# Distributed Algorithms for n-Ship Collision Avoidance

Kim, Donggyun

(Degree)
博士（海事科学）

(Date of Degree)
2016-09-25

(Date of Publication)
2017-09-01

(Resource Type)
doctoral thesis

(Report Number)
甲第6757号

(URL)
https://hdl.handle.net/20.500.14094/D1006757

# Doctoral Dissertation

## Distributed Algorithms for *n*-Ship Collision Avoidance

(*n* 船衝突回避のための分散アルゴリズム)

August 2016

Graduate School of Maritime Sciences, Kobe University

DONGGYUN KIM

KOBE UNIVERSITY

# *Abstract*

Graduate School of Maritime Sciences

Doctor of Philosophy

**Distributed Algorithms for $n$-Ship Collision Avoidance**

by Donggyun Kim

As vital transportation carriers in trade, ships have the advantage of stability, economy, and bulk capacity over trucks, trains, and airplanes. Even though navigation technology has been developing year by year, ship collision still accounts for a large percentage of maritime accidents. Their loss and cost due to collisions and other accidents exceed those of any other mode of transportation. It is very difficult for officers to ascertain routes that will avoid collisions, especially when multiple ships travel the same waters.

Ship collision avoidance involves helping ships find routes that will best enable them to avoid a collision. When more than two ships encounter one another, the procedure becomes more complex since a slight change in course by one ship might cause a "butterfly effect" in the whole system. To prevent ship collisions many ways have been suggested, such as COLREGs, ship domain, fuzzy theory, and genetic algorithm. These methods work well in one-on-one situations, but are more difficult to apply in multiple-ship situations.

To support the need to find safe routes for ship traveling in crowded waters, I apply the Distributed Algorithm. The Distributed Algorithm does not require a server, which could be came a bottleneck in the system. The objective of this thesis is to reduce the ship collision risk in multiple-ship situations. To do that, I suggest three kinds of Distributed Algorithms, such as Distributed Local Search Algorithm (DLSA), Distributed Tabu Search Algorithm (DTSA) and Distributed Stochastic Search Algorithm (DSSA). Along with the development of the Distributed Algorithms, I also suggest a new cost function that considers both safety and efficiency to find a safe course.

First, I introduce the DLSA. The main purpose of DLSA is to reduce collision risk among multiple ships. DLSA is a distributed algorithm in which multiple ships communicate with each other within a certain area. DLSA computes a cost based on the information received from neighboring ships. By exchanging information on, for example, *next-intended courses* within a certain area among ships, ships having the maximum reduction in collision risk change courses simultaneously until all ships approach a destination without collision. In DLSA, I assume that ships can exchange information with each other (using a communication device such as the Automatic Identification System) to cooperatively establish routes to avoid collisions. More specifically, when multiple ships meet, the ship that can reduce collision risk most significantly has the right to choose its next course. Where there is a tie in the maximum risk reduction, the one with the highest priority has the right to choose its next course. These choices are then relayed to their neighboring ships as their current courses. Each individual ship computes its collision risk based on the information on current courses that it receives from the neighboring ships.

Second, I show DTSA to make up for the weak points of DLSA. DLSA works well empirically, but, it is sometimes trapped in Quasi-Local Minimum (QLM) that prevents a ship from changing course even when at risk of collision. To deal with this issue, therefore, I developed a new distributed algorithm called the DTSA. DTSA uses a Tabu Search (TS) to escape from QLM. There are several types of memory, such as short-term, intermediate-term and long-term. By using memory to prohibit certain moves, TS searches for global optimization rather than local optimization. In short-term memory, the recently selected or visited solutions are put in tabu list. Therefore, an agent can search new solution in spite of worse one. Only a short-term memory may be enough to solve a given problem in conventional local search methods. The intermediate and long-term memory structures are used for solving more complex problems. For ship collision avoidance, I use only short-term memory structure. DTSA enables a ship to search for a new course compulsorily when trapped in QLM, to allow it to escape. In this way, DTSA enables ships to find shorter paths to their destinations while avoiding collisions.

Third, I introduce DSSA to compensate the problems of DLSA and DTSA. The common drawback of DLSA and DTSA are that it takes a relatively large number of messages for the ships to coordinate their actions. This could be fatal, especially in cases of emergency, where quick decisions should be made. DSSA enables each

ship to change her *next-intended course* in a stochastic manner immediately after receiving all of the intentions from the neighboring ships. It allows each ship to reduce the number of messages compared to DLSA and DTSA.

Forth, I propose new cost function. For a course, two things have to be taken into consideration, i.e. *collision risk* and *relative angle* are related to the safety and the efficiency for a certain course, respectively. Otherwise, a ship will be in a dangerous situation or she will go a long way round. Therefore, the cost function is made up of two parts.

Finally, to know the performance of DLSA, DTSA, and DSSA, I made experiments in various situations. For the performance benchmark, I computed the sailing distance, the average cost, the number of exchanged messages. As the results of experiments, I categorized the results according to each variable.

# Contents

**Abstract** i

**List of Figures** vi

**List of Tables** viii

**Abbreviations** ix

**Acknowledgements** x

**1 INTRODUCTION** 1
- 1.1 Ship collision . . . 1
  - 1.1.1 Ship's characteristics . . . 3
  - 1.1.2 Statistics of ship collision . . . 6
  - 1.1.3 Issues . . . 8
- 1.2 Thesis outline . . . 9

**2 Background and Related work** 10
- 2.1 Introduction . . . 10
- 2.2 COLREG . . . 11
- 2.3 Computational methodology for ship collision avoidance . . . 13
  - 2.3.1 Ship domain . . . 13
  - 2.3.2 Fuzzy theory . . . 16
  - 2.3.3 Genetic algorithm . . . 17

**3 Distributed Collision Avoidance** 19
- 3.1 Framework and terminology . . . 19
  - 3.1.1 Framework . . . 19
  - 3.1.2 Terminology . . . 20
- 3.2 Cost and improvement . . . 24

**4 Distributed Local Search Algorithm** 28
- 4.1 Introduction . . . 28
- 4.2 Local Search . . . 29
- 4.3 Distributed Local Search Algorithm for Ship Collision Avoidance . . . 31
- 4.4 Experiments . . . 37

4.4.1 Classification by Candidate course . . . 39
4.4.2 Classification by Detection range . . . 40
4.4.3 Classification by Safety domain . . . 42
4.4.4 Classification by Timestep . . . 42
4.5 Conclusion . . . 45

**5 Distributed Tabu Search Algorithm 47**
5.1 Introduction . . . 47
5.2 Tabu Search . . . 48
5.3 Distributed Tabu Search Algorithm for Ship Collision Avoidance . . . 50
5.4 Experiments . . . 54
5.4.1 1st experiment . . . 54
5.4.2 2nd experiment . . . 56
5.4.3 3rd experiment . . . 57
5.4.4 4th experiment . . . 57
5.4.5 5th experiment . . . 58
5.5 Conclusion . . . 61

**6 Distributed Stochastic Search Algorithm 62**
6.1 Introduction . . . 62
6.2 Motivation . . . 63
6.3 Distirbuted stochastic algorithm . . . 64
6.4 Detail . . . 66
6.5 Comparison DSSA with DLSA and DTSA . . . 70
6.6 Experiments . . . 72
6.6.1 Four-ship Encounter . . . 73
6.6.1.1 Cooperative and non-cooperative situation . . . 74
6.6.2 Twelve-ship Encounter . . . 75
6.6.3 One hundred-ship Encounter . . . 77
6.7 Conclusion . . . 78

**7 Conclusions and Future work 79**
7.1 Research summary . . . 79
7.2 Future work . . . 81

**Bibliography 82**

# List of Figures


1.1 Several types of ships, (a)Bulk carrier 'Sabrina I', (b)Oil tanker 'AbQaiq', (c)LNG tanker 'Puteri Firus Satu'. . . . 4
1.2 World largest container ship 'Oscar'. . . . 4
1.3 Comparison between Oscar and Tokyo tower. . . . 5
1.4 Oil on to the rocky shores (left) and Clean-up for an oiled shoreline (right) in Alaska, US. . . . 8
1.5 Composition of the thesis. . . . 9

2.1 The relationship between ships. . . . 13
2.2 Multiple-ship situations. . . . 13
2.3 Coordinate transformation. . . . 14
2.4 The change of $f(t)$. . . . 14
2.5 The change of the relative bearing I. . . . 17
2.6 The change of the relative bearing II. . . . 17

3.1 Framework. . . . 20
3.2 Basic terms. . . . 21
3.3 Limit of maneuvering course. . . . 22
3.4 The change of neighboring ships depending on the *detection range*. . . . 22
3.5 Multiple-ship situations. . . . 23
3.6 How to proceed to destination. . . . 23
3.7 Collision risk and relative angle . . . . 24
3.8 Candidate courses and the angle heading for a destination. . . . 25
3.9 Numerical example for computing costs and improvements. . . . 26

4.1 CSP example of color mapping . . . . 32
4.2 CSP example of chess . . . . 33
4.3 Flowchart of Distributed Local Search Algorithm. . . . 33
4.4 The process of Distributed Local Search Algorithm. . . . 34
4.5 Situations involving ships having different destinations. . . . 37
4.6 Success ratio by Candidate course. . . . 39
4.7 Average message and distance. . . . 39
4.8 Average cost. . . . 40
4.9 Success ratio by Detection range. . . . 41
4.10 Average message and distance by Candidate course. . . . 41
4.11 Average cost by Candidate course. . . . 41

4.12 Success ratio by Safety domain. . . . . 42
4.13 Average message and distance by Safety domain. . . . . 43
4.14 Average cost by Safety domain. . . . . 43
4.15 Success ratio by Timestep. . . . . 44
4.16 Average message and distance. . . . . 44
4.17 Average cost. . . . . 44

5.1 Sketch of Distributed Tabu Search Algorithm. . . . . 51
5.2 The process of Distributed Tabu Search Algorithm. . . . . 52
5.3 Situation for 1st experiment. . . . . 55
5.4 Result for 1st experiment. . . . . 56
5.5 Situation for 2nd experiment. . . . . 56
5.6 Result for 2nd experiment. . . . . 57
5.7 Situation for 3rd experiment. . . . . 58
5.8 Result for 3rd experiment. . . . . 58
5.9 Situation for 4th experiment. . . . . 59
5.10 Result for 4th experiment. . . . . 59
5.11 Situation for 5th experiment. . . . . 60
5.12 Result for 5th experiment. . . . . 60

6.1 Procedure for DSSA . . . . . 67
6.2 How to exchange messages for DLSA and DTSA . . . . . 71
6.3 How to exchange messages for DSSA . . . . . 71
6.4 Simulated trajectories by DSSA-C. . . . . 72
6.5 Simulated trajectories by DSSA-D. . . . . 73
6.6 Simulated trajectories by DSSA-E. . . . . 73
6.7 Simulated trajectories of four ships by Non-Cooperative (a, left) and Cooperative (b, right). . . . . 74
6.8 Result of simulation for Four-ship Encounter. . . . . 75
6.9 Simulated trajectories of twelve-ship by DSSA-A. . . . . 76
6.10 Result of simulation for Twelve-ship Encounter. . . . . 76
6.11 Initialization (a, left) and trajectories (b, right) for 100-ship encounter instance. . . . . 77
6.12 Result of simulation for One hundred-ship Encounter. . . . . 77

# List of Tables


1.1 Number of maritime accidents as of April 30, 2016 . . . . . 6
1.2 Number of Vessels by types as of April 30, 2016 . . . . . 7
1.3 Number of marine incidents as of April 30, 2016 . . . . . 7
1.4 Top 20 Major Spills Table . . . . . 8

4.1 Variable . . . . . 37
4.2 Result of Experiment . . . . . 38

5.1 Comparison of DLSA and DTSA. . . . . 50
5.2 Candidate course by Index used in experiments. . . . . 55
5.3 List of neighboring ships for experiment 1. . . . . 55

6.1 Five possible strategies for DSAs . . . . . 65
6.2 Comparison of DLSA, DTSA, and DSSA. . . . . 70

# Abbreviations

| | |
|---|---|
| **AIS** | **A**utomatic **I**dentification **S**ystem |
| **ARPA** | **A**utomatic **R**adar **P**lotting **A**ids |
| **CB** | **C**hangeable **B**earing |
| **COLREGs** | The International Regulations for Preventing Collisions at Sea |
| **CPA** | **C**loest **P**oint of **A**pproach |
| **CSP** | **C**onstraint **S**atisfaction **P**roblem |
| **DA** | **D**istributed **A**lgorithm |
| **DCPA** | **D**istance to **C**loest **P**oint of **A**pproach |
| **DisCSP** | **Dis**tributed **C**onstraint **S**atisfaction **P**roblem |
| **DLSA** | **D**istributed **L**ocal **S**earch **A**lgorithm |
| **DSSA** | **D**istributed **S**tochastic **S**earch **A**lgorithm |
| **DTSA** | **D**istributed **T**abu **S**earch **A**lgorithm |
| **DWT** | **D**ead **W**eight **T**onnage |
| **ECDIS** | **E**lectronic **C**hart **D**isplay and **I**nformation **S**ystem |
| **NOAA** | **N**ational **O**ceanic and **A**tmospheric **A**dministration |
| **OOW** | **O**fficer **O**f the **W**atch |
| **QLM** | **Q**uasi **L**ocal **M**inimum |
| **RADAR** | **RA**dio **D**etection **A**nd **R**anging |
| **TCPA** | **T**ime to **C**loest **P**oint of **A**pproach |
| **TEU** | **T**wenty-foot **E**quivalent **U**nit |
| **VCD** | **V**ariation of **C**ompass **D**egree |
| **VTS** | **V**essel **T**raffic **S**ervices |

# *Acknowledgements*

I would like to express my special appreciation to my supervisor Prof. Katsutoshi Hirayama for supporting my Ph.D and for his patience. Katsutoshi provided me with every bit of guidance, and assistance during my doctoral course. Again, I appreciate all his contributions of time, effort, and encouragement. His advice on the research has been priceless.

I would like to thank to Prof. Tenda Okimoto, for supporting me. Tenda is a faithful advisor and one of the smartest people.

I also would like to thank to Prof. Yamamura Saburo with a benevolent smile. Yamanura is kind and gentle.

I would like to thank to the member of Intelligent Informatics Laboratory for their helpful advice, especially Daisuke Hatano, and Kenta Hanada. All of you have been there to support me.

I would like to thank the graduate school of maritime sciences at Kobe University, especially Prof. Tomoya Horiguchi, Prof. Akira Sou, Prof. Kohei Hirono, Prof. Nobuyoshi Kouguchi, Prof. Mikio Takebayashi, Prof. Wayne Rossman and Prof. Shigeru Yoshida, for the substantial influence that their courses have had on my research.

A special thanks to my family. Words cannot express how grateful I am to my mother, father, and brother. Your prayer for me was what sustained me thus far.

At the end I would like express appreciation to Yuka for her love, support, and sacrifices. Thank you for cheering me up.

