Ý nghĩa của việc tìm kiếm cục bộ thông qua các di chuyển Chemotaxis đảm bảo rằng tất cả các tuyến đường có thể đều được kiểm tra.

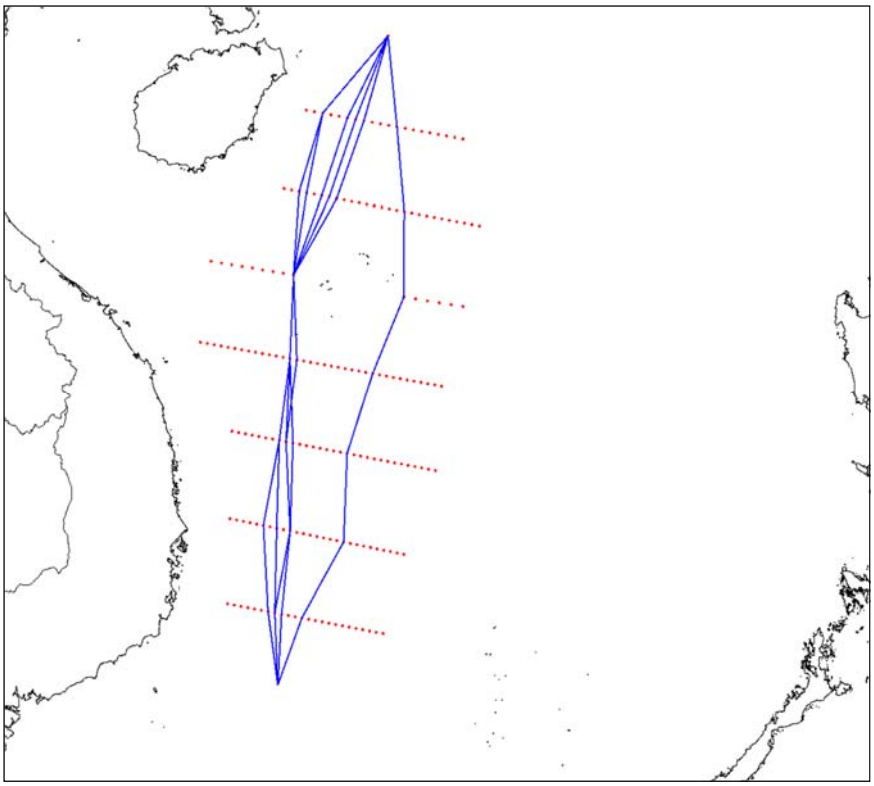
Tuy vậy, sau khi thực hiện các di chuyển Chemotaxis, vi khuẩn chuyển đến các vị trí tương ứng với các tuyến hàng hải tốt hơn rất nhiều. Nếu không có các

quá trình khác như Sinh sản (Reproduction), Triệt tiêu (Elimination) hoặc Phân tán (Dispersal) sau một số đủ lớn các bước di chuyển chemotaxis, nhiều khả năng vi khuẩn sẽ ở vị trí ứng với các giá trị tối ưu cục bộ.

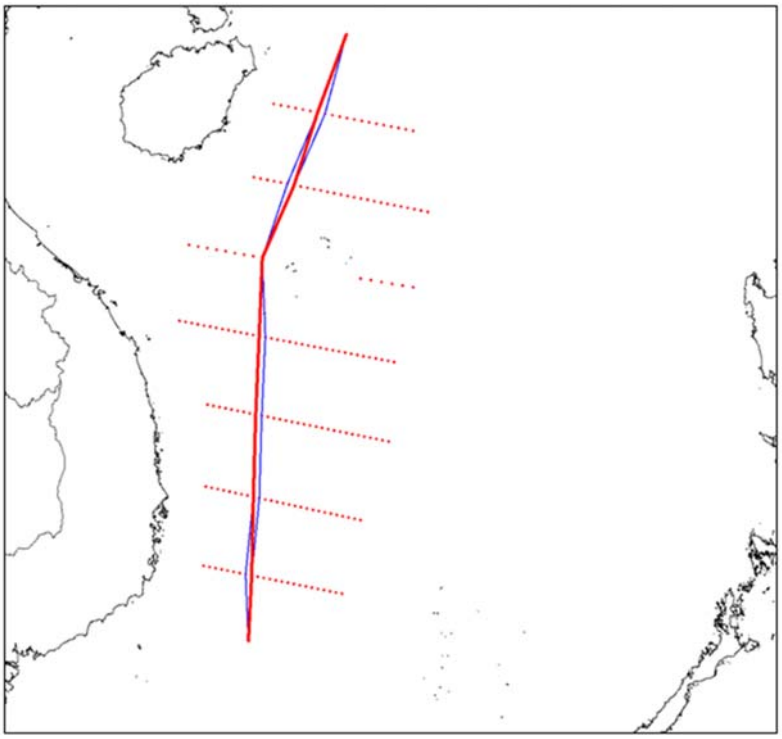
Quá trình lựa chọn (thiết lập) tuyến đường chạy tàu tối ưu (đường đi có lợi nhất) bằng thuật toán vi khuẩn cải tiến được mô tả qua các hình vẽ sau:



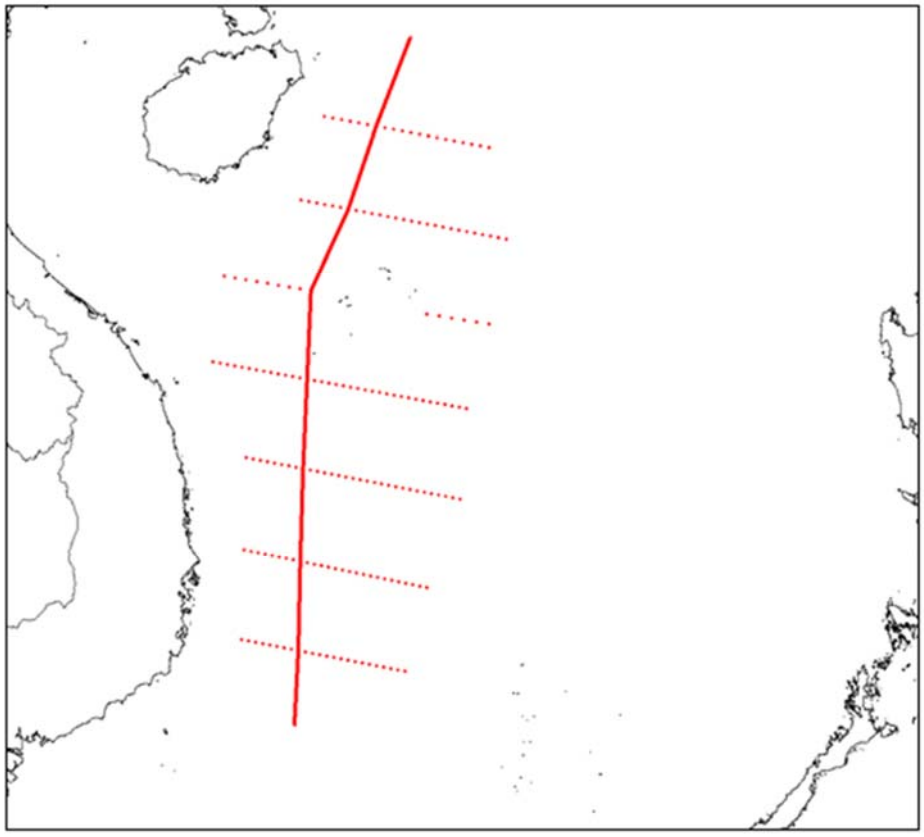
Hình 4.7. Một số tuyến được khởi tạo ngẫu nhiên trong không gian tìm kiếm



Hình 4.8. Các tuyến ngẫu nhiên sau khi được chỉnh sửa trong không gian tìm kiếm



Hình 4.9 Tuyến ngẫu nhiên sau vòng lặp thứ i



Hình 4.10 Tuyến ngẫu nhiên được lựa chon sau n vòng lặp
(tuyến đường có lợi nhất)

4.3.5. Một số điều chỉnh (cải tiến) để tăng hiệu quả lựa chọn của thuật toán vi khuẩn

4.3.5.1. Thay đổi chiều dài bước di chuyển (Swim length)

Việc lựa chọn chiều dài bước di chuyển chemotaxis có ý nghĩa quan trọng bậc nhất đối với hiệu quả tìm kiếm của thuật toán vi khuẩn, bởi vì:

- Bước di chuyển dài sẽ làm cho vi khuẩn có thể tiến nhanh tới miền chứa giá trị tối ưu. Tuy nhiên, khi vi khuẩn đã bắt đầu tiếp cận giá trị này, nếu tiếp tục thực hiện các bước di chuyển dài thì vi khuẩn sẽ "nhảy" quanh giá trị tối ưu thay vì tiến tới gần giá trị này hơn;

- Ngược lại, nếu bước di chuyển ngắn, cần nhiều thời gian để vi khuẩn có thể tiếp cận miền chứa giá trị tối ưu. Tuy nhiên, khi đã ở gần giá trị này, bước di chuyển ngắn giúp vi khuẩn có thể tiến tới giá trị tối ưu một cách chắc chắn.

Một nhược điểm nữa cần kể tới của bước di chuyển ngắn là vi khuẩn hầu như không có khả năng thoát ra ngoài vùng hấp dẫn của các giá trị tối ưu cục bộ.

Từ những điều này, để việc tìm tuyến đường hàng hải tối ưu nhiên liệu just in time bằng thuật toán vi khuẩn có hiệu quả, NCS sử dụng một cơ chế thay đổi chiều dài bước di chuyển chemotaxis như sau:

- Bước 1: Đặt giá trị độ dài bước di chuyển lớn (large Swim length – d_{large}) để vi khuẩn có thể nhanh chóng tiếp cận miền tối ưu;

- Bước 2: Nếu không thực hiện được bước di chuyển sau một số lần thử chọn trước, bước di chuyển được đưa về giá trị trung bình d_{medium} để vi khuẩn có thể tiếp tục tiến tới miền tối ưu;

- Bước 3: Nếu việc thử với bước di chuyển trung bình không thành công sau một số lần thử, giảm tiếp bước di chuyển tới giá trị nhỏ hơn d_{small};

- Bước 4: Nếu không tìm được vị trí tốt hơn cho vi khuẩn sau 1 số lần thử, chyển trở lại bước 1 để loại trừ khả năng vi khuẩn rơi vào vùng hấp dẫn của 1 giá trị tối ưu cục bộ;

Nhờ sự thay đổi độ dài bước di chuyển này, các vi khuẩn có khả năng tiến tới vị trí ứng với giá trị tối ưu một cách nhanh chóng và bền vững. Đồng thời, vi khuẩn cũng có khả năng thoát khỏi vùng hấp dẫn của các giá trị tối ưu cục bộ.

4.3.5.2 Trao đổi thông tin giữa các cá thể vi khuẩn

Mục tiêu của việc thực hiện trao đổi thông tin giữa các cá thể vi khuẩn là để cho các cá thể vi khuẩn có thể hợp tác nhằm tăng hiệu quả của việc tìm kiếm giá trị tối ưu.

Trong đề tài luận án này, khoảng cách giữa 2 cá thể vi khuẩn S_1 và S_2 được định nghĩa như sau:

$$S_1 = [P_1(1), P_1(2), \cdots, P_1(i), \cdots, P_1(N)]$$
$$S_2 = [P_2(1), P_2(2), \cdots, P_2(i), \cdots, P_2(N)]$$
$$\textit{then} \qquad (4.11)$$

$$d_{S_1-S_2} = \sqrt{\sum_{i=1}^{N}(P_1(i) - P_2(i))^2}$$

Các vi khuẩn được gọi là ở gần (neighbors) nếu khoảng cách giữa chúng nhỏ hơn một giá trị do ta chọn. Đương nhiên, để việc tìm kiếm hiệu quả, ta không mong muốn các vi khuẩn chiếm cùng một vị trí hoặc ở vị trí quá gần nhau. Vì vậy, giá trị chi phí bổ xung (dQ) được vi khuẩn sử dụng để quyết định việc có chuyển tới vị trí ứng với tuyến mới là S' hay không. dQ được tính như sau:

Let

$\{S_1, S_2, ..., S_M\}$ *to be the set of neighbor bacteria of* S

$\{d_1, d_2, ..., d_M\}$ *are distances from S to its neighbor bacteria*

$\{Q_1, Q_2, ..., Q_M\}$ *are routed*$_1, d_2, ..., d_M\}$ *are costs coresponding to* S_i

Define

$$dQ(S) = \sum_{k=1}^{M}\left[-Q_{expellant}\left(1-\frac{d_k}{d_{\max}}\right)\right]$$

Similarly,

$$dQ(S') = \sum_{k=1}^{M'}\left[-Q_{expellant}\left(1-\frac{d'_k}{d_{\max}}\right)\right] \qquad (4.12)$$

Then

$$dQ(S,S') = dQ(S') - dQ(S)$$

$$if\ (Q_{S'} < Q_S)\ and\ (Q_{S'} < Q_i(for\ all\ i = 1\ to\ M))\ then\ dQ(S,S') = 0$$

Giá trị d_{max} đặc trưng cho một vùng được gọi là vùng đẩy nhau (expellant region) xung quanh mỗi cá thể vi khuẩn và $Q_{expellant}$ là lực đẩy lớn nhất. Cơ chế này cho phép các vi khuẩn phân bổ xung quanh tuyến tối ưu, nhờ vậy tuyến này có thể được xác định nhanh chóng. Đồng thời, lực đẩy này giúp tập hợp vi khuẩn không bị hút hết vào một vùng tối ưu cục bộ.

4.3.6. Tổng thể thuật toán xác định tuyến đường tối ưu nhiên liệu dựa trên nguyên tắc just in time "tàu đến cảng kịp lúc"

Vận dụng các phân tích và phương pháp được trình bày ở trên, NCS đã xây dựng chương trình xác định tuyến đường hàng hải tối ưu nhiên liệu just in time cho tàu dựa trên thuật toán vi khuẩn cải tiến bằng ngôn ngữ lập trình VB 2010.

Kỹ thuật lập trình không phải là mối quan tâm chủ yếu của để tài này, vì vậy chương trình được viết chưa hẳn là tối ưu về tốc độ tính toán hay về việc sử dụng tài nguyên máy tính. Tuy nhiên, chương trình vẫn đảm bảo tuyến đường được tính toán hiệu quả và nhanh chóng (trong phạm vi vài chục giây).

Thuật toán tổng thể có thể được viết như sau (dưới dạng code giả - pseudo-code):

A. Initialization

Initialize_Grid (N_line, N_point, D_point);

For *bac = 1* ***to*** *Ns*

Initialize_Bacterium(B(bac));

Next *bac*

B. Evolution

For *cycles = 1* ***to*** *N_cyc*

For *bac = 1* ***to*** *Ns*

For *chemo = 1* ***to*** *Nc*

Perform_Chemotatic_Move(B(bac));

Next *chemo*

If (Number_of_ Unsuccessful_ Move N_size_converted_to_[large/medium/small]) then

Convert_move_length_from_[large/medium/small]to[medium/small/large]();

End if

Next *bac*

Sort_the_Bacteria_and_Cost_Arrays_by_Ascending_RouteCost(B(Ns), Q(Ns));

For *die_no = 1* ***to*** *Nr*

If (Chemotatic_Move_of_Bacterium_Count(B(die_no))> N_steps_to_die) then (*)

Kill_bacterium(B(die_no));

B(die_no) = Reprocude_Bacterium (B (Ns - die_no));

End if

***Next** die_no*

***For** disperse_no = 1 **to** Nd*

rand = produce_random_interger ()

Initialize_Bacterium(B(rand));

***Next** disperse_no*

***Next** cycles*

C. Termination

Sort_the_Bacteria_and_Cost_Arrays_by_Ascending_RouteCost(B(Ns), Q(Ns));

Return B (1);

Trong đó, các thông số thiết kế của thuật toán có thể được định nghĩa và giải thích như sau:

N_line: Số đường (gồm các nút) trên mạng lưới

N_point: Số nút trên một đường

D_point: Khoảng cách giữa các nút trên đường

N_cyc: Số vòng lặp của thuật toán (hay số thế hệ vi khuẩn được sử dụng trong thuật toán)

Ns: Số vi khuẩn trong tập hợp vi khuẩn

B(Ns): Tập hợp vi khuẩn

Q(Ns): Mảng các giá trị chi phí ứng với mỗi cá thể vi khuẩn

Nr: Số vi khuẩn bị chết trong mỗi vòng lặp (cũng chính là số vi khuẩn thực hiện quá trình sinh sản)

Nd: Số vi khuẩn bị triệt tiêu - eliminated (cũng bằng số vi khuẩn được phân tán – dispersed) trong mỗi chu kỳ

Nc: Số di chuyển chemotaxis của một cá thể vi khuẩn trong mỗi vòng lặp

N_size_converted_to_large: Số lần thử di chuyển không thành công liên tục trước khi bước di chuyển được chuyển từ nhỏ (d_small) sang lớn (d_large)

N_size_converted_to_medium: Số lần thử di chuyển không thành công liên tục trước khi bước di chuyển được chuyển từ lớn (d_large) sang trung bình (d_medium)

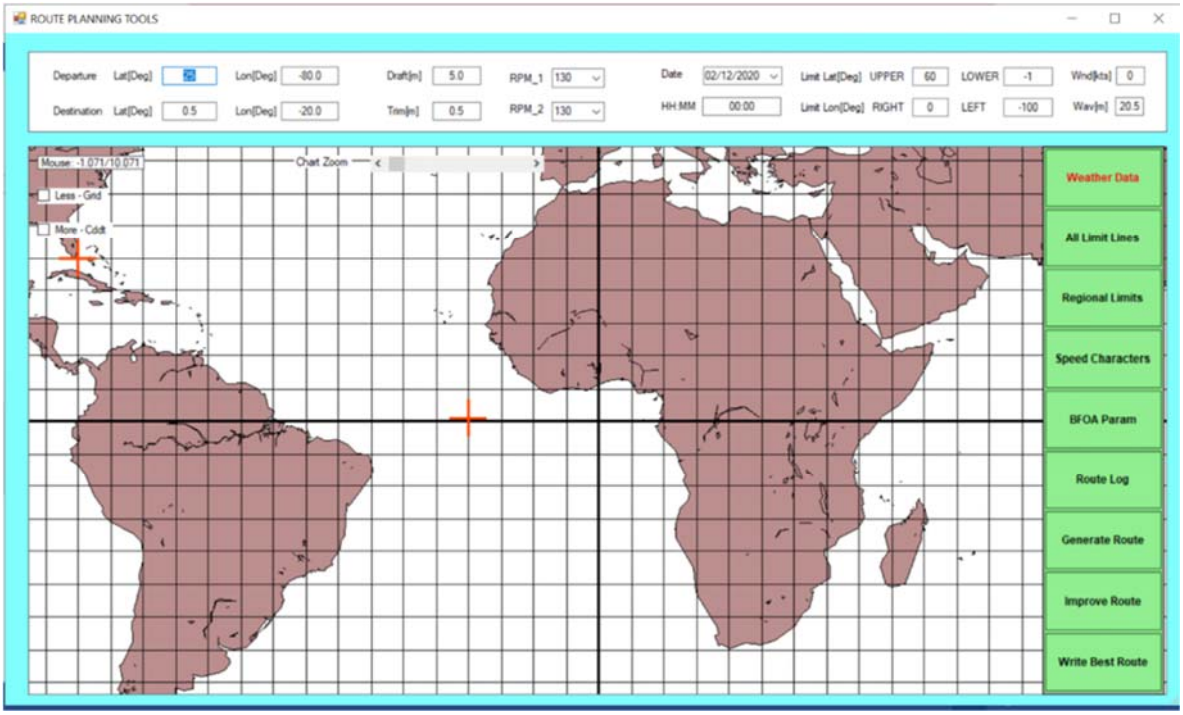
N_size_converted_to_small: Số lần thử di chuyển không thành công liên tục trước khi bước di chuyển được chuyển từ trung bình (d_medium) sang nhỏ (d_small)

N_steps_to_die: Số bước di chuyển một cá thể vi khuẩn cần thực hiện tới khi vi khuẩn được gọi là trưởng thành

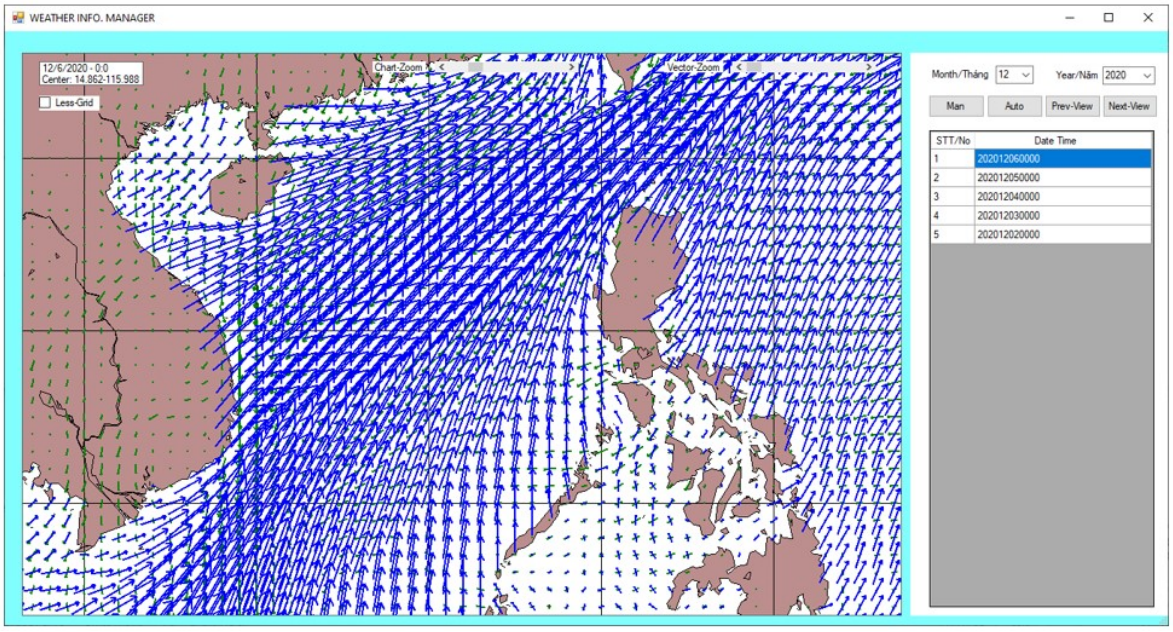
Lưu ý rằng vi khuẩn không "Chết" trước khi nó đã "trưởng thành", tức là ta cho vi khuẩn thực hiện việc tìm kiếm giá trị tối ưu cục bộ ít nhất một số lần nhất định trước khi loại bỏ vi khuẩn đó. Điều này cho phép vùng tìm kiếm quanh vi khuẩn được khai thác triệt để trong quá trình tìm kiếm giá trị tối ưu.

4.4. Xây dựng phần mềm tính toán và mô hình mô phỏng kết quả tính toán tuyến đường chạy tàu tối ưu nhiên liệu dựa trên nguyên tắc just in time "tàu đến cảng kịp lúc" bằng thuật toán vi khuẩn cải tiến

Hình 4.9 Giao diện đăng nhập phần mềm tính toán tuyến đường chạy tàu tối ưu just in time bằng thuật toán vi khuẩn cải tiến



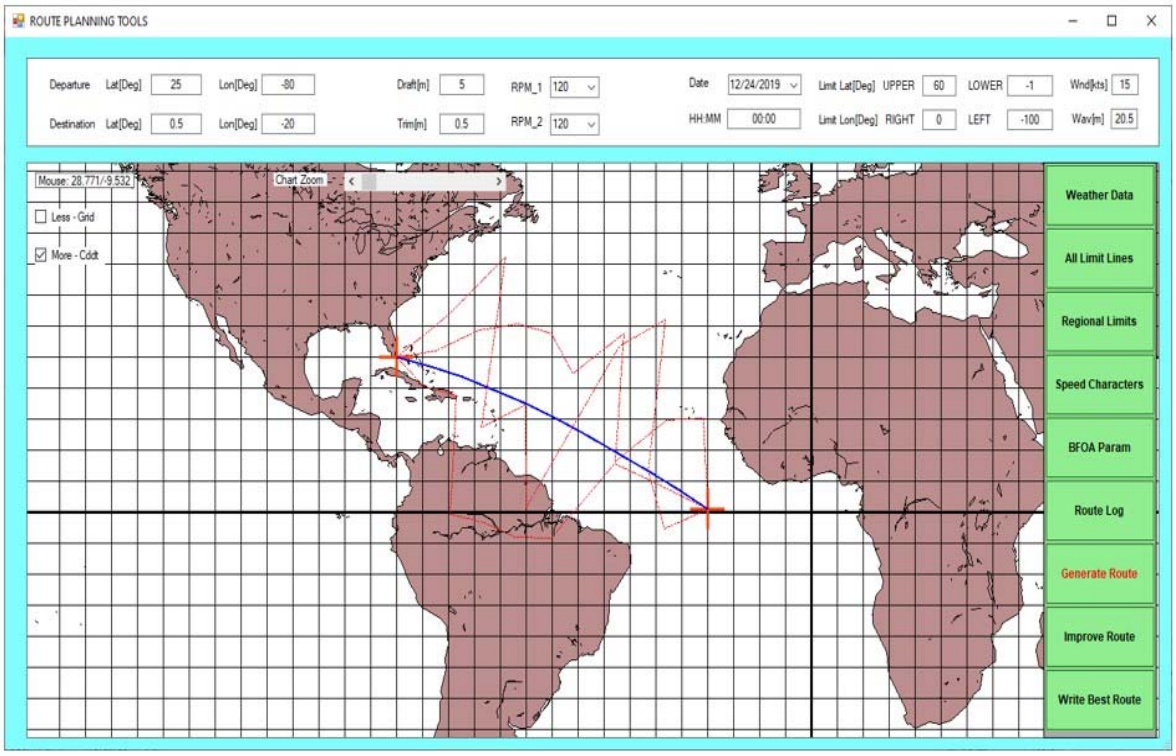
Hình 4.10. Giao diện phần mềm tính toán tuyến đường chạy tàu tối ưu just in time bằng thuật toán vi khuẩn cải tiến



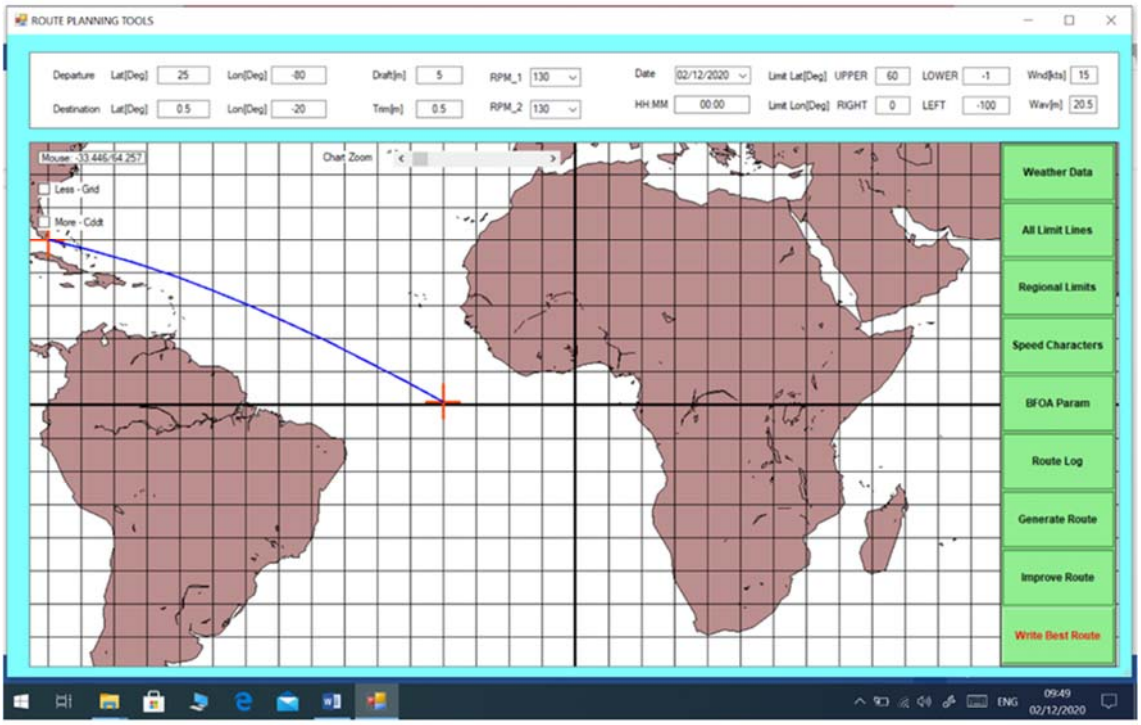
Hình 4.11 Giao diện phần mềm khi cập nhật thông tin thời tiết dạng số

Sử dụng phần mềm tính toán tuyến đường chạy tàu tối ưu nhiên liệu dựa trên nguyên tắc just in time bằng thuật toán vi khuẩn cải tiến, người Sỹ quan Hàng hải cần:

- Chọn các chế độ vòng tua máy (vòng quay chân vịt - rpm);
- Nhập các hạn chế thời tiết;
- Nhập các vĩ độ, kinh độ giới hạn;
- Nhập thời gian yêu cầu tàu đến;
- Nhập các đường hạn chế;
- Chạy phần mềm tính toán tuyến đường;
- Kiểm tra tuyến đường tính toán, tính lại để nâng cao chất lượng (nếu cần);
- Kiểm tra để thay đổi các điều kiện giới hạn nếu thời gian tàu đến không đáp ứng yêu cầu.