# Chapter 1

# INTRODUCTION

## 1.1 Ship collision

The ship is a watercraft to transport passenger or cargo from one to another. She has long been used throughout the world. There are many kinds of ships depending on the purpose, such as bulk carrier, tanker, container, LNG (liquified natural gas), and submarine. As vital transportation carriers in trade, ships have the advantage of stability, economy, and bulk capacity over airplanes, trucks, and trains. Even so, their loss and cost due to collisions and other accidents exceed those of any other mode of transportation. The size and speed of ships is rapidly increasing in order to boost economic efficiency. However, navigation technology has been developing year after year, ship collision still accounts for a large percentage of maritime accidents [1]. Ship collision is a physical impact between ships or a ship and a floating or fixed objects. If ships collide, the damage and cost can be astronomical. There are huge impacts on our life, economy and environments. It is very difficult for officers to ascertain routes that will avoid collisions, especially when multiple ships proceed the same waters.

To prevent ship collisions many ways have been suggested, e.g., lookouts, radar, and VHF radio. The 1972 COLREGs which is the regulation for preventing collision between ships. It specifies navigation rules to be followed by all ships at sea to prevent collisions. However, it would be very hard to describe all possible conditions in the form of rules due to the complexity of the actual marine environment. On top of that, it would be a big burden for an officer to consider many different variables to apply to the rules in time-pressed situations.

Technologically speaking, many related studies have been conducted. The term "Ship domain" involves that area surrounding a ship that the navigator wants to keep other ships clear of. Ship domain alone is not sufficient, however, for enabling one or more ships to simultaneously determine the collision risk for all of the ships concerned. More advanced methodologies, such as fuzzy theory, and genetic algorithm, have been proposed [2–8]. Fuzzy theory is useful in helping ships avoid collision in that fuzzy theory may define whether collision risk is based on Distance to Closest Point of Approach (DCPA), Time to Closest Point of Approach (TCPA), or relative bearing - algorithms that are difficult to apply to more than two ships simultaneously. These methods work well in one-on-one situations, but are more difficult to apply in multiple-ship situations.

However, in reality, collisions between ships frequently occur. This is partly due to the ever increasing size and speed of ships each year. A primary cause of ship collisions is officer error. OOW have generally some expertise in finding safe routes that will avoid ship collisions; however, particularly when shipping lanes are crowded and many ships encounter each other simultaneously, finding such routes is especially difficult for officers. The need to repeat this task throughout the voyage multiplies the risk of human error.

Ship collision avoidance involves helping ships find routes that will best enable them to avoid a collision. When more than two ships encounter one another, the procedure becomes more complex since a slight change in course by one ship might cause a "butterfly effect" in the whole system. To support the need to find safe routes for ship traveling in crowded waters, I propose the Distributed Algorithms such as Distributed Local Search Algorithm (DLSA), Distributed Tabu Search Algorithm (DTSA) and Distributed Stochastic Search Algorithm (DSSA). Along with the development of the Distributed Algorithms, I also suggest a new cost function that considers both safety and efficiency. By adjusting a weight factor of the cost function, a ship can consider both.

The main purpose of DLSA is to reduce collision risk among multiple ships. DLSA is a distributed algorithm in which multiple ships communicate with each other within a certain area. DLSA computes cost based on the information received from neighboring ships. By exchanging information on, for example, *next-intended courses* within a certain area among ships, ships having the maximum reduction in collision risk change courses simultaneously until all ships approach a destination without collision. In DLSA, I assume that ships can exchange information

with each other (using a communication device such as the Automatic Identification System (AIS)) to cooperatively establish routes to avoid collisions. More specifically, when multiple ships meet, the ship that can reduce collision risk most significantly has the right to choose its next course. Where there is a tie in the maximum risk reduction, the one with the highest priority has the right to choose its next course. These choices are then relayed to their neighboring ships as their current courses. Each individual ship computes its collision risk based on the information on current courses that it receives from the neighboring ships. This process is repeated until the collision risk disappears.

DLSA works well empirically, however, it is sometimes trapped in Quasi Local Minimum (QLM) that prevents a ship from changing course even when at risk of collision. To deal with this issue, I developed a new distributed algorithm called the Distributed Tabu Search Algorithm (DTSA). DTSA uses a tabu list to escape from QLM. DTSA enables a ship to search for a new course compulsorily when trapped in QLM, to allow it to escape.

The common drawback of these algorithms is that it takes a relatively large number of messages for the ships to coordinate their actions. This could be fatal, especially in cases of emergency, where quick decisions should be made. Distributed Stochastic Search Algorithm (DSSA) enables each ship to change her next-intended course in a stochastic manner immediately after receiving all of the intentions from the neighboring ships. It allows each ship to change her next-intended course in a stochastic manner immediately after receiving all of the intentions from the neighboring ships.

To know the performance of the Distributed Algorithms, I made experiments to compare DLSA, DTSA, and DSSA. I computed the average distance and cost, and the number of exchanged messages on the performance benchmark. Experiments results showed that in most cases, the proposals apply well in ship collision avoidance among multiple ships.

### 1.1.1 Ship's characteristics

There is a big difference between land and sea when an object moves, i.e. *buoyancy*. It is possible to make a ship bigger and bigger. Thus, ships can carry lots of cargoes easily to far away countries. Now more than ever, the size and speed of ships are


FIGURE 1.1: Several types of ships, (a)Bulk carrier 'Sabrina I', (b)Oil tanker 'AbQaiq', (c)LNG tanker 'Puteri Firus Satu'.


FIGURE 1.2: World largest container ship 'Oscar'.

rapidly increasing in order to boost economic efficiency. These cause the issues of those resulting from characteristics of ship:

- **High speed**: Normally, the speed of a container ship is around 24 knots (≈44 kilometers/hour). When it considers the ship's size, the force of inertia is very big. Also, there is no brake to stop like car. In other words, a ship cannot change her heading and speed easily. When changing a course, therefore, a ship has to decide next course carefully.

- **Massiveness**: There are different types and sizes of ships to meet the demands of transportation. For the fuel efficiency, furthermore, the sizes of

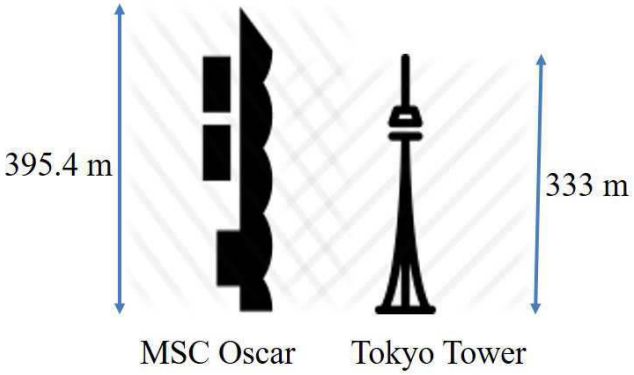



FIGURE 1.3: Comparison between Oscar and Tokyo tower.

ships are highly diversified. The containerization has allowed container shipping companies to achieve of economy of scale. The more the containers, the less the transportation cost. Cargo ships can be categorized by capacity. For example, there are various types of ships as the followings:

- **Aframax**: Aframax is medium-sized oil tankers.

- **Capesize**: Capesize ships are large and ultra cargo ships. They are categorized by Very Large Ore Carriers (VLOC) and Very Large Bulk Carriers (VLBC).

- **Chinamax**: Chinamax ships are the largest bulk carriers with 400,000 DeadWeight Tonnage (DWT)

- **Handymax**: Handymax are small-sized ships with 60,000 DWT.

- **Handysize**: Handysize are small-sized ships between 15,000 and 35,000 DWT.

For example, MSC "Oscar" is one of the largest container ships in the world as shown in Figure 1.2. The length is 395.4 meters and the tonnage is 197,362 DWT. The capacity for containers is 19,224 Twenty-foot Equivalent Unit(TEU). Compared to Tokyo tower that the length is 333 meters, she is longer and heavier. Therefore, when a ship starts to move, it should be decided carefully. And once she starts to move, it is hard to stop.

- **Automation**: For support an officer, there are many on-board machines, such as AIS, RADAR, ARPA, Electronic Chart Display and Information System (ECDIS), gyro compass, autopilot and communications equipment. When target ship penetrates the safety domain of own ship, an ARPA warns

an officer for dangerous situation. An autopilot can control the trajectory of a ship without constant control by an officer.

### 1.1.2 Statistics of ship collision

Table 1.1 shows the number of maritime accidents as of 2016 in Japan [9]. The total number of marine accidents is on a downward tendency. Among that, the number of collisions occupied the largest percentage of total accidents. The reasons due to a maneuvering mistake are also forming big part of marine accidents, such as contact (striking by an external object, but not another ship), grounding (striking the sea bottom), and capsizing.

TABLE 1.1: Number of maritime accidents as of April 30, 2016

| Year | Collision | Contact | Grounding | Sinking | Flooding | Capsizing | Fire | Explosion | Missing | Damage | Casualty | Others | Total |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 2016 | 48 | 23 | 38 | 0 | 8 | 9 | 5 | 0 | 0 | 5 | 37 | 0 | 173 |
| 2015 | 239 | 95 | 196 | 5 | 11 | 57 | 38 | 3 | 0 | 22 | 122 | 0 | 788 |
| 2014 | 266 | 115 | 213 | 7 | 11 | 61 | 35 | 1 | 0 | 37 | 150 | 3 | 899 |
| 2013 | 265 | 144 | 210 | 10 | 25 | 49 | 33 | 2 | 0 | 38 | 163 | 2 | 941 |
| 2012 | 246 | 132 | 264 | 5 | 21 | 55 | 44 | 2 | 0 | 34 | 155 | 0 | 958 |
| 2011 | 282 | 145 | 264 | 12 | 18 | 57 | 32 | 1 | 0 | 23 | 142 | 1 | 977 |
| 2010 | 356 | 180 | 369 | 15 | 18 | 50 | 35 | 2 | 0 | 26 | 146 | 0 | 1197 |
| 2009 | 325 | 174 | 431 | 16 | 19 | 58 | 42 | 3 | 0 | 38 | 217 | 2 | 1325 |
| 2008 | 181 | 101 | 255 | 12 | 4 | 28 | 15 | 3 | 0 | 30 | 61 | 0 | 690 |

Table 1.2 shows the number of vessels by types as of 2016. The total number of vessels exceeded 1,000 and it is showing a reduction trend from 2009. Among that, fishing boat recorded the largest percentage. And cargo ship and pleasure boat followed after that.

Table 1.3 shows the number of marine incidents as of 2016. Most of incidents are done by loss of control, such as machinery failure.

Table 1.2: Number of Vessels by types as of April 30, 2016

| Year | Passenger ship | Cargo ship | Tanker | Fishing vessel | Tug & Pushing boat | Recreational fishing boat | Angler tender boat | Work vessel | Barge & Lighter | Public-service ship | Pleasure boat | Personal water craft | Others | Total |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 2016 | 19 | 40 | 14 | 73 | 16 | 8 | 0 | 7 | 11 | 4 | 39 | 0 | 6 | 237 |
| 2015 | 47 | 172 | 59 | 347 | 49 | 29 | 7 | 25 | 39 | 8 | 244 | 49 | 13 | 1088 |
| 2014 | 49 | 213 | 63 | 390 | 81 | 39 | 5 | 34 | 55 | 16 | 224 | 67 | 16 | 1252 |
| 2013 | 47 | 206 | 64 | 427 | 93 | 39 | 4 | 35 | 69 | 20 | 243 | 62 | 18 | 1327 |
| 2012 | 61 | 272 | 58 | 413 | 84 | 30 | 8 | 36 | 57 | 13 | 221 | 48 | 8 | 1309 |
| 2011 | 49 | 268 | 91 | 451 | 83 | 36 | 6 | 26 | 48 | 16 | 217 | 43 | 20 | 1354 |
| 2010 | 80 | 382 | 98 | 500 | 116 | 52 | 6 | 44 | 76 | 22 | 225 | 64 | 16 | 1681 |
| 2009 | 79 | 437 | 72 | 535 | 146 | 35 | 5 | 34 | 96 | 35 | 228 | 63 | 22 | 1787 |
| 2008 | 40 | 253 | 42 | 251 | 77 | 25 | 4 | 24 | 54 | 11 | 120 | 31 | 6 | 938 |

Table 1.3: Number of marine incidents as of April 30, 2016

| | Loss of control | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Year | Machinery failure | Listing | Insufficient fuel | Stranded | Safety obstruction | Navigation obstruction | Others | Total |
| 2016 | 17 | 0 | 1 | 0 | 1 | 5 | 0 | 24 |
| 2015 | 81 | 0 | 3 | 4 | 4 | 11 | 0 | 103 |
| 2014 | 89 | 1 | 2 | 15 | 0 | 24 | 0 | 131 |
| 2013 | 98 | 2 | 6 | 7 | 3 | 25 | 0 | 141 |
| 2012 | 108 | 0 | 5 | 5 | 4 | 35 | 0 | 157 |
| 2011 | 96 | 1 | 6 | 10 | 1 | 35 | 0 | 149 |
| 2010 | 80 | 0 | 3 | 16 | 0 | 38 | 0 | 137 |
| 2009 | 102 | 0 | 3 | 33 | 0 | 59 | 0 | 197 |
| 2008 | 53 | 1 | 0 | 34 | 8 | 87 | 0 | 183 |

FIGURE 1.4: Oil on to the rocky shores (left) and Clean-up for an oiled shoreline (right) in Alaska, US.

TABLE 1.4: Top 20 Major Spills Table

| Position | Ship name | Year | Location | Spill Size |
|---|---|---|---|---|
| 1 | ATLANTIC EMPRESS | 1979 | Off Tobago, West Indies | 287,000 |
| 2 | ABT SUMMER | 1991 | 700 nautical miles off Angola | 260,000 |
| 3 | CASTILLO DE BELLVER | 1983 | Off Saldanha Bay, South Africa | 252,000 |
| 4 | AMOCO CADIZ | 1978 | Off Brittany, France | 223,000 |
| 5 | HAVEN | 1991 | Genoa, Italy | 144,000 |
| 6 | ODYSSEY | 1988 | 700 nautical miles off Nova Scotia, Canada | 132,000 |
| 7 | TORREY CANYON | 1967 | Scilly Isles, UK | 119,000 |
| 8 | SEA STAR | 1972 | Gulf of Oman | 115,000 |
| 9 | IRENES SERENADE | 1980 | Navarino Bay, Greece | 100,000 |
| 10 | URQUIOLA | 1976 | La Coruna, Spain | 100,000 |
| 11 | HAWAIIAN PATRIOT | 1977 | 300 nautical miles off Honolulu | 95,000 |
| 12 | INDEPENDENTA | 1979 | Bosphorus, Turkey | 94,000 |
| 13 | JAKOB MAERSK | 1975 | Oporto, Portugal | 88,000 |
| 14 | BRAER | 1993 | Shetland Islands, UK | 85,000 |
| 15 | AEGEAN SEA | 1992 | La Coruna, Spain | 74,000 |
| 16 | SEA EMPRESS | 1996 | Milford Haven, UK | 72000 |
| 17 | KHARK 5 | 1989 | 120 nautical miles off Atlantic coast of Morocco | 70,000 |
| 18 | NOVA | 1985 | Off Kharg Island, Gulf of Iran | 70,000 |
| 19 | KATINA P | 1992 | Off Maputo, Mozambique | 67,000 |
| 20 | PRESTIGE | 2002 | Off Galicia, Spain | 63,000 |
| 35 | EXXON VALDEZ | 1989 | Prince William Sound, Alaska, USA | 37,000 |
| 131 | HEBEI SPIRIT | 2007 | South Korea | 11,000 |

### 1.1.3 Issues

As shown statistics above, the collision betweens ships is big part of ship accidents. Once ship collision happened, there are huge impacts, such as the loss of human life, the destruction of the environment and the local community. Especially, the destruction of the environment is done by the oil spill of large tanker. Table 1.4 shows the top 20 major oil spills. For example, the *exxon valdez* oil spill occurred in Alaska, US. About 38,000 to 42,000 $m^2$ were spilled into the Alaska. Until 2010, there still remains an estimated 23,000 US gallons($\approx$ 87,064 liters). Until 2014, National Oceanic and Atmospheric Administration (NOAA) reported that between 16,000 and 21,000 gallons ($\approx$ 60,566 $\sim$ 79,493 liters) of oil still remained

[10]. Thus, to prevent ship accidents including collision between ships is one of the most important issues.