Hình 4.12 Giao diện phần mềm mô tả các tuyến đường ngẫu nhiên được tính toán trong không gian tìm kiếm



Hình 4.13 Giao diện phần mềm tuyến đường chạy tàu tối ưu nhiên liệu just in time "tàu đến cảng kịp lúc" được tính toán

4.5. Kết luận chương 4

Kết thúc chương 4, NCS đạt được các mục tiêu nghiên cứu chính sau:

- Nghiên cứu kỹ thuật toán vi khuẩn cổ điển và các cải tiến của thuật toán vi khuẩn để ứng dụng hiệu quả cho nhiều lĩnh vực khác nhau;

- Xây dựng hàm mục tiêu tuyến đường chạy tàu tối ưu nhiên liệu dựa trên nguyên tắc just in time "tàu đến cảng kịp lúc"

- Xây dựng sơ đồ tổng quát tính toán tuyến đường chạy tàu tối ưu nhiên liệu dựa trên nguyên tắc just in time;

- Xây dựng thuật toán BFO cải tiến (thích nghi) tính toán tuyến đường chạy tàu tối ưu nhiên liệu dựa trên nguyên tắc just in time;

- Xây dựng phần mềm tính toán và mô hình mô phỏng kết quả tính toán tuyến đường chạy tàu tối ưu nhiên liệu dựa trên nguyên tắc just in time.

KẾT LUẬN VÀ HƯỚNG NGHIÊN CỨU TIẾP THEO

Kết luận

Đề tài luận án tiến sĩ "Nghiên cứu xây dựng thuật toán ngẫu nhiên tính toán tuyến đường và kế hoạch chạy tàu tối ưu trên cơ sở ảnh hưởng của các yếu tố thời tiết" nhằm tăng hiệu quả sử dụng năng lượng và giảm phát thải khí nhà kính từ tàu biển là một đề tài mang tính thời sự, cấp thiết và thực tiễn cao. Sản phẩm nghiên cứu của đề tài luận án có ý nghĩa khoa học, thực tiễn và thể hiện được tính mới của một đề tài luận án tiến sĩ.

NCS trình bày đề tài luận án mạch lạc; bố cục đề tài chặt chẽ, logic; nội dung đề tài đầy đủ, rõ ràng; tài liệu tham khảo trích dẫn trung thực.

Kết thúc đề tài luận án, NCS đạt được một số kết quả nghiên cứu chính như sau:

(1) Xây dựng phần mềm thu thập và xử lý thông tin thời tiết phục vụ tính toán tuyến đường và kế hoạch chạy tàu tối ưu, cụ thể:

- Trích xuất, tổng hợp và phân tích bản tin sóng toàn cầu, bản tin gió toàn cầu được định dạng Grib 2 từ cơ sở dữ liệu của Viện nghiên cứu phát triển bền vững khí quyển nhân loại thuộc Đại học Kyoto, Nhật Bản (gọi tắt là RISH – Research, Institute for Sustainable Humanoshere);

- Trích xuất, tổng hợp và phân tích dữ liệu dòng chảy đại dương theo thời gian thực được định dạng netCDF từ cơ sở dữ liệu của Dự án nghiên cứu, phân tích dòng chảy đại dương theo thời gian thực, Phòng thí nghiệm sức đẩy phản lực, Viện Công nghệ California, Mỹ (gọi tắt là OSCAR – Ocean Surface Current Analysis Real-time).

(2) Ứng dụng phương pháp bình phương nhỏ nhất và ngôn ngữ lập trình Visual basic xây dựng phần mềm tổng hợp, phân tích đặc tính thay đổi tốc độ và đặc tính nhiên liệu tàu biển trong từng điều kiện hành hải cụ thể phục vụ tính toán tuyến đường và kế hoạch chạy tàu tối ưu.

- NCS xác định đặc tính thay đổi tốc độ và đặc tính tiêu thụ nhiên liệu tàu biển trong từng điều kiện hành hải cụ thể bằng phương pháp bình phương nhỏ nhất;

- NCS xây dựng phần mềm tổng hợp, phân tích đặc tính thay đổi tốc độ và đặc tính tiêu thụ nhiên liệu tàu biển trong từng điều kiện hành hải cụ thể phục vụ tính toán tuyến đường và kế hoạch chạy tàu tối ưu.

(3) Nghiên cứu xây dựng thuật toán vi khuẩn cải tiến để tính toán tuyến đường chạy tàu tối ưu nhiên liệu dựa trên nguyên tắc just in time "tàu đến cảng kịp lúc".

- NCS xây dựng hàm mục tiêu cho bài toán tính toán tuyến đường chạy tàu tối ưu nhiên liệu dựa trên nguyên tắc just in time "tàu đến cảng kịp lúc";

- NCS xây dựng sơ đồ tổng quát cho bài toán tính toán tuyến đường chạy tàu tối ưu nhiên liệu dựa trên nguyên tắc just in time "tàu đến cảng kịp lúc";

- NCS đưa ra một số cải tiến (điều chỉnh) cho thuật toán vi khuẩn để tăng hiệu quả lựa chọn tuyến đường tối ưu nhiên liệu dựa trên nguyên tắc just in time "tàu đến cảng kịp lúc";

- NCS xây dựng phần mềm tính toán và mô hình mô phỏng kết quả tính toán tuyến đường chạy tàu tối ưu nhiên liệu dựa trên nguyên tắc just in time "tàu đến cảng kịp lúc".

Hướng nghiên cứu tiếp theo

Một số vấn đề NCS dự kiến cần nghiên cứu thêm, gồm:

- Xây dựng cơ chế trao đổi thông tin về tuyến đường chạy tàu khuyến cáo tối ưu nhiên liệu dựa trên nguyên tắc just in time "tàu đến cảng kịp lúc" giữa tàu và bờ (trích xuất các điểm chuyển hướng và chế độ máy trên từng đoạn tuyến);

- Nghiên cứu, xác định điều kiện sóng tới hạn (độ cao sóng tới hạn) để đảm bảo an toàn cho tàu, hàng hóa và con người;

- Nghiên cứu, thử nghiệm, đánh giá thực tế các sản phẩm nghiên cứu của đề tài luận án tiến tới áp dụng hiệu quả cho đội tàu biển Việt Nam.

DANH MỤC CÁC CÔNG TRÌNH KHOA HỌC ĐÃ CÔNG BỐ LIÊN QUAN ĐẾN ĐỀ TÀI LUẬN ÁN

1. NCS. Đặng Quang Việt, TS. Nguyễn Thanh Sơn. *Xây dựng hệ thống hỗ trợ hàng hải tối ưu cho đội tàu biển Việt Nam nhằm tăng hiệu quả sử dụng năng lượng và giảm phát thải khí nhà kính từ tàu biển*. Tạp chí Giao thông vận tải, ISSN 2354 - 0818, Số tháng 9/2021, tr.132 - 135;
2. NCS. Đặng Quang Việt, TS. Nguyễn Thanh Sơn. *Xây dựng chương trình khai thác bản tin gió, sóng của Rish và bản tin dòng chảy của Oscar phục vụ tính toán phương án chạy tàu tối ưu*. Tạp chí Giao thông vận tải, ISSN 2354-0818, Số tháng 10/2021, tr. 127 - 131;
3. NCS. Đặng Quang Việt, TS. Nguyễn Thanh Sơn. *Nghiên cứu, xây dựng phần mềm theo dõi, đánh giá đặc tính thay đổi tốc độ tàu biển phục vụ xây dựng hệ thống hỗ trợ hàng hải tối ưu hoạt động cho đội tàu biển Việt Nam.* Tạp chí Giao thông vận tải, ISSN 2354-0818, Số tháng 11/2021, tr. 110 - 114;
4. Đang Quang Viet, Nguyen Minh Đuc & Phan Van Hung. *Assessment of potentially cutting GHG emissions from shipping in relation to energy to energy consuption trends using Fuzzy Analytic Hierarchy Process.* Australian Journal of Maritime & Ocean Affairs, Published online: 08 Nov 2021.
5. Phan Van Hung, Đang Quang Viet, Le Thanh Đat. *Optimal weather routing based on adaptive bacterial foraging algorithm for vessel.* Journal of Technology & Innovation (JTIN), ISSN: 2773-6202 (Online), Available online: 20 May 2022.

6. Phan Van Hung, Đang Quang Viet, Nguyen Minh Đuc, Le Thanh Đat. *Ship routing optimization using bacterial foraging optimization algorithm for safety and efficient navigation.* International Journal of Electrical and Computer Engineering, Vol 13, No 2: April 2023.

DANH MỤC TÀI LIỆU THAM KHẢO

I. TÀI LIỆU THAM KHẢO TIẾNG VIỆT

[1] Nghị quyết số 09 – NQ/TƯ, ngày 09/02/2007 về chiến lược biển Việt Nam đến năm 2020;

[2] Nghị quyết số 36 – NQ/TƯ, ngày 22/10/2018 về chiến lược phát triển bền vững kinh tế biển Việt Nam đến năm 2030, tầm nhìn đến 2045 nhằm đưa Việt Nam trở thành quốc gia mạnh về biển, giàu từ biển;

[3] Bộ luật Hàng hải Việt Nam 2015;

[4] Thông tư số 40/2018/TT – BGTVT, ngày 29/06/2018 "Quy định về thu thập và báo cáo tiêu thụ nhiên liệu của tàu biển Việt Nam";

[5] QCVN 26/2018/BGTVT "Quy chuẩn kỹ thuật quốc gia về các hệ thống ngăn ngừa ô nhiễm biển của tàu" và Thông tư số 09/2019/TT – BGTVT, ngày 31/03/2019 "Ban hành quy chuẩn kỹ thuật quốc gia về các hệ thống ngăn ngừa ô nhiễm tàu biển của tàu";

[6] Khoa Hàng hải, Trường Đại học Hàng hải Việt Nam. "*Giáo trình Vô tuyến điện 3*", 2020;

[7] Khoa Hàng hải, Trường Đại học Hàng hải Việt Nam. "*Giáo trình địa văn Hàng hải III*";

[8] Khoa Hàng hải, Trường Đại học Hàng hải Việt Nam. "*Tài liệu học tập khí tượng hải dương*";

[9] TS. Nguyễn Minh Đức, Đại học Hàng hải Việt Nam, giáo trình toán chuyên đề. "*Phương pháp bình phương nhỏ nhất*", 2013;

[10] TS. Nguyễn Minh Đức, Đại học Hàng hải Việt Nam, đề tài NCKH cấp trường. "*Xây dựng chương trình tính toán đường đi tối ưu dựa trên các thông tin thời tiết*", 2013;

[11] NCS. Đặng Quang Việt, TS. Nguyễn Thanh Sơn. "*Xây dựng hệ thống hỗ trợ hàng hải tối ưu cho đội tàu biển Việt Nam nhằm tăng hiệu quả sử dụng năng lượng và giảm phát thải khí nhà kính từ tàu biển"*. Tạp chí Giao thông vận tải, ISSN 2354 - 0818, Số tháng 9/2021, tr.132 - 135;

[12] Phạm Ngọc Hà, Nguyễn Minh Đức. *"Tổng hợp thông tin thời tiết để dự đoán sự trôi dạt của vật thể trong Tìm kiếm cứu nạn"*. Tạp chí Khoa học – Công nghệ Hàng hải, ISSN 1859 – 316X, số 51 – 8/2017, tr. 31 – 35;

[13] Phạm Ngọc Hà, Trần Hải Triều, Bùi Duy Tùng, Nguyễn Minh Đức. *"Thuật toán vi khuẩn sửa đổi tính toán phương án tìm kiếm tối ưu trên biển cho một tàu tìm cứu"*. Tạp chí Khoa học – Công nghệ Hàng hải, ISSN 1859 – 316X, số 57 – 1/2019, tr. 31 – 35;

[14] Ths. Nguyễn Thị Huệ, Bộ môn Cơ bản cơ sở, Đại học Dân lập Hải Phòng, bài báo khoa học. "*Phương pháp bình phương cực tiểu*", 2013;

[15] Phạm Ngọc Hà, Bùi Thị Phương Thảo. *"Nghiên cứu ứng dụng thuật toán vi khuẩn xây dựng tuyến đường phối hợp tìm kiếm tối ưu cho nhiều tàu Tìm kiếm cứu nạn"*. Tạp chí Giao thông vận tải, ISSN 2354 – 0818, số 10/2019, tr. 106 -109;

[16] NCS. Đặng Quang Việt, TS. Nguyễn Thanh Sơn. *"Xây dựng chương trình khai thác bản tin gió, sóng của Rish và bản tin dòng chảy của Oscar phục vụ tính toán phương án chạy tàu tối ưu"*. Tạp chí Giao thông vận tải, ISSN 2354-0818, Số tháng 10/2021, tr. 127 - 131;

[17] NCS. Đặng Quang Việt, TS. Nguyễn Thanh Sơn. *"Nghiên cứu, xây dựng phần mềm theo dõi, đánh giá đặc tính thay đổi tốc độ tàu biển phục vụ xây dựng hệ thống hỗ trợ hàng hải tối ưu hoạt động cho đội tàu biển Việt Nam"*. Tạp chí Giao thông vận tải, ISSN 2354-0818, Số tháng 11/2021, tr. 110 - 114;

[18] NCS, Phạm Ngọc Hà, Trường ĐH Giao thông vận tải, Thành phố Hồ Chí Minh, Luận án tiến sĩ. "*Nghiên cứu lý thuyết dự đoán quỹ đạo trôi dạt và tính toán tuyến đường tối ưu cho phương tiện gặp nạn trong vùng biển Ninh Thuận – Kiên Giang*";

[19] PGS, TS. Nguyễn Minh Đức, Trường Đại học Hàng hải Việt Nam. *"Xây dựng giải pháp tổ chức, quản lý, khai thác Vận tải biển theo hướng tiết kiệm năng lượng, giảm phát thải phù hợp với quy định phụ lục VI, Công ước MARPOL"*. Dự án bảo vệ môi trường cấp Bộ, mã số: GG 191003, 2020;

II. TÀI LIỆU THAM KHẢO TIẾNG ANH

[20] C. Ying et al., "A Fast Bacterial Swarming Algorithm for high-dimensional function optimization", IEEE World Congress on Computational Intelligence, pp. 3135-3140, 2008;

[21] H. Chen, Y. Zhu, K. Hu, "Adaptive Bacterial Foraging Optimization", available at "http://www.hindawi.com/journals/aaa/2011/108269;

[22] K. M. Passino, "Biomimicry of bacterial foraging for distributed optimization and control", IEEE Control Systems Magazine, Vol. 22, pp.52–67, 2002;

[23] LauraWalther, Anisa Rizvanolli, Mareike Wendebourg, Carlos Jahn. *"Modeling and Optimization Algorithms in Ship Weather Routing"*, International Journal of e_Navigation and Maritime Economy, Vol 4, pp. 31 – 45, 2016;

[24] M.D. Nguyen et al. "*Multi-Scale Automatic Route Planning Algorithms for Sea-Going Vessel"*, AMFUF 2013;

[25] Y. Liu and K. M. Passino, *"Biomimicry of social foraging bacteria for distributed optimization: Models, principles, and emergent behaviors"*, Journal of Optimization Theory and Applications, Vol. 115, pp. 603–628, 2002;

[26] Raphael Zaccone, Massimo Figari, Michele Martelli. *"An Optimization Tool For Ship Route Planning In Real Weather Scenarios"*, International Offshore and Polar Engineering Conference (ISOPE), pp. 738 – 744, 2018;

[27] Rohrs, J, Christensen, K, Hole, L, Brostrom, G, Drivdal, M, Sundby, S, 2012. *"Observation based evaluation of Surface wave effects on currents and trajectory forcasts"*. To appears in Ocean Dynam, 14 pp, doi: 10.1007/s 10236 – 012 – 0576 – y;

[28] E. Kobayashi, T. Asajima & N. Sueyoshi. "*Advanced Navigation Route Optimization for an Oceangoing Vessel*";

[29] Hanssen, G.L, James, R.W. "*Optimum Ship Routing*". The Institute of Navigation, 13 (1960);

[30] Papadakis, N.A. "*On the Minimal Time Vessel Routing Problem*", Ph. D, Thesis, Department of Naval Architecture and Marine Engineering. The Uni of Michigan, Ann Arbor Mich (1988);

[31] Parkis, A.N, Papadakis, N.A. "*New Models for Minimal Time Ship Weather Routing*", SNAM Transaction, 96 (1988);

[32] "ENCs, ECDIS & S_100". International Hydrographic Organization;

[33] Admiralty List of Radio Signals, Vol 5, NP 283;

[34] Admiralty List of Radio Signal, Vol 3, NP 285;

[35] C.S. Nilsson, Department of Defence Science and Technology Organization, Australia. "*The effect of weather on a ship's speed*";

[36] Christiaan Heij, Sabine Knapp, Econometric Institute, Erasmis University Rotterdam, Econometric Institute Report 2014-15. "*Effects of wind strength and wave height on ship incident risk: Regional trends and seasonality*";

[37] Haper H.L. (1974-1976). "*The method of least squares and some alternatives*";

[38] Plackett, R.L. (1972). *"The discovery of the method of least squares"*;

[39] Marco Dorigo and Thomas Stutze. *"Ant Colony Optimization"*, MIT press, 2004;

[40] Mitchell, Melanie (1996). *"An Introduction to Genetic Algorithms"*, Cambridge, MA, MIT press;

[41] M. Tripathy, S. Mishra, et al., *"Transmission loss reduction based on FACTS and bacteria foraging algorithm"*, Proceedings of the 9th International Conference on Parallel Problem Solving from Nature (PPSN '06), Vol. 4193, pp. 222–231, 2006;

[42] S. Mishra, *"A hybrid least square-fuzzy bacterial foraging strategy for harmonic estimation"*, IEEE Transactions on Evolutionary Computation, Vol. 9, No. 1, pp. 61–73, 2005;

[43] Hagiwara, H (1989). "*Weather routing of (Sail-assisted) motor vessels*". PhD thesis. Delft;

[44] James, R.W (1957). "*Application of wave forecast to marine navigation*", U.S Navy Hudrographic office, Washington;

[45] P. Krata & J. Szlapczynska, Maritime University, Gdynia, Poland. "*Weather Hazard Avoidance in Modeling Safety of Motor-driven Ship for Multicriteria*";

[46] Chen, H. "*A Stochstic Dynamic Program for minimum Cost Ship Route*", Ph. D, Thesis, Department of Ocean Engineering, Masachusetts Institute of Technology (1978);

[47] California Institute of Technology, 2009, All rights reserved. *"Ocean Surface Current Analysis (OSCAR) Third Degree Resolution User's Hand Book";*

[48] Bonjean, F., Lagerloef, G. S. E. *"Diagnostic model and analysis of the surface currents in the tropical Pacific Ocean"*, J. Phys. Oceanogr., vol. 32, 2938 – 2954, 2002;

[49] W. Zhao, H. Wang, J. Geng, et al. *"Multi-Objective Weather Routing Algorithm for Ships Based on Hybrid Particle Swarm Optimization"*. J. Ocean Univ. China, Vol 21, pp. 28 – 38, Oct, 2018, doi: 10.1016/j. ijpe. 2018. 07. 014;

[50] Tsou, Ming-Cheng & Hsueh, Chao-Kuang. "*The study of ship collision avoidance route planning by ant colony algorithm*" (2010);

[51] A. P. Teixeira & C. Guedes Soares, "*AIS Based Shipping Routes Using the Dijkstra Algorithm"* (2019);

[52] M.D. Nguyen et al, *Automatic collision avoiding system for ship in congested water and at open sea*, ICAIS 2012;

[53] J. Zheng et al. *"A voyage with minimal fuel consumption for cruise ships"*. J. Clean. Prod, vol. 215, pp. 144 - 153, Apr. 2019, doi: 10.1016/j.jclepro.2019.01.032;

[54] M.D. Nguyen, Tamaru Hitoi "*A study on an Automatic Navigation System Basing on Radar and AIS data*", World Congress 2009 – International Association of Institute of Navigation;

[55] L. Walther, A. Rizvanolli, M. Wendebourg, and C. Jahn. *"Modelling and Optimization Algorithms in Ship Weather Routing"*. Int. J. E-Navig. Marit. Econ, vol. 4, pp. 31 – 45, Jun. 2016, doi: 10.1016/j.oceaneng.2017.12.049;

[56] L. Yang, G. Chen, J. Zhao, and N. G. M. Rytter. *"Ship Speed Optimization Considering Ocean Currents to Enhance Environmental Sustainability in Maritme Shipping"*. Sustainalbility, vol. 12, no. 9, p. 3649, May. 2020, doi: 10.3390/su 12093649;

[57] M.D. Nguyen, Tokyo University of Marine Science and Technology, Doctor thesis "*A study on an integrated collision avoiding system for merchant ships*";

[58] Dang Quang Viet, Nguyen Minh Duc & Phan Van Hung. "*Assessment of potentially cutting GHG emissions from shipping in relation to energy to energy consuption trends using Fuzzy Analytic Hierarchy Process"*. Australian Journal of Maritime & Ocean Affairs, Published online: 08 Nov 2021.

[59] Phan Văn Hưng, Đặng Quang Việt, Lê Thành Đạt. *"Optimal weather routing based on adaptive bacterial foraging algorithm for vessel"*. Journal of Technology & Innovation (JTIN), ISSN: 2773-6202 (Online), Available online: 20 May 2022.

[60] Phan Văn Hưng, Đặng Quang Việt, Nguyễn Minh Đức, Lê Thành Đạt. *"Ship routing optimization using bacterial foraging optimization*

algorithm for safety and efficient navigation". International Journal of Electrical and Computer Engineering, Vol 13, No 2: April 2023;

[61] C. Guo, H. Tang, B. Niu, C. B. Lee. *"A survey of bacterial foraging optimization"*. Neurocomputing, Vol. 452, pp. 728 – 746, Sept, 2021, doi: 10. 1016/J.neucom.2020.06.142;

[62] J. Dow and R. Brown. *"Isochrone convergence imaging of pulse echoes"*. Ultrasonics, vol. 15, no. 3, pp. 129 – 135, May. 1977, doi: 10. 1016/0041-624X(77)90030-0;

[63] J. Hu, Z. Cheng, G. Zhong, and Z. Huang. *"A Calculation Method and Its Application of Bus Isochrones"*. J. Transp. Syst. Eng. Inf. Tachnol, vol. 13, no. 3, pp. 99 – 104, Jun. 2013, doi: 10. 1016/S1570-6672(13)60111-7;

[64] P. Ramond, J. Perez. *"New Methods of Isochrone Mechanics"*. Journal of Mathematical Physics, American Institute of Physics (AIP), 2021, 62, pp. 112704, doi: 10.1063/5.0056957;

[65] L. Jun, et al. *"Analysis and Improvement of the Bacterial Foraging Optimization Algorithm"*. Journal of Computing Science and Engineering, Vol. 8, no 1, pp. 1-10, 2014, doi: 10.5626/JCSE.2014.8.1.1;

[66] Abd-Elazim, S. M, Ali, E. S, 2012. *"Bacterrial Foraging Optimization Algorithm based SVC damping controller design for power system stability enhancement"*. International Journal of Electrical Power & Energy Systems, 43, pp. 933-940. https://doi.org/10.1016/j.ijepes.2012.06.048;

[67] Corradu, A. Oneto, L. Baldi, F. Anguita, D, 2017. *"Vessels fuel consumption forecast and trim optimization"*. A data analytics

perspective, Ocean Engineering 130, pp. 351-370, https://doi.org/10.1016/j.oceaneng.2016.11.05;

[68] International Convention for the Safety of Life at Sea (SOLAS), 1974/ 2004;

[69] The International Convention for the Prevention of Pollution from ships, 1973 as modified by the protocol of 1978, or "MARPOL 73/78";

[70] International Convention on Loadline, 1966/ 1988;

[71] International Convention on Standards of Training, Certification and Watchkeeping for Seafarers, 1978/ 2010;

[72] IMO Resolution A 893 (21) – Guidelines for Voyage Planning;

[73] IMO (2018) Admiralty – Ocean Passages for the World (Vol1), NP 136;

III. TÀI LIỆU THAM KHẢO TỪ INTERNET

[74] https://database.rish.kyoto-u.ac.jp/arch/jmadata/data/gpv/original/;

[75] https://www.esr.org/research/oscar/;

[76] https://www.windy.com;

[77] https://www.imo.org;

[78] https://en.wikipedia.org/wiki/Dynamic-programming;

[79] https://www.inmarsat.com;

[80] https://www.sciencedirect.com/topics/engineering/exhaustive-search;

[81] https://en.wikipedia.org/wiki/hill_climbing;

[82] https://www.passageweather.com;

[83] https://www.nchmf.gov.vn;

[84] https://www.fugroweather.com;

[85] https://www.offshoreweather.com;

[86] https://www.geeksForgeeks.org/dynamic-programming;