## 1.2 Thesis outline

In Chapter 2, it gives related works on ship collision avoidance, and Chapter 3 provides the outline of distributed ship collision avoidance including the overall framework, basic terminologies, and new cost function to compute collision risk. Chapter 4 gives the details of DLSA to prevent collision in multiple-ship situations. Chapter 5 illustrates the DTSA to remedy DLSA's shortcomings. Chapter 6 gives the motivation and details of DSSA. Chapter 7 concludes with a brief summary. Figure 1.5 shows the composition of the thesis.

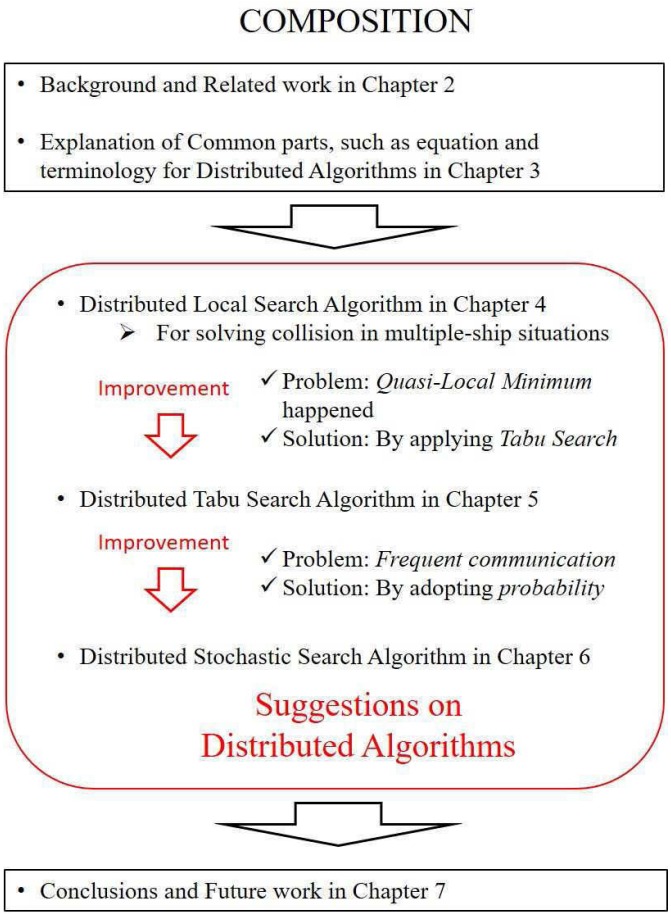



FIGURE 1.5: Composition of the thesis.

# Chapter 2

# Background and Related work

## 2.1 Introduction

In Chapter 2, I introduce a brief summary of what have been suggested and the details for supporting methods for ship collision avoidance.

Szlapczynski suggested a new approach to collision avoidance by evolutionary algorithms [11, 12]. He tried to find optimal sets of safe trajectories in multi-ship encounter situations. In this approach, the fitness function is computed by the sum of the fitness of trajectories. The fitness of each trajectory considers the way loss, target ships and obstacles. He revised the algorithm for application to restricted visibility situations and focused on compliance with COLREGS Rule 19 [13]. A new violation penalty was added for penetration of ship domain, the difference for altering course, distance from target ship, and etc. The methods proposed in these papers had good results. However, he focused on the application of a centralized system, such as VTS. If many ships encounter each other outside of VTS control, it is difficult to apply these methods. Moreover, the number of ships used in experiments was less than five. Lamb and Hunt used Poisson distribution to compute the probability of multiple encounters [14]. The traffic flow is assumed to be uniformly distributed across the lane. The probability for various situations is computed by vessel type, speed, domain radius, and lane traffic flow during crossing situations. Lamb and Hunt later revised their method by adding three options: the relationship between ships, maneuvering angle (which is changeable), and speed reduction [15]. They also took into account not only the first ship, but also the second ship at risk to avoid collision. Although their methods are related

with multiple encounters, their focus is on how a ship finds a safe course in multiple encounters, namely a one-to-many situation. Hornauer et al. proposed a decentralized trajectory optimization algorithm to avoid collision between ships that are partly cooperating with each other [16, 17]. The movement for non-cooperative ships is computed by a Bayesian model using the data from the AIS. The probability of the estimated position for a passive ship that predicts the trajectories by historic probabilistic models is accurately computed. The computed trajectory is reasonable when three ships encounter each other. However, any new explicit algorithm among cooperating ships has not been provided in these papers.

## 2.2 COLREG

There are many methods for preventing ship collisions at sea. From a regulation point of view, the 1972 Convention on the International Regulations for Preventing Collisions at Sea (COLREGs) compels or recommends that ships follow specific regulations [18]. COLREGs, published by the International Maritime Organization (IMO), set out navigation rules, compelling most oceangoing ships to obey COLREG rule such as navigational lights, traffic laws of the waterways, and the buoyage system. COLREGs is composed of the followings:

- **Part A - General**: This part defines the meaning of terms, an applicable object and scope.
- **Part B - Steering and sailing**: This part defines the relationship between ships, how to pass narrow channels or avoid collision.
- **Part C - Lights and shapes**: In this part, to let other ships know a ship's situation, the installation requirements for light and shapes are defined.
- **Part D - Sound and light signals**: To let other ships know a ship's situation, this part defines the sound and light signals, such as warning and distress signals.
- **Part E - Exemption**: This part defines the details of exemption.
- **Annexes**: If any articles are modified, the changes are recorded.

When a ship encounters with other ship at sea, there are three possible relationship of position, such as *head-on*, *crossing*, and *overtaking* situations. By COLREGs, each situation is defined as the followings [18]:

- **Head-on situation**

  - "When two power-driven vessels are meeting on reciprocal or nearly reciprocal courses so as to involve risk of collision each shall alter her course to starboard so that each shall pass on the port side of the other".

  - "When a vessel is in any doubt as to whether such a situation exists she shall assume that it does exist and act accordingly".

- **Crossing situation**

  - "When two power-driven vessels are crossing so as to involve risk of collision, the vessel which has the other on her own starboard side shall keep out of the way and shall, if the circumstances of the case admit, avoid crossing ahead of the other vessel".

- **Overtaking situation**

  - "A vessel shall be deemed to be overtaking when coming up with another vessel from a direction more than 22.5 degrees abaft her beam, that is, in such a position with reference to the vessel she is overtaking, that at night she would be able to see only the stern light of that vessel but neither of her sidelights".

Figure 2.1 shows the relationship and how to avoid collision between ships. According to COLREGs, the ship at port (on the left side) of target ship gives way. It is easy to avoid ship collisions by following COLREGs if there is enough time or few ships involved.

In multiple-ship situations as shown in Figure 2.2, however, it may give a big burden to officers to avoid collisions. Whether COLREGs apply to ship collision avoidance depends on the situation and it requires much navigational experience among a ship's officers.

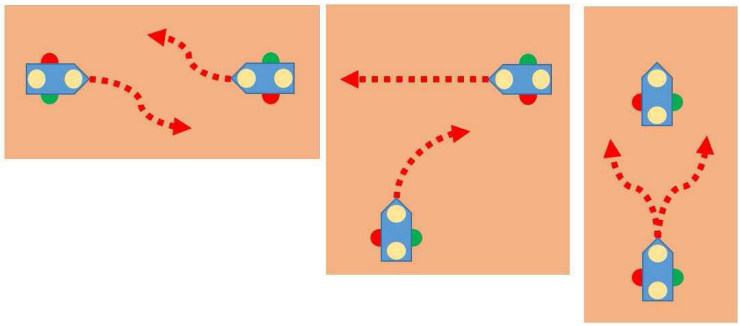


FIGURE 2.1: The relationship between ships.

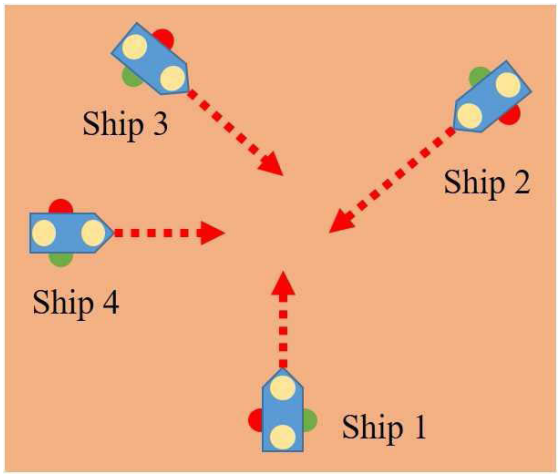



FIGURE 2.2: Multiple-ship situations.

## 2.3 Computational methodology for ship collision avoidance

From a technological point of view, several algorithms are used in ship collision avoidance, such as ship domain [4, 5], fuzzy theory [6], and genetic algorithm (GA) [19].

### 2.3.1 Ship domain

The idea of a ship domain is the important terms for navigation. The ship domain algorithm computes a collision risk depending on whether the ship's safety domain is penetrated. Several ship collision avoidance algorithms are based on the ship domain concept, such as circle, ellipse and octagon [5]. In Fujii's paper, the

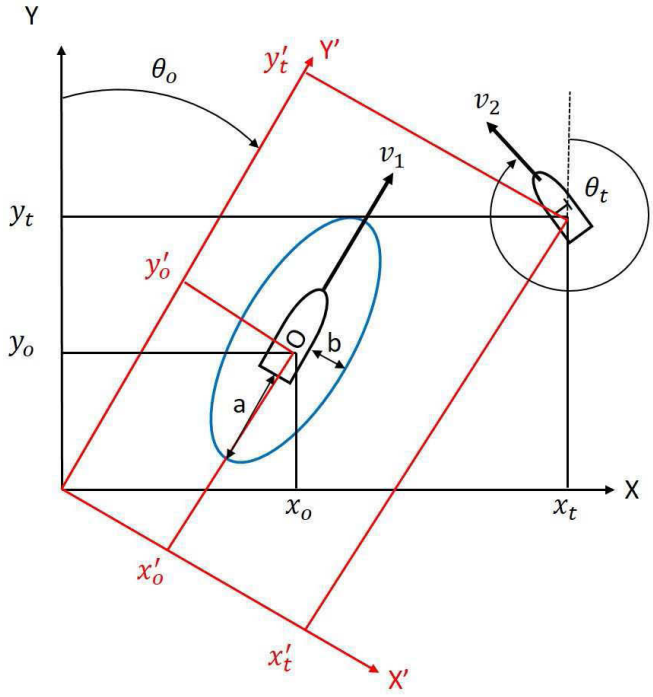



FIGURE 2.3: Coordinate transformation.

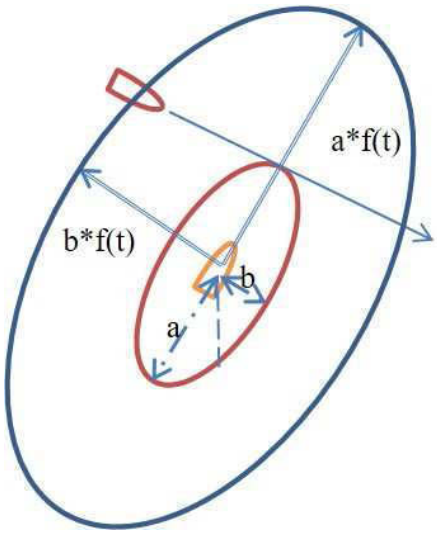



FIGURE 2.4: The change of $f(t)$.

collision risk was defined by $f(t)$ as *temporary approach factor*. Fig. 2.3 shows the coordinates for ellipse-shaped domain.

New coordinates (red line) are as follows:

$$\begin{aligned} x'_o &= x_o \cos\psi_o - y_o \sin\psi_o, \quad y'_o = x_o \sin\psi_o + y_o \cos\psi_o \\ x'_t &= x_t \cos\psi_o - y_t \sin\psi_o, \quad y'_t = x_t \sin\psi_o + y_t \cos\psi_o \end{aligned} \tag{2.1}$$

where $x_o$ and $y_o$ are the x and y coordinates for own ship, respectively. $x_t$ and $y_t$ are the x and y coordinates for target ship, respectively

If two ships are placed as shown in Figure 2.4, the relationship between home and target ship's position can be formulated by an equation of an ellipse.

$$\begin{aligned} \frac{(x'_t - x'_o)}{(af)^2} + \frac{(y'_t - y'_o)}{(bf)^2} &= 1 \\ f^2 = \frac{(x'_t - x'_o)}{a^2} + \frac{(y'_t - y'_o)}{b^2} & \end{aligned} \tag{2.2}$$

The equation 2.2 can be replaced by substituting the equation 2.1.

$$\begin{aligned} f^2 = \frac{((x_t - x_o)\cos\psi_o + ((-y_t + y_o)\sin\psi_o)^2}{a^2} \\ + \quad \frac{((x_t - x_o)\sin\psi_o + ((y_t - y_o)\cos\psi_o)^2}{b^2} \end{aligned} \tag{2.3}$$

$$\begin{aligned} x_o(t) &= x_o(t-1) + V_o \sin\psi_o t, \quad y_o(t) = y_o(t-1) + V_o \cos\psi_o t \\ x_t(t) &= x_t(t-1) + V_t \sin\psi_t t, \quad y_t(t) = y_t(t-1) + V_t \cos\psi_t t \end{aligned} \tag{2.4}$$

New quadratic equation on the time series can be induced by putting the equation 2.4 into the equation 2.3.

$$f^2(t) = At^2 + Bt + C \tag{2.5}$$

And thus:

$$f'(t) = \frac{2At + B}{2\sqrt{At^2 + Bt + C}} \tag{2.6}$$

The equation 2.6 equals to zero for:

$$T_{min} = -\frac{B}{2A} \tag{2.7}$$

where $T_{min}$ is the remaining time until the moment of the smallest ellipse-shaped figure.

$$f_{min} = \sqrt{-\frac{B^2}{4A} + C} \tag{2.8}$$

where $f_{min}$ is a minimum value among $f(t)$ as shown in Figure 2.4.