PHỤ LỤC

DANH MỤC MÃ CODE

1. Tính toán tuyến đường chạy tàu

```
Imports System.IO

Public Class CTMPRouteForShowing
    Public wpsLat() As Double
    Public wpslon() As Double
    Public travTime() As Double
    Public travDist() As Double
    Public preceding As String

    Public Sub New()
    End Sub

    Public Sub New(ByRef st As String)
        Dim v() As String = Split(st, ";")

        preceding = v(0)
        If v.Length < 2 Then
            wpsLat = Nothing
            wpslon = Nothing
            Exit Sub
        End If


        Dim No As Integer = Val(v(1))
        If v.Length <> 2 + No * 2 + (No - 1) * 2 Then
            wpsLat = Nothing
            wpslon = Nothing
            Exit Sub
        End If


        If No > 0 Then
            ReDim wpsLat(No - 1)
            ReDim wpslon(No - 1)
            ReDim travTime(No - 2)
            ReDim travDist(No - 2)

            Dim count As Integer = 2
            For I As Integer = 0 To wpsLat.Length - 1
                wpsLat(I) = Val(v(count))
                count += 1

                wpslon(I) = Val(v(count))
                count += 1
            Next

            For I As Integer = 0 To travTime.Length - 1
                travTime(I) = Val(v(count))
                count += 1

                travDist(I) = Val(v(count))
                count += 1
```

```
            Next

        End If
    End Sub


    Public Function ToStringFormat() As String
        Dim st As String = preceding & ";" & wpsLat.Length
        For I As Integer = 0 To wpsLat.Length - 1
            st = st & ";" & wpsLat(I) & ";" & wpslon(I)
        Next

        For I As Integer = 0 To wpsLat.Length - 2
            st = st & ";" & travTime(I) & ";" & travDist(I)
        Next

        Return st
    End Function

    Public Function TotalTime() As Double
        Dim retn As Double = 0
        If IsNothing(travTime) = False Then
            For I As Integer = 0 To travTime.Length - 1
                retn += travTime(I)
            Next
        End If

        Return retn
    End Function



    Public Function TotalDist() As Double
        Dim retn As Double = 0
        If IsNothing(travDist) = False Then
            For I As Integer = 0 To travDist.Length - 1
                retn += travDist(I)
            Next
        End If

        Return retn
    End Function
End Class


Public Class CSuggestRouteList
    Public routelist() As CTMPRouteForShowing
    Public tmpFilename As String

    Public Sub Init(ByVal filename As String)
        tmpFilename = filename

        Dim st() As String = mGlbReadStringsFromFile(filename)
        If IsNothing(st) = True Then
            routelist = Nothing
            Exit Sub
        End If

        ReDim routelist(st.Length - 1)
```

```
        Dim count As Integer = 0
        For I As Integer = 0 To st.Length - 1
            routelist(count) = New CTMPRouteForShowing(st(I))
            If IsNothing(routelist(count).wpsLat) = False Then
                count += 1
            End If
        Next

        If count > 0 Then
            ReDim Preserve routelist(count - 1)
        Else
            routelist = Nothing
        End If
    End Sub

    Public Sub DeleteARoute(ByVal idx As Integer)
        If routelist.Length = 1 Then
            routelist = Nothing
        Else
            For I As Integer = idx To routelist.Length - 2
                routelist(I) = New CTMPRouteForShowing(routelist(I +
1).ToStringFormat)
            Next
            ReDim Preserve routelist(routelist.Length - 2)
        End If
        WriteToFile()
    End Sub


    Public Sub WriteToFile()
        Dim stWrite As StreamWriter = New StreamWriter(tmpFilename, False)
        If IsNothing(routelist) = False Then
            For I As Integer = 0 To routelist.Length - 1
                stWrite.WriteLine(routelist(I).ToStringFormat)
            Next
        End If
        stWrite.Close()
    End Sub


    'Public Sub WriteBestROuteToFile(ByVal filename As String)
    '    Dim stWrite As StreamWriter = New StreamWriter(filename, True)
    '    Dim st As String = mGlbTimeOfUpdateToString() & ";" &
bacteria(qualitySort(0)).WPs.Length
    '    For I As Integer = 0 To bacteria(qualitySort(0)).WPs.Length - 1
    '        st = st & ";" & bacteria(qualitySort(0)).WPs(I).lat & ";" &
bacteria(qualitySort(0)).WPs(I).lon
    '    Next

    '    For I As Integer = 0 To bacteria(qualitySort(0)).WPs.Length - 2
    '        st = st & ";" & bacteria(qualitySort(0)).travTime(I) & ";" &
bacteria(qualitySort(0)).dist(I)
    '    Next
    '    stWrite.WriteLine(st)

    '    stWrite.Close()
    'End Sub

End Class
```

```
Public Class C8PlannedRouteList
    Public routes() As C8Route
    Public routeNames() As String

    Public Function mGlbFcnGetPlannedRouteFilename(ByVal IMO As String) As String
        Dim retn As String = mGlbFcnGetVesselFolderName() &
IO.Path.DirectorySeparatorChar & "plannedroutefile"
        Return retn
    End Function

    Public Sub Init()
        Dim filename As String = mGlbFcnGetPlannedRouteFilename(mGlbShipInfo.IMO)
        Dim stRead As StreamReader = New StreamReader(filename)
        Dim st() As String = Nothing

        Try
            Dim tmp As String = stRead.ReadToEnd()
            st = Split(tmp, vbCrLf)
            stRead.Close()
        Catch ex As Exception
            MsgBox(ex.ToString)
        End Try

        If IsNothing(st) = False Then
            Init(st)
        Else
            routeNames = Nothing
            routes = Nothing
        End If
    End Sub


    Public Sub Init(ByRef st() As String)
        ReDim routes(st.Length - 1)
        ReDim routeNames(st.Length - 1)
        Dim v() As String
        Dim tmp As Integer

        For I As Integer = 0 To routes.Length - 1
            v = Split(st(I), ";")
            routeNames(I) = v(1)

            tmp = (v.Length - 4) / 6

            ReDim routes(I).WPs(tmp)
            ReDim routes(I).dir(tmp - 1)
            ReDim routes(I).dist(tmp - 1)
            ReDim routes(I).travRPM(tmp - 1)
            ReDim routes(I).travTime(tmp - 1)

            For J As Integer = 0 To routes(I).WPs.Length - 1
                If J < routes(I).WPs.Length - 1 Then
                    routes(I).WPs(J) = New C8WayPoint(Val(v(2 + J * 6)), Val(v(2 + J
* 6 + 1)))
                    routes(I).dir(J) = Val(v(2 + J * 6 + 2))
                    routes(I).dist(J) = Val(v(2 + J * 6 + 3))
```

```
                    routes(I).travRPM(J) = Val(v(2 + J * 6 + 4))
                    routes(I).travTime(J) = Val(v(2 + J * 6 + 5))
                Else
                    routes(I).WPs(J) = New C8WayPoint(Val(v(2 + J * 6)), Val(v(2 + J
* 6 + 1)))
                End If
            Next
        Next

        Dim tmpRoute As C8Route = New C8Route()
        Dim tmpName As String
        For I As Integer = 0 To routes.Length - 2
            For J As Integer = I + 1 To routes.Length - 1
                If String.Compare(routeNames(I), routeNames(J), True) > 0 Then
                    tmpRoute.CopyFrom(routes(I))
                    routes(I).CopyFrom(routes(J))
                    routes(J).CopyFrom(tmpRoute)

                    tmpName = routeNames(I)
                    routeNames(I) = routeNames(J)
                    routeNames(J) = tmpName
                End If
            Next
        Next
    End Sub

    Public Function AddRoute(ByRef aRoute As C8Route, ByRef routename As String) As
Integer
        Dim idx As Integer = 0
        If IsNothing(routes) = True Then
            idx = routes.Length
            For I As Integer = 0 To routeNames.Length - 1
                If String.Compare(routename, routeNames(I), True) = 0 Then
                    idx = -1
                End If
            Next
        End If

        If idx = -1 Then Return -1

        ReDim Preserve routeNames(idx)
        ReDim Preserve routes(idx)
        routeNames(idx) = routename.Replace(";", ",")
        routes(idx) = New C8Route()
        routes(idx).CopyFrom(aRoute)
        SaveToFile()

        Return 1
    End Function

    Public Function DeleteRoute(ByVal routeName As String) As String
        Dim idx As Integer = -1

        If IsNothing(routes) = False Then
            For I As Integer = 0 To routes.Length - 1
                If String.Compare(routeName, routeNames(I), True) = 0 Then
                    idx = I
                    Exit For
```

```
                End If
            Next
        End If

        If idx > -1 Then
            DeleteRoute(idx)
            Return 1
        Else
            Return -1
        End If
    End Function


    Public Sub DeleteRoute(ByVal idx As Integer)
        If IsNothing(routes) = True Then Exit Sub
        If routes.Length = 1 Then
            routes = Nothing
            routeNames = Nothing
            SaveToFile()

        Else
            For I As Integer = idx To routeNames.Length - 2
                routes(I).CopyFrom(routes(I + 1))
                routeNames(I) = routeNames(I + 1)
            Next

            ReDim Preserve routes(routes.Length - 2)
            ReDim Preserve routeNames(routeNames.Length - 2)
            SaveToFile()
        End If
    End Sub

    Public Sub SaveToFile()
    End Sub
End Class


Public Class C8RouteExplorer
    Public weatherDB As C8WeatherDB

    'Public RpmAndSpeedCharUsed As C8ShipSpeedCharSet

    Public normSpdChar As C3NormSpeedForAllRPM
    Public standardRpmDraftTrim As C7StandardRpmDraftTrim
    Public totalSpdChar As C7TotalShipSpeedChar

    Public limitlines() As C8LimitLine
    Public limitWaveHeight As Double
    Public limitWndSpeed As Double
    Public limitLowLat As Double
    Public limitUpLat As Double
    Public limitLeftLon As Double
    Public limitRightLon As Double

    Public draft As Double
    Public trm As Double
    Public rpm1 As Double
    Public rpm2 As Double
    Public fuelTonsConsume1 As Double
```

```
    Public fuelTonsConsume2 As Double

    Public routeCal As C8RouteMathCal
    Public IsPassingDayLine As Integer
    Public lonDiff As Double
    Public depWP As C8WayPoint
    Public desWP As C8WayPoint
    Public depTime As Date
    Public timeStep As Integer = 2

    Public maxSwimLength As Integer = 4
    Public baseshiftDist As Double = 2

    Public bacteria() As C8Route
    Public RpmShiftPointIdx() As Integer

    Public TimeAllowToReachDest_Hour As Double
    Public FuelAllowToReachDest_Ton As Double

    Public qualitySort() As Integer
    Public bacNew() As Integer


    Public searchDir As Integer ' 1:Long - 90     2:Lat-0         3:Diag-45
    Public rumlineDist As Double


    Public CS As Double
    Public SN As Double

    Public Sub New()
    End Sub


    Public Sub InitExplorer(ByRef limits() As C8LimitLine, ByVal limUpLat As Double,
ByVal limLowLat As Double, ByVal limLftLon As Double, ByVal limRgtLon As Double,
ByVal limWav As Double, ByVal limWnd As Double, ByRef stTime As Date, ByRef stWP As
C8WayPoint, ByRef enWP As C8WayPoint, ByRef tmpRPM1 As Double, ByRef tmpRPM2 As
Double, ByRef tmpDraft As Double, ByRef tmpTrim As Double)
        trm = tmpTrim
        draft = tmpDraft
        rpm1 = tmpRPM1
        rpm2 = tmpRPM2

        standardRpmDraftTrim = New C7StandardRpmDraftTrim()
        standardRpmDraftTrim.Init()

        normSpdChar = New C3NormSpeedForAllRPM()
        normSpdChar.Init(standardRpmDraftTrim)

        totalSpdChar = New C7TotalShipSpeedChar()
        totalSpdChar.Init()

        routeCal = New C8RouteMathCal()
        depWP = New C8WayPoint(stWP)
        desWP = New C8WayPoint(enWP)

        depTime = DateAdd(DateInterval.Hour, 0, stTime)
```

```
limitLowLat = limLowLat
limitUpLat = limUpLat

If limitUpLat <= routeCal.Max(depWP.lat, desWP.lat) + 5.5 Then
    limitUpLat = routeCal.Max(depWP.lat, desWP.lat) + 5.5
End If

If limitLowLat >= routeCal.Min(depWP.lat, desWP.lat) - 5.5 Then
    limitLowLat = routeCal.Min(depWP.lat, desWP.lat) - 5.5
End If

limitLeftLon = limLftLon
limitRightLon = limRgtLon

limitWaveHeight = limWav
limitWndSpeed = limWnd


lonDiff = enWP.lon - stWP.lon
Dim isW As Integer = 0
If lonDiff >= 180 Then
    lonDiff -= 360
    IsPassingDayLine = 1

ElseIf lonDiff < -180 Then
    lonDiff += 360
    IsPassingDayLine = 1

End If

If IsPassingDayLine = 1 Then
    If depWP.lon < 0 Then depWP.lon += 360
    If desWP.lon < 0 Then desWP.lon += 360

    If limitLeftLon < 0 Then limitLeftLon += 360
    If limitRightLon < 0 Then limitRightLon += 360
End If


If limitLeftLon >= routeCal.Min(depWP.lon, desWP.lon) - 0.5 Then
    limitLeftLon = routeCal.Min(depWP.lon, desWP.lon) - 0.5
End If

If limitRightLon <= routeCal.Max(depWP.lon, desWP.lon) + 0.5 Then
    limitRightLon = routeCal.Max(depWP.lon, desWP.lon) + 0.5
End If

Dim tmpDir As Double = routeCal.Direction_rad(depWP, desWP) / routeCal.toRad
If tmpDir > 180 Then tmpDir -= 180

If tmpDir <= 20 Or tmpDir >= 160 Then
    searchDir = 1
ElseIf tmpDir >= 70 And tmpDir <= 110 Then
    searchDir = 2
Else
    searchDir = 3
End If

rumlineDist = routeCal.Distance(depWP, desWP) / 60
```

```
'=============================================================================
        InitLimits(limits)
    End Sub


    Public Sub InitWeatherDB(ByRef orgWeatherDB As C8BaseWeatherDB)
        weatherDB = New C8WeatherDB()

        Dim lat0 As Double = routeCal.Min(depWP.lat, desWP.lat)
        lat0 = CType(lat0 - 20, Integer)
        If lat0 < -70 Then lat0 = -70

        Dim lat1 As Double = routeCal.Max(depWP.lat, desWP.lat)
        lat1 = CType(lat1 + 20, Integer)
        If lat1 > 70 Then lat1 = 70

        Dim latNo As Integer = (lat1 - lat0) / 0.5

        Dim lon0 As Integer = Math.Floor(routeCal.Min(depWP.lon, desWP.lon))
        If lonDiff < 10 Then lon0 = lon0 - 10
        Dim lon1 As Integer = Math.Floor(routeCal.Max(depWP.lon, desWP.lon))
        If lonDiff < 10 Then lon1 = lon1 + 10
        Dim lonNo As Integer = (lon1 - lon0) / 0.5

        weatherDB.InitWholeMessagesFromBaseWeatherMessage(orgWeatherDB, depTime,
lat0, latNo, lon0, lonNo)

     End Sub


    Public Sub InitLimits(ByRef orgLimits() As C8LimitLine)
        If IsNothing(orgLimits) = True Then
            limitlines = Nothing
            Exit Sub
        End If


        ReDim limitlines(orgLimits.Length - 1)
        Dim count As Integer = 0

        For I As Integer = 0 To orgLimits.Length - 1
            If IsLimitLineInvolved(orgLimits(I)) > 0 Then
                limitlines(count) = New C8LimitLine(orgLimits(I))
                limitlines(count).GetLimitLonLat()
                count += 1
            End If
        Next
        If count > 0 Then
            ReDim Preserve limitlines(count - 1)
        Else
            limitlines = Nothing
        End If
    End Sub


    Public Function IsLimitLineInvolved(ByRef alimLine As C8LimitLine) As Integer
        If limitLeftLon < limitRightLon Then
```

```
            For I As Integer = 0 To alimLine.wps.Length - 1
                If alimLine.wps(I).lon >= limitLeftLon And alimLine.wps(I).lon <=
limitRightLon Then
                    Return 1
                End If
            Next

        Else
            For I As Integer = 0 To alimLine.wps.Length - 1
                If alimLine.wps(I).lon >= limitLeftLon Or alimLine.wps(I).lon <=
limitRightLon Then
                    Return 1
                End If
            Next
        End If

        Return 0
    End Function

    Public Function CheckLimitsValid(ByRef wp1 As C8WayPoint, ByRef wp2 As
C8WayPoint, ByRef wp3 As C8WayPoint) As Boolean
        If wp2.lat > limitUpLat Or wp2.lat < limitLowLat Then Return False
        For I As Integer = 0 To limitlines.Length - 1
            If limitlines(I).IsCross(wp1, wp2, routeCal) = True Then
                Return False
            End If

            If limitlines(I).IsCross(wp2, wp3, routeCal) = True Then
                Return False
            End If
        Next

        Return True
    End Function



    Public Sub RouteExplore(ByVal NoOfBac As Integer, ByVal NoOfSwim As Integer,
ByVal NoOfSlide As Integer, ByVal NoOfCouling As Integer, ByVal NeighbourDist As
Double, ByRef NoOfRenew As Integer, ByVal NoOfGeneration As Integer)
        InitSwarm(NoOfBac)


        For I As Integer = 0 To NoOfGeneration
            BacteriaSwarmSearch(NoOfSwim, NoOfSlide)
            QualitySorting()
            BacterialSwarming(NoOfCouling, NeighbourDist, NoOfRenew)
        Next

        QualitySorting()
    End Sub



    Public Sub ImproveRoute(ByVal NoOfSwim As Integer, ByVal NoOfSlide As Integer,
ByVal NoOfCouling As Integer, ByVal NeighbourDist As Double, ByRef NoOfRenew As
Integer, ByVal NoOfGeneration As Integer)
        Dim tmp As Double = bacteria(0).Quality
        For I As Integer = 0 To NoOfGeneration
```

```
            BacteriaSwarmSearch(NoOfSwim, NoOfSlide)
            QualitySorting()
            BacterialSwarming(NoOfCouling, NeighbourDist, NoOfRenew)
        Next

        Dim tmpp As Double = bacteria(0).Quality - tmp
        QualitySorting()
    End Sub


    Public Function InitSwarm(ByVal NoOfBac As Integer) As Boolean
        ReDim bacteria(NoOfBac - 1)
        ReDim qualitySort(NoOfBac - 1)
        ReDim bacNew(NoOfBac - 1)
        ReDim RpmShiftPointIdx(NoOfBac - 1)

        Dim count As Integer = 0
        For I As Integer = 0 To bacteria.Length * 14
            If RandomInitABacteria(bacteria(count), count) = True Then
                count += 1
            Else
                Dim tt As Double
                tt = 0
            End If

            If count >= bacteria.Length Then
                Exit For
            End If
        Next

        If count > 0 Then
            ReDim Preserve bacteria(count - 1)
            ReDim Preserve qualitySort(count - 1)
            ReDim Preserve bacNew(count - 1)
            ReDim Preserve RpmShiftPointIdx(count - 1)


            'Dim tmpTime As Date = depTime
            'For I As Integer = 0 To bacteria.Length - 1
            '    ReDim bacteria(I).dir(bacteria(I).WPs.Length - 2)
            '    ReDim bacteria(I).dist(bacteria(I).WPs.Length - 2)
            '    ReDim bacteria(I).travTime(bacteria(I).WPs.Length - 2)
            '    ReDim bacteria(I).travRPM(bacteria(I).WPs.Length - 2)

            '    tmpTime = depTime

            '    For J As Integer = 0 To bacteria(I).WPs.Length - 2
            '        bacteria(I).dir(J) = routeCal.Direction_rad(bacteria(I).WPs(J),
bacteria(I).WPs(J + 1))
            '        bacteria(I).dist(J) = routeCal.Distance(bacteria(I).WPs(J),
bacteria(I).WPs(J + 1))

            '        If J <= RPMShiftWPidx Then
            '            bacteria(I).travTime(J) = routeCal.CalculateTime(tmpTime,
bacteria(I).WPs(J), bacteria(I).WPs(J + 1), bacteria(I).dist(J), bacteria(I).dir(J),
VssSpdChar(0), weatherDB, limitWaveHeight, limitWndSpeed)
            '            bacteria(I).travRPM(J) = RPMUsed(0)
            '        Else
```

```
            '            bacteria(I).travTime(J) = routeCal.CalculateTime(tmpTime,
bacteria(I).WPs(J), bacteria(I).WPs(J + 1), bacteria(I).dist(J), bacteria(I).dir(J),
VssSpdChar(1), weatherDB, limitWaveHeight, limitWndSpeed)
            '            bacteria(I).travRPM(J) = RPMUsed(1)
            '        End If
            '        tmpTime = DateAdd(DateInterval.Hour, bacteria(I).travTime(J),
tmpTime)
            '    Next

            '    bacteria(I).CalculateQuality()
            'Next


            Return True
        Else

            bacteria = Nothing
            qualitySort = Nothing
            bacNew = Nothing
            RpmShiftPointIdx = Nothing

            Return False
        End If
    End Function



    Public Function IsCrossLimits(ByRef wp1 As C8WayPoint, ByRef wp2 As C8WayPoint)
As Boolean
        If IsNothing(limitlines) = True Then Return False


        If (wp1.lon > 150 And wp2.lon < -150) Then
            Dim wp3 As C8WayPoint = New C8WayPoint(wp1)
            Dim wp4 As C8WayPoint = New C8WayPoint(wp2)
            wp3.lon = wp1.lon - 360
            wp4.lon = wp2.lon + 360

            For I As Integer = 0 To limitlines.Length - 1
                If limitlines(I).IsCross(wp1, wp4, routeCal) = True Then
                    Return True
                End If

                If limitlines(I).IsCross(wp2, wp3, routeCal) = True Then
                    Return True
                End If
            Next

        ElseIf (wp2.lon > 150 And wp1.lon < -150) Then
            Dim wp3 As C8WayPoint = New C8WayPoint(wp1)
            Dim wp4 As C8WayPoint = New C8WayPoint(wp2)
            wp3.lon = wp1.lon + 360
            wp4.lon = wp2.lon - 360

            For I As Integer = 0 To limitlines.Length - 1
                If limitlines(I).IsCross(wp1, wp4, routeCal) = True Then
                    Return True
                End If
```

```
                If limitlines(I).IsCross(wp2, wp3, routeCal) = True Then
                    Return True
                End If
            Next

        Else
            For I As Integer = 0 To limitlines.Length - 1
                If limitlines(I).IsCross(wp1, wp2, routeCal) = True Then
                    Return True
                End If
            Next

        End If

        Return False
    End Function



    Public Function IsCrossLimits(ByRef wp1 As C8WayPoint, ByRef wp2 As C8WayPoint,
ByRef wp3 As C8WayPoint) As Boolean
        If IsCrossLimits(wp1, wp2) = True Then Return True
        If IsCrossLimits(wp2, wp3) = True Then Return True

        Return False
    End Function



    Public Sub WriteBestROuteToFile(ByVal filename As String)
        Dim stWrite As StreamWriter = New StreamWriter(filename, True)
        Dim st As String = mGlbTimeOfUpdateToString() & ";" &
bacteria(qualitySort(0)).WPs.Length
        For I As Integer = 0 To bacteria(qualitySort(0)).WPs.Length - 1
            st = st & ";" & bacteria(qualitySort(0)).WPs(I).lat & ";" &
bacteria(qualitySort(0)).WPs(I).lon
        Next

        For I As Integer = 0 To bacteria(qualitySort(0)).WPs.Length - 2
            st = st & ";" & bacteria(qualitySort(0)).travTime(I) & ";" &
bacteria(qualitySort(0)).dist(I)
        Next
        stWrite.WriteLine(st)

        stWrite.Close()
    End Sub


    Public Function RandomInitABacteria(ByRef bac As C8Route, ByVal bacIdx As
Integer) As Boolean
        bac = New C8Route()
        Dim NoOfDivide As Integer = 0
        NoOfDivide = rumlineDist / 50
        If NoOfDivide < 10 Then NoOfDivide = 10

        Dim dlon As Double
        Dim dlat As Double

        ReDim bac.WPs(NoOfDivide)
```

```
        dlon = lonDiff / NoOfDivide
        dlat = (desWP.lat - depWP.lat) / NoOfDivide

        bac.WPs(0) = New C8WayPoint(depWP.lat, depWP.lon)
        bac.WPs(bac.WPs.Length - 1) = New C8WayPoint(desWP.lat, desWP.lon)


        If searchDir = 2 Then
            If RandomInitBacteriaCase0(bac, NoOfDivide, dlat, dlon) < 1 Then Return
False
        ElseIf searchDir = 1 Then
            If RandomInitBacteriaCase1(bac, NoOfDivide, dlat, dlon) < 1 Then Return
False
        Else
            If RandomInitBacteriaCase2(bac, NoOfDivide, dlat, dlon) < 1 Then Return
False
        End If

        ReDim bac.dir(bac.WPs.Length - 2)
        ReDim bac.dist(bac.WPs.Length - 2)
        ReDim bac.travTime(bac.WPs.Length - 2)
        ReDim bac.travRPM(bac.WPs.Length - 2)
        ReDim bac.fuelTpH(bac.WPs.Length - 2)


        Dim tmpTime As Date = DateAdd(DateInterval.Minute, 0, depTime)
        RpmShiftPointIdx(bacIdx) = bac.WPs.Length * 0.7

        For J As Integer = 0 To bac.WPs.Length - 2
            bac.dir(J) = routeCal.Direction_rad(bac.WPs(J), bac.WPs(J + 1))
            bac.dist(J) = routeCal.Distance(bac.WPs(J), bac.WPs(J + 1))

            If J <= RpmShiftPointIdx(bacIdx) Then
                bac.travRPM(J) = rpm1
                bac.fuelTpH(J) = fuelTonsConsume1
            Else
                bac.travRPM(J) = rpm2
                bac.fuelTpH(J) = fuelTonsConsume2
            End If

            bac.travTime(J) = routeCal.CalculateTime(tmpTime, bac.WPs(J), bac.WPs(J
+ 1), bac.dist(J), bac.dir(J), totalSpdChar, weatherDB, limitWaveHeight,
limitWndSpeed, bac.travRPM(J), draft, trm, normSpdChar)
            If bac.travTime(J) >= mGlbInfinite Then
                Return False
            End If

            tmpTime = DateAdd(DateInterval.Hour, bac.travTime(J), tmpTime)
        Next

        bac.CalculateQuality(TimeAllowToReachDest_Hour, FuelAllowToReachDest_Ton)

        Return True
    End Function


    Public Function RandomInitBacteriaCase0(ByRef bac As C8Route, ByRef NoOfDivide
As Integer, ByRef dlat As Double, ByRef dlon As Double) As Integer
        Dim count As Integer
```

```
        Dim tmp As Double = Rnd()
        Dim udb As Integer = 1

        If tmp < 0.4 Then
            udb = 1
        ElseIf tmp > 0.6 Then
            udb = 2
        Else
            udb = 3
        End If

        CS = 1
        SN = 0

        For I As Integer = 1 To NoOfDivide - 1
            count = 0
            bac.WPs(I) = New C8WayPoint(depWP.lat + I * dlat, depWP.lon + I * dlon)


            While count < 100
                count += 1

                If udb = 1 Then
                    bac.WPs(I).lat += (limitUpLat - bac.WPs(I).lat) * Rnd()
                ElseIf udb = 2 Then
                    bac.WPs(I).lat += (limitLowLat - bac.WPs(I).lat) * Rnd()
                Else
                    If Rnd() > 0.5 Then
                        bac.WPs(I).lat += (limitUpLat - bac.WPs(I).lat) * Rnd()
                    Else
                        bac.WPs(I).lat += (limitLowLat - bac.WPs(I).lat) * Rnd()
                    End If
                End If

                If I < NoOfDivide - 1 Then
                    If IsCrossLimits(bac.WPs(I - 1), bac.WPs(I)) = False Then
                        Exit While
                    End If
                Else
                    If IsCrossLimits(bac.WPs(I - 1), bac.WPs(I), bac.WPs(I + 1)) =
False Then
                        Exit While
                    End If
                End If
            End While

            If count = 100 Then Return 0
        Next

        Return 1
    End Function

    Public Function RandomInitBacteriaCase1(ByRef bac As C8Route, ByRef NoOfDivide
As Integer, ByRef dlat As Double, ByRef dlon As Double) As Integer
        Dim count As Integer
        Dim tmp As Double = Rnd()
        Dim udb As Integer = 1

        If tmp < 0.4 Then
```

```
            udb = 1
        ElseIf tmp > 0.6 Then
            udb = 2
        Else
            udb = 3
        End If

        CS = 0
        SN = 1

        For I As Integer = 1 To NoOfDivide - 1
            count = 0
            bac.WPs(I) = New C8WayPoint(depWP.lat + I * dlat, depWP.lon + I * dlon)


            While count < 100
                count += 1

                If udb = 1 Then
                    bac.WPs(I).lon += (limitRightLon - bac.WPs(I).lon) * Rnd()
                ElseIf udb = 2 Then
                    bac.WPs(I).lon += (limitLeftLon - bac.WPs(I).lon) * Rnd()
                Else
                    If Rnd() > 0.5 Then
                        bac.WPs(I).lon += (limitRightLon - bac.WPs(I).lon) * Rnd()
                    Else
                        bac.WPs(I).lon += (limitLeftLon - bac.WPs(I).lon) * Rnd()
                    End If
                End If

                If I < NoOfDivide - 1 Then
                    If IsCrossLimits(bac.WPs(I - 1), bac.WPs(I)) = False Then
                        Exit While
                    End If
                Else
                    If IsCrossLimits(bac.WPs(I - 1), bac.WPs(I), bac.WPs(I + 1)) =
False Then
                        Exit While
                    End If
                End If
            End While

            If count = 100 Then Return 0
        Next
        Return 1
    End Function


    Public Function RandomInitBacteriaCase2(ByRef bac As C8Route, ByRef NoOfDivide
As Integer, ByRef dlat As Double, ByRef dlon As Double) As Integer
        Dim count As Integer
        Dim tmp As Double = Rnd()
        Dim udb As Integer = 1

        If tmp < 0.4 Then
            udb = 1
        ElseIf tmp > 0.6 Then
            udb = 2
        Else
```

```
            udb = 3
        End If


        Dim dir As Double = routeCal.Direction_rad(depWP, desWP)
        dir = dir + Math.PI / 2
        CS = Math.Cos(dir)
        SN = Math.Sin(dir)

        For I As Integer = 1 To NoOfDivide - 1
            count = 0
            bac.WPs(I) = New C8WayPoint(depWP.lat + I * dlat, depWP.lon + I * dlon)

            While count < 100
                count += 1

                If udb = 1 Then
                    tmp = rumlineDist * 0.4 * Rnd()
                    bac.WPs(I).lat += tmp * CS
                    bac.WPs(I).lon += tmp * SN

                    If bac.WPs(I).lon < limitLeftLon Or bac.WPs(I).lon >
limitRightLon Or bac.WPs(I).lat < limitLowLat Or bac.WPs(I).lat > limitUpLat Then
                        bac.WPs(I).lat -= tmp * CS
                        bac.WPs(I).lon -= tmp * SN
                    End If

                ElseIf udb = 2 Then
                    tmp = -rumlineDist * 0.4 * Rnd()
                    bac.WPs(I).lat += tmp * CS
                    bac.WPs(I).lon += tmp * SN

                    If bac.WPs(I).lon < limitLeftLon Or bac.WPs(I).lon >
limitRightLon Or bac.WPs(I).lat < limitLowLat Or bac.WPs(I).lat > limitUpLat Then
                        bac.WPs(I).lat -= tmp * CS
                        bac.WPs(I).lon -= tmp * SN
                    End If

                Else
                    If Rnd() > 0.5 Then
                        tmp = rumlineDist * 0.4 * Rnd()
                        bac.WPs(I).lat += tmp * CS
                        bac.WPs(I).lon += tmp * SN

                        If bac.WPs(I).lon < limitLeftLon Or bac.WPs(I).lon >
limitRightLon Or bac.WPs(I).lat < limitLowLat Or bac.WPs(I).lat > limitUpLat Then
                            bac.WPs(I).lat -= tmp * CS
                            bac.WPs(I).lon -= tmp * SN
                        End If

                    Else
                        tmp = -rumlineDist * 0.4 * Rnd()
                        bac.WPs(I).lat += tmp * CS
                        bac.WPs(I).lon += tmp * SN

                        If bac.WPs(I).lon < limitLeftLon Or bac.WPs(I).lon >
limitRightLon Or bac.WPs(I).lat < limitLowLat Or bac.WPs(I).lat > limitUpLat Then
                            bac.WPs(I).lat -= tmp * CS
                            bac.WPs(I).lon -= tmp * SN
```

```
                        End If

                    End If
                End If


                If I < NoOfDivide - 1 Then
                    'bac.WPs(I - 1) = New C8WayPoint(20, -55)
                    'bac.WPs(I) = New C8WayPoint(15, -52)

                    If IsCrossLimits(bac.WPs(I - 1), bac.WPs(I)) = False Then
                        Exit While
                    Else
                        'Dim tt As Integer
                        'tt = 111
                    End If
                Else
                    If IsCrossLimits(bac.WPs(I - 1), bac.WPs(I), bac.WPs(I + 1)) =
False Then
                        Exit While
                    End If
                End If
            End While

            If count = 100 Then Return 0
        Next

        Return 1
    End Function

    Public Sub BacteriaSwarmSearch(ByVal NoOfSwim As Integer, ByVal NoOfSlide As
Integer)
        For I As Integer = 0 To bacteria.Length - 1
            BacteriaSearch(bacteria(I), I, NoOfSwim, NoOfSlide)
        Next
    End Sub

    Public Sub BacteriaSearch(ByRef bac As C8Route, ByRef bacIdx As Integer, ByVal
NoOfSwim As Integer, ByVal NoOfSlide As Integer)
        Dim Quality As Double = bac.Quality
        Dim travtimes(bac.travTime.Length - 1) As Double
        Dim travdists(bac.travTime.Length - 1) As Double
        Dim travdir(bac.travTime.Length - 1) As Double
        Dim travRPM(bac.travTime.Length - 1) As Double
        Dim fuelCons(bac.travTime.Length - 1) As Double

        Dim stIdx As Integer
        Dim len As Integer

        Quality = bac.Quality
        For K As Integer = 0 To bac.dir.Length - 1
            travdir(K) = bac.dir(K)
            travdists(K) = bac.dist(K)
            travtimes(K) = bac.travTime(K)
            travRPM(K) = bac.travRPM(K)
            fuelCons(K) = bac.fuelTpH(K)
        Next

        Dim tmpDist As Double
```

```
        For I As Integer = 0 To NoOfSwim
            stIdx = 1 + Rnd() * (bac.WPs.Length - 3)
            If stIdx < 1 Then stIdx = 1
            If stIdx > bac.WPs.Length - 2 Then stIdx = bac.WPs.Length - 2

            len = -1 + Rnd() * (maxSwimLength + 1)
            If len < 1 Then len = 1
            If len > maxSwimLength Then len = maxSwimLength

            If stIdx + len > bac.WPs.Length - 1 Then
                len = bac.WPs.Length - 1 - stIdx
            End If

            Dim tmpLatShifts(len - 1) As Double
            Dim tmpLonShifts(len - 1) As Double

            For J As Integer = 0 To len - 1
                tmpDist = baseshiftDist * (Rnd() - 0.5)
                tmpLatShifts(J) = CS * tmpDist
                tmpLonShifts(J) = SN * tmpDist
            Next

            'End If

            'If stIdx = 6 And len = 1 Then
            '    If tmpLatShifts(0) < 0 Then
            '        len = len * 1
            '    End If
            'Else
            '    Exit For
            'End If

            Dim isValid As Boolean
            Dim tmpTime As Date

            For J As Integer = 0 To NoOfSlide
                isValid = BacSwim(bac, stIdx, len, tmpLonShifts, tmpLatShifts,
baseshiftDist)

                If isValid = True Then
                    For K As Integer = stIdx - 1 To stIdx + len - 1
                        bac.dir(K) = routeCal.Direction_rad(bac.WPs(K), bac.WPs(K +
1))

                        bac.dist(K) = routeCal.Distance(bac.WPs(K), bac.WPs(K + 1))
                    Next

                    For K As Integer = stIdx - 1 To bac.travTime.Length - 1
                        tmpTime = depTime
                        For G As Integer = 0 To stIdx
                            tmpTime = DateAdd(DateInterval.Hour, bac.travTime(G),
tmpTime)
                        Next

                        If K <= RpmShiftPointIdx(bacIdx) Then
                            bac.travRPM(K) = rpm1
                            bac.fuelTpH(K) = fuelTonsConsume1
                        Else
                            bac.travRPM(K) = rpm2
                            bac.fuelTpH(K) = fuelTonsConsume2
```

```
                        End If

                        bac.travTime(K) = routeCal.CalculateTime(tmpTime,
bac.WPs(K), bac.WPs(K + 1), bac.dist(K), bac.dir(K), totalSpdChar, weatherDB,
limitWaveHeight, limitWndSpeed, bac.travRPM(K), draft, trm, normSpdChar)
                    Next

                    tmpDist = travdists.Length
                    bac.CalculateQuality(TimeAllowToReachDest_Hour,
FuelAllowToReachDest_Ton)
                End If

                If bac.Quality < Quality And isValid = True Then
                    Quality = bac.Quality
                    For K As Integer = 0 To bac.dir.Length - 1
                        travdir(K) = bac.dir(K)
                        travdists(K) = bac.dist(K)
                        travtimes(K) = bac.travTime(K)
                        travRPM(K) = bac.travRPM(K)
                        fuelCons(K) = bac.fuelTpH(K)
                    Next

                Else
                    bac.Quality = Quality
                    For K As Integer = 0 To bac.dir.Length - 1
                        bac.dir(K) = travdir(K)
                        bac.dist(K) = travdists(K)
                        bac.travTime(K) = travtimes(K)
                        bac.travRPM(K) = travRPM(K)
                        bac.fuelTpH(K) = fuelCons(K)
                    Next

                    For K As Integer = stIdx To stIdx + len - 1
                        bac.WPs(K).lat -= tmpLatShifts(K - stIdx)
                        bac.WPs(K).lon -= tmpLonShifts(K - stIdx)
                    Next

                    Exit For
                End If

            Next

        Next
    End Sub


    Public Function BacSwim(ByRef bac As C8Route, ByRef stIdx As Integer, ByRef
NoOfPointChange As Integer, ByRef lonshiftdists() As Double, ByRef latshiftdists()
As Double, ByRef baseShiftDist As Double) As Boolean
        For I As Integer = stIdx To stIdx + NoOfPointChange - 1
            bac.WPs(I).lon += lonshiftdists(I - stIdx)
            bac.WPs(I).lat += latshiftdists(I - stIdx)
        Next

        For I As Integer = stIdx To stIdx + NoOfPointChange - 1
            If (bac.WPs(I).lon < limitLeftLon) Or (bac.WPs(I).lon > limitRightLon)
Or (bac.WPs(I).lat < limitLowLat) Or (bac.WPs(I).lat > limitUpLat) Then
                Return False
            End If
```

```
            If I < stIdx + NoOfPointChange - 1 Then
                If IsCrossLimits(bac.WPs(I - 1), bac.WPs(I)) = True Then
                    Return False
                End If
            Else
                If IsCrossLimits(bac.WPs(I - 1), bac.WPs(I), bac.WPs(I + 1)) = True
Then
                    Return False
                End If
            End If
        Next

        Return True
    End Function



    Public Sub BacterialSwarming(ByVal NoOfCoupling As Integer, ByVal
neibourDistance As Double, ByVal NoOfRenewing As Integer)
        For I As Integer = 0 To bacNew.Length - 1
            bacNew(I) = 0
        Next

        BacterialSwarmingAndBirthing(NoOfCoupling, neibourDistance)
        BacterialEliminationAndReplicate(NoOfRenewing)
        BacterialReplicate(NoOfRenewing)
    End Sub


    Public Sub BacterialEliminationAndReplicate(ByVal NoOfRenew As Integer)
        Dim count As Integer
        Dim idx As Integer

        For I As Integer = 0 To NoOfRenew
            count = 0
            While count < 4
                count += 1

                idx = qualitySort(bacteria.Length - 1 - Rnd() * bacteria.Length *
0.35)
                If bacNew(idx) = 0 Then
                    bacNew(idx) = 1

                    Dim tmpNewBacOk As Boolean = False
                    For tt As Integer = 0 To 5
                        tmpNewBacOk = RandomInitABacteria(bacteria(idx), idx)
                        If tmpNewBacOk = True Then Exit For
                    Next

                    If tmpNewBacOk = False Then
                        bacteria(idx).CopyFrom(bacteria(qualitySort(0)))
                        RpmShiftPointIdx(idx) = RpmShiftPointIdx(qualitySort(0))
                    End If

                    Exit While
                End If
```

```
            End While
        Next
    End Sub


    Public Sub BacterialReplicate(ByVal NoOfRenew As Integer)
        Dim count As Integer
        Dim idx As Integer
        Dim idxOrg As Integer

        For I As Integer = 0 To NoOfRenew
            count = 0
            While count < 4
                count += 1

                idx = qualitySort(bacteria.Length - 1 - Rnd() * bacteria.Length *
0.35)
                If bacNew(idx) = 0 Then
                    bacNew(idx) = 1
                    idxOrg = qualitySort(Rnd() * bacteria.Length * 0.2)

                    bacteria(idx).CopyFrom(bacteria(idxOrg))
                    RpmShiftPointIdx(idx) = RpmShiftPointIdx(idxOrg)
                    Exit While
                End If
            End While
        Next
    End Sub



    Public Sub ChildBacFromMomAndDad(ByRef child As C8Route, ByVal childIdx As
Integer, ByVal childRPMShiftIdx As Integer, ByRef mom As C8Route, ByRef dad As
C8Route)
        ReDim child.WPs(mom.WPs.Length - 1)

        For I As Integer = 0 To child.WPs.Length - 1
            child.WPs(I) = New C8WayPoint((mom.WPs(I).lat + dad.WPs(I).lat) / 2,
(mom.WPs(I).lon + dad.WPs(I).lon) / 2)
        Next

        ReDim child.travRPM(child.WPs.Length - 2)
        ReDim child.travTime(child.WPs.Length - 2)
        ReDim child.dir(child.WPs.Length - 2)
        ReDim child.dist(child.WPs.Length - 2)
        ReDim child.fuelTpH(child.WPs.Length - 2)

        For I As Integer = 0 To child.WPs.Length - 2
            child.dir(I) = routeCal.Direction_rad(child.WPs(I), child.WPs(I + 1))
            child.dist(I) = routeCal.Distance(child.WPs(I), child.WPs(I + 1))
        Next

        RpmShiftPointIdx(childIdx) = childRPMShiftIdx

        Dim tmpTime As Date
        tmpTime = depTime

        For J As Integer = 0 To child.WPs.Length - 2
            If J <= RpmShiftPointIdx(childIdx) Then
```

```
                child.travRPM(J) = rpm1
                child.fuelTpH(J) = fuelTonsConsume1
            Else
                child.travRPM(J) = rpm2
                child.fuelTpH(J) = fuelTonsConsume2
            End If

            child.travTime(J) = routeCal.CalculateTime(tmpTime, child.WPs(J),
child.WPs(J + 1), child.dist(J), child.dir(J), totalSpdChar, weatherDB,
limitWaveHeight, limitWndSpeed, child.travRPM(J), draft, trm, normSpdChar)
            tmpTime = DateAdd(DateInterval.Hour, child.travTime(J), tmpTime)
        Next

        child.CalculateQuality(TimeAllowToReachDest_Hour, FuelAllowToReachDest_Ton)
    End Sub

    Public Sub BacterialSwarmingAndBirthing(ByVal NoOfCoupling As Integer, ByVal
neibourDistance As Double)
        Dim momIdx As Integer
        Dim dadIdx As Integer
        Dim childIdx As Integer
        Dim neiborIdx() As Integer

        For I As Integer = 0 To NoOfCoupling
            dadIdx = qualitySort(Rnd() * 0.3 * qualitySort.Length)

            ReDim neiborIdx(20)
            Dim count As Integer = 0

            For J As Integer = 0 To bacteria.Length * 0.3 - 1
                If qualitySort(J) <> dadIdx Then
                    If routeCal.QuickRouteToRouteDistance(bacteria(dadIdx),
bacteria(qualitySort(J)), searchDir) < neibourDistance Then
                        neiborIdx(count) = qualitySort(J)
                        count += 1
                    End If
                End If

                If count >= neiborIdx.Length Then
                    Exit For
                End If
            Next

            If count > 0 Then
                momIdx = neiborIdx(Rnd() * (count - 1))
                If momIdx < 0 Then momIdx = 0

                childIdx = qualitySort(bacteria.Length - 1 - Rnd() * bacteria.Length
* 0.3)

                If IsNothing(bacteria(momIdx).WPs) = True Or
IsNothing(bacteria(dadIdx).WPs) = True Then
                    count = count
                End If

                bacteria(childIdx) = New C8Route()

                Dim rmpShiftIdx As Integer = (RpmShiftPointIdx(dadIdx) +
RpmShiftPointIdx(momIdx)) / 2
```

```
                ChildBacFromMomAndDad(bacteria(childIdx), childIdx, rmpShiftIdx,
bacteria(momIdx), bacteria(dadIdx))


                Dim isChildBacValid As Boolean = True
                For J As Integer = 0 To bacteria(childIdx).WPs.Length - 2
                    If IsCrossLimits(bacteria(childIdx).WPs(J),
bacteria(childIdx).WPs(J + 1)) = True Then
                        isChildBacValid = False
                        Exit Sub
                    End If
                Next

                If isChildBacValid = False Then
                    bacteria(childIdx).CopyFrom(bacteria(momIdx))
                    RpmShiftPointIdx(childIdx) = RpmShiftPointIdx(momIdx)
                End If

                bacNew(childIdx) = 1
            End If
        Next
    End Sub

    Public Sub QualitySorting()
        For I As Integer = 0 To bacteria.Length - 1
            qualitySort(I) = I
        Next

        Dim tmp As Integer
        For I As Integer = 0 To bacteria.Length - 2
            For J As Integer = I + 1 To bacteria.Length - 1
                If bacteria(qualitySort(I)).Quality >
bacteria(qualitySort(J)).Quality Then
                    tmp = qualitySort(I)
                    qualitySort(I) = qualitySort(J)
                    qualitySort(J) = tmp
                End If
            Next
        Next

        Dim tmpp As Double = 999999.9
        For I As Integer = 0 To bacteria.Length - 1
            If tmpp > bacteria(I).Quality Then
                tmpp = bacteria(I).Quality
            End If
        Next

        tmp = 1
    End Sub


End Class


Public Class C8RouteSolution
    Public route As C8Route
    Public legTravelTime() As Double
    Public RPMused As Double
```

```
    Public Sub New()
    End Sub

    Public Sub CopyFrom(ByRef org As C8RouteSolution)
        With org
            ReDim legTravelTime(.legTravelTime.Length - 1)
            For I As Integer = 0 To legTravelTime.Length - 1
                legTravelTime(I) = .legTravelTime(I)
            Next

            route = New C8Route(.route)
            RPMused = .RPMused

        End With
    End Sub

End Class



Public Class C8WayPoint
    Public lat As Double
    Public lon As Double

    Public Sub New()
    End Sub

    Public Sub New(ByRef _lat As Double, ByRef _lon As Double)
        lat = _lat
        lon = _lon
    End Sub

    Public Sub New(ByRef org As C8WayPoint)
        lat = org.lat
        lon = org.lon
    End Sub

    Public Sub CopyFrom(ByRef org As C8WayPoint)
        lat = org.lat
        lon = org.lon
    End Sub
End Class


Public Class C8Route
    Public WPs() As C8WayPoint
    Public dist() As Double
    Public dir() As Double
    Public travTime() As Double
    Public travRPM() As Double
    Public fuelTpH() As Double
    Public Quality As Double

    Public Sub New()
    End Sub

    Public Sub New(ByRef org As C8Route)
        ReDim WPs(org.WPs.Length - 1)
        For I As Integer = 0 To WPs.Length - 1
```

```
        WPs(I) = New C8WayPoint(org.WPs(I))
    Next

    ReDim dist(org.dist.Length - 1)
    For I As Integer = 0 To dist.Length - 1
        dist(I) = org.dist(I)
    Next

    ReDim dir(org.dir.Length - 1)
    For I As Integer = 0 To dir.Length - 1
        dir(I) = org.dir(I)
    Next

    ReDim travTime(org.travTime.Length - 1)
    For I As Integer = 0 To travTime.Length - 1
        travTime(I) = org.travTime(I)
    Next


    ReDim travRPM(org.travRPM.Length - 1)
    For I As Integer = 0 To travRPM.Length - 1
        travRPM(I) = org.travRPM(I)
    Next

    ReDim fuelTpH(org.fuelTpH.Length - 1)
    For I As Integer = 0 To fuelTpH.Length - 1
        fuelTpH(I) = org.fuelTpH(I)
    Next


    Quality = org.Quality
End Sub

Public Sub CopyFrom(ByRef org As C8Route)
    ReDim WPs(org.WPs.Length - 1)
    For I As Integer = 0 To WPs.Length - 1
        WPs(I) = New C8WayPoint(org.WPs(I))
    Next

    ReDim dist(org.dist.Length - 1)
    For I As Integer = 0 To dist.Length - 1
        dist(I) = org.dist(I)
    Next

    ReDim dir(org.dir.Length - 1)
    For I As Integer = 0 To dir.Length - 1
        dir(I) = org.dir(I)
    Next


    ReDim travTime(org.travTime.Length - 1)
    For I As Integer = 0 To travTime.Length - 1
        travTime(I) = org.travTime(I)
    Next


    ReDim travRPM(org.travRPM.Length - 1)
    For I As Integer = 0 To travRPM.Length - 1
        travRPM(I) = org.travRPM(I)
```

```
        Next

        ReDim fuelTpH(org.fuelTpH.Length - 1)
        For I As Integer = 0 To fuelTpH.Length - 1
            fuelTpH(I) = org.fuelTpH(I)
        Next

        Quality = org.Quality
    End Sub


    Public Sub CalculateQuality(ByRef TimeAllow_H As Double, ByRef FuelAllow_T As
Double)
        If TimeAllow_H <= 1 Or FuelAllow_T < 1 Then
            CalculateQualityTimeOnly()
        Else
            CalculateQualityWithFuel(TimeAllow_H, FuelAllow_T)
        End If
    End Sub


    Public Sub CalculateQualityTimeOnly()
        Dim retn As Double = 0
        For I As Integer = 0 To travTime.Length - 1
            retn += travTime(I)
        Next

        Quality = retn
    End Sub


    Public Sub CalculateQualityWithFuel(ByRef TimeMustReach As Double, ByRef
targetFuelConsumeTons As Double)
        Dim retn As Double = 0
        Dim tmpTime As Double = 0
        Dim tmpFuel As Double = 0

        For I As Integer = 0 To travTime.Length - 1
            tmpTime += travTime(I)
            tmpFuel += travTime(I) * fuelTpH(I)
        Next

        If tmpTime >= TimeMustReach Then
            Quality = tmpTime / TimeMustReach + 9999.9

        ElseIf tmpTime <= 0.85 * TimeMustReach Then
            Quality = 0.85 + 3 * tmpFuel / targetFuelConsumeTons

        Else
            Quality = tmpTime / TimeMustReach + 3 * tmpFuel / targetFuelConsumeTons
        End If
    End Sub

End Class


Public Class C8WeatherPoint
    Public wnddir As Double
    Public wndspd As Double
```

```
    Public wavdir As Double
    Public wavHei As Double
    Public crrdir As Double
    Public crrspd As Double

    Public Sub New()
    End Sub

    Public Sub New(ByRef org As C8WeatherPoint)
        With org
            wnddir = .wnddir
            wndspd = .wndspd
            wavdir = .wavdir
            wavHei = .wavHei
            crrdir = .crrdir
            crrspd = .crrspd
        End With
    End Sub

    Public Sub CopyFrom(ByRef org As C8WeatherPoint)
        With org
            wnddir = .wnddir
            wndspd = .wndspd
            wavdir = .wavdir
            wavHei = .wavHei
            crrdir = .crrdir
            crrspd = .crrspd
        End With
    End Sub

    Public Sub FromString(ByRef st As String)
        Dim v() As String = Split(st, ";")
        If v.Length <> 8 Then Exit Sub

        wnddir = Val(v(2))
        wndspd = Val(v(3))
        wavdir = Val(v(4))
        wavHei = Val(v(5))
        crrdir = Val(v(6))
        crrspd = Val(v(7))
    End Sub
End Class


Public Class C8WeatherMessage
    Public lat0 As Double = -70
    Public lon0 As Double = -180
    Public dLat As Double = 0.5
    Public dLon As Double = 0.5
    Public latNo As Double
    Public lonNo As Double
    Public wPts(,) As C8WeatherPoint

    Public Sub New()
        latNo = 140 / dLat - 1
        lonNo = 360 / dLon - 1
        ReDim wPts(latNo - 1, lonNo - 1)

        For I As Integer = 0 To latNo - 1
```

```
            For J As Integer = 0 To lonNo - 1
                wPts(I, J) = New C8WeatherPoint()
            Next
        Next
    End Sub

    Public Sub Init(ByVal wdate As Date, ByVal hh As Integer, ByVal mm As Integer)
        Dim filename As String = mGlbWeatherFilename(wdate, hh, mm)
        Init(filename)
    End Sub


    Public Sub Init(ByVal filename As String)
        Dim st As String
        Dim stdat() As String = Nothing

        Try
            Dim stRead As StreamReader = New StreamReader(filename)
            st = stRead.ReadToEnd()
            stRead.Close()

            If st <> "" Then
                stdat = Split(st, vbCrLf)
                If stdat.Length < 3 Then
                    stdat = Nothing
                End If
            End If

        Catch ex As Exception
            stdat = Nothing
        End Try

        Dim v() As String
        Dim latIdx As Integer
        Dim lonIdx As Integer

        If IsNothing(stdat) = False Then
            For I As Integer = 0 To stdat.Length - 1
                v = Split(stdat(I), ";")
                If v.Length = 11 Then
                    latIdx = (Val(v(0)) - lat0) / dLat
                    lonIdx = (Val(v(1)) - lon0) / dLon

                    If latIdx >= 0 And latIdx < latNo And lonIdx >= 0 And lonIdx <
lonNo Then
                        With wPts(latIdx, lonIdx)
                            .wnddir = Val(v(5))
                            .wndspd = Val(v(6))
                            .crrdir = Val(v(7))
                            .crrspd = Val(v(8))
                            .wavdir = Val(v(9))
                            .wavHei = Val(v(10))
                        End With
                    End If

                End If

                ''If v.Length = 8 Then
                ''    latIdx = (Val(v(0)) - lat0) / dLat
```

```
                ''    lonIdx = (Val(v(1)) - lon0) / dLon

                ''    If latIdx >= 0 And latIdx < latNo And lonIdx >= 0 And lonIdx <
lonNo Then
                ''        With wPts(latIdx, lonIdx)
                ''            .wnddir = Val(v(2))
                ''            .wndspd = Val(v(3))
                ''            .crrdir = Val(v(4))
                ''            .crrspd = Val(v(5))
                ''            .wavdir = Val(v(6))
                ''            .wavHei = Val(v(7))
                ''        End With
                ''    End If

                ''End If


            Next
        End If
    End Sub

    Public Sub GetWeatherPoint(ByVal lat As Double, ByVal lon As Double, ByRef
weatherPoint As C8WeatherPoint)
        Dim latIdx As Integer
        Dim lonIdx As Integer

        'If lat < lat0 Then
        '    latIdxDown = 0
        '    latIdxUpp = 0

        'Else
        '    latIdxDown = Math.Floor((lat - lat0) / dLat)
        '    If latIdxDown >= latNo - 1 Then
        '        latIdxDown = latNo - 1
        '        latIdxUpp = latNo - 1
        '    Else
        '        latIdxUpp = latIdxDown + 1
        '    End If
        'End If

        'If lon < -180 Then
        '    lon += 360
        'ElseIf lon > 180 Then
        '    lon -= 360
        'End If


        If lon < lon0 Then
            lonIdx = 0
        Else
            lonIdx = (lon - lon0) / dLon
            If lonIdx >= lonNo - 1 Then lonIdx = lonNo - 1
        End If

        If lat < lat0 Then
            latIdx = 0
        Else
            latIdx = (lat - lat0) / dLat
            If latIdx >= latNo - 1 Then latIdx = latNo - 1
```

```
        End If

        weatherPoint.CopyFrom(wPts(latIdx, lonIdx))
    End Sub

End Class


Public Class C8BaseWeatherDB
    Public timeArray() As Date
    Public weatherMessages() As C8WeatherMessage
    Public routeCal As C8RouteMathCal

    Public Sub New()
        routeCal = New C8RouteMathCal()
    End Sub

    Public Function DateFromString(ByRef yyyymmddhhMM As String) As Date
        Dim y As Integer = Val(yyyymmddhhMM.Substring(0, 4))
        Dim m As Integer = Val(yyyymmddhhMM.Substring(4, 2))
        Dim d As Integer = Val(yyyymmddhhMM.Substring(6, 2))
        Dim h As Integer = Val(yyyymmddhhMM.Substring(8, 2))
        Dim min As Integer = Val(yyyymmddhhMM.Substring(10, 2))

        Dim retn As Date = New Date(y, m, d, h, min, 0)
        Return retn
    End Function


    Public Function InitAvailableBaseWeatherData(ByVal startTime As Date, Optional
ByVal timeSpanDays As Double = 10) As String
        Dim filenames() As String = IO.Directory.GetFiles(mGlbWeatherFolderName())
        If IsNothing(filenames) = True Then Return Nothing
        Dim wDateStrings(filenames.Length - 1) As String

        Dim count As Integer = 0
        Dim tmpStringTime As String
        Dim tmpIdx As Integer
        Dim tmpStart As Double = startTime.Year * 10 ^ 8 + startTime.Month * 10 ^ 6
+ startTime.Day * 10 ^ 4
        Dim tmpEnd As Double = tmpStart + timeSpanDays * 10 ^ 4

        For I As Integer = 0 To filenames.Length - 1
            tmpIdx = filenames(I).LastIndexOf(IO.Path.DirectorySeparatorChar)
            tmpStringTime = filenames(I).Substring(tmpIdx + 2)

            If filenames(I).Chars(tmpIdx + 1) = "w" And tmpStringTime.Length = 12
Then
                If tmpStart <= Val(tmpStringTime) And tmpEnd >= Val(tmpStringTime)
Then

                    filenames(count) = filenames(I)
                    wDateStrings(count) = tmpStringTime
                    count += 1

                End If
            End If
        Next
```

```
        If count > 0 Then
            ReDim Preserve filenames(count - 1)
            ReDim Preserve wDateStrings(count - 1)

            For I As Integer = 0 To filenames.Length - 1
                weatherMessages(I) = New C8WeatherMessage()
                weatherMessages(I).Init(filenames(I))
                timeArray(I) = DateFromString(wDateStrings(I))
            Next

            Return 1
        Else
            weatherMessages = Nothing
            timeArray = Nothing
            Return -1
        End If
    End Function


    Public Sub GetWeatherMessage(ByRef atTime As Date, ByRef resWeatherMessage As
C8WeatherMessage)
        Dim idx As Integer = -1
        Dim idxUp As Integer

        If atTime <= timeArray(0) Then
            idx = 0
            idxUp = 0

        ElseIf atTime >= timeArray(timeArray.Length - 1) Then
            idx = timeArray.Length - 1
            idxUp = idx

        Else
            For I As Integer = 0 To timeArray.Length - 2
                If atTime >= timeArray(I) And atTime <= timeArray(I + 1) Then
                    idx = I
                    idxUp = idx + 1
                End If
            Next
        End If

        Dim lat As Double
        Dim lon As Double

        If idx = idxUp Then
            For I As Integer = 0 To resWeatherMessage.latNo - 1
                For J As Integer = 0 To resWeatherMessage.lonNo - 1
                    lat = resWeatherMessage.lat0 + I * 0.5
                    lon = resWeatherMessage.lat0 + J * 0.5
                    If lon > 180 Then lon -= 180

                    weatherMessages(idx).GetWeatherPoint(lat, lon,
resWeatherMessage.wPts(I, J))
                Next
            Next

        Else
            Dim retnW1 As C8WeatherPoint = New C8WeatherPoint()
```

```
            Dim retnW2 As C8WeatherPoint = New C8WeatherPoint()
            Dim fract As Double = DateDiff(DateInterval.Hour, timeArray(idx),
atTime) / DateDiff(DateInterval.Hour, atTime, timeArray(idxUp))

            For I As Integer = 0 To resWeatherMessage.latNo - 1
                For J As Integer = 0 To resWeatherMessage.lonNo - 1
                    lat = resWeatherMessage.lat0 + I * 0.5
                    lon = resWeatherMessage.lat0 + J * 0.5
                    If lon > 180 Then lon -= 180

                    weatherMessages(idx).GetWeatherPoint(lat, lon, retnW1)
                    weatherMessages(idxUp).GetWeatherPoint(lat, lon, retnW2)
                    resWeatherMessage.wPts(I, J) = WeatherInterpolate(retnW1,
retnW2, fract)
                Next
            Next
        End If
    End Sub


    Public Sub GetWeatherPoint(ByRef atTime As Date, ByRef lat As Double, ByRef lon
As Double, ByRef weatherPoint As C8WeatherPoint)
        Dim idx As Integer = -1
        Dim idxUp As Integer

        If atTime <= timeArray(0) Then
            idx = 0
            idxUp = 0

        ElseIf atTime >= timeArray(timeArray.Length - 1) Then
            idx = timeArray.Length - 1
            idxUp = idx

        Else
            For I As Integer = 0 To timeArray.Length - 2
                If atTime >= timeArray(I) And atTime <= timeArray(I + 1) Then
                    idx = I
                    idxUp = idx + 1
                End If
            Next
        End If

        If idx = idxUp Then
            weatherMessages(idx).GetWeatherPoint(lat, lon, weatherPoint)
        Else
            Dim retnW1 As C8WeatherPoint = New C8WeatherPoint()
            Dim retnW2 As C8WeatherPoint = New C8WeatherPoint()
            Dim fract As Double = DateDiff(DateInterval.Hour, timeArray(idx),
atTime) / DateDiff(DateInterval.Hour, atTime, timeArray(idxUp))
            weatherMessages(idx).GetWeatherPoint(lat, lon, retnW1)
            weatherMessages(idxUp).GetWeatherPoint(lat, lon, retnW2)
            weatherPoint = WeatherInterpolate(retnW1, retnW2, fract)
        End If
    End Sub

    Public Function WeatherInterpolate(ByRef w1 As C8WeatherPoint, ByRef w2 As
C8WeatherPoint, ByRef fract As Double) As C8WeatherPoint
        Dim retn As C8WeatherPoint = New C8WeatherPoint()
        With retn
```

```
            .wnddir = DirectionInterpolate(w1.wnddir, w2.wnddir, fract)
            .wndspd = ValueInterpolate(w1.wndspd, w2.wndspd, fract)

            .wavdir = DirectionInterpolate(w1.wavdir, w2.wavdir, fract)
            .wavHei = ValueInterpolate(w1.wavHei, w2.wavHei, fract)

            .crrdir = DirectionInterpolate(w1.crrdir, w2.crrdir, fract)
            .crrspd = ValueInterpolate(w1.crrspd, w2.crrspd, fract)
        End With

        Return retn
    End Function


    Public Function ValueInterpolate(ByRef v1 As Double, ByRef v2 As Double, ByRef