If $f_{min}$ is greater than one, no other ship is located within its safety domain. If it equals one, target ship will be located on the border of the ship domain. If $f_{min}$ is less than one, the ship domain has been penetrated by target ship, and if this happens, the ship must alter its course to keep other ships out of her ship domain.

### 2.3.2 Fuzzy theory

The fuzzy theory computes the membership function for a collision risk [20]. To compute collision risk, several parameters - Variation of Compass Degree (VCD), Time of Closest Point of Approach (TCPA), and Distance to Closest Point of Approach (DCPA) - are used. Figures 2.5 and 2.6 show the collision risk depending on the change of the relative bearing. If the change of the relative bearing has constant bearing, the target ship or an object are getting closer but maintaing the same relative bearing. If it continues, the collision will be happened. The VCD is a parameter whether to check the change of the relative bearing. If the VCD equals zero, there are collision risks on the current course. Otherwise, ships can pass each other safely.

$$VCD_i = |Bearing_i - Bearing_{i-1}| \tag{2.9}$$

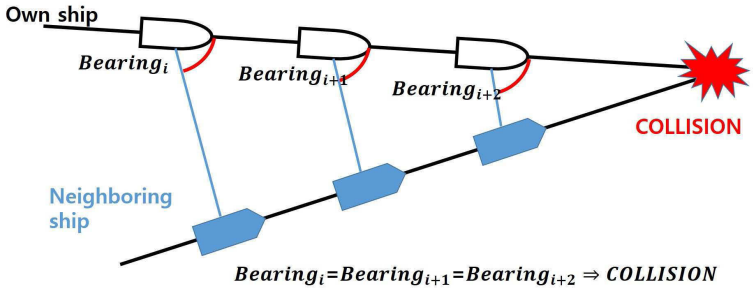



FIGURE 2.5: The change of the relative bearing I.

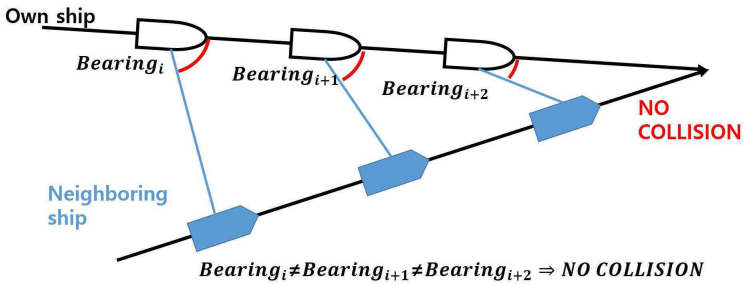



FIGURE 2.6: The change of the relative bearing II.

### 2.3.3 Genetic algorithm

The genetic algorithm (GA) is based on the principle of evolution, that is, survival of the fittest. Tsou and Hsueh used GA to find the safest and shortest path that also complied with COLREGs[19]. The fitness function is defined as the distance from the turning point to the original route. Then, they used GA to get the shortest collision avoidance route. Equation 2.10 is the fitness function.

$$Distance = \min_{i=1}^{n}\{D_{r_i} + D_{s_i}\} \tag{2.10}$$

where $D_{r_i}$ is the distance of navigational restoration, $D_{s_i}$ is the distance after collision avoidance.

$$D_{s_i} = f^3(X_1)V_o, \quad D_{r_i} = f^3(X_2)V_o \tag{2.11}$$

$\min f^3(X_1)$ is the navigation time after collision avoidance. $\min f^3(X_2)$ is the navigation time of navigational restoration.

As chromosome constitution, there are four parameters - *avoidance time*, *turning angle*, *restoration time* and *limited angle*. They found optimum routes under three situations in which a ship can encounter a target ship. Considering that the real-time collision avoidance, they limited the number of the population, the crossover and the mutation rate.

# Chapter 3

# Distributed Collision Avoidance

In Chapter 3, I introduce common parts of the distributed algorithms. First, I explain the framework how to work the distributed algorithms and the terminology. I also suggest new cost function.

## 3.1 Framework and terminology

### 3.1.1 Framework

Distributed ship collision avoidance is made up of two procedures: control and search. A framework of these procedures is given in Figure 3.1. When a ship arrives at her destination, this procedure is terminated. For the control procedure, the ship decides whether to proceed to the next position. If the ship does not have any neighboring ship within a certain area, namely *detection range*, and also has not yet arrived at her destination, she moves to the next position. For the search procedure, a ship tries to avoid collision by running a distributed algorithm when she confirms that there is a collision risk. If every ship finds a solution, or if the computational time exceeds a certain time limit, they move to the next positions. A time limit on the computational time is set for all ships exchange messages with each other to figure out safe courses. When the time has elapsed, all ships move to the next positions to check whether a collision happened on the spot. The ships alternate the search and control procedures until they arrive at their destinations.

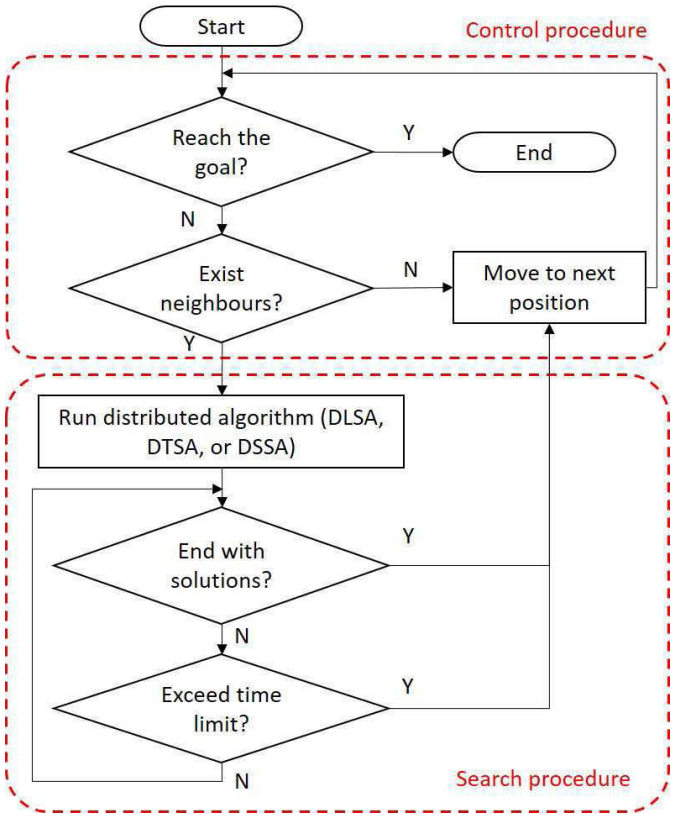



FIGURE 3.1: Framework.

### 3.1.2 Terminology

Figure 3.2 illustrates these basic terms. The home ship located at the center has a *detection range* to detect neighboring ships. The home ship can exchange messages with the neighboring ships, but not with the ships located outside the detection range. The home ship tries to keep a safety domain between herself and the neighboring ship. If that safety domain is penetrated, it is considered they collide with each other. The meaning of the terms is as followings:

- **Home ship**: A ship that focus on.
- **T**: The maximum length of time (in minutes) for which the home ship plans her future positions.
- **Detection range**: The area in which the home ship can communicate with other ships.

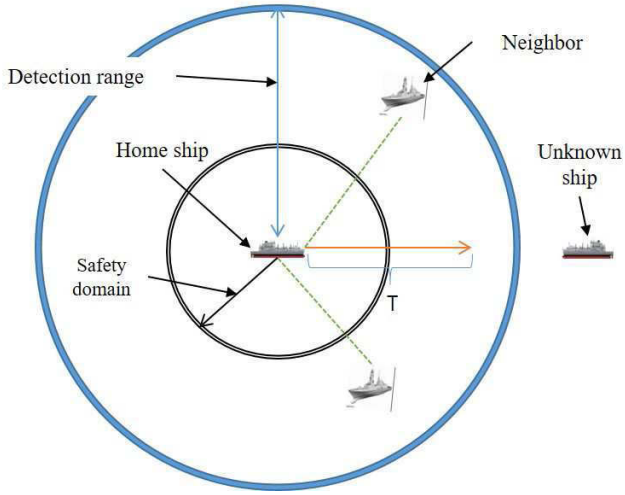



FIGURE 3.2: Basic terms.

- **Neighbor**: A ship located within the detection range. The home ship can exchange messages only with her neighbors.
- **Safety domain**: The area that the home ship prohibits a neighboring ship from penetrating.
- **Ok? message**: It includes information of position.
- **Improvement message**: It includes the number how much the cost is reduced.
- **ID**: Identification is given at initial state. It is used when improvement is same with neighboring ship's one. A ship with higher priority ID has the right to choose next course.
- **Candidate course**: Considering the maneuvering ability of ships, the altering course is restricted as shown in Figure 3.3. It has to be considered with the characteristics of ship, such as speed, tactical circle and the traffic condition.

Due to the characteristics of ship, a ship has a restriction to change maximum course as shown in Figure 3.3. The figures show candidate courses for a ship. For example, when maximum changeable course is 10 degrees, a ship can choose one of them, such as starboard 10 degrees, forward, and port 10 degrees. Figure 3.4 shows the change of neighboring ships depending on the *detection range* of home

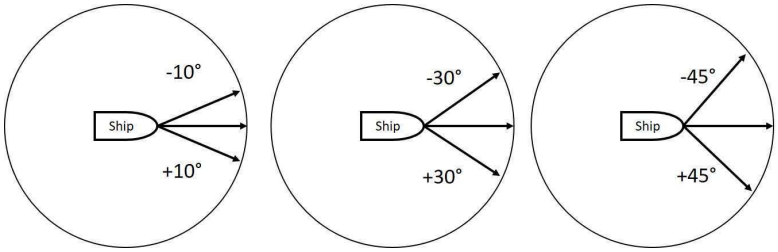



FIGURE 3.3: Limit of maneuvering course.

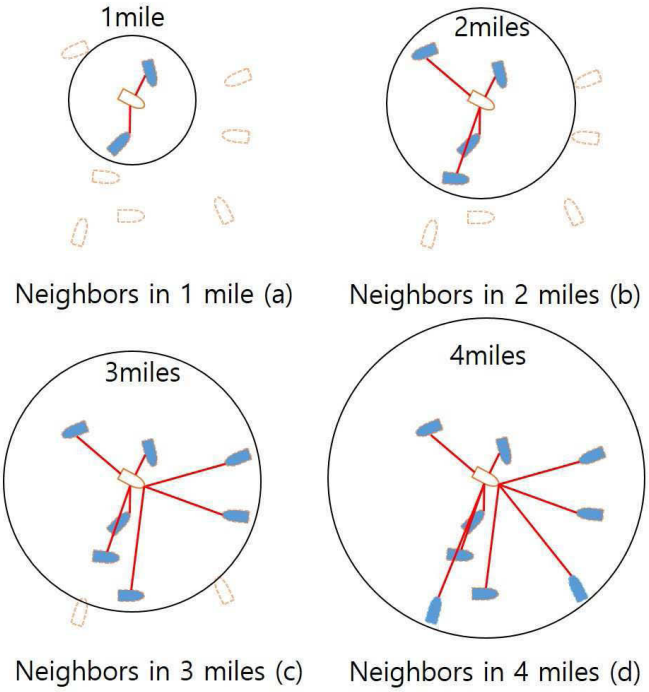



FIGURE 3.4: The change of neighboring ships depending on the *detection range*.

ship at center. The larger the size of the *detection range*, the more the number of neighboring ships.

Figure 3.5 shows multiple-ship situations. Each ship has her own local view called *detection range* (red circles). The home ship can exchange messages with a neighboring ship, but not with a ship located outside the *detection range*. Figure 3.6 shows how to proceed to destination. To arrive destination, home ship exchanges messages with neighboring ships until she finds a safe course. And then home ship proceeds to next position. Home ship repeats this process until she arrives at the destination.

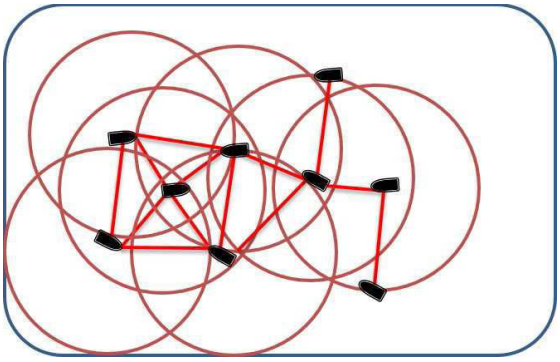


FIGURE 3.5: Multiple-ship situations.

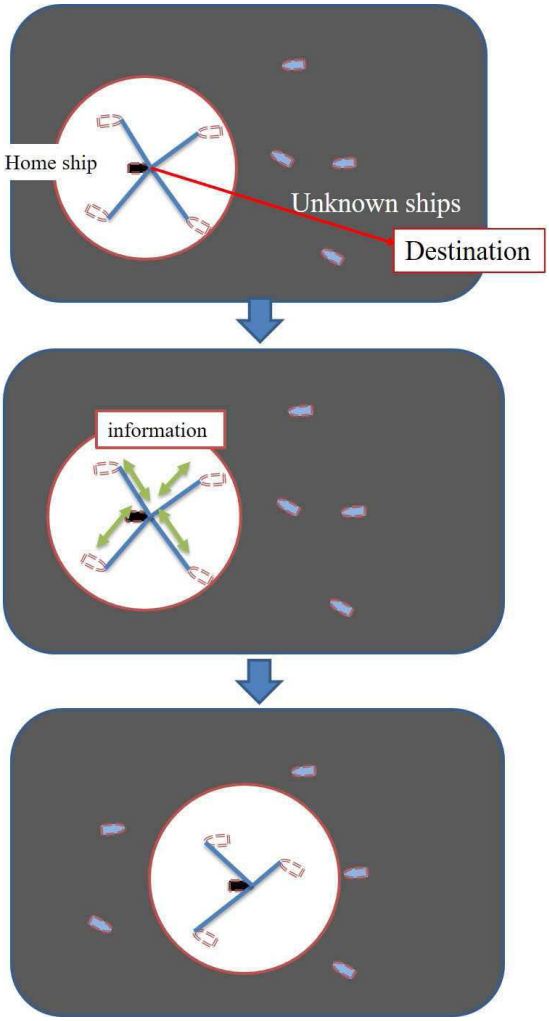



FIGURE 3.6: How to proceed to destination.

Suppose, for example, that a ship sails the ocean at 12 nautical miles per hour. Due to the restriction on ship movement, a sailing ship cannot change her course abruptly. Once she selects a course, she must follow it for a certain period of time. The ship is required to consider changing her course every three minutes (every 0.6 nautical miles). The ship plans her future positions in 15 minutes on the basis of current positions, headings, and speeds of herself and her neighbors. Note that this is done every three minutes through communication with neighboring ships. The *Ok?* message includes the information for position. The *improvement* message includes the number how much the cost is reduced. The ship exchanges both messages with neighboring ships. When $T$ gets larger, a ship becomes more proactive.

## 3.2 Cost and improvement

For a candidate course, two things have to be considered, i.e. *collision risk* against a neighboring ship and *relative angle* between a candidate course and destination as shown in Figure 3.7. I propose a cost function considering both of them. The cost function is used for all distributed algorithms that will be shown in the Chapter. 4, 5, 6.