fract As Double) As Double
        Dim retn As Double = v1 + (v2 - v1) * fract
        Return retn
    End Function

    Public Function DirectionInterpolate(ByRef dir1 As Double, ByRef dir2 As Double,
ByRef fract As Double) As Double
        Dim retn As Double = dir1 + routeCal.DirectionDiff_deg(dir1, dir2) * fract
        If retn < 0 Then retn += 360
        If retn >= 360 Then retn -= 360
        Return retn
    End Function

End Class


Public Class C8WeatherDB
    Public timeStep As Double = 2
    Public timeArray() As Date
    Public weatherMessages() As C8WeatherMessage

    Public mathCal As C8RouteMathCal

    Public Sub New()
        mathCal = New C8RouteMathCal()
    End Sub


    Public Sub InitMessage(ByRef lat As Double, ByRef lon As Double, ByRef _latNo As
Integer, ByRef _lonNo As Integer)
        For I As Integer = 0 To weatherMessages.Length - 1
            weatherMessages(I) = New C8WeatherMessage()
            weatherMessages(I).lat0 = lat
            weatherMessages(I).lon0 = lon
            weatherMessages(I).dLat = 0.5
            weatherMessages(I).dLon = 0.5
            weatherMessages(I).latNo = _latNo
            weatherMessages(I).lonNo = _lonNo
            ReDim weatherMessages(I).wPts(_latNo - 1, _lonNo - 1)

            For latIdx As Integer = 0 To _latNo - 1
                For lonIdx As Integer = 0 To _lonNo - 1
                    weatherMessages(I).wPts(latIdx, lonIdx) = New C8WeatherPoint()
                Next
```

```
            Next
        Next
    End Sub


    Public Sub GetWeatherPoint(ByRef atTime As Date, ByRef lat As Double, ByRef lon
As Double, ByRef weatherPoint As C8WeatherPoint)
        Dim idx As Integer = -1

        If atTime <= timeArray(0) Then
            idx = 0
        ElseIf atTime >= timeArray(timeArray.Length - 1) Then
            idx = timeArray.Length - 1
        Else
            idx = DateDiff(DateInterval.Hour, timeArray(0), atTime) / timeStep
        End If

        weatherMessages(idx).GetWeatherPoint(lat, lon, weatherPoint)
    End Sub


    Public Sub InitAMessageFromTwoBaseMessage(ByRef wmsg1 As C8WeatherMessage, ByRef
t1 As Date, ByRef wmsg2 As C8WeatherMessage, ByRef t2 As Date, ByRef wmsgRes As
C8WeatherMessage, ByRef tRes As Date)
        Dim fac As Double
        If t1 = t2 Then
            If tRes < t1 Then
                fac = 0
            Else
                fac = 1
            End If
        Else
            fac = DateDiff(DateInterval.Minute, t1, tRes) /
DateDiff(DateInterval.Minute, t1, t2)
        End If

        Dim latIdxBase(wmsgRes.latNo - 1) As Integer
        Dim lonIdxBase(wmsgRes.lonNo - 1) As Integer
        Dim tmpLon As Double

        For I As Integer = 0 To wmsgRes.latNo - 1
            latIdxBase(I) = (wmsgRes.lat0 + wmsgRes.dLat * I - wmsg1.lat0) /
wmsg1.dLat - 1
            If latIdxBase(I) < 0 Then latIdxBase(I) = 0
            If latIdxBase(I) > wmsg1.latNo - 1 Then latIdxBase(I) = wmsg1.latNo - 1
        Next

        For I As Integer = 0 To wmsgRes.lonNo - 1
            tmpLon = wmsgRes.lon0 + wmsgRes.dLon * I
            If tmpLon > 180 Then tmpLon -= 360

            lonIdxBase(I) = (tmpLon - wmsg1.lon0) / wmsg1.dLon - 1
            If lonIdxBase(I) < 0 Then lonIdxBase(I) = 0
            If lonIdxBase(I) > wmsg1.lonNo - 1 Then lonIdxBase(I) = wmsg1.lonNo - 1
        Next

        For I As Integer = 0 To wmsgRes.latNo - 1
            For J As Integer = 0 To wmsgRes.lonNo - 1
```

```
                mathCal.WeatherPointInterpolate(wmsg1.wPts(latIdxBase(I),
lonIdxBase(J)), wmsg2.wPts(latIdxBase(I), lonIdxBase(J)), fac, wmsgRes.wPts(I, J))
            Next
        Next
    End Sub


    Public Sub InitWholeMessagesFromBaseWeatherMessage(ByRef baseWeatherBD As
C8BaseWeatherDB, ByRef startTime As Date, ByRef lat0 As Double, ByRef latNo As
Integer, ByRef lon0 As Double, ByRef lonNo As Integer, Optional ByVal timeSpanInHour
As Double = 240, Optional ByVal timeInterval As Double = 2)
        Dim tmpNo As Integer = timeSpanInHour / timeInterval
        ReDim weatherMessages(tmpNo)
        ReDim timeArray(tmpNo)

        For I As Integer = 0 To timeArray.Length - 1
            timeArray(I) = DateAdd(DateInterval.Hour, timeInterval * I, startTime)
        Next
        InitMessage(lat0, lon0, latNo, lonNo)

        If IsNothing(baseWeatherBD.weatherMessages) = True Then Exit Sub

        With baseWeatherBD
            For I As Integer = 0 To weatherMessages.Length - 1
                If timeArray(I) <= .timeArray(0) Then
                    InitAMessageFromTwoBaseMessage(.weatherMessages(0),
.timeArray(0), .weatherMessages(0), .timeArray(0), weatherMessages(I), timeArray(I))
                ElseIf timeArray(I) >= .timeArray(baseWeatherBD.timeArray.Length -
1) Then

InitAMessageFromTwoBaseMessage(.weatherMessages(.weatherMessages.Length - 1),
.timeArray(.weatherMessages.Length - 1), .weatherMessages(.weatherMessages.Length -
1), .timeArray(.weatherMessages.Length - 1), weatherMessages(I), timeArray(I))
                Else

                    For J As Integer = 0 To .timeArray.Length - 2
                        If timeArray(I) >= .timeArray(J) And timeArray(I) <
.timeArray(J + 1) Then
                            InitAMessageFromTwoBaseMessage(.weatherMessages(J),
.timeArray(J), .weatherMessages(J + 1), .timeArray(J + 1), weatherMessages(I),
timeArray(I))
                            Exit For
                        End If
                    Next
                End If
            Next
        End With

    End Sub

End Class




Public Class C8LimitLineSet
    Public limitlines() As C8LimitLine
    Public matCal As C8RouteMathCal
```

```
    Public Sub New()
        matCal = New C8RouteMathCal()
    End Sub


''=============================================================================
=
    Public Sub Init(ByRef minLat As Double, ByRef maxLat As Double, ByRef minLon As
Double, ByRef maxLon As Double)
        Dim filename As String = mGlbLimitLinesFilename(mGlbShipInfo.IMO)
        Dim stDat() As String = mGlbReadStringsFromFile(filename)
        InitLimits(stDat, minLat, maxLat, minLon, maxLon)
    End Sub


    Public Sub InitLimits(ByRef st() As String, ByRef minLat As Double, ByRef maxLat
As Double, ByRef minLon As Double, ByRef maxLon As Double)
        If IsNothing(st) = True Then
            limitlines = Nothing
        Else
            Dim tmpLims(st.Length - 1) As C8LimitLine
            For I As Integer = 0 To st.Length - 1
                tmpLims(I) = New C8LimitLine(st(I))
            Next

            For I As Integer = 0 To st.Length - 1
                If tmpLims(I).Id = "" Then
                    tmpLims(I).isDeleted = 1
                End If
            Next

            For I As Integer = 1 To st.Length - 1
                For J As Integer = 0 To I - 1
                    If String.Compare(tmpLims(I).Id, tmpLims(J).Id, True) = 0 Then
'And tmpperf(I).hour = tmpperf(J).hour Then
                        tmpLims(J).isDeleted = 1
                    End If
                Next
            Next

            Dim count As Integer = 0
            For I As Integer = 0 To tmpLims.Length - 1
                If tmpLims(I).isDeleted <> 1 Then
                    count += 1
                End If
            Next

            If count = 0 Then
                limitlines = Nothing
            Else

                ReDim limitlines(count - 1)
                count = 0
                For I As Integer = 0 To tmpLims.Length - 1
                    If tmpLims(I).isDeleted <> 1 Then
                        limitlines(count) = New C8LimitLine(tmpLims(I))
                        count += 1
```

```
                    End If
                Next
            End If
        End If

        If IsNothing(limitlines) = False Then
            Dim wp1 As C8WayPoint = New C8WayPoint(minLat, minLon)
            Dim wp2 As C8WayPoint = New C8WayPoint(maxLat, minLon)
            Dim wp3 As C8WayPoint = New C8WayPoint(maxLat, maxLon)
            Dim wp4 As C8WayPoint = New C8WayPoint(minLat, maxLon)
            Dim limCount As Integer = 0
            Dim iscross As Integer

            For I As Integer = 0 To limitlines.Length - 1
                iscross = -1

                For J As Integer = 0 To limitlines(I).wps.Length - 1
                    With limitlines(I).wps(J)
                        If .lat >= minLat And .lat <= maxLat And .lon <= maxLon And
.lon >= minLon Then
                            iscross = 1
                            Exit For
                        End If
                    End With
                Next

                If iscross <= 0 Then
                    If limitlines(I).IsCross(wp1, wp2, matCal) = True Then
                        iscross = 1
                    End If
                End If

                If iscross <= 0 Then
                    If limitlines(I).IsCross(wp2, wp3, matCal) = True Then
                        iscross = 1
                    End If
                End If

                If iscross <= 0 Then
                    If limitlines(I).IsCross(wp3, wp4, matCal) = True Then
                        iscross = 1
                    End If
                End If

                If iscross <= 0 Then
                    If limitlines(I).IsCross(wp4, wp1, matCal) = True Then
                        iscross = 1
                    End If
                End If


                If iscross > 0 Then
                    If limCount <> I Then
limitlines(limCount).CopyFrom(limitlines(I))
                    limCount += 1
                End If
            Next

            If limCount > 0 Then
```

```
                ReDim Preserve limitlines(limCount - 1)
            Else
                limitlines = Nothing
            End If
        End If
    End Sub


''=============================================================================
=
    Public Sub Init()
        Dim filename As String = mGlbLimitLinesFilename()
        Dim stDat() As String = mGlbReadStringsFromFile(filename)

        If mGlbLonLeft >= mGlbLonRight Or mGlbLatDown >= mGlbLatUp Then
            InitLimits(stDat)
        Else
            InitLimits(stDat, mGlbLatDown, mGlbLatUp, mGlbLonLeft, mGlbLonRight)
        End If
    End Sub


    Public Sub InitLimits(ByRef st() As String)
        If IsNothing(st) = True Then
            limitlines = Nothing
        Else
            Dim tmpLims(st.Length - 1) As C8LimitLine
            For I As Integer = 0 To st.Length - 1
                tmpLims(I) = New C8LimitLine(st(I))
            Next

            For I As Integer = 0 To st.Length - 1
                If tmpLims(I).Id = "" Then
                    tmpLims(I).isDeleted = 1
                End If
            Next

            For I As Integer = 1 To st.Length - 1
                For J As Integer = 0 To I - 1
                    If String.Compare(tmpLims(I).Id, tmpLims(J).Id, True) = 0 Then
'And tmpperf(I).hour = tmpperf(J).hour Then
                        tmpLims(J).isDeleted = 1
                    End If
                Next
            Next

            Dim count As Integer = 0
            For I As Integer = 0 To tmpLims.Length - 1
                If tmpLims(I).isDeleted <> 1 Then
                    count += 1
                End If
            Next

            If count = 0 Then
                limitlines = Nothing
            Else

                ReDim limitlines(count - 1)
```

```
            count = 0
            For I As Integer = 0 To tmpLims.Length - 1
                If tmpLims(I).isDeleted <> 1 Then
                    limitlines(count) = New C8LimitLine(tmpLims(I))
                    count += 1
                End If
            Next
        End If
    End If
End Sub



Public Sub AppendToLimitsFile(ByRef tmpperf As C8LimitLine)
    Dim st As String = mGLBAppendToTotalRecordFile(tmpperf.ToStringFormat)

    Dim filename As String = mGlbLimitLinesFilename()
    mGlbAppendDataToFile(filename, st)
End Sub


Public Function Change(ByRef tmpLimit As C8LimitLine) As Integer
    AppendToLimitsFile(tmpLimit)
    For I As Integer = 0 To limitlines.Length - 1
        If limitlines(I).Id = tmpLimit.Id Then
            limitlines(I).CopyFrom(tmpLimit)
            Return I
        End If
    Next
    Return -1
End Function


Public Function Delete(ByVal idx As Integer) As Integer
    If idx < 0 Or idx >= limitlines.Length Then Return -1

    limitlines(idx).isDeleted = 1
    AppendToLimitsFile(limitlines(idx))

    If idx > -1 And limitlines.Length = 1 Then
        limitlines = Nothing
        Return 0
    End If

    If idx > -1 Then
        For I As Integer = idx To limitlines.Length - 2
            limitlines(I).CopyFrom(limitlines(I + 1))
        Next
        ReDim Preserve limitlines(limitlines.Length - 2)
    End If

    Return idx
End Function



Public Function Add(ByRef tmpLimitLine As C8LimitLine) As Integer
    AppendToLimitsFile(tmpLimitLine)
    Dim tmp As Integer = 0
```

```
        If IsNothing(limitlines) = False Then
            tmp = limitlines.Length
        End If

        ReDim Preserve limitlines(tmp)
        limitlines(tmp) = New C8LimitLine(tmpLimitLine)
        Return tmp
    End Function



    Public Function IsNewSet(ByRef id As String) As Boolean
        If IsNothing(limitlines) = True Then Return 1
        For I As Integer = 0 To limitlines.Length - 1
            If String.Compare(limitlines(I).Id, id, True) = 0 Then
                Return -1
            End If
        Next
        Return 1
    End Function

    Public Function ProduceNewName() As String
        If IsNothing(limitlines) = False Then Return "line_0001"

        Dim retn As String = "line"
        Dim stIdx As Integer = 1
        Dim isNew As Boolean = False

        While isNew = False
            isNew = True
            retn = "line_" & stIdx.ToString.PadLeft(4, "0")

            For I As Integer = 0 To limitlines.Length - 1
                If String.Compare(retn, limitlines(I).Id, True) = 0 Then
                    isNew = False
                    Exit For
                End If
            Next
        End While

        Return retn
    End Function
End Class



Public Class C8LimitLine
    Public Id As String
    Public isDeleted As Integer
    Public wps() As C8WayPoint
    Public MaxLat As Double
    Public MinLat As Double
    Public MaxLon As Double
    Public MinLon As Double

    Public Sub New()
    End Sub
```

```
Public Sub New(ByRef st As String)
    Dim v() As String = Split(st, ";")
    If v.Length < 5 Then
        Id = ""
        isDeleted = 1
        Exit Sub

    Else
        Id = v(3)
        isDeleted = Val(v(4))

        If isDeleted = 1 Then
            wps = Nothing
            Exit Sub
        End If

        If v.Length < 13 Then
            Id = ""
            isDeleted = 1
            Exit Sub
        End If

        MaxLat = Val(v(5))
        MinLat = Val(v(6))
        MaxLon = Val(v(7))
        MinLon = Val(v(8))

        Dim tmp As Integer = (v.Length - 9) / 2
        ReDim wps(tmp - 1)
        For I As Integer = 0 To tmp - 1
            wps(I) = New C8WayPoint(Val(v(9 + I * 2)), Val(v(9 + I * 2 + 1)))
        Next
    End If
End Sub


Public Function ToStringFormat()
    Dim retn As String = "limitline;" & Id & ";" & isDeleted
    If isDeleted = 1 Then
        Return retn
    Else
        retn = retn & ";" & MaxLat & ";" & MinLat & ";" & MaxLon & ";" & MinLon
        For I As Integer = 0 To wps.Length - 1
            retn = retn & ";" & wps(I).lat & ";" & wps(I).lon
        Next

    End If
    Return retn
End Function

Public Sub New(ByRef org As C8LimitLine)
    ReDim wps(org.wps.Length - 1)
    For I As Integer = 0 To wps.Length - 1
        wps(I) = New C8WayPoint(org.wps(I))
    Next

    Id = org.Id
    MaxLat = org.MaxLat
    MinLat = org.MinLat
```

```
        MaxLon = org.MaxLon
        MinLon = org.MinLon
        isDeleted = org.isDeleted
    End Sub

    Public Sub CopyFrom(ByRef org As C8LimitLine)
        ReDim wps(org.wps.Length - 1)
        For I As Integer = 0 To wps.Length - 1
            wps(I) = New C8WayPoint(org.wps(I))
        Next

        Id = org.Id
        MaxLat = org.MaxLat
        MinLat = org.MinLat
        MaxLon = org.MaxLon
        MinLon = org.MinLon
        isDeleted = org.isDeleted
    End Sub


    Public Function IsCross(ByRef wp1 As C8WayPoint, ByRef wp2 As C8WayPoint, ByRef
mathcal As C8RouteMathCal) As Boolean
        If mathcal.Max(wp1.lat, wp2.lat) < MinLat Then Return False
        If mathcal.Min(wp1.lat, wp2.lat) > MaxLat Then Return False
        If mathcal.Max(wp1.lon, wp2.lon) < MinLon Then Return False
        If mathcal.Min(wp1.lon, wp2.lon) > MaxLon Then Return False

        For I As Integer = 0 To wps.Length - 2
            If mathcal.doIntersect(wp1, wp2, wps(I), wps(I + 1)) = True Then
                Return True
            End If
        Next
        Return False
    End Function


    Public Sub GetLimitLonLat()
        MaxLon = -9999
        MinLon = 9999
        MinLat = 9999
        MaxLat = -9999
        For I As Integer = 0 To wps.Length - 1
            If MaxLon < wps(I).lon Then MaxLon = wps(I).lon
            If MinLon > wps(I).lon Then MinLon = wps(I).lon
            If MaxLat < wps(I).lat Then MaxLat = wps(I).lat
            If MinLat > wps(I).lat Then MinLat = wps(I).lat

        Next
    End Sub
End Class

Public Class C8RouteMathCal
    Public latArray() As Double
    Public departArray() As Double
    Dim latStep As Double = 0.001
    Public toRad As Double = Math.PI / 180

    Public Sub New()
        CreateDepartArray()
```

```
    End Sub

    Public Function QuickRouteToRouteDistance(ByRef rte1 As C8Route, ByRef rte2 As
C8Route, ByRef searchdir As Integer) As Double
        Dim retn As Double = 0
        Select Case searchdir
            Case 0
                For I As Integer = 0 To rte1.WPs.Length - 1
                    retn += rte1.WPs(I).lat - rte2.WPs(I).lat
                Next
            Case 1
                For I As Integer = 0 To rte1.WPs.Length - 1
                    retn += rte1.WPs(I).lon - rte2.WPs(I).lon
                Next
            Case Else
                For I As Integer = 0 To rte1.WPs.Length - 1
                    retn += (rte1.WPs(I).lat - rte2.WPs(I).lat) * 5 / 3
                Next
        End Select

        Return Math.Abs(retn)
    End Function


    Public Function FindCrossingPoint(ByRef P11 As C8WayPoint, ByRef P12 As
C8WayPoint, ByRef ChkP21 As C8WayPoint, ByRef ChkP22 As C8WayPoint) As C8WayPoint
        Dim retn As C8WayPoint = New C8WayPoint()
        retn = FindCrossingPoint(P11.lon, P11.lat, P12.lon, P12.lat, ChkP21.lon,
ChkP21.lat, ChkP22.lon, ChkP22.lat)

        Dim tmpDX As Double = Math.Round((retn.lon - ChkP21.lon) * (retn.lon -
ChkP22.lon), 7)
        Dim tmpDY As Double = Math.Round((retn.lat - ChkP21.lat) * (retn.lat -
ChkP22.lat), 7)

        If retn.lon > -999999.9 Then
            If tmpDX > 0 Or tmpDY > 0 Then
                retn.lon = -999999.9
                retn.lat = -999999.9
            End If
        End If

        Return retn
    End Function


    Public Function FindCrossingPoint(ByRef x1 As Double, ByRef y1 As Double, ByRef
x2 As Double, ByRef y2 As Double, ByRef x3 As Double, ByRef y3 As Double, ByRef x4
As Double, ByRef y4 As Double) As C8WayPoint
        Dim retn As C8WayPoint = New C8WayPoint()
        Dim x12 As Double = x1 - x2
        Dim x34 As Double = x3 - x4
        Dim y12 As Double = y1 - y2
        Dim y34 As Double = y3 - y4

        Dim c As Double = x12 * y34 - y12 * x34

        If (Math.Abs(c) < 0.00000001) Then
            retn.lon = -999999.9
```

```
            retn.lat = -999999.9
        Else
            Dim a = x1 * y2 - y1 * x2
            Dim b = x3 * y4 - y3 * x4
            retn.lon = (a * x34 - b * x12) / c
            retn.lat = (a * y34 - b * y12) / c

            If x3 = x4 Then retn.lon = x3
            If y3 = y4 Then retn.lat = y3
        End If

        Return retn
    End Function


    ' Given three colinear points p, q, r, the function checks if
    ' point q lies on line segment 'pr'
    Public Function Max(ByRef a As Double, ByRef b As Double) As Double
        If a < b Then Return b
        Return a
    End Function

    Public Function Min(ByRef a As Double, ByRef b As Double) As Double
        If a < b Then Return a
        Return b
    End Function

    Public Function onSegment(ByRef p As C8WayPoint, ByRef q As C8WayPoint, ByRef r
As C8WayPoint) As Boolean
        If q.lon <= Max(p.lon, r.lon) And q.lon >= Min(p.lon, r.lon) And q.lat <=
Max(p.lat, r.lat) And q.lat >= Min(p.lat, r.lat) Then
            Return True
        End If
        Return False
    End Function

    ' To find orientation of ordered triplet (p, q, r).
    ' The function returns following values
    ' 0 --> p, q and r are colinear
    ' 1 --> Clockwise
    ' 2 --> Counterclockwise
    Public Function orientation(ByRef p As C8WayPoint, ByRef q As C8WayPoint, ByRef
r As C8WayPoint) As Integer
        ' See https:'www.geeksforgeeks.org/orientation-3-ordered-Dim as C8WayPoints/
        ' for details of below formula.
        Dim vv As Double = (q.lat - p.lat) * (r.lon - q.lon) - (q.lon - p.lon) *
(r.lat - q.lat)

        If (vv = 0) Then Return 0 'colinear

        If vv > 0 Then
            Return 1   'clock wise
        Else
            Return 2    'counter-clock
        End If
    End Function

    ' The main function that returns true if line segment 'p1q1'
    ' and 'p2q2' intersect.
```

```
    Public Function doIntersect(ByRef p1 As C8WayPoint, ByRef q1 As C8WayPoint,
ByRef p2 As C8WayPoint, ByRef q2 As C8WayPoint) As Boolean
        ' Find the four orientations needed for general and
        ' special cases
        Dim o1 As Integer = orientation(p1, q1, p2)
        Dim o2 As Integer = orientation(p1, q1, q2)
        Dim o3 As Integer = orientation(p2, q2, p1)
        Dim o4 As Integer = orientation(p2, q2, q1)

        ' General case
        If (o1 <> o2 And o3 <> o4) Then Return True

        ' Special Cases
        ' p1, q1 and p2 are colinear and p2 lies on segment p1q1
        If (o1 = 0 And onSegment(p1, p2, q1)) Then Return True

        ' p1, q1 and q2 are colinear and q2 lies on segment p1q1
        If (o2 = 0 And onSegment(p1, q2, q1)) Then Return True

        ' p2, q2 and p1 are colinear and p1 lies on segment p2q2
        If (o3 = 0 And onSegment(p2, p1, q2)) Then Return True

        ' p2, q2 and q1 are colinear and q1 lies on segment p2q2
        If (o4 = 0 And onSegment(p2, q1, q2)) Then Return True

        Return False ' Doesn't fall in any of the above cases
    End Function


    'Public Sub CreateDepartArray()
    '    ReDim latArray(140000)
    '    ReDim departArray(140000)
    '    Dim tmpAbsLat As Double

    '    latArray(0) = -70

    '    For I As Integer = 0 To latArray.Length - 1
    '        latArray(I) = latArray(0) + latStep * I
    '        tmpAbsLat = Math.Abs(latArray(I))

    '        If latArray(I) <= 0 Then
    '            departArray(I) = -Departure(tmpAbsLat)
    '        Else
    '            departArray(I) = Departure(tmpAbsLat)
    '        End If

    '        'departArray(I) = Departure(latArray(I))
    '    Next

    '    'Dim ddd As Double = departArray(140000)
    'End Sub

    'Public Function GetDepartute(ByRef lat As Double) As Double
    '    If lat < latArray(0) Then Return departArray(0)
    '    If lat >= latArray(latArray.Length - 1) Then Return
departArray(departArray.Length - 1)

    '    Dim idx As Integer = Math.Floor((lat - latArray(0)) / latStep)
```

```
    '    Return (departArray(idx) + (departArray(idx + 1) - departArray(idx)) * (lat
- latArray(idx)) / latStep)
    'End Function


    Public Sub CreateDepartArray()
        ReDim latArray(70000)
        ReDim departArray(70000)
        'Dim tmpAbsLat As Double

        latArray(0) = 0

        For I As Integer = 0 To latArray.Length - 1
            latArray(I) = latArray(0) + latStep * I
            departArray(I) = Departure(latArray(I))
        Next

        'Dim ddd As Double = departArray(140000)
    End Sub

    Public Function GetDepartute(ByRef lat_deg As Double) As Double
        Dim lat As Double = Math.Abs(lat_deg)
        Dim retn As Double
        If lat >= latArray(latArray.Length - 1) Then
            retn = departArray(departArray.Length - 1)
        Else
            Dim idx As Integer = Math.Floor((lat - latArray(0)) / latStep)
            retn = departArray(idx) + (departArray(idx + 1) - departArray(idx)) *
(lat - latArray(idx)) / latStep
        End If

        If lat_deg > 0 Then
            Return retn
        Else
            Return -retn
        End If
    End Function


    Public Function Departure(ByRef lat_deg As Double) As Double
        Dim retn As Double = 7915.7045 * Math.Log10(Math.Tan(Math.PI / 4 + lat_deg *
Math.PI / 360))
        Return retn
    End Function


    Public Function Distance(ByRef wp1 As C8WayPoint, ByRef wp2 As C8WayPoint) As
Double
        If wp1.lat = wp2.lat Then
            Return (Math.Abs(wp1.lon - wp2.lon) * 60 * Math.Cos(wp1.lat * toRad))
        End If

        Dim retn As Double = (wp2.lat - wp1.lat) * 60
        Dim C As Double = Direction_rad(wp1, wp2)

        If Math.Abs(retn / Math.Cos(C)) > 999999.0 Then
            Dim dd As Double = 1
        End If
```

```
        Return (Math.Abs(retn / Math.Cos(C)))
    End Function

    Public Function Direction_rad(ByRef wp1 As C8WayPoint, ByRef wp2 As C8WayPoint)
As Double
        Dim tmpX As Double = (wp2.lon - wp1.lon)
        If tmpX > 180 Then
            tmpX -= 360
        ElseIf tmpX < -180 Then
            tmpX += 360
        End If

        tmpX *= 60

        Dim tmpY As Double = GetDepartute(wp2.lat) - GetDepartute(wp1.lat)
        Dim retn As Double = Math.Atan2(tmpX, tmpY)
        If retn < 0 Then retn += 2 * Math.PI
        Return retn
    End Function



    Public Function NewPosition(ByRef orgWP As C8WayPoint, ByRef C_deg As Double,
ByRef Dist_NM As Double) As C8WayPoint
        Dim retn As C8WayPoint = New C8WayPoint()
        Dim cs As Double = Math.Cos(C_deg * toRad)
        Dim sn As Double = Math.Cos(C_deg * toRad)
        retn.lat += Dist_NM * cs / 60
        retn.lon += Dist_NM * sn * Math.Cos((retn.lat + orgWP.lat) * toRad / 2) / 60
        Return retn
    End Function



    Public Function CalculateTime(ByVal timeAtWp1 As Date, ByRef wp1 As C8WayPoint,
ByRef wp2 As C8WayPoint, ByRef dist As Double, ByRef dir As Double, ByRef
spdchar_forDTR As C7TotalShipSpeedChar, ByRef weatherDB As C8WeatherDB, ByRef
limitWavHeight As Double, ByRef limitWndSpd As Double, ByRef rpm As Double, ByRef
draft As Double, ByRef trm As Double, ByRef normSpdChar As C3NormSpeedForAllRPM) As
Double
        Dim retn As Double = 0
        Dim actSpd As Double
        Dim dtime As Double = 1

        Dim weatherDat As C8WeatherPoint = New C8WeatherPoint()
        Dim tmpDistTravel As Double = 0
        Dim tmpRemainDist As Double = dist
        Dim tmp As Double

        Dim tmpCurrPos As C8WayPoint = New C8WayPoint(wp1)
        Dim tmpTime As Date = timeAtWp1

        Dim relWndDir As Double
        Dim WndSpd As Double
        Dim relWavDir As Double
        Dim Wavhgt As Double


        While True
```

```
            weatherDB.GetWeatherPoint(tmpTime, tmpCurrPos.lat, tmpCurrPos.lon,
weatherDat)
            If weatherDat.wavHei >= limitWavHeight Or weatherDat.wndspd >=
limitWndSpd Then
                retn = mGlbInfinite
                Return retn
            End If

            actSpd = spdchar_forDTR.GetSpeedValue(normSpdChar, rpm, draft, trm,
WndSpd, relWndDir, Wavhgt, relWavDir)
            actSpd += weatherDat.crrspd * Math.Cos(toRad * DirectionDiff_deg(dir,
weatherDat.crrdir))

            If actSpd <= 0 Then
                retn = mGlbInfinite
                Return retn
            End If

            tmp = tmpRemainDist / actSpd

            If tmp <= dtime Then
                retn += tmp
                timeAtWp1 = DateAdd(DateInterval.Hour, tmp, tmpTime)
                Exit While

            Else
                tmpRemainDist -= dtime * actSpd
                tmpDistTravel += dtime * actSpd
                retn += dtime
                tmpTime = DateAdd(DateInterval.Hour, dtime, tmpTime)
                tmpCurrPos = NewPosition(tmpCurrPos, dir, tmpDistTravel)

            End If
        End While

        Return retn
    End Function



    Public Function DirectionDiff_deg(ByRef base As Double, ByRef bear As Double) As
Double
        Dim retn As Double = bear - base
        If retn < -180 Then retn += 360
        If retn > 180 Then retn -= 360

        If retn < 0 Then retn = -retn
        Return retn
    End Function


    Public Function ValueInterpolate(ByRef v1 As Double, ByVal v2 As Double, ByRef
fac As Double) As Double
        Dim retn As Double = v1 + (v2 - v1) * fac
        Return retn
    End Function

    Public Function DirectionInterpolate(ByRef dir1 As Double, ByVal dir2 As Double,
ByRef fac As Double) As Double
```

```
        Dim diff As Double = DirectionDiff_deg(dir1, dir2)
        Dim retn As Double = dir1 + diff * fac
        If retn < 0 Then
            retn += 360
        ElseIf retn >= 360 Then
            retn -= 360
        End If
        Return retn
    End Function

    Public Sub WeatherPointInterpolate(ByRef w1 As C8WeatherPoint, ByRef w2 As
C8WeatherPoint, ByRef fac As Double, ByRef wRes As C8WeatherPoint)
        If fac <= 0 Then
            wRes.CopyFrom(w1)
        ElseIf fac >= 1 Then
            wRes.CopyFrom(w2)
        Else
            wRes.wndspd = ValueInterpolate(w1.wndspd, w2.wndspd, fac)
            wRes.crrspd = ValueInterpolate(w1.crrspd, w2.crrspd, fac)
            wRes.wndspd = ValueInterpolate(w1.wavHei, w2.wavHei, fac)
            wRes.wnddir = DirectionInterpolate(w1.wnddir, w2.wnddir, fac)
            wRes.wavdir = DirectionInterpolate(w1.wavdir, w2.wavdir, fac)
            wRes.crrdir = DirectionInterpolate(w1.crrdir, w2.crrdir, fac)
        End If
    End Sub

End Class
```