Given current positions, headings, and speeds of neighboring ships, a ship computes the cost for each candidate course. A candidate course is chosen from a

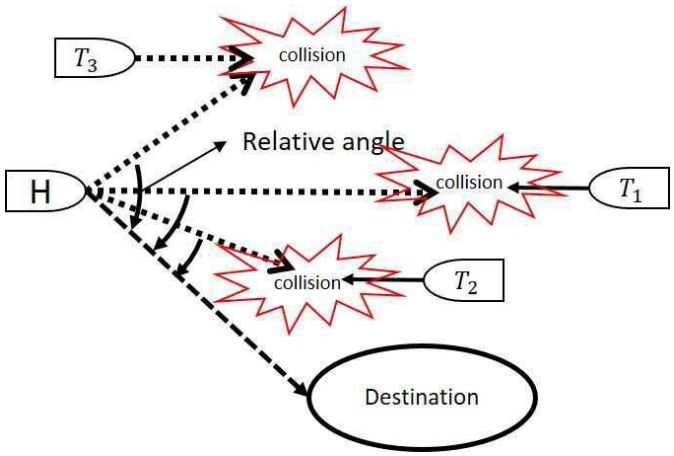



FIGURE 3.7: Collision risk and relative angle

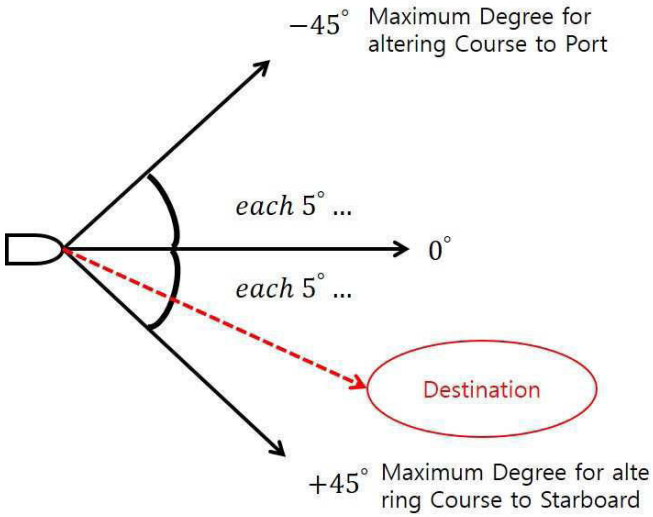



FIGURE 3.8: Candidate courses and the angle heading for a destination.

discrete set of angles as shown in Figure 3.8. In consideration of typical ship maneuvering, it ranges from 45 degree on the port side (-45 degrees) to 45 degrees on the starboard side (+45 degrees) in step of 5 degrees. If the angle heading for a destination exists in these bounds, it is also included as a candidate course.

Equation 3.1 shows the *collision risk*, where *crs* and $j$ mean a candidate course and a neighboring ship, respectively, and *self* means the home ship. It *self* will collide with ship $j$ in $T$ when choosing a course *crs*, and $CR_{self}$ for *crs* and $j$ is computed as $T$ divided by TCPA. It becomes zero, otherwise. Equation 3.2 computes the $COST$ for a course *crs*, which is made up of two parts: first, the sum of $CR_{self}$ over the neighboring ships at risk for *crs*, and second, the relative angle between *crs* and a destination. The $\alpha$ is a weight factor that controls the relationship between the safety and efficiency. In equation 3.2, the front part, $CR_{self}$, is for the safety against target ships, and the rest part is for the efficiency, considering the destination. If $\alpha$ gets larger, a ship places more emphasis on safety than efficiency. On the other hand, the ship goes long way round. For all distributed algorithms, I set the value of $\alpha$ to one. During the search, a ship tentatively selects one course as her *next-intended course* that causes some cost computed by equation 3.2. However, she may be able to reduce the cost by changing it to another course. Equation 3.3 computes the largest reduction in costs as $improvement_{self}$. A ship always tries to select the course that gives the largest reduction in costs. A ship is always aware of absolute angles $\theta_{heading}$ and $\theta_{dest}$ for her heading and destination,

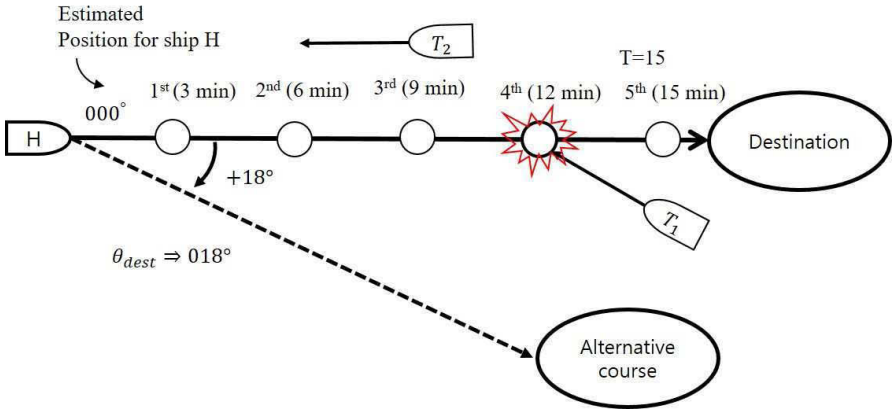



FIGURE 3.9: Numerical example for computing costs and improvements.

respectively. As shown in Equation 3.4, a course for the destination is computed by $\theta_{dest}$-$\theta_{heading}$ to be added into a set of candidate courses only if the course is within the bounds on alterable angles.

Figure 3.9 is an example of how to compute $COST_{self}$ and $improvement_{self}$ for the home ship (H). Ships $T_1$ and $T_2$ are neighboring ships of the home ship. Ship $T_2$ is contented with the current course. However, the home ship will collide with ship $T_1$ after 12 minutes (four time steps) later with her current course. The cost for 000° is computed by COST(000°) = 5/4 + 18°/180° = 1.35, while the cost for 045° is COST(045°) = 0 + 27°/180° = 0.15, and the cost for 018° (the course for the destination) is COST(018°) = 0 + 0°/180° = 0. The $improvement_{self}$ for the home ship is thus computed by $improvement_{self} = \max_{crs}\{\text{COST}(000°) - \text{COST}(crs)\} =$ 1.35, since the cost for 018° is clearly minimum among the candidate courses. Ship $T_2$ is ruled out for computing the cost because ship $T_2$ has nothing to do with any collision.

$$CR_{self}(crs, j) \equiv \begin{cases} \frac{T}{TCPA_{self}(crs,j)}, & \text{if } self \text{ will collide with ship } j \\ 0, & \text{otherwise} \end{cases} \tag{3.1}$$

$$COST_{self}(crs) \equiv \alpha \sum_{j \in Neighbors} CR_{self}(crs, j) + \frac{\theta_{dest} - \theta_{self}(crs)}{180°} \tag{3.2}$$

$$improvement_{self} \equiv \max_{crs}\{COST_{self}(next\_intended\_course) - COST_{self}(crs)\} \tag{3.3}$$

$$\theta_{dest} \equiv \begin{cases} \theta_{dest} - \theta_{heading}, & \text{if } |\theta_{dest} - \theta_{heading}| < 45^\circ \\ \text{empty}, & \text{otherwise} \end{cases} \tag{3.4}$$

where $crs \in \{-45^\circ, -40^\circ, \cdots, -5^\circ, 0^\circ, +5^\circ, \cdots, +40^\circ, +45^\circ\} \cup \{\theta_{dest}\}$

# Chapter 4

# Distributed Local Search Algorithm

## 4.1 Introduction

One of the many methods for preventing ship collisions at sea from a regulations point of view is the COLREGs. From a technological point of view, several algorithms are used in ship collision avoidance, including those of ship domain, fuzzy theory, ant colony algorithm and neural networks. These algorithms use the distance to the DCPA and TCPA as parameters. The ship domain algorithm calculates collision risk well, depending on whether the ship safety area is penetrated. Fuzzy theory is said to enable machines to "reason flexible" the as humans beings do [20, 21] and was well applicable in ship collision avoidance. These algorithms define collision risk mostly in one-to-one ship situations, however, and are computed by using only local information without online communication among ships. In this sense, it is said that most precedent algorithms are suited to a centralized system in which every computation is done by a server without online communication among participants, i.e., ship personnel. To deal with multiple-ship situations, which are more complex, however, algorithms must be suited to distributed systems that enable individual ships to exchange information online with neighboring ships. Indeed, exchanging such information online is important, but one further important consideration for individual ships is to know the intentions of other ships because individual ships otherwise cannot respond promptly to actions by other ships. In other words, a ship typically needs ten to fifteen times its length

for stopping to avoid a collision. A ship 200 meters long, for example, needs two to three kilometers for stopping. Therefore I consider it important to enable individual ships to exchange information on both facts and intentions to avoid collisions in multiple-ship situations. In Chapter 4, I assume that individual ships exchange intentions with neighboring ships using an AIS. The AIS is a machine used on board to displays information such as the speed, bearing, and position of neighboring ships in real time. Based on AIS, I introduce a Distributed Local Search Algorithm (DLSA) for solving the ship collision problem effectively in multiple-ship situations by taking into account the intentions of neighboring ships[22]. A distributed local search involves individual agents satisfying constraints by exchanging information with other agents. This pure distributed algorithm does not require a server, which could be a bottleneck in the system. One objective of the work is to reduce the ship collision risk in multiple-ship situations by using a DLS. It is difficult to quantitatively compare the DLSA with precedent ship collision avoidance algorithms because, as mentioned above, precedent ones are designed assuming one-to-one ship situations. Of course, precedent ones may be applicable to multiple-ship situations, in principle, by reducing this situation to a set of one-to-ones. I, however, consider that such a method would end up in immediate failure, due to its higher computational complexity. In contrast, the DLSA deals naturally with multiple-ship situations and make the system robust by enabling individual ships to react to dynamic changes in the environment.

In Chapter 4, I explain the background of my work. Furthermore, I show how DLSA is applied to ship collision avoidance, explaining variables, procedures for the proposed algorithm, and experimental results.

## 4.2 Local Search

Local search is a metaheuristic method for solving optimization problems and incompletely satisfiability algorithms. It may find the solution to a problem or fail even if the problem is satisfiable. Local search is applied to many problems, e.g., the traveling salesman problem or the nurse scheduling problem. The local search is a centralized system in which every computation is done at a central location or computer. It is easy to design a whole system if a server knows all information for all agents. The LS is a iterative improvement algorithm that keeps a 'current'

state and tries to improve it [23, 24]. This process is repeated until no further improvement solution can be found. Algorithm 4.1 shows the pseudocode for LS.

**Algorithm 4.1** Local Search

```
1: Set x_0 as initial state
2: while do not satisfy or reach maximum iteration do
3:     for each node N do
4:         Evaluate the neighbors of N
5:         Select one of the neighbors of x_{i+1}
6:         Move to x_{i+1}
7:     end for
8: end while
```

There are typical algorithms as the followings:

- **Hill-climbing**: It is like that climbing mountains in thick fog with amnesia. It tries to find a solution to maximize or minimize a target function $f(x)$.

  - Simple hill-climbing: It chooses the closest move as initial step. If there are any changes that improve $f(x)$, it chooses next move.

  - Stochastic hill-climbing: It does not choose next move, deterministically. It chooses next move by probability $p$, or keeps current move probability *1-p*.

- **Simulated annealing**: It is a probabilistic method to find out the approximating global optimum. It accepts the worse solutions for expanding the search space. Therefore, it is no need to consider local-minimum that happens in hill-climbing.

- **Genetic algorithm**: It is a heuristic search algorithm by imitating natural evolution. The main idea came from Charles Darwin of "survival of the fittest". Each generation consists of a population of strings like DNA.

Algorithm 4.2 shows the pseudocode for simulated annealing [25]. Algorithm 4.3 shows the genetic algorithm [26].

If, for some reason, e.g. a server is broken, it is not possible to maintain a system. Compared to a local search, the DLSA searches locally for an approximation solution by different agents. The DLSA does not have a server and need not use a computer. This means that individual agents may solve a certain problem by satisfying constraints. This is why it is flexible in a system failure and adds less load in computation.

**Algorithm 4.2** Simulated annealing

Set initial solution $S_{current}$ as $S_{best}$
**for** $i$=1:iteration **do**
    $S_i \leftarrow S_{neighbor}$
    $temperature_{current} \leftarrow$ ComputeTemperature($i$, $MaxTemp$)
    $\Delta \leftarrow Cost(S_{current})$-$Cost(S_i)$
    **if** $\Delta < 0$ **then**
        $S_{current} \leftarrow S_i$
        **if** $Cost(S_i) \leq Cost(S_{best})$ **then**
            $S_{best} \leftarrow S_i$
        **end if**
    **else if** exp $\frac{Cost(S_{current})-Cost(S_i)}{temp_{current}} >$ rand(0,1) **then**
        $S_{current} \leftarrow S_i$
    **end if**
**end for**
return $S_{best}$

**Algorithm 4.3** Genetic algorithm

Set $population_{current} \leftarrow population_0$
ComputePopulation($population_{current}$)
$S_{best} \leftarrow BestSolution(population_{current})$
**while** do not satisfy stop condition **do**
    $Parents \leftarrow$ SelectionParent($Population$, $Population_{size}$)
    $Childere \leftarrow \varnothing$
    **for** $Parent_1, Parent_2 \in Parents$ **do**
        $Child_1, Child_2 \leftarrow Crossover(Parent_1, Parent_2, P_{crossover})$
        $Children \leftarrow Mutation(Child_1, P_{mutation})$
        $Children \leftarrow Mutation(Child_2, P_{mutation})$
    **end for**
    ComputePopulation($Children$)
    $S_{best} \leftarrow BestSolution(Children)$
    Set $population_{current} \leftarrow$ Change($population_{current}, Children$)
**end while**
return $S_{best}$

## 4.3 Distributed Local Search Algorithm for Ship Collision Avoidance

A constraint satisfaction problem (CSP) is a mathematical problem defined as a set of objects that are consistent with the assignment of values to variables. A CSP is composed of $n$ variables $x_1, \ldots, x_n$, and a set of constraints. And each variable can choose a value from finite and discrete domains $D_1, \ldots, D_n$. It needs to a value for each of the variables that satisfies all constraints. A typical example

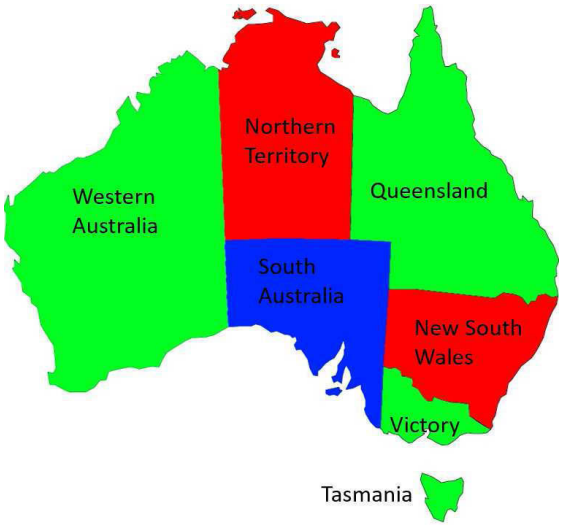



FIGURE 4.1: CSP example of color mapping

of CSP is color mapping as shown in Fig. 4.1. Each territory in Australia is defined by variables $WA$, $NT$, $Q$, $SA$, $V$, $NSW$, and $T$. The objective is to color a map that is divided by seven places so that the adjacency of each other does not have same color. Each variable chooses red, green and blue from its domain. For other example, there is the eight queens puzzle, in which eight chess queens are placed on an 8 X 8 chessboard so that no more than two queens may attack each other. The domain is the 64 positions on the chessboard where each queen can choose one. The variables are eight queens, $x_1, x_2, \ldots x_8$. The constraint is that no queen may attack another queen. It can organize the constraints and know that two queens do not place in the same column. Each queen can be represented as $x_1 = 3$, $x_2 = 6$, $x_3 = 4$, $x_4 = 2$, $x_5 = 8$, $x_6 = 5$, $x_7 = 7$, and $x_8 = 1$. $x_i$ cannot attack $x_j$ $(i \neq j)$. Along a diagonal, all queens have to satisfy $|i - j| \neq |x_i - x_j|$. In this case, the number of representation equals to $8^8(16,777,216)$. There are 93 solutions that satisfies all constraints.

The DLSA, is a distributed constraint satisfaction problem (DisCSP), used for solving a problem or satisfying all constraints by multiple agents [27–30]. The DisCSP is consists of a set of agents, $1, 2, \ldots, k$ and a set of CSPs, $P_1, P_2, \ldots, P_k$.

Figure 4.3 is the flowchart for DLSA. First, a ship sets current course as *next-intended course*. Each ship exchanges information to compute $COST_{self}$ and $improvement_{self}$. If every ship is contented with *next-intended course*, then this

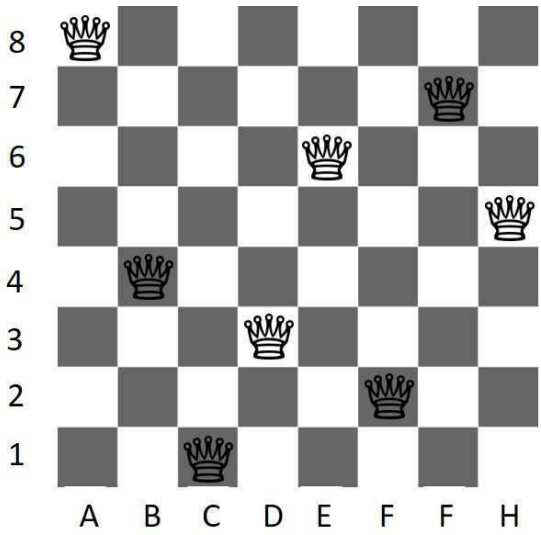



FIGURE 4.2: CSP example of chess

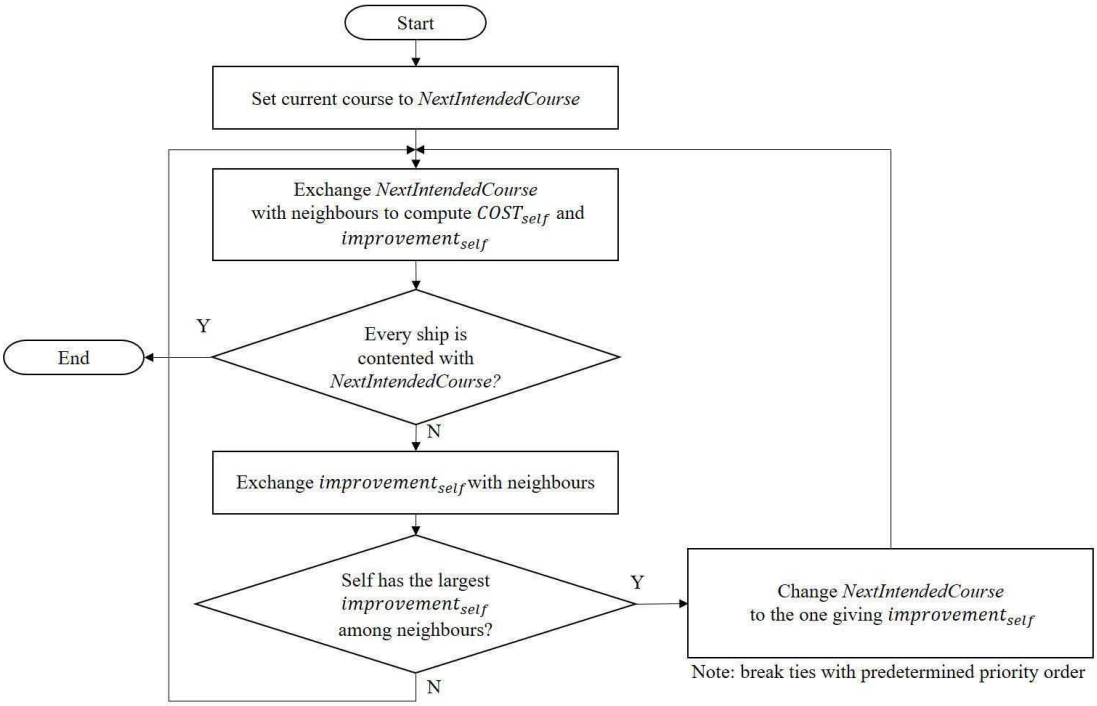



FIGURE 4.3: Flowchart of Distributed Local Search Algorithm.

process ends. Otherwise, they exchange an $improvement_{self}$ message with neighboring ships. A ship which has the largest value of improvement chooses new *next-intended course*.

Figure 4.4 shows the process of the DLSA. This describes how to exchange messages and decides highest priority ship among ships. Assume three ships are

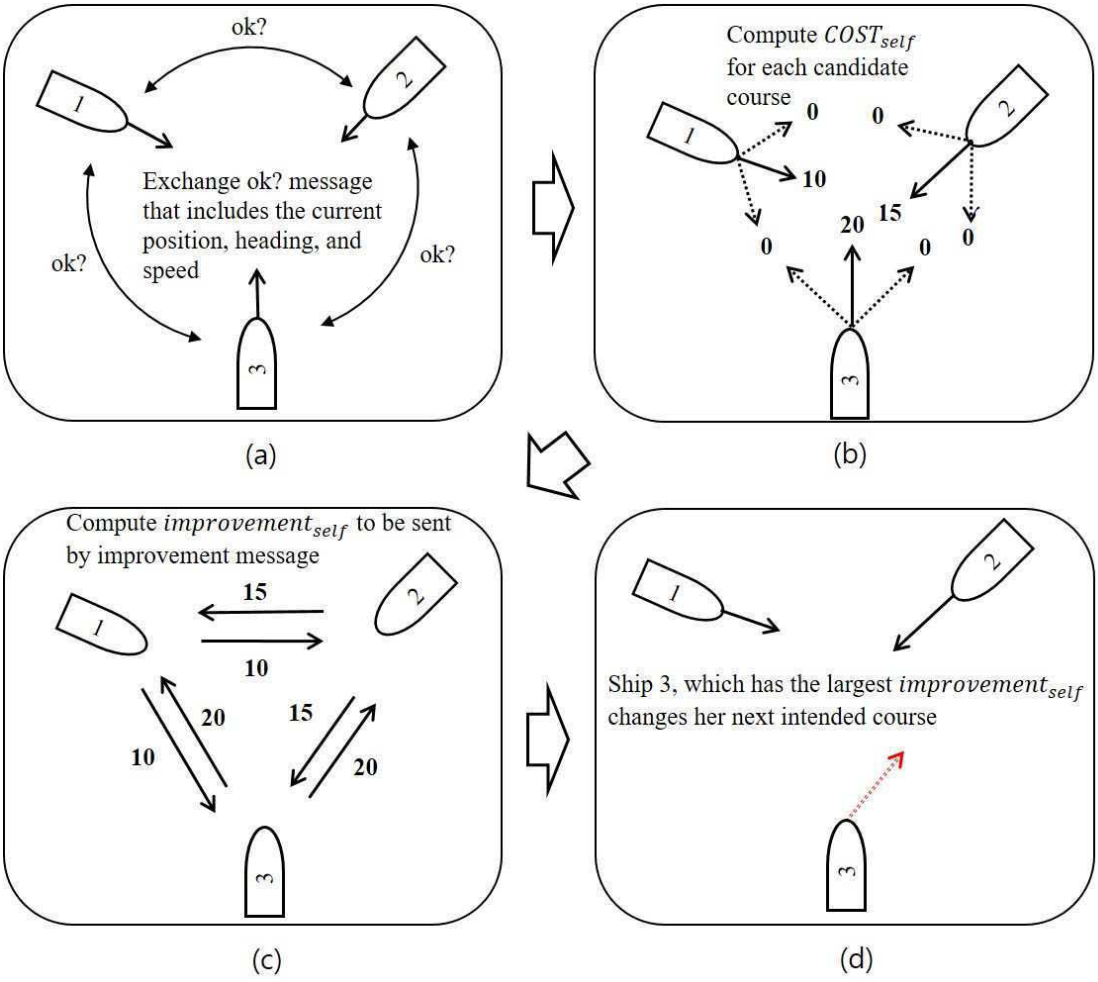



FIGURE 4.4: The process of Distributed Local Search Algorithm.

encountered as shown in Fig. 4.4(a). A ship checks her position for whether she has arrived at a destination. The process will be repeated until a ship arrives at her destination. A ship searches a vicinity whether target ships exist. If so, target ships are added in the neighboring list. Each ship exchanges *ok?* message with target ships. They compute the cost for each candidate course as shown in Fig. 4.4(b). If there are collision risks between ships, they exchange $improvement_{self}$ message as shown in Fig. 4.4(c). Each ship compares the value of improvement of herself and target ships. A ship that has the biggest improvement has a right to choose *next-intended course*. Otherwise, she keeps the current course as shown in Fig. 4.4(d). If more than two ships have same the value of improvement, the ties are broken by the ID of ships that is given randomly in an initial situation. This process continues until the collision risks disappear. If collision risk has disappeared, a ship proceeds to the next position and checks whether its position is

the destination.

Algorithm 4.4 is the pseudocode for DLSA. A ship checks whether she has arrived at a destination (step 5). If she has not arrived at a destination, she starts to search a vicinity (step 10). If there are neighboring ships in *detection range*, she adds neighboring ships to neighboring ship's list (step 11). She sends *ok?* messages to the ships that registered in neighboring ship's list and receives *ok?* messages from the neighboring ships (step 17-18). Based on the *ok?* message, she computes *cost* and *improvement* (step 20). She sends her *improvement* message to the neighboring ships and receives *improvement* messages from the neighboring ships (step 22-23). From step 25 to 33, the process chooses a ship which has the biggest improvement. And the ship can select *next-intended course*. If there are more than two ships that have the same *improvement*, the tie will be broken by the ID (step 30-32). After deciding *next-intended course*, a ship $i$ proceeds to the next position (step 35). She checks whether the current position is the same as the destination. If she has not arrived at the destination, she starts to search a vicinity (step 40). This process repeats until a ship $i$ arrives at the destination.

**Algorithm 4.4** Distributed Local Search Algorithm for Ship Collision Avoidance

```
1: improve_i ≡ maximum improvement of i
2: neighs_i ≡ neighbors of i
3: estPsn_i ≡ estimated position of i
4: ID_i ≡ identification of i
5: if i arrives at destination then
6:     arrived_i=TRUE
7: else
8:     arrived_i=FALSE
9: end if
10: Search a vicinity with Detection range
11: if neighs_i exist then
12:     Add neighs_i to NeighsList_i
13: end if
14: while arrived_i = FALSE do
15:     while computation time < limited time do
16:         for neighs_i in neighsList_i do
17:             send ok?(estPsn_i) to neighs_i
18:             add ok?(estPsn_j) to ShipView_i
19:         end for
20:         compute cost_i and improve_i
21:         for neighs_i in neighsList_i do
22:             send improve_i to neighs_i
23:             add improve_j to ShipImprove_i
24:         end for
25:         if improve_i < max(ShipImprove_i) then
26:             i keeps current next_intended_course
27:         else if improve_i > max(ShipImprove_i) then
28:             i chooses new next_intended_course
29:         else
30:             if ID_i > ID_j of NeighsList_i then
31:                 i chooses new next_intended_course
32:             end if
33:         end if
34:     end while
35:     i proceeds to next position
36:     if i arrives at destination then
37:         arrived_i=TRUE
38:     else
39:         NeighsList_i = empty
40:         search a vicinity with detection range
41:         if neighs_i exist then
42:             add neighs_i to NeighsList_i
43:         end if
44:     end if
45: end while
```

## 4.4 Experiments

The experiments are done by five ships and four variables - *Candidate course*, *Detection range*, *Safety domain* and *Timestep* - to determine how much each variable affects this algorithm. In one situation, all variables are used by changing their values from each domain. There are 120 situations in which ships do not have the same destination as shown in Figure 4.5.

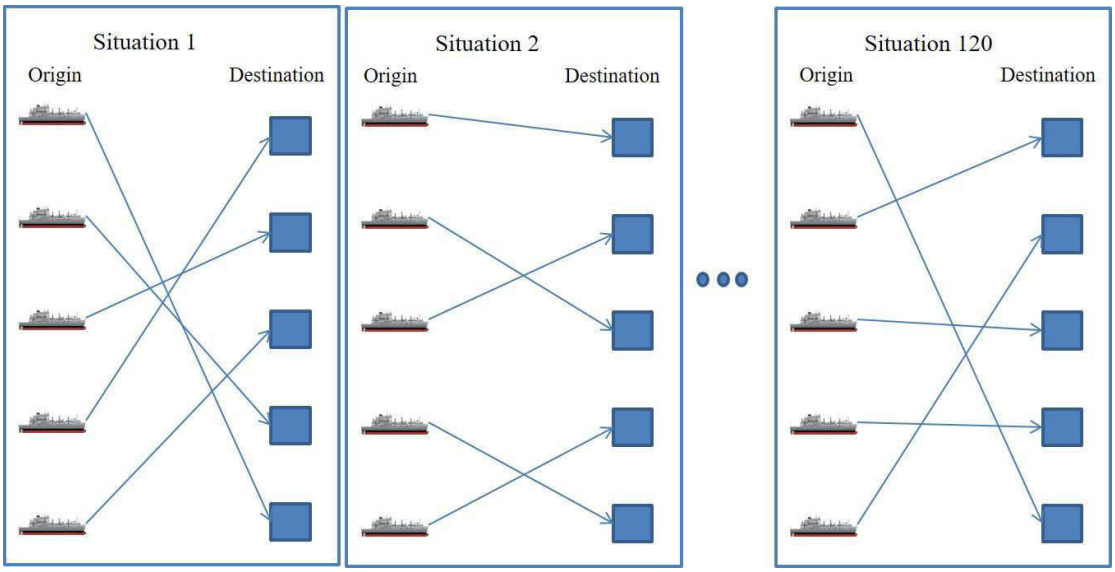



FIGURE 4.5: Situations involving ships having different destinations.