2. Tính toán kế hoach khai thác tàu tối ưu

```
Public Class CFleetPlanning
End Class



Public Class CCARGOCONTRACT
    Public cargoes() As CCargo

    Public Sub New()
    End Sub


    Public Sub CopyFrom(ByRef org As CCARGOCONTRACT)
        If IsNothing(org.cargoes) = True Then
            cargoes = Nothing
        Else
            ReDim cargoes(org.cargoes.Length - 1)
            For I As Integer = 0 To cargoes.Length - 1
                cargoes(I) = New CCargo()
                cargoes(I).CopyFrom(org.cargoes(I))
            Next
        End If
    End Sub


    Public Function AddNewShip(ByRef newCargo As CCargo) As Integer
        Dim st As String = newCargo.ToStringFormat()

        If IsNothing(cargoes) = True Then
```

```
        ReDim cargoes(0)
        cargoes(0) = New CCargo()
        cargoes(0).InitFromStr(st)
        Return 1
    End If

    For I As Integer = 0 To cargoes.Length - 1
        If String.Compare(cargoes(I).code, newCargo.code, True) = 0 Then
            Return -1
        End If
    Next

    ReDim Preserve cargoes(cargoes.Length)
    cargoes(cargoes.Length - 1) = New CCargo()
    cargoes(cargoes.Length - 1).InitFromStr(st)
    Return 1
End Function


Public Function ChangeShipData(ByRef newCargo As CCargo) As Integer
    Dim st As String = newCargo.ToStringFormat()
    Dim idx As Integer = -1

    If IsNothing(cargoes) = False Then
        For I As Integer = 0 To cargoes.Length - 1
            If String.Compare(cargoes(I).code, newCargo.code, True) = 0 Then
                idx = I
                Exit For
            End If
        Next
    End If

    If idx < 0 Then Return -1
    cargoes(idx) = New CCargo()
    cargoes(idx).InitFromStr(st)
    Return 1
End Function



Public Function DeleteShip(ByVal idx As Integer) As Integer
    If IsNothing(cargoes) = True Then Return -1
    If idx < 0 Or idx >= cargoes.Length Then Return -1

    If cargoes.Length = 1 Then
        cargoes = Nothing
        Return -1
    End If

    Dim st As String
    For I As Integer = idx To cargoes.Length - 2
        st = cargoes(I + 1).ToStringFormat()
        cargoes(I) = New CCargo()
        cargoes(I).InitFromStr(st)
    Next

    ReDim Preserve cargoes(cargoes.Length - 2)
    Return 1
End Function
```

```
    Public Sub InitCargoContract(ByRef st() As String)
        ReDim cargoes(st.Length - 1)
        Dim count As Integer = 0

        For I As Integer = 0 To cargoes.Length - 1
            cargoes(count) = New CCargo()
            cargoes(count).InitFromStr(st(I))
            If cargoes(count).code <> "" Then count += 1
        Next

        If count = 0 Then
            cargoes = Nothing
            Exit Sub
        Else
            If count < cargoes.Length Then
                ReDim Preserve cargoes(count - 1)
            End If
        End If

        Dim st1 As String
        Dim st2 As String
        For I As Integer = 0 To cargoes.Length - 2
            For J As Integer = I + 1 To cargoes.Length - 1
                If cargoes(I).code > cargoes(J).code Then
                    st1 = cargoes(I).ToStringFormat()
                    st2 = cargoes(2).ToStringFormat()

                    cargoes(I).InitFromStr(st2)
                    cargoes(J).InitFromStr(st1)
                End If
            Next
        Next
    End Sub
End Class



Public Class CCargo
    Public code As String
    Public type As String

    Public minimumWeight As Double
    Public maximumWeight As Double

    'Public weight As Double
    'Public IsWeightMatter As Integer

    Public frmPort As String
    Public earlyDueDate As Date
    Public lateDueDate As Date

    'Public dueDateReceipt As Date
    'Public IsDateMatter As Integer

    Public toPort As String
```

```
    Public Sub New()
    End Sub


    Public Sub CopyFrom(ByRef org As CCargo)
        With org
            code = .code
            type = .type
            minimumWeight = .minimumWeight
            maximumWeight = .maximumWeight

            'weight = .weight
            'IsWeightMatter = .IsWeightMatter

            frmPort = .frmPort
            earlyDueDate = DateAdd(DateInterval.Day, 0, .earlyDueDate)
            lateDueDate = DateAdd(DateInterval.Day, 0, .lateDueDate)

            'dueDateReceipt = .dueDateReceipt
            'IsDateMatter = .IsDateMatter

            toPort = .toPort
        End With
    End Sub

    Public Sub InitFromStr(ByRef st As String)
        Dim v() As String = Split(st, ";")
        code = v(0)
        type = v(1)

        minimumWeight = Val(v(2))
        maximumWeight = Val(v(2))

        'weight = Val(v(2))
        'IsWeightMatter = Val(v(3))

        frmPort = v(4)
        earlyDueDate = mGlbFcnUlongStringToDate(v(5))
        lateDueDate = mGlbFcnUlongStringToDate(v(5))

        'dueDateReceipt = mGlbFcnUlongStringToDate(v(5))
        'IsWeightMatter = Val(v(6))

        toPort = v(7)
    End Sub


    Public Function ToStringFormat() As String
        'Dim st As String = String.Format("{0};{1};{2};{3};{4};{5};{6};{7}", code,
type, weight, IsWeightMatter, frmPort, mGlbFcnDateToUlongString(dueDateReceipt),
IsDateMatter, toPort)
        Dim st As String = String.Format("{0};{1};{2};{3};{4};{5};{6};{7}", code,
type, minimumWeight, maximumWeight, frmPort, mGlbFcnDateToUlongString(earlyDueDate),
mGlbFcnDateToUlongString(lateDueDate), toPort)
        Return st
    End Function
End Class
```

```
Public Class CCargoTypeList
    Public types() As String

    Public Sub New()
    End Sub

    Public Sub Init(ByRef st As String)
        If st = "" Then
            types = Nothing
            Exit Sub
        End If

        Dim v() As String = Split(st, ";")

        ReDim types(v.Length - 1)
        For I As Integer = 0 To v.Length - 1
            types(I) = v(I)
        Next
    End Sub


    Public Function ToStringFormat() As String
        If IsNothing(types) = True Then Return ""

        Dim retn As String = types(0)
        For I As Integer = 1 To types.Length - 1
            types(I) = retn & ";" & types(I)
        Next

        Return retn
    End Function

    Public Function AddNewType(ByVal newtype As String) As Integer
        If IsNothing(types) = True Then
            ReDim types(0)
            types(0) = newtype
            Return 1
        End If

        For I As Integer = 0 To types.Length - 1
            If String.Compare(newtype, types(I), True) = 0 Then
                Return -1
            End If
        Next


        ReDim Preserve types(types.Length)
        types(types.Length - 1) = newtype

        Return 1
    End Function



    Public Function DeleteType(ByVal idx As Integer) As Integer
        If IsNothing(types) = True Then
            Return 1
        End If
```

```
        If idx < 0 Or idx >= types.Length Then
            Return -1
        End If

        If types.Length = 1 Then
            types = Nothing
            Return 1
        End If

        For I As Integer = idx To types.Length - 2
            types(I) = types(I + 1)
        Next
        ReDim Preserve types(types.Length - 2)

        Return 1
    End Function
End Class


Public Class CPortList
    Public ports() As String

    Public Sub New()
    End Sub

    Public Sub Init(ByRef st As String)
        If st = "" Then
            ports = Nothing
            Exit Sub
        End If

        Dim v() As String = Split(st, ";")

        ReDim ports(v.Length - 1)
        For I As Integer = 0 To v.Length - 1
            ports(I) = v(I)
        Next
    End Sub


    Public Function ToStringFormat() As String
        If IsNothing(ports) = True Then Return ""

        Dim retn As String = ports(0)
        For I As Integer = 1 To ports.Length - 1
            ports(I) = retn & ";" & ports(I)
        Next

        Return retn
    End Function

    Public Function AddNewport(ByVal newport As String) As Integer
        If IsNothing(ports) = True Then
            ReDim ports(0)
            ports(0) = newport
            Return 1
        End If

        For I As Integer = 0 To ports.Length - 1
```

```
            If String.Compare(newport, ports(I), True) = 0 Then
                Return -1
            End If
        Next


        ReDim Preserve ports(ports.Length)
        ports(ports.Length - 1) = newport

        Return 1
    End Function



    Public Function Deleteport(ByVal idx As Integer) As Integer
        If IsNothing(ports) = True Then
            Return 1
        End If

        If idx < 0 Or idx >= ports.Length Then
            Return -1
        End If

        If ports.Length = 1 Then
            ports = Nothing
            Return 1
        End If

        For I As Integer = idx To ports.Length - 2
            ports(I) = ports(I + 1)
        Next
        ReDim Preserve ports(ports.Length - 2)

        Return 1
    End Function
End Class



Public Class CFLEETCAPACITY
    Public ships() As CShipDataForTrips
    Public shipUse() As Integer

    Public Sub New()
    End Sub


    Public Sub SetShipUsage(ByRef usg() As Integer)
        ReDim shipUse(ships.Length - 1)
        For I As Integer = 0 To shipUse.Length - 1
            shipUse(I) = usg(I)
        Next
    End Sub


    Public Function AddNewShip(ByRef newShip As CShipDataForTrips) As Integer
        Dim st As String = newShip.ToStringFormat()

        If IsNothing(ships) = True Then
```

```
        ReDim ships(0)
        ships(0) = New CShipDataForTrips()
        ships(0).Init(st)
        Return 1
    End If

    For I As Integer = 0 To ships.Length - 1
        If String.Compare(ships(I).shipIMO, newShip.shipIMO, True) = 0 Then
            Return -1
        End If
    Next

    ReDim Preserve ships(ships.Length)
    ships(ships.Length - 1) = New CShipDataForTrips()
    ships(ships.Length - 1).Init(st)
    Return 1
End Function


Public Function ChangeShipData(ByRef newShip As CShipDataForTrips) As Integer
    Dim st As String = newShip.ToStringFormat()
    Dim idx As Integer = -1

    If IsNothing(ships) = False Then
        For I As Integer = 0 To ships.Length - 1
            If String.Compare(ships(I).shipIMO, newShip.shipIMO, True) = 0 Then
                idx = I
                Exit For
            End If
        Next
    End If

    If idx < 0 Then Return -1
    ships(idx) = New CShipDataForTrips()
    ships(idx).Init(st)
    Return 1
End Function



Public Function DeleteShip(ByVal idx As Integer) As Integer
    If IsNothing(ships) = True Then Return -1
    If idx < 0 Or idx >= ships.Length Then Return -1

    If ships.Length = 1 Then
        ships = Nothing
        Return -1
    End If

    Dim st As String
    For I As Integer = idx To ships.Length - 2
        st = ships(I + 1).ToStringFormat()
        ships(I) = New CShipDataForTrips()
        ships(I).Init(st)
    Next

    ReDim Preserve ships(ships.Length - 2)
    Return 1
End Function
```

```
    Public Sub InitFleet(ByRef st() As String)
        ReDim ships(st.Length - 1)
        Dim count As Integer = 0

        For I As Integer = 0 To ships.Length - 1
            ships(count) = New CShipDataForTrips()
            ships(count).Init(st(I))
            If ships(count).shipIMO <> "" Then count += 1
        Next

        If count = 0 Then
            ships = Nothing
            Exit Sub
        Else
            If count < ships.Length Then
                ReDim Preserve ships(count - 1)
            End If
        End If

        Dim st1 As String
        Dim st2 As String
        For I As Integer = 0 To ships.Length - 2
            For J As Integer = I + 1 To ships.Length - 1
                If ships(I).shipIMO > ships(J).shipIMO Then
                    st1 = ships(I).ToStringFormat()
                    st2 = ships(2).ToStringFormat()

                    ships(I).Init(st2)
                    ships(J).Init(st1)
                End If
            Next
        Next
    End Sub
End Class


Public Class CShipDataForTrips
    Public shipIMO As String
    Public DWT As Double
    Public avgResttime As Double

    Public availFrom As Date
    Public availFromPlace As String

    Public availTo As Date
    Public availToPlace As String

    Public dailyOffServiceFuelCost As Double
    Public portPossible() As String
    Public cargoPossible() As String
    Public cweights() As Double
    Public spds() As Double
    Public fuelcons() As Double

    Public Sub New()
    End Sub
```

```
Public Sub Init(ByRef st As String)
    Try
        Dim v() As String = Split(st, ";")
        shipIMO = v(0)

        DWT = Val(v(1))
        avgResttime = Val(v(2))
        availFrom = mGlbFcnUlongStringToDate(v(3))
        availFromPlace = v(5)

        availTo = mGlbFcnUlongStringToDate(v(6))
        availToPlace = v(7)

        dailyOffServiceFuelCost = Val(v(8))

        Dim idx As Integer = 9
        Dim No As Integer

        '--------------------------------
        No = Val(v(idx))
        idx += 1

        If No > 0 Then
            ReDim portPossible(No - 1)
            For I As Integer = 0 To No - 1
                portPossible(I) = v(idx)
                idx += 1
            Next

        Else
            portPossible = Nothing
        End If

        '--------------------------------
        No = Val(v(idx))
        idx += 1

        If No > 0 Then
            ReDim cargoPossible(No - 1)
            For I As Integer = 0 To No - 1
                cargoPossible(I) = v(idx)
                idx += 1
            Next

        Else
            cargoPossible = Nothing
        End If


        '--------------------------------
        No = Val(v(idx))
        idx += 1

        If No > 0 Then
            ReDim cweights(No - 1)
            ReDim spds(No - 1)
            ReDim fuelcons(No - 1)

            For I As Integer = 0 To No - 1
```

```
                    cweights(I) = v(idx)
                    spds(I) = v(idx + 1)
                    fuelcons(I) = v(idx + 2)
                    idx += 3
                Next

            Else
                cweights = Nothing
                spds = Nothing
                fuelcons = Nothing
            End If

        Catch ex As Exception
            shipIMO = ""
        End Try
    End Sub


    Public Function ToStringFormat() As String
        Dim retn As String = shipIMO & ";" & DWT & ";" & avgResttime & ";" &
mGlbFcnDateToUlongString(availFrom) & ";" & availFromPlace & ";" &
mGlbFcnDateToUlongString(availTo) & ";" & availToPlace & ";" & ";" &
dailyOffServiceFuelCost
        If IsNothing(portPossible) = False Then
            retn = retn & ";" & portPossible.Length
            For I As Integer = 0 To portPossible.Length - 1
                retn = retn & ";" & portPossible(I)
            Next
        Else
            retn = retn & ";0"
        End If

        If IsNothing(cargoPossible) = False Then
            retn = retn & ";" & cargoPossible.Length
            For I As Integer = 0 To cargoPossible.Length - 1
                retn = retn & ";" & cargoPossible(I)
            Next
        Else
            retn = retn & ";0"
        End If

        If IsNothing(cweights) = False Then
            retn = retn & ";" & cweights.Length
            For I As Integer = 0 To cweights.Length - 1
                retn = retn & ";" & cweights(I) & ";" & spds(I) & ";" & fuelcons(I)
            Next
        Else
            retn = retn & ";0"
        End If

        Return retn
    End Function


    Public Function GetBallastSpeedAndFuelConsumption() As Double()
        Dim cargoweight As Double = (cweights(0) * 2 + cweights(cweights.Length -
1)) / 3
        Return GetTravelSpeedAndFuelConsumption(cargoweight)
    End Function
```

```
    Public Function GetTravelSpeedAndFuelConsumption(ByVal cargoweight As Double) As
Double()
        Dim retn(2) As Double
        If cargoweight < cweights(0) Then
            retn(0) = spds(0)
            retn(1) = fuelcons(0)

        ElseIf cargoweight >= cweights(cweights.Length - 1) Then
            retn(0) = spds(cweights.Length - 1)
            retn(1) = fuelcons(cweights.Length - 1)

        Else
            Dim idx As Integer = -1
            For I As Integer = 0 To cweights.Length - 2
                If cweights(I) <= cargoweight And cweights(I + 1) >= cargoweight
Then
                    idx = I
                    Exit For
                End If
            Next

            Dim fct As Double = (cargoweight - cweights(idx)) / (cweights(idx + 1) -
cweights(idx))
            retn(0) = spds(idx) + (spds(idx + 1) - spds(idx)) * fct
            retn(0) = fuelcons(idx) + (fuelcons(idx + 1) - fuelcons(idx)) * fct
        End If

        Return retn
    End Function

End Class


Public Class CPortToPortDistance
    Public ports() As String
    Public portportDist(,) As Double

    Public Sub New(ByRef portlist As CPortList)
        If IsNothing(portlist.ports) = True Then
            ports = Nothing
            portportDist = Nothing
        End If

        ReDim ports(portlist.ports.Length - 1)
        For I As Integer = 0 To ports.Length - 1
            ports(I) = portlist.ports(I)
        Next

        ReDim portportDist(ports.Length - 1, ports.Length - 1)
        For I As Integer = 0 To ports.Length - 1
            For J As Integer = 0 To ports.Length - 1
                If I = J Then
                    portportDist(I, J) = 0
                Else
                    portportDist(I, J) = 1000
                End If
```

```
            Next
        Next
    End Sub


    Public Sub Init(ByRef st() As String)
        Dim v() As String
        Dim idx1 As Integer
        Dim idx2 As Integer

        For I As Integer = 0 To st.Length - 1
            v = Split(st(I), ";")
            If v.Length <> 3 Then Continue For

            idx1 = -1
            idx2 = -1

            For J As Integer = 0 To ports.Length - 1
                If String.Compare(ports(I), v(0), True) = 0 Then idx1 = J
                If String.Compare(ports(I), v(1), True) = 0 Then idx2 = J
            Next

            If idx1 >= 0 And idx2 > 0 Then
                portportDist(idx1, idx2) = Val(v(2))
            End If
        Next
    End Sub


    Public Function ToStrings() As String()
        Dim retn(portportDist.Length - 1) As String
        Dim idx As Integer = 0

        For I As Integer = 0 To ports.Length - 1
            For J As Integer = 0 To ports.Length - 1
                retn(idx) = String.Format("{0};{1};{2}", ports(I), ports(J),
portportDist(I, J))
            Next
        Next

        Return retn
    End Function

    Public Function GetPortToPortDistance(ByRef stPort As String, ByRef enPort As
String) As Double
        Dim idx1 As Integer = -1
        Dim idx2 As Integer = -1

        For I As Integer = 0 To ports.Length - 1
            If String.Compare(stPort, ports(I), True) = 0 Then idx1 = I
            If String.Compare(enPort, ports(I), True) = 0 Then idx2 = I
        Next

        If idx1 >= 0 And idx2 >= 0 Then
            Return portportDist(idx1, idx2)
        Else
            Return 0
        End If
    End Function
```

```
End Class
```