TABLE 4.1: Variable

| Variable | Value |
|---|---|
| Candidate course | {−10°, 0°,+10°}, {−30°, 0°,+30°}, {−45°, 0°,+45°}, {−45°, −30°,−10°, 0°, +10°, +30°, +45°} |
| Detection range | {10}, {20} nautical miles |
| Safety domain | {0.5}, {1} nautical miles |
| Timestep | {5}, {10} |
| Speed | {12} knots |

Experiments are done 3,840 times totally. The number of experiments are a combination of *candidate course* (four kinds), *detection range* (two kinds), *safety domain* (two kinds), *timestep* (two kinds), and 120 kinds of situations (120 kinds). Table 4.1 shows the values for each variable. During one round, individual ships exchange *ok?* and *improvement* messages with neighboring ships. The *ok?* message, which has ship position information, is sent to neighboring ships. Individual

Table 4.2: Result of Experiment

| Total Experiment | 3,840 times | |
|---|---|---|
| Result | Success | 3,720 times (96.875%) |
| | Failure | 120 times (3.125%) |

ships use *ok?* messages to get the maximum reduction collision risk by changing course. The maximum reduction cost is sent again to neighboring ships, and the ship with the highest maximum reduction cost has the right to choose its next possible course. This process is considered to be one round. Until subsequent courses, which have no collision risk, are chosen, this round is repeated.

All experiments are done by Matlab R2013b and a PC (Core i7-4790K, 4 cores, 8 threads, 16 GB memory and Windows 10 Professional).

In 3,840 runs (combinations of different variables, *Candidate course*, *Detection range*, *Safety domain* and *Timestep*) of the experiment, it succeeded 3,720 times (96.875%) and failed 120 times (3.225%).

To evaluate the performance of DLSA, the followings are computed:

- **Success ratio**: The meaning of success is that all ships arrive at their destination without collision. The success ratio is the ratio between the number of successful results and total number of experiments.
- **Average exchanged messages**: The number of total exchanged messages, e.g. *ok?* and *improvement* messages is divided by the total number of experiments.
- **Average distance**: The sailing distance is a ship's route from origin to destination. The average distance is the figure that the sum of the sailing distance for all experiments divided by the number of situations that result in success.
- **Cost**: It is the cost for the course that is chosen by a ship for proceeding to next position.

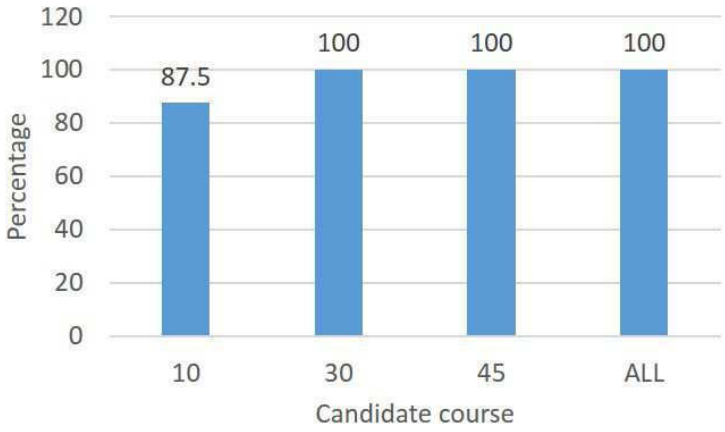



FIGURE 4.6: Success ratio by Candidate course.

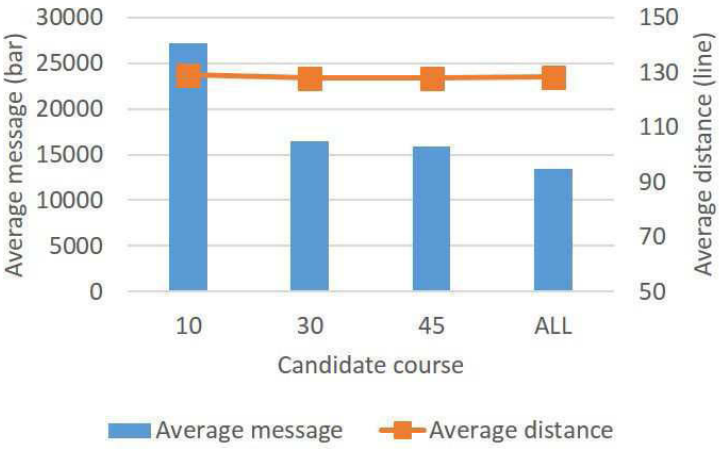



FIGURE 4.7: Average message and distance.

### 4.4.1 Classification by Candidate course

I categorized the success percentage, average message, average distance, and cost according to *candidate course*. Figure 4.6 shows the success percentage according to *candidate course*. The bar indicates the success percentage. The number below the figures represents the candidate courses. The 10 denotes $\{-10°, 0°, +10°\}$. ALL denotes $\{-45°, -30°, -10°, 0°, +10°, +30°, +45°\}$. As the value of domain of candidate course increased, the success percentage also increased. In the case of 10, the success percentage recorded lowest as 87.5%. Compared than other candidate courses, because, in the case of 10, the range of alternatives is narrow.

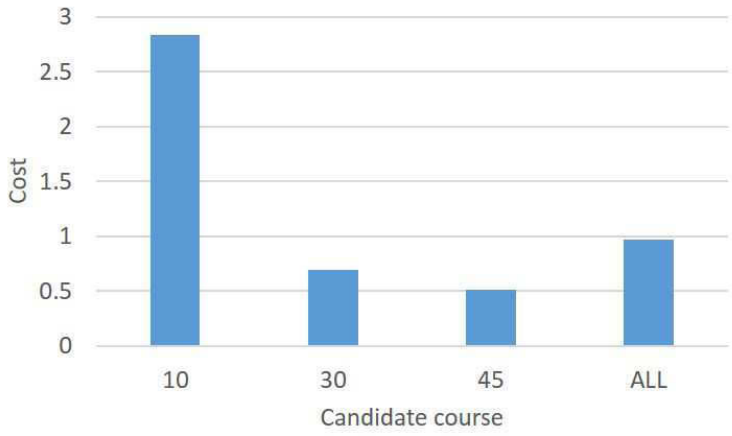



FIGURE 4.8: Average cost.

Figure 4.7 shows the average message and distance according to *candidate course*. The bar and line mean the average message and distance, respectively. As the value of domain of candidate course increased, the average message decreased. The lowest average message showed at ALL candidate course. In terms of average distance, all candidate courses showed a similar result.

Figure 4.8 shows the cost. In the case of 10, it recorded highest cost. In the case of 45, it recorded lowest cost.

### 4.4.2 Classification by Detection range

I categorized the success percentage, average message, average distance, and cost according to *detection range* Figure 4.9 shows the success percentage depending on *detection range*. The numbers below the figure, such as 10 and 20, indicate the *detection range* in nautical miles. The results for the success percentage are same as 96.875% (1,860 times).

Figure 4.10 shows the average message and distance. As the value of domain of detection range increased, the average message also increased. If the detection range enlarges, the search space expands. Therefore, a ship can exchange message with more neighboring ships. The average distance showed similar result regardless of the *detection range*.

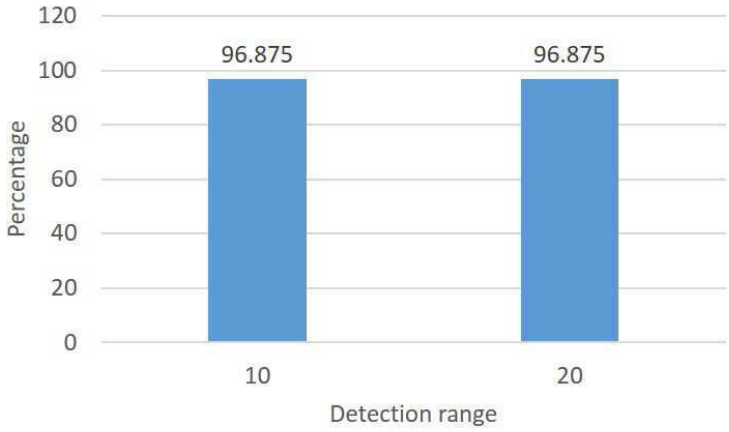



FIGURE 4.9: Success ratio by Detection range.

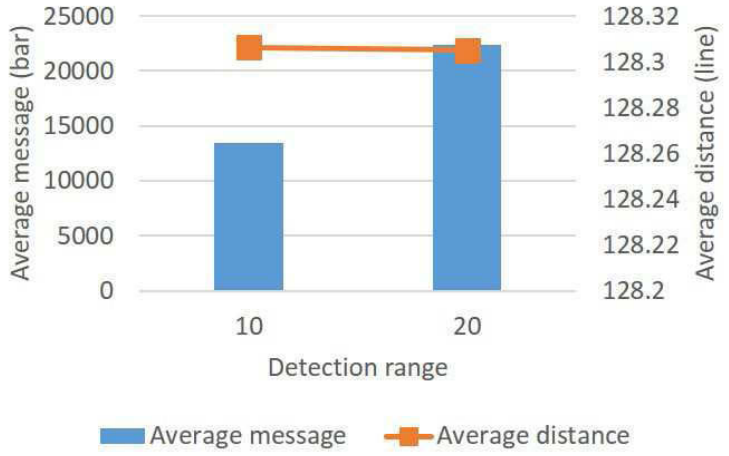



FIGURE 4.10: Average message and distance by Candidate course.

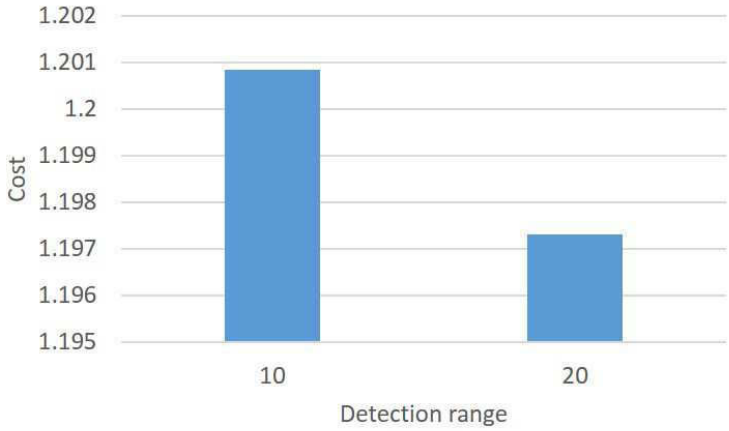



FIGURE 4.11: Average cost by Candidate course.

Figure 4.11 shows the cost. As the value of domain of detection range increased, the average message decreased. The *detection range* gets lager, a ship can detect target ship in advance. Therefore, a ship can avoid target ship with lower cost.

### 4.4.3 Classification by Safety domain

I categorized the success percentage, average message, average distance, and cost according to *safety domain* Figure 4.12 shows the success percentage depending on safety domain. The numbers below the figure mean the *safety domain*. The case of 0.5 recorded higher percentage than the case of 1. As the safety domain increased, the probability of collision also increased.

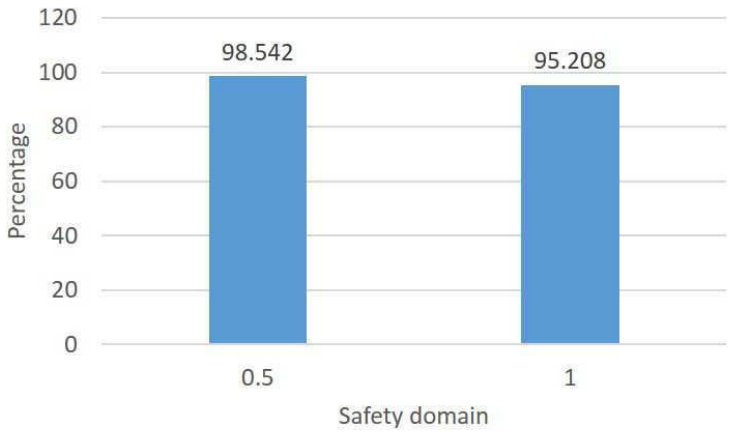



FIGURE 4.12: Success ratio by Safety domain.

Figure 4.13 shows the average message and distance. As the *safety domain* increased, the average message also increased. The average distance showed similar results.

Figure 4.14 shows the cost. The cost of the case of 1 recorded twice as much as the case of 0.5. If the *safety domain* enlarges, a ship react positively.

### 4.4.4 Classification by Timestep

I categorized the success percentage, average message, average distance, and cost according to *timestep* Figure 4.15 shows the success percentage depending on


FIGURE 4.13: Average message and distance by Safety domain.

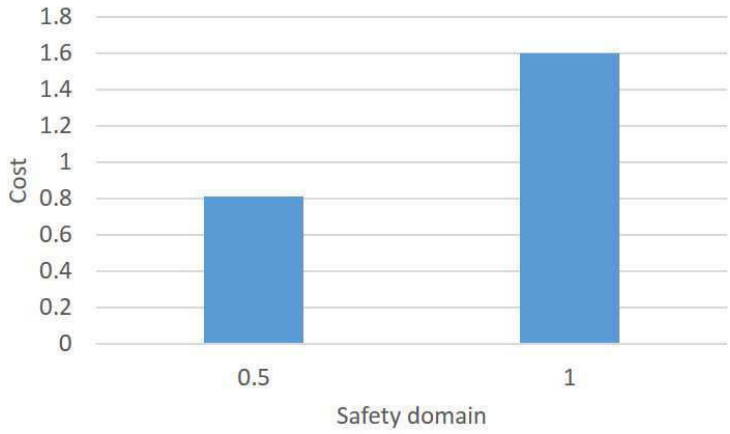



FIGURE 4.14: Average cost by Safety domain.

*timestep*. The both results of success percentage showed similar percentages. Therefore, it can infer that the timestep does not have a large impact on the collision avoidance in regard to the prevention of ship collision.

Figure 4.16 shows the average message and distance. As the domain of the *timestep* increased, the average message decreased. In the case of 10 *timestp*, the average message is decreased by approximately 22% compared to 5 *timestep*. With regard to the average message, the longer the *timestep*, the less the communication between ships. In terms of average distance, both 5 and 10 *timestep* showed the similar results.

Figure 4.17 shows the cost for *timestep*. As the domain of the timestep increased, the cost decreased. If the *timestep* gets longer, a ship can estimate the position

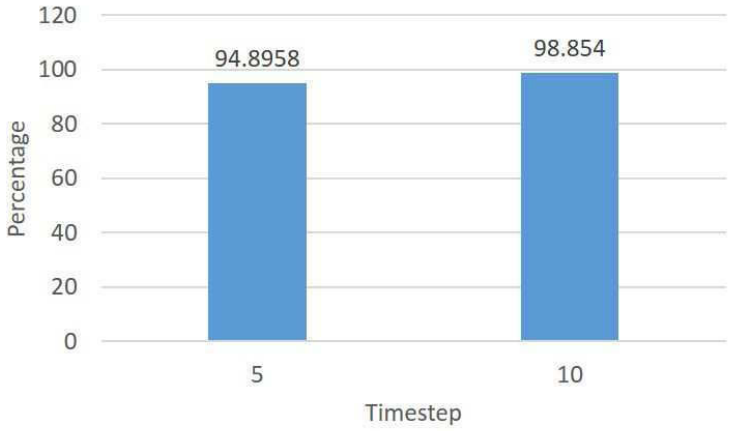



FIGURE 4.15: Success ratio by Timestep.