3. Tính toán đặc tính điều động và tiêu thụ nhiên liệu của tàu

```
Imports System.Text
Imports System.IO



''NHAP LIEU, THAY RPM BANG NORMINAL SPEED
''THAY FUELCONSUME DONG NHAT BANG GIA TRI 1 (ONE)
''THAY DRAFT DONG NHAT BANG GIA TRI 1 (ONE)
''THAY MONTHSSINCELASTDRYDOCK DONG NHAT BANG GIA TRI 1 (ONE)
''THAY TRIM DONG NHAT BANG GIA TRI 0 (ZERO)
''THEO DO, THAY GIA TRI MAX, MIN CUA RMP THANH GIA TRI TRONG KHOANG  0 - 50
''THAY GIA TRI MAX, MIN CUA FUELCONSUME  THANH GIA TRI TRONG KHOANG  0 - 2
''NHUNG PHAN KHAC TINH SAU
''02 THONG SO CAN NHAP LA LEARN-RATE,


Public Class CParallelANN
    Public anns() As CShipPerformanceANN
    Public fisrtTrainingRetn() As Double
    Public lastTrainingRetn() As Double

    Public NoOfHiddenNeural As Integer
    Public learnRate As Double
    Public NoOf1000Repear As Integer
    Public NoOfReservedTestSample As Integer

    Public sDraft As Double
    Public sTrim As Double
    Public sRPM As Double
    Public sTrimStep As Double = 0.3
    Public sDraftStep As Double = 0.5
    Public sRPMStep As Double = 10


    Public Sub New(ByVal drft As Double, ByVal trm As Double, ByVal rpm As Double,
ByVal tmpNoOfHiddenNeural As Integer, ByVal tmpLearningRate As Double, ByVal
tmpNoOfRepeatx1000 As Integer, ByVal tmpNoOfSampleForTest As Integer)
        sDraft = drft
        sTrim = trm
        sRPM = rpm

        NoOfHiddenNeural = tmpNoOfHiddenNeural
        learnRate = tmpLearningRate
        NoOf1000Repear = tmpNoOfRepeatx1000
        NoOfReservedTestSample = tmpNoOfSampleForTest

        Dim paramfile As String = GetParamFileName()
        Dim st As String = ""
        Dim dats() As String = Nothing

        Try
            Dim stRead As StreamReader = New StreamReader(paramfile)
            st = stRead.ReadToEnd()
```

```
            stRead.Close()
        Catch ex As Exception
            st = ""
        End Try

        If st <> "" Then
            dats = Split(st, vbCrLf)
        Else
            dats = Nothing
        End If

        If IsNothing(dats) = False Then
            If dats.Length <> mGlbWndDirs.Length + 1 Then
                dats = Nothing
            Else
                Dim v() As String = Split(dats(0), ";")
                If v.Length <> 3 Then
                    dats = Nothing
                Else
                    If Val(v(1)) <> NoOfHiddenNeural Then
                        dats = Nothing
                    End If
                End If
            End If
        End If

        ReDim anns(mGlbWndDirs.Length - 1)
        ReDim fisrtTrainingRetn(mGlbWndDirs.Length - 1)
        ReDim lastTrainingRetn(mGlbWndDirs.Length - 1)

        For I As Integer = 0 To anns.Length - 1
            If IsNothing(dats) = True Then
                anns(I) = New CShipPerformanceANN(mGlbWndDirs(I), "", learnRate,
NoOf1000Repear, NoOfReservedTestSample, NoOfHiddenNeural)
            Else
                anns(I) = New CShipPerformanceANN(mGlbWndDirs(I), dats(I + 1),
learnRate, NoOf1000Repear, NoOfReservedTestSample, NoOfHiddenNeural)
            End If
        Next
    End Sub



    Public Sub ReNew(ByVal tmpNoOfHiddenNeural As Integer, ByVal tmpLearningRate As
Double, ByVal tmpNoOfRepeatx1000 As Integer, ByVal tmpNoOfSampleForTest As Integer)
        NoOfHiddenNeural = tmpNoOfHiddenNeural
        learnRate = tmpLearningRate
        NoOf1000Repear = tmpNoOfRepeatx1000
        NoOfReservedTestSample = tmpNoOfSampleForTest

        ReDim anns(mGlbWndDirs.Length - 1)
        ReDim fisrtTrainingRetn(mGlbWndDirs.Length - 1)
        ReDim lastTrainingRetn(mGlbWndDirs.Length - 1)

        For I As Integer = 0 To anns.Length - 1
            anns(I) = New CShipPerformanceANN(mGlbWndDirs(I), "", learnRate,
NoOf1000Repear, NoOfReservedTestSample, NoOfHiddenNeural)
        Next
    End Sub
```

```
    Public Function GetFolderName() As String
        Return "NeuralData"
    End Function

    Public Function GetParamFileName() As String
        Dim retn As String = GetFolderName() & IO.Path.DirectorySeparatorChar &
"NN_Param_"
        Dim tmp As Integer

        tmp = sRPM
        retn = retn & tmp.ToString

        tmp = sDraft * 10
        retn = retn & tmp.ToString

        tmp = sTrim * 10
        retn = retn & tmp.ToString

        Return retn
    End Function


    Public Function GetNormalizeFileName() As String
        Dim retn As String = GetFolderName() & IO.Path.DirectorySeparatorChar &
"NormalizeParam.txt"
        Return retn
    End Function


    Public Function SaveNetWorkParameter(ByVal filename As String) As Boolean
        Dim stWrite As StreamWriter = New StreamWriter(filename)
        Dim st As String

        st = ""
        For I As Integer = 0 To anns(0).ANN.LayerCount - 1
            st = st & anns(0).ANN.Layers(I).NeuronCount & ";"
        Next
        st = st.Substring(0, st.Length - 1)
        stWrite.WriteLine(st)

        For KK As Integer = 0 To anns.Length - 1

            st = ""
            If anns(KK).TrainSuccess = 1 Then
                For I As Integer = 0 To anns(KK).ANN.LayerCount - 1
                    For J As Integer = 0 To anns(KK).ANN.Layers(I).NeuronCount - 1
                        st = st & ";" & anns(KK).ANN.Layers(I).Neurons(J).Bias
                        For K As Integer = 0 To
anns(KK).ANN.Layers(I).Neurons(J).DendriteCount - 1
                            st = st & ";" &
anns(KK).ANN.Layers(I).Neurons(J).Dendrites(K).Weight
                        Next
                    Next
                Next
            Else
                st = "Unsuccessful"
            End If
```

```
            st = st.Substring(1)
            stWrite.WriteLine(st)
        Next

        stWrite.Close()
        Return True
    End Function



    Public Sub RunANN(ByVal isReInit As Integer)
        If isReInit = 0 Then
            For I As Integer = 0 To anns.Length - 1
                anns(I).ANN.InitializeNetworkRandomly()
            Next
        End If

        For I As Integer = 0 To anns.Length - 1
            Dim retn() As Double = anns(I).RunNN()
            fisrtTrainingRetn(I) = retn(0)
            lastTrainingRetn(I) = retn(retn.Length - 1)
        Next

        Dim paramfile As String = GetParamFileName()
        SaveNetWorkParameter(paramfile)
    End Sub


    Public Sub InitANNAndTrainingDataSet(ByRef workingrecords As
CShipPerformanceRecordList, ByRef timeofrecord As Date)
        Dim NormFileName As String = GetNormalizeFileName()
        Dim NormSet As CNormalizeSet = New CNormalizeSet()
        NormSet.LoadNormDataFromFile(NormFileName)

        Dim tmpWndDirStep As Double = (mGlbWndDirs(1) - mGlbWndDirs(0)) / 2

        Dim tmpWorkingRecords As CShipPerformanceRecordList = New
CShipPerformanceRecordList()
        For I As Integer = 0 To anns.Length - 1
            ReDim tmpWorkingRecords.perf(1000)
            Dim tmpCount As Integer = 0

            For J As Integer = 0 To workingrecords.perf.Length - 1
                With workingrecords.perf(J)
                    If .sRPM >= sRPM - sRPMStep And .sRPM <= sRPM + sRPMStep And
.sDraft >= sDraft - sDraftStep And .sDraft <= sDraft + sDraftStep And .sTrim >=
sTrim - sTrimStep And .sTrim <= sTrim + sTrimStep Then

                        If .WndDir_deg >= anns(I).WndRelativeDir - tmpWndDirStep And
.WndDir_deg <= anns(I).WndRelativeDir + tmpWndDirStep Then
                            tmpWorkingRecords.perf(tmpCount) = New
CShipPerformanceRecord(workingrecords.perf(J))
                            tmpCount += 1
                        End If

                    End If
                End With
            Next
```

```
            If tmpCount >= 10 Then
                ReDim Preserve tmpWorkingRecords.perf(tmpCount - 1)
            Else
                tmpWorkingRecords.perf = Nothing
            End If

            anns(I).TrainingDataRecords = New CTrainingDataSet(NormSet,
tmpWorkingRecords, timeofrecord, NoOfReservedTestSample)
        Next
    End Sub

End Class



Public Class CShipPerformanceANN
    Public layList As List(Of Integer)
    Public lrnRate As Double = 0.01
    Public ANN As NeuralNet.NeuralNetwork
    Public TrainingDataRecords As CTrainingDataSet
    Public NoOf_1000_Repeat As Integer = 20
    Public NoOfInp As Integer
    Public NoOfOut As Integer
    Public NoOfhiden As Integer
    Public NoOfSampleReservedForTesting As Integer = 0

    Public WndRelativeDir As Double
    Public TrainSuccess As Integer


    Public Sub New(ByVal wnddir As Double, ByVal biasweight As String, ByVal
learnRate As Double, ByVal NoOfRepeatx1000 As Integer, ByVal ReservedForTesting As
Integer, ByVal NoOfHiddenNeural As Integer, Optional ByVal NoOfInputNeural As
Integer = 7, Optional ByVal NoOfOutNeural As Integer = 1)
        NoOfInp = NoOfInputNeural
        NoOfOut = NoOfOutNeural
        NoOfhiden = NoOfHiddenNeural

        NoOf_1000_Repeat = NoOfRepeatx1000
        lrnRate = learnRate
        NoOfSampleReservedForTesting = ReservedForTesting

        layList = New List(Of Integer)
        layList.Add(NoOfInputNeural)
        layList.Add(NoOfHiddenNeural)
        layList.Add(NoOfOut)
        ANN = New NeuralNet.NeuralNetwork(lrnRate, layList)

        WndRelativeDir = wnddir

        Dim loaded As Boolean = ANN.LoadNetWorkParameter(biasweight)
        If loaded = False Then
            ANN.InitializeNetworkRandomly()
            TrainSuccess = 0
        Else
            TrainSuccess = 1
        End If
    End Sub
```

```
    Public Function GetOutPutSpeedErrorForTheTrainingSet() As Double()
        Dim tmpInput As List(Of Double) = New List(Of Double)
        Dim tmpOutput As List(Of Double) = New List(Of Double)

        Dim retn(TrainingDataRecords.InputDat.Length - 1) As Double


        For I As Integer = 0 To TrainingDataRecords.InputDat.Length - 1
            tmpInput = New List(Of Double)
            tmpOutput = New List(Of Double)

            With TrainingDataRecords.InputDat(I)
                'tmpInput.Add(.monthsSinceLastDryDock)
                tmpInput.Add(.prop_rpm)
                tmpInput.Add(.draft_m)
                tmpInput.Add(.trim_m)
                tmpInput.Add(.wind_deg)
                tmpInput.Add(.wind_kts)
                tmpInput.Add(.wave_deg)
                tmpInput.Add(.wave_m)

                tmpOutput.Add(.spd_kts)
                'tmpOutput.Add(.fdo_tph)
            End With

            Dim rOut As List(Of Double) = ANN.Execute(tmpInput)
            Dim tmpRetn(0) As Double
            tmpRetn(0) = rOut.Item(0)
            'tmpRetn(1) = rOut.Item(1)

            Dim retn1() As Double
            retn1 =
TrainingDataRecords.NormSet.NormalizedOutput_2_ShipSpdAndFuelcomsume(tmpRetn(0),
tmpRetn(1))
            'retn1 =
TrainingDataRecords.NormSet.NormalizedOutput_2_ShipSpdAndFuelcomsume(tmpRetn(0),
tmpRetn(1))

            tmpRetn(0) = tmpOutput.Item(0)
            tmpRetn(1) = tmpOutput.Item(1)
            Dim retn2() As Double =
TrainingDataRecords.NormSet.NormalizedOutput_2_ShipSpdAndFuelcomsume(tmpRetn(0),
tmpRetn(1))

            retn(I) = retn1(0) - retn2(0)
        Next

        Return retn
    End Function



    Public Function GetOutPutSpeedData(ByRef workingStatus As
CShipPerformanceRecord) As Double
        Dim opernationalSet As CShipOperationDataSet =
TrainingDataRecords.GetNormalizeOpertionalDataSet(workingStatus)
        Dim tmpInput As List(Of Double) = New List(Of Double)
```

```
        Dim tmpOutput As List(Of Double) = New List(Of Double)

        tmpInput = New List(Of Double)
        tmpOutput = New List(Of Double)

        With opernationalSet
            tmpInput.Add(.monthsSinceLastDryDock)
            tmpInput.Add(.prop_rpm)
            tmpInput.Add(.draft_m)
            tmpInput.Add(.trim_m)
            tmpInput.Add(.wind_deg)
            tmpInput.Add(.wind_kts)
            tmpInput.Add(.wave_deg)
            tmpInput.Add(.wave_m)

            tmpOutput.Add(.spd_kts)
            tmpOutput.Add(.fdo_tph)
        End With

        Dim rOut As List(Of Double) = ANN.Execute(tmpInput)
        Dim retn(1) As Double
        retn(0) = rOut.Item(0)
        retn(1) = rOut.Item(1)

        Dim retn1() As Double
        retn1 =
TrainingDataRecords.NormSet.NormalizedOutput_2_ShipSpdAndFuelcomsume(retn(0),
retn(1))
        Return retn1(0)
    End Function


    Public Function RunNN() As Double()
        Return TrainANN(TrainingDataRecords)
    End Function


    Public Function TrainANN(ByRef trainSet As CTrainingDataSet) As Double()
        Dim tmpInput As List(Of Double) = New List(Of Double)
        Dim tmpOutput As List(Of Double) = New List(Of Double)
        Dim NoOfCheckingStep As Integer = 1000

        Dim retn(NoOf_1000_Repeat - 1) As Double

        For K As Integer = 0 To NoOf_1000_Repeat - 1
            For Rep As Integer = 0 To 1000
                For I As Integer = 0 To trainSet.InputDat.Length - 1
                    tmpInput = New List(Of Double)
                    tmpOutput = New List(Of Double)

                    With trainSet.InputDat(I)
                        'tmpInput.Add(.monthsSinceLastDryDock)
                        tmpInput.Add(.prop_rpm)
                        tmpInput.Add(.draft_m)
                        tmpInput.Add(.trim_m)
                        tmpInput.Add(.wind_deg)
                        tmpInput.Add(.wind_kts)
                        tmpInput.Add(.wave_deg)
                        tmpInput.Add(.wave_m)
```

```
                    tmpOutput.Add(.spd_kts)
                    'tmpOutput.Add(.fdo_tph)
                End With

                ANN.Train(tmpInput, tmpOutput)
            Next
        Next

        retn(K) = SquareErrorSumUp(trainSet)
    Next

    Return retn
End Function



    Public Function TestSetSquareErrorSumUp(ByRef trainSet As CTrainingDataSet) As
Double
        If IsNothing(trainSet.TestDat) = True Then Return 999999

        Dim retn As Double = 0
        Dim tmpInput As List(Of Double) = New List(Of Double)
        Dim tmpOutput As List(Of Double) = New List(Of Double)

        For I As Integer = 0 To trainSet.TestDat.Length - 1
            tmpInput = New List(Of Double)
            tmpOutput = New List(Of Double)

            With trainSet.TestDat(I)
                'tmpInput.Add(.monthsSinceLastDryDock)
                tmpInput.Add(.prop_rpm)
                tmpInput.Add(.draft_m)
                tmpInput.Add(.trim_m)
                tmpInput.Add(.wind_deg)
                tmpInput.Add(.wind_kts)
                tmpInput.Add(.wave_deg)
                tmpInput.Add(.wave_m)

                tmpOutput.Add(.spd_kts)
                'tmpOutput.Add(.fdo_tph)

                Dim rOut As List(Of Double) = ANN.Execute(tmpInput)
                retn += (rOut.Item(0) - tmpOutput.Item(0)) ^ 2
                'retn += (rOut.Item(0) - tmpOutput.Item(0)) ^ 2 + (rOut.Item(1) -
tmpOutput.Item(1)) ^ 2
            End With
        Next
        Return retn
    End Function



    Public Function SquareErrorSumUp(ByRef trainSet As CTrainingDataSet) As Double
        Dim retn As Double = 0
        Dim tmpInput As List(Of Double) = New List(Of Double)
        Dim tmpOutput As List(Of Double) = New List(Of Double)

        'Dim tmpErr(trainSet.InputDat.Length - 1) As Double
```

```
        For I As Integer = 0 To trainSet.InputDat.Length - 1
            tmpInput = New List(Of Double)
            tmpOutput = New List(Of Double)

            With trainSet.InputDat(I)
                'tmpInput.Add(.monthsSinceLastDryDock)
                tmpInput.Add(.prop_rpm)
                tmpInput.Add(.draft_m)
                tmpInput.Add(.trim_m)
                tmpInput.Add(.wind_deg)
                tmpInput.Add(.wind_kts)
                tmpInput.Add(.wave_deg)
                tmpInput.Add(.wave_m)

                tmpOutput.Add(.spd_kts)
                'tmpOutput.Add(.fdo_tph)

                Dim rOut As List(Of Double) = ANN.Execute(tmpInput)
                retn += (rOut.Item(0) - tmpOutput.Item(0)) ^ 2
                'retn += (rOut.Item(0) - tmpOutput.Item(0)) ^ 2 + (rOut.Item(1) -
tmpOutput.Item(1)) ^ 2

                'tmpErr(I) = (rOut.Item(0) - tmpOutput.Item(0))
            End With
        Next
        Return retn
    End Function


    Public Sub GetData()
    End Sub
End Class


Public Class CNormalizeValue
    Public MinValue As Double
    Public MaxValue As Double

    Public Sub New(ByVal valMin As Double, ByVal valMax As Double)
        MinValue = valMin
        MaxValue = valMax
    End Sub
End Class


Public Class CTrainingDataSet
    Public mGlbOrgDate As Date = New Date(2010, 1, 1)
    Public InputDat() As CShipOperationDataSet
    Public TestDat() As CShipOperationDataSet
    Public NormSet As CNormalizeSet

    Public Sub New()
    End Sub


    'Public Sub New(ByVal NormFileName As String)
    '    NormSet = New CNormalizeSet()
    '    NormSet.LoadNormDataFromFile(NormFileName)
    'End Sub
```

```
    Public Sub New(ByRef orgNormSet As CNormalizeSet, ByRef workingrecords As
CShipPerformanceRecordList, ByRef TimeOfRecord As Date, ByVal NoOfTestSample As
Integer)
        NormSet = New CNormalizeSet()
        NormSet.CopyFrom(orgNormSet)

        ReDim TestDat(10000)
        Dim testCount As Integer

        ReDim InputDat(10000)
        Dim count As Integer = 0

        For I As Integer = 0 To workingrecords.perf.Length - 1
            If workingrecords.perf(I).doR >= TimeOfRecord Then
                If I Mod 5 >= NoOfTestSample Then
                    InputDat(count) = New CShipOperationDataSet()
                    With workingrecords.perf(I)
                        InputDat(count).monthsSinceLastDryDock = 1 ' Val(v(0))
                        InputDat(count).prop_rpm = .sRPM 'Val(v(1))
                        InputDat(count).draft_m = 1 ' Val(v(2))
                        InputDat(count).trim_m = 0  ' Val(v(3))
                        InputDat(count).wind_deg = .WndDir_deg ' Val(v(4))
                        InputDat(count).wind_kts = .WndSpd_kts ' Val(v(5))
                        InputDat(count).wave_deg = .WavDir_deg ' Val(v(6))
                        InputDat(count).wave_m = .WavHgt_m  ' Val(v(7))
                        InputDat(count).spd_kts = .SOW ' Val(v(8))
                        InputDat(count).fdo_tph = 1 ' Val(v(9))
                    End With

                    count += 1


                    If count > InputDat.Length - 10 Then
                        ReDim Preserve InputDat(count + 100)
                    End If

                Else
                    TestDat(testCount) = New CShipOperationDataSet()
                    With workingrecords.perf(I)
                        TestDat(testCount).monthsSinceLastDryDock = 1 ' Val(v(0))
                        TestDat(testCount).prop_rpm = .sRPM 'Val(v(1))
                        TestDat(testCount).draft_m = 1 ' Val(v(2))
                        TestDat(testCount).trim_m = 0  ' Val(v(3))
                        TestDat(testCount).wind_deg = .WndDir_deg ' Val(v(4))
                        TestDat(testCount).wind_kts = .WndSpd_kts ' Val(v(5))
                        TestDat(testCount).wave_deg = .WavDir_deg ' Val(v(6))
                        TestDat(testCount).wave_m = .WavHgt_m  ' Val(v(7))
                        TestDat(testCount).spd_kts = .SOW ' Val(v(8))
                        TestDat(testCount).fdo_tph = 1 ' Val(v(9))
                    End With

                    testCount += 1


                    If testCount > TestDat.Length - 10 Then
                        ReDim Preserve TestDat(testCount + 100)
                    End If
```

```
                End If
            End If

        Next

        If count > 0 Then
            ReDim Preserve InputDat(count - 1)
        Else
            InputDat = Nothing
        End If


        If testCount > 0 Then
            ReDim Preserve TestDat(testCount - 1)
        Else
            TestDat = Nothing
        End If

        NormalizeInputData()
        NormalizeTestData()
    End Sub


    'Public Sub New(ByVal NormFileName As String, ByVal DatFileName As String)
    '    NormSet = New CNormalizeSet()
    '    NormSet.LoadNormDataFromFile(NormFileName)

    '    LoadDataFile(DatFileName)
    '    NormalizeInputData()
    'End Sub

    Public Sub NormalizeInputData()
        Dim retn As CShipOperationDataSet = New CShipOperationDataSet()

        If IsNothing(InputDat) = False Then
            For I As Integer = 0 To InputDat.Length - 1
                retn = NormSet.ShipOperationData_2_NormalizeValue(InputDat(I))
                InputDat(I).CopyFrom(retn)
            Next
        End If

    End Sub

    Public Sub NormalizeTestData()
        Dim retn As CShipOperationDataSet = New CShipOperationDataSet()

        If IsNothing(TestDat) = False Then
            For I As Integer = 0 To TestDat.Length - 1
                retn = NormSet.ShipOperationData_2_NormalizeValue(TestDat(I))
                TestDat(I).CopyFrom(retn)
            Next
        End If

    End Sub

    Public Function GetNormalizeOpertionalDataSet(ByRef workingstatusDetail As
CShipPerformanceRecord) As CShipOperationDataSet
        Dim retn As CShipOperationDataSet = New CShipOperationDataSet()
        Dim retn1 As CShipOperationDataSet
```

```
        retn.monthsSinceLastDryDock = 1 ' Val(v(0))
        retn.prop_rpm = workingstatusDetail.sRPM 'Val(v(1))
        retn.draft_m = 1 ' Val(v(2))
        retn.trim_m = 0  ' Val(v(3))
        retn.wind_deg = workingstatusDetail.WndDir_deg ' Val(v(4))
        retn.wind_kts = workingstatusDetail.WndSpd_kts ' Val(v(5))
        retn.wave_deg = workingstatusDetail.WavDir_deg ' Val(v(6))
        retn.wave_m = workingstatusDetail.WavHgt_m  ' Val(v(7))
        retn.spd_kts = workingstatusDetail.SOW ' Val(v(8))
        retn.fdo_tph = 1 ' Val(v(9))

        retn1 = NormSet.ShipOperationData_2_NormalizeValue(retn)

        Return retn1
    End Function


    Public Sub LoadDataFile(ByVal DatFileName As String)
        Dim stRead As StreamReader = New StreamReader(DatFileName)
        ReDim InputDat(10000)
        Dim count As Integer = 0
        Dim st As String
        Dim v() As String

        While Not stRead.EndOfStream
            st = stRead.ReadLine()
            v = Split(st, ";")

            If v.Length = 10 Then
                InputDat(count) = New CShipOperationDataSet(v)
                count += 1
            End If
        End While
        stRead.Close()

        If count > 0 Then
            ReDim Preserve InputDat(count - 1)
        Else
            InputDat = Nothing
        End If
    End Sub

    Public Sub AddASet(ByRef operationDat As CShipOperationDataSet, ByVal
DatFileName As String)
        Dim st As String
        Dim stWrite As StreamWriter = New StreamWriter(DatFileName, True)
        With operationDat
            st = String.Format("{0};{1};{2};{3};{4};{5};{6};{7};{8};{9}",
.monthsSinceLastDryDock, .prop_rpm, .draft_m, .trim_m, .wind_deg, .wind_kts,
.wave_deg, .wave_m, .spd_kts, .fdo_tph)
        End With
        stWrite.WriteLine(st)
        stWrite.Close()
    End Sub

    Public Sub InitFromWorkingStatus(ByRef workingStatusCollect As
CShipPerformanceRecordList)
        ReDim InputDat(workingStatusCollect.perf.Length - 1)
```

```
        For I As Integer = 0 To workingStatusCollect.perf.Length - 1
            InputDat(I) = New CShipOperationDataSet()
            InputDat(I).draft_m = workingStatusCollect.perf(I).sDraft
            InputDat(I).trim_m = workingStatusCollect.perf(I).sTrim
            InputDat(I).prop_rpm = workingStatusCollect.perf(I).sRPM

            InputDat(I).fdo_tph = 1

            InputDat(I).wind_deg = workingStatusCollect.perf(I).WndDir_deg
            InputDat(I).wind_kts = workingStatusCollect.perf(I).WndSpd_kts
            InputDat(I).wave_deg = workingStatusCollect.perf(I).WavDir_deg
            InputDat(I).wave_m = workingStatusCollect.perf(I).WavHgt_m
            InputDat(I).spd_kts = workingStatusCollect.perf(I).SOW
        Next
    End Sub

End Class


Public Class CShipOperationDataSet
    Public monthsSinceLastDryDock As Double
    Public wind_deg As Double
    Public wind_kts As Double
    Public wave_deg As Double
    Public wave_m As Double
    Public prop_rpm As Double
    Public draft_m As Double
    Public trim_m As Double
    Public spd_kts As Double
    Public fdo_tph As Double

    Public Sub New()
    End Sub

    Public Sub CopyFrom(ByRef orgSet As CShipOperationDataSet)
        With orgSet
            monthsSinceLastDryDock = .monthsSinceLastDryDock
            prop_rpm = .prop_rpm
            draft_m = .draft_m
            trim_m = .trim_m
            wind_deg = .wind_deg
            wind_kts = .wind_kts
            wave_deg = .wave_deg
            wave_m = .wave_m
            spd_kts = .spd_kts
            fdo_tph = .fdo_tph
        End With
    End Sub

    Public Sub New(ByRef v() As String)
        monthsSinceLastDryDock = Val(v(0))
        prop_rpm = Val(v(1))
        draft_m = Val(v(2))
        trim_m = Val(v(3))
        wind_deg = Val(v(4))
        wind_kts = Val(v(5))
        wave_deg = Val(v(6))
        wave_m = Val(v(7))
        spd_kts = Val(v(8))
```

```
        fdo_tph = Val(v(9))
    End Sub
End Class

Public Class CNormalizeSet
    Public NormMonthsSinceLastDryDock As CNormalizeValue
    Public NormRPM As CNormalizeValue
    Public NormSpd As CNormalizeValue
    Public NormTrim As CNormalizeValue
    Public NormDraft As CNormalizeValue
    Public NormWindSpd As CNormalizeValue
    Public NormWindDir As CNormalizeValue
    Public NormWaveHgt As CNormalizeValue
    Public NormWaveDir As CNormalizeValue
    Public NormFuelConsum As CNormalizeValue


    Public Sub New()
    End Sub

    Public Sub CopyFrom(ByRef org As CNormalizeSet)
        With org
            NormMonthsSinceLastDryDock = New
CNormalizeValue(.NormMonthsSinceLastDryDock.MinValue,
.NormMonthsSinceLastDryDock.MaxValue)
            NormRPM = New CNormalizeValue(.NormRPM.MinValue, .NormRPM.MaxValue)
            NormSpd = New CNormalizeValue(.NormSpd.MinValue, .NormSpd.MaxValue)
            NormTrim = New CNormalizeValue(.NormTrim.MinValue, .NormTrim.MaxValue)
            NormDraft = New CNormalizeValue(.NormDraft.MinValue,
.NormDraft.MaxValue)
            NormWindSpd = New CNormalizeValue(.NormWindSpd.MinValue,
.NormWindSpd.MaxValue)
            NormWindDir = New CNormalizeValue(.NormWindDir.MinValue,
.NormWindDir.MaxValue)
            NormWaveHgt = New CNormalizeValue(.NormWaveHgt.MinValue,
.NormWaveHgt.MaxValue)
            NormWaveDir = New CNormalizeValue(.NormWaveDir.MinValue,
.NormWaveDir.MaxValue)
            NormFuelConsum = New CNormalizeValue(.NormFuelConsum.MinValue,
.NormFuelConsum.MaxValue)
        End With
    End Sub

    Public Sub LoadNormDataFromFile(ByVal filename As String)
        If My.Computer.FileSystem.FileExists(filename) = False Then
            MsgBox("Không tìm được File Normalize Data")
            Exit Sub
        End If

        Dim stRead As StreamReader = New StreamReader(filename)
        Dim st As String = stRead.ReadToEnd()
        stRead.Close()

        Dim stdat() As String = Split(st, vbCrLf)

        If IsNothing(stdat) = False Then
            LoadData(stdat)
        End If
    End Sub
```

```
    Public Sub LoadData(ByRef st() As String)
        Dim v() As String

        For I As Integer = 0 To st.Length - 1
            v = Split(st(I), ";")

            If v.Length = 3 Then
                If String.Compare(v(0), "RPM", True) = 0 Then
                    NormRPM = New CNormalizeValue(Val(v(1)), Val(v(2)))
                ElseIf String.Compare(v(0), "MonthsSinceLastDryDock", True) = 0 Then
                    NormMonthsSinceLastDryDock = New CNormalizeValue(Val(v(1)),
Val(v(2)))
                ElseIf String.Compare(v(0), "Trim_m", True) = 0 Then
                    NormTrim = New CNormalizeValue(Val(v(1)), Val(v(2)))
                ElseIf String.Compare(v(0), "Draft_m", True) = 0 Then
                    NormDraft = New CNormalizeValue(Val(v(1)), Val(v(2)))
                ElseIf String.Compare(v(0), "WindDir_deg", True) = 0 Then
                    NormWindDir = New CNormalizeValue(Val(v(1)), Val(v(2)))
                ElseIf String.Compare(v(0), "WindSpd_kts", True) = 0 Then
                    NormWindSpd = New CNormalizeValue(Val(v(1)), Val(v(2)))
                ElseIf String.Compare(v(0), "WaveHgt_m", True) = 0 Then
                    NormWaveHgt = New CNormalizeValue(Val(v(1)), Val(v(2)))
                ElseIf String.Compare(v(0), "WaveDir_deg", True) = 0 Then
                    NormWaveDir = New CNormalizeValue(Val(v(1)), Val(v(2)))
                ElseIf String.Compare(v(0), "Spd_kts", True) = 0 Then
                    NormSpd = New CNormalizeValue(Val(v(1)), Val(v(2)))
                ElseIf String.Compare(v(0), "FuelConsume_tph", True) = 0 Then
                    NormFuelConsum = New CNormalizeValue(Val(v(1)), Val(v(2)))
                End If
            End If
        Next
    End Sub


    Public Function CalculateNormalizeValue(ByVal inputVal As Double, ByRef
NormValue As CNormalizeValue) As Double
        If inputVal < NormValue.MinValue Then Return 0
        If inputVal > NormValue.MaxValue Then Return 1

        Dim retn As Double = (inputVal - NormValue.MinValue) / (NormValue.MaxValue -
NormValue.MinValue)
        Return retn
    End Function


    Public Function CalculateValue(ByVal outputVal As Double, ByRef NormValue As
CNormalizeValue) As Double
        Dim retn As Double = NormValue.MinValue + outputVal * (NormValue.MaxValue -
NormValue.MinValue)
        Return retn
    End Function


    Public Function ShipOperationData_2_NormalizeValue(ByRef operationDat As
CShipOperationDataSet) As CShipOperationDataSet
        Dim retn As CShipOperationDataSet = New CShipOperationDataSet()
```

```
        retn.monthsSinceLastDryDock =
CalculateNormalizeValue(operationDat.monthsSinceLastDryDock,
NormMonthsSinceLastDryDock)
        retn.trim_m = CalculateNormalizeValue(operationDat.trim_m, NormTrim)
        retn.draft_m = CalculateNormalizeValue(operationDat.draft_m, NormDraft)
        retn.prop_rpm = CalculateNormalizeValue(operationDat.prop_rpm, NormRPM)
        retn.wind_deg = CalculateNormalizeValue(operationDat.wind_deg, NormWindDir)
        retn.wind_kts = CalculateNormalizeValue(operationDat.wind_kts, NormWindSpd)
        retn.wave_m = CalculateNormalizeValue(operationDat.wave_m, NormWaveHgt)
        retn.wave_deg = CalculateNormalizeValue(operationDat.wave_deg, NormWaveDir)
        retn.spd_kts = CalculateNormalizeValue(operationDat.spd_kts, NormSpd)
        retn.fdo_tph = CalculateNormalizeValue(operationDat.fdo_tph, NormFuelConsum)
        Return retn
    End Function



    Public Function NormalizedOutput_2_ShipSpd(ByRef normalizedSpd As Double) As
Double()
        Dim retn(1) As Double
        retn(0) = CalculateValue(normalizedSpd, NormSpd)
        Return retn
    End Function


    Public Function NormalizedOutput_2_ShipSpdAndFuelcomsume(ByRef normalizedSpd As
Double, ByRef normalizedFuelComsume As Double) As Double()
        Dim retn(1) As Double
        retn(0) = CalculateValue(normalizedSpd, NormSpd)
        retn(1) = CalculateValue(normalizedFuelComsume, NormFuelConsum)
        Return retn
    End Function

End Class

Public Class NeuralNet
    Shared r As New Random

    Public Class Dendrite
        Dim _weight As Double

        Property Weight As Double
            Get
                Return _weight
            End Get
            Set(ByVal value As Double)
                _weight = value
            End Set
        End Property

        Public Sub New()
            Me.Weight = r.NextDouble()
        End Sub
    End Class

    Public Class Neuron
        Dim _dendrites As New List(Of Dendrite)
        Dim _dendriteCount As Integer
        Dim _bias As Double
```

```
    Dim _value As Double
    Dim _delta As Double

    Public Property Dendrites As List(Of Dendrite)
        Get
            Return _dendrites
        End Get
        Set(ByVal value As List(Of Dendrite))
            _dendrites = value
        End Set
    End Property

    Public Property Bias As Double
        Get
            Return _bias
        End Get
        Set(ByVal value As Double)
            _bias = value
        End Set
    End Property

    Public Property Value As Double
        Get
            Return _value
        End Get
        Set(ByVal value As Double)
            _value = value
        End Set
    End Property

    Public Property Delta As Double
        Get
            Return _delta
        End Get
        Set(ByVal value As Double)
            _delta = value
        End Set
    End Property

    Public ReadOnly Property DendriteCount As Integer
        Get
            Return _dendrites.Count
        End Get
    End Property

    Public Sub New()
        Me.Bias = r.NextDouble()
    End Sub
End Class

Public Class Layer
    Dim _neurons As New List(Of Neuron)
    Dim _neuronCount As Integer

    Public Property Neurons As List(Of Neuron)
        Get
            Return _neurons
        End Get
        Set(ByVal value As List(Of Neuron))
```

```
            _neurons = value
        End Set
    End Property

    Public ReadOnly Property NeuronCount As Integer
        Get
            Return _neurons.Count
        End Get
    End Property

    Public Sub New(ByVal neuronNum As Integer)
        _neuronCount = neuronNum
    End Sub
End Class

Public Class NeuralNetwork
    Dim _layers As New List(Of Layer)
    Dim _learningRate As Double

    Public Property Layers As List(Of Layer)
        Get
            Return _layers
        End Get
        Set(ByVal value As List(Of Layer))
            _layers = value
        End Set
    End Property

    Public Property LearningRate As Double
        Get
            Return _learningRate
        End Get
        Set(ByVal value As Double)
            _learningRate = value
        End Set
    End Property

    Public ReadOnly Property LayerCount As Integer
        Get
            Return _layers.Count
        End Get
    End Property

    Sub New(ByVal LearningRate As Double, ByVal nLayers As List(Of Integer))
        If nLayers.Count < 2 Then Exit Sub

        Me.LearningRate = LearningRate

        For ii As Integer = 0 To nLayers.Count - 1

            Dim l As Layer = New Layer(nLayers(ii) - 1)
            Me.Layers.Add(l)

            For jj As Integer = 0 To nLayers(ii) - 1
                l.Neurons.Add(New Neuron())
            Next

            For Each n As Neuron In l.Neurons
                If ii = 0 Then n.Bias = 0
```

```
                    If ii > 0 Then
                        For k As Integer = 0 To nLayers(ii - 1) - 1
                            n.Dendrites.Add(New Dendrite)
                        Next
                    End If

                Next

            Next
        End Sub

        Function Execute(ByVal inputs As List(Of Double)) As List(Of Double)
            If inputs.Count <> Me.Layers(0).NeuronCount Then
                Return Nothing
            End If

            For ii As Integer = 0 To Me.LayerCount - 1
                Dim curLayer As Layer = Me.Layers(ii)

                For jj As Integer = 0 To curLayer.NeuronCount - 1
                    Dim curNeuron As Neuron = curLayer.Neurons(jj)

                    If ii = 0 Then
                        curNeuron.Value = inputs(jj)
                    Else
                        curNeuron.Value = 0
                        For k = 0 To Me.Layers(ii - 1).NeuronCount - 1
                            curNeuron.Value = curNeuron.Value + Me.Layers(ii -
1).Neurons(k).Value * curNeuron.Dendrites(k).Weight
                        Next k

                        curNeuron.Value = TransferFunction(curNeuron.Value +
curNeuron.Bias)
                    End If

                Next
            Next

            Dim outputs As New List(Of Double)
            Dim la As Layer = Me.Layers(Me.LayerCount - 1)
            For ii As Integer = 0 To la.NeuronCount - 1
                outputs.Add(la.Neurons(ii).Value)
            Next

            Return outputs
        End Function


        Private Function TransferFunction(ByVal Value As Double) As Double
            Return mySigmoid(Value)
        End Function


        Private Function myLinear(ByVal Value As Double) As Double
            Return Value
        End Function
```

```
        Private Function mySigmoid(ByVal Value As Double) As Double
            Return 1 / (1 + Math.Exp(Value * -1))
        End Function

        Public Overrides Function ToString() As String
            Dim nstr As New StringBuilder()

            For Each l As Layer In Me.Layers
                nstr.AppendLine("--+-- Layer")
                nstr.AppendLine("  |   Neurons: " & l.NeuronCount)

                For Each n As Neuron In l.Neurons

                    nstr.AppendLine("  |--+-- Neuron")
                    nstr.AppendLine("  |  |   Bias: " & n.Bias)
                    nstr.AppendLine("  |  |   Delta: " & n.Delta)
                    nstr.AppendLine("  |  |   Value: " & n.Value)
                    nstr.AppendLine("  |  |   Dendrites: " & n.DendriteCount)

                    For Each d As Dendrite In n.Dendrites
                        nstr.AppendLine("  |  |--+-- Dendrite")
                        nstr.AppendLine("  |  |  |   Weight: " & d.Weight)
                    Next

                Next
            Next

            nstr.Append("====== EONN ======")
            Return nstr.ToString
        End Function

        Public Function Train(ByVal inputs As List(Of Double), ByVal outputs As
List(Of Double)) As Boolean
            If inputs.Count <> Me.Layers(0).NeuronCount Or outputs.Count <>
Me.Layers(Me.LayerCount - 1).NeuronCount Then
                Return False
            End If

            Execute(inputs)

            For ii = 0 To Me.Layers(Me.LayerCount - 1).NeuronCount - 1
                Dim curNeuron As Neuron = Me.Layers(Me.LayerCount - 1).Neurons(ii)

                curNeuron.Delta = curNeuron.Value * (1 - curNeuron.Value) *
(outputs(ii) - curNeuron.Value)

                For jj = Me.LayerCount - 2 To 1 Step -1
                    For kk = 0 To Me.Layers(jj).NeuronCount - 1
                        Dim iNeuron As Neuron = Me.Layers(jj).Neurons(kk)

                        iNeuron.Delta = iNeuron.Value *
                                        (1 - iNeuron.Value) * Me.Layers(jj +
1).Neurons(ii).Dendrites(kk).Weight *
                                        Me.Layers(jj + 1).Neurons(ii).Delta
                    Next kk
                Next jj
            Next ii
```

```
            For ii = Me.LayerCount - 1 To 0 Step -1
                For jj = 0 To Me.Layers(ii).NeuronCount - 1
                    Dim iNeuron As Neuron = Me.Layers(ii).Neurons(jj)
                    iNeuron.Bias = iNeuron.Bias + (Me.LearningRate * iNeuron.Delta)

                    For kk = 0 To iNeuron.DendriteCount - 1
                        iNeuron.Dendrites(kk).Weight = iNeuron.Dendrites(kk).Weight
+ (Me.LearningRate * Me.Layers(ii - 1).Neurons(kk).Value * iNeuron.Delta)
                    Next kk
                Next jj
            Next ii

            Return True
        End Function









        Public Function LoadNetWorkParameter(ByVal params As String) As Boolean
            Dim v() As String = Split(params, ";")
            Dim idx As Integer = 0

            Try
                For I As Integer = 0 To Me.LayerCount - 1
                    For J As Integer = 0 To Me.Layers(I).NeuronCount - 1
                        Me.Layers(I).Neurons(J).Bias = Val(v(idx))
                        idx += 1

                        For K As Integer = 0 To
Me.Layers(I).Neurons(J).DendriteCount - 1
                            Me.Layers(I).Neurons(J).Dendrites(K).Weight =
Val(v(idx))
                            idx += 1
                        Next
                    Next
                Next

            Catch ex As Exception
                Return False
            End Try

            Return True
        End Function

        'Public Function LoadNetWorkParameter(ByVal filename As String) As Boolean
        '    Dim stRead As StreamReader = New StreamReader(filename)
        '    Dim st As String
        '    Dim v() As String

        '    st = stRead.ReadLine()
        '    v = Split(st, ";")
        '    If Me.LayerCount <> v.Length Then
        '        MsgBox("Cấu trúc mạng Neural không đúng")
```

```
'        stRead.Close()
'        Return False
'    End If

'    For I As Integer = 0 To Me.LayerCount - 1
'        If Val(v(I)) <> Me.Layers(I).NeuronCount Then
'            MsgBox("Cấu trúc mạng Neural không đúng")
'            stRead.Close()
'            Return False
'        End If
'    Next


'    sua cho nay, ghi moi cai 1 dong cho cac huong gio tuong doi
'    tên file bao gom rpm, draft, trim
'    For I As Integer = 0 To Me.LayerCount - 1
'        For J As Integer = 0 To Me.Layers(I).NeuronCount - 1
'            st = stRead.ReadLine()
'            v = Split(st, ";")
'            Me.Layers(I).Neurons(J).Bias = Val(v(0))

'            For K As Integer = 0 To Me.Layers(I).Neurons(J).DendriteCount -
1
'                Me.Layers(I).Neurons(J).Dendrites(K).Weight = Val(v(K + 1))
'            Next
'        Next
'    Next

'    stRead.Close()
'    Return True
'End Function


    Public Sub InitializeNetworkRandomly()
        For I As Integer = 0 To Me.LayerCount - 1
            For J As Integer = 0 To Me.Layers(I).NeuronCount - 1
                Me.Layers(I).Neurons(J).Bias = Rnd()

                For K As Integer = 0 To Me.Layers(I).Neurons(J).DendriteCount -
1
                    Me.Layers(I).Neurons(J).Dendrites(K).Weight = Rnd()
                Next
            Next
        Next
    End Sub


    Public Sub Draw(ByVal hDC As Graphics, ByVal startX As Integer, ByVal startY
As Integer, ByVal scale As Integer, ByVal hspace As Integer, ByVal vspace As
Integer, ByVal iColor As Color, ByVal hColor As Color, ByVal oColor As Color)
        Dim i As Integer
        Dim k As Integer
        Dim j As Integer

        Dim x As Integer
        Dim y As Integer

        hDC.Clear(Color.White)
        For i = 0 To Me.LayerCount - 1
```

```
                x = startX - hspace * (Me.Layers(i).NeuronCount / 2)
                y = startY - (vspace * i)

                For k = 0 To Me.Layers(i).NeuronCount - 1
                    Dim b As SolidBrush
                    Dim p As Pen
                    Select Case i
                        Case 0
                            b = New SolidBrush(iColor)
                            p = New Pen(iColor)
                            hDC.DrawEllipse(p, x, y, scale, scale)
                            hDC.FillEllipse(b, x, y, scale, scale)
                        Case Me.LayerCount - 1
                            b = New SolidBrush(oColor)
                            p = New Pen(oColor)
                            hDC.DrawEllipse(p, x, y, scale, scale)
                            hDC.FillEllipse(b, x, y, scale, scale)
                        Case Else
                            b = New SolidBrush(hColor)
                            p = New Pen(hColor)
                            hDC.DrawEllipse(p, x, y, scale, scale)
                            hDC.FillEllipse(b, x, y, scale, scale)
                    End Select
                    b.Dispose()
                    p.Dispose()

                    If i > 0 Then
                        Dim denX1 As Integer = x + (scale / 2)
                        Dim denY1 As Integer = y + scale
                        Dim denX2 As Integer = (startX - hspace * (Me.Layers(i -
1).NeuronCount / 2)) + (scale / 2)
                        Dim denY2 As Integer = y + vspace
                        For j = 0 To Me.Layers(i).Neurons(k).DendriteCount - 1
                            hDC.DrawLine(Pens.Black, denX1, denY1, denX2, denY2)
                            denX2 = denX2 + hspace
                        Next
                    End If
                    x = x + hspace

                Next
            Next
        End Sub

    End Class
End Class


Public Class CANNCalculate
End Class
```