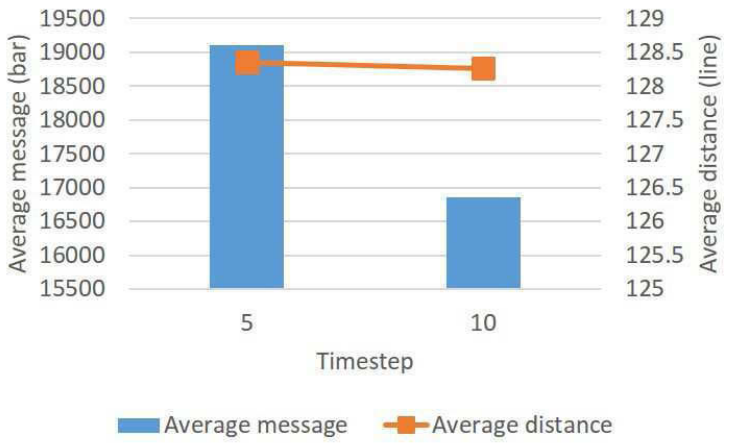



FIGURE 4.16: Average message and distance.

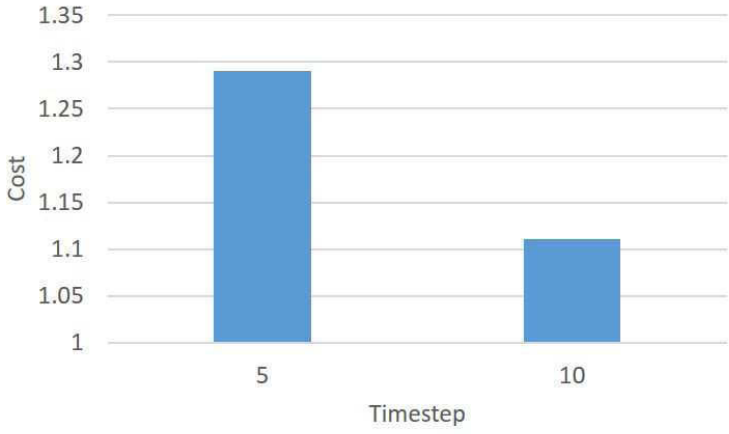



FIGURE 4.17: Average cost.

of target ships for longer. It signifies that if a ship is well aware of the position of neighboring ships, the collision risk decreases.

## 4.5 Conclusion

I have shown that using a DLSA is applicable in multiple-ship situations. Until now there has been no study that analyzed relationship between ship collision and the variables, such as *Timestep*, *Safety domain*, *Detection range*, and *Candidate course*. The experiments have shown, through the simulation, how individual ships can avoid neighboring ships. Furthermore, the experiments showed the effects of given variables - *Timestep*, *Safety domain*, *Detection range*, and *Candidate course* in different situations.

- **Candidate course**: it showed a correlation between ship collision and the ability to turning. Except the case of 10, no collision is found in all candidate courses. The case of 10 indicating $\{-10°,0°,+10°\}$ recorded lowest success percentage as 87.5%. When a ship tries to avoid a collision, she only can alter her course as 10 degrees maximumly. In the case of 45 indicating $\{-45°,0°,+45°\}$, however, a ship can alter her course as 45 degrees at a time. Of course, when a ship alters her course, a speed should be taken into consideration for safety.

  In the case of 10, it recorded highest average message and cost. A frequent maneuvering a ship can be a cause of increasing the number of exchanged messages. It will be obstructive of the decision for ship collision avoidance.

- **Detection range**: The cases of 10 and 20 showed same success percentages. It needs to adjust the size of *detection range*. The number of average messages for the case of 20 increased about 60% than the case of 10. The bigger the size of detection range, the more the average messages.

  The average cost for the case of 10 recorded a little higher than the case of 20. A ship can aware of collision risk depending on the *detection range*.

- **Safety domain**: The case of 0.5 recorded higher success ratio than the case of 1. It demonstrates a ship that requires lager safety domain has to consider a course carefully.

The bigger the size of safety domain, the more not only the average message but also distance.

- **Timestep**: A ship can estimate the next positions of target ships depending on the size of *Timestep*. The cases of 5 and 10 showed similar results. It is no required more than necessary.

  The bigger the size of *timestep*, the less not only the average message, but also the average cost.

In the "failure" case, it occupied almost 3% of this result, so the algorithm must be improved efficiency. The case of the failure occurred mostly in the case of 10 candidate course. The high speed boat such as a ground effect vehicle (GEV) has a restricted maneuvering angle. Therefore, it requires to enlarges the size of *detection range* or change the size of *safety domain*.

In future work, it needs to expand the values of individual variable domains. Because there are various types of ships such as container ship, bulk carrier and fishing boat. Each individual ship has different safety domain, speed and turning circle. To satisfy various restriction, it requires to find a proper value for a given situation.

# Chapter 5

# Distributed Tabu Search Algorithm

## 5.1 Introduction

There are many methods for preventing ship collisions at sea. COLREGs, ship domain [4, 5], fuzzy theory [6], and genetic algorithm [19]. The ship domain algorithm computes collision risk depending on whether the ships safety domain is penetrated. The fuzzy theory computes the membership function for collision risk. The genetic algorithm is based on the principle of evolution, that is, survival of the fittest. Tsou [19] used genetic algorithm to find the safest and shortest path that also complied with COLREGs. The fitness function is defined as the distance from the turning point to the original route. As chromosome constitution, there are four parameters - avoidance time, turning angle, restoration time and limited angle. They found optimum routes under three situations in which a ship can encounter a target ship. As mentioned previously, these works well in one-on-one situations, but, with multiple ships collisions may be difficult to avoid. To solve this problem, I suggested DLSA in Chapter 4. DLSA is flexible during a system failure. DLSA is easily applied to ship collision avoidance in multiple-ships situations. All ships can chart their course freely. They prefer a course that will allow them to reach their destination safely and quickly. A certain sea area, such as an entry port, crossing area, or narrow area has no option but to be crowded because all ships will travel in a similar pattern. In addition, each individual ship must find a solution by itself using local information. However, according

to the recent study, it is sometimes trapped in Quasi Local Minimum (QLM) that prevents a ship from changing course even when at risk of collision. To deal with this issue, I propose a new distributed algorithm called the Distributed Tabu Search Algorithm (DTSA)[31]. DTSA enables a ship to search for a new course compulsorily when trapped in QLM, to allow it to escape.

In Chapter 5, I explain the tabu search, and the application of tabu search for ship collision avoidance and experimental results.

## 5.2 Tabu Search

Tabu search (TS) was invented by Glover in 1986 [32]. TS was proposed to overcome local optima and it has been made to meta heuristic search method along with genetic algorithm, simulated annealing and ant colony algorithm. TS is being used in integer programming, scheduling, routing, and the traveling sales-man problem. By using memory to prohibit certain moves, TS searches for global optimization rather than local optimization. There are several kinds of memory structures, such as short, intermediate, and long-term memory. The short-term memory prohibits a solution (move) from being selected in the tabu list. The intermediate-term memory may lead to bias moves toward promising areas. The long-term memory guides to new search areas for diversity. In conventional problems, application of the short-term memory only is sufficient. As one of the features, it allows to choose a solution even though the solution is worse than the current solution. This method enables precedent local search to overcome local minimum. A combinatorial optimization problem can be represented in the following:

$$\text{Min. } f(x) : x \in X \ \ in \ \ R_n \tag{5.1}$$

$f(x)$ is the objective function which may be linear or nonlinear. The $x \in X$ that is the condition constrains $x$ to discrete values.

Let me explain hill climbing heuristic algorithm for easy understanding of TS.

**Step 0: Initialization**

- Choose an initial value $x \in X$.

**Step 1: Comparison**

- Choose some $s \in S(x)$ such that $f(s(x)) < f(x)$
- If there is no such $s$, $x$ is a local optimum and stop. Otherwise,

**Step 2: Update**

- Let $x$ be $s(x)$ and go to Step 1.

A value is chosen for $x$. $S$ is the set of neighbors of $X$. The value $x$ searches the vicinity and checks whether any neighbors $s$ which is satisfying the equation in Step 2 exist. If there are no neighbors $s$, the value $x$ is a local optimum. Otherwise, the value $x$ is substituted with the neighbor $s$, and go to Step 2.

The main problem of hill climbing heuristic algorithm is that the local optimum obtained at the stopping point that there is no moves to improve current state. TS enables a problem to search improving moves and to do not falling back a local optimum again.

**Step 0: Initialization**

- Choose an initial value $x \in X$. Let $x^*$ and $k$(iteration counter) be $x$ and 0, respectively. $T$ is empty.

**Step 1: Searching and Tabu list**

- If $S(x) - T$ is empty, go to Step 3. Otherwise, set $k := k + 1$ and choose $s_k \in S(x) - T$ such that $s_k(x) = \mathit{optimum}(s(x) : s \in S(x) - T)$

**Step 2: Update**

- Let $x$ be $s_k(x)$. If $c(x) < c(x^*)$, let $x^*$ be $x$.

**Step 3: Termination condition**

- If $k$ exceeds the iterations or $x^*$ was last improved, or if the case that directly reaching this step from Step 2 happened, then stop. Otherwise, update T and go to Step 2.

Table 5.1: Comparison of DLSA and DTSA.

| | DLSA(Kim, 2014) | DTSA(Kim, 2015) |
|---|---|---|
| Solution for QLM | None | **Tabu** |
| Solution for endless loop | Mutual exclusion with neighbors | Mutual exclusion with neighbors |
| #opportunities of message exchange per round | Twice | Twice |

A subset $T$ of $S$ is the elements called *tabu moves*. And $T$ is defined as $T(x) = \{s \in S : s(x)\}$ violates the tabu conditions. For Step 0, the value $x$ is initialized and let $x$ be the best solution found. The iteration counter $k$ sets to zero and leave the $T$ empty. If there is no candidate value, go to Step 3. Otherwise, the iteration $k$ is increased by one. And the value $x$ chooses a neighbor having optimum value. For step 2, the value $x$ moves to next point and the optimum value may be changed if $c(x) < c(x^*)$. For step 3, the termination condition is checked by the iteration $k$ and optimum value $x^*$.

## 5.3 Distributed Tabu Search Algorithm for Ship Collision Avoidance

In Chapter 5, I use TS to escape QLM, which prevents a ship in risk of collision from changing course. DLSA suffers from QLM, in which a ship cannot change its course even though a collision risk still exists. To solve this problem, we applied tabu search technique, where the ship in QLM puts her current course in a tabu list to prohibit herself from selecting that course for a certain period of time. DTSA enables individual ships to choose another course compulsorily. Table 5.1 shows the difference between DLSA and DTSA.

Figure 5.1 shows the procedure for DTSA. The whole framework is essentially the same as DLSA; only the QLM procedure (dotted red box) is added. All ships repeat this process until they arrive at their destination. Each ship checks for whether it has arrived the destination. If not, the ship searches the vicinity to find a neighboring ship. The ship exchanges an *ok?* and *improvement* messages with its neighbors. The ship with the highest improvement chooses the *next-intended course*. If there is no collision, all individuals move to the next position. If not, the ship exchanges the exchanged information with its neighboring ship. This

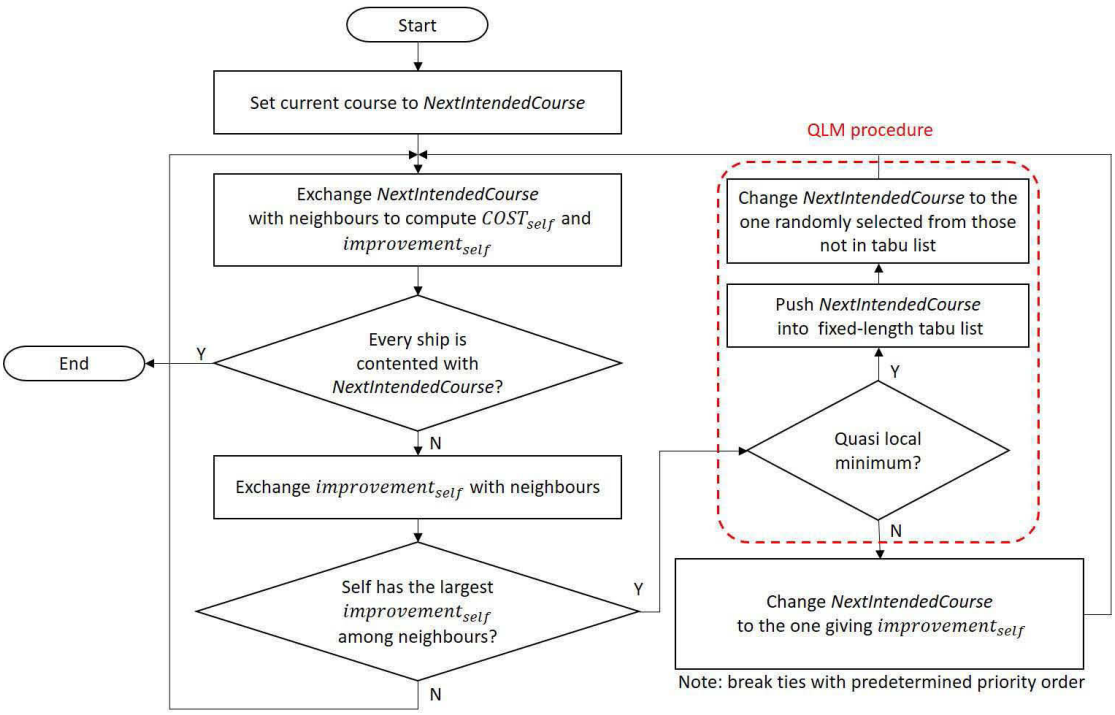



FIGURE 5.1: Sketch of Distributed Tabu Search Algorithm.

process is repeated until all ships are contended with their *next-intended courses*. If QLM occurs, a ship calls the QLM procedure, in which she randomly chooses an alternative course excepting any courses in the tabu list. This process will be recurred until QLM is resolved. If the collision risk has disappeared, all ships move to the next position.

Figure 5.2 shows the process of DTSA. Assume four ships encounter with each other as shown in Figure 5.2 (a). Ships 2 and 4 are now satisfying current course. Ships 1 and 3 have the risk of collision at current course. Even though ships 1 and 3 alter their course to avoid collision, there still exist collision risk with ships 2 and 4. The current courses of ships 1 and 3 are recorded in the *tabu list* to prevent a ship from choosing current course as shown in Figure 5.2 (b). Ships 1 and 3 choose a course randomly except *tabu list* as shown in Figure 5.2 (c). After exchanging messages with each other, ships 2 and 4 start to search a course with minimum cost as shown in Figure 5.2 (d).

Algorithm 5.5 shows the pseudocode for DTSA. A greater part of the algorithm are the same as DLSA. The procedure for QLM (step 36) is only added. If QLM happened, a ship calls a procedure QLM (step 49-70). The current *next_intended_course* is put into *tabu list*. The ship chooses a candidate course randomly except a course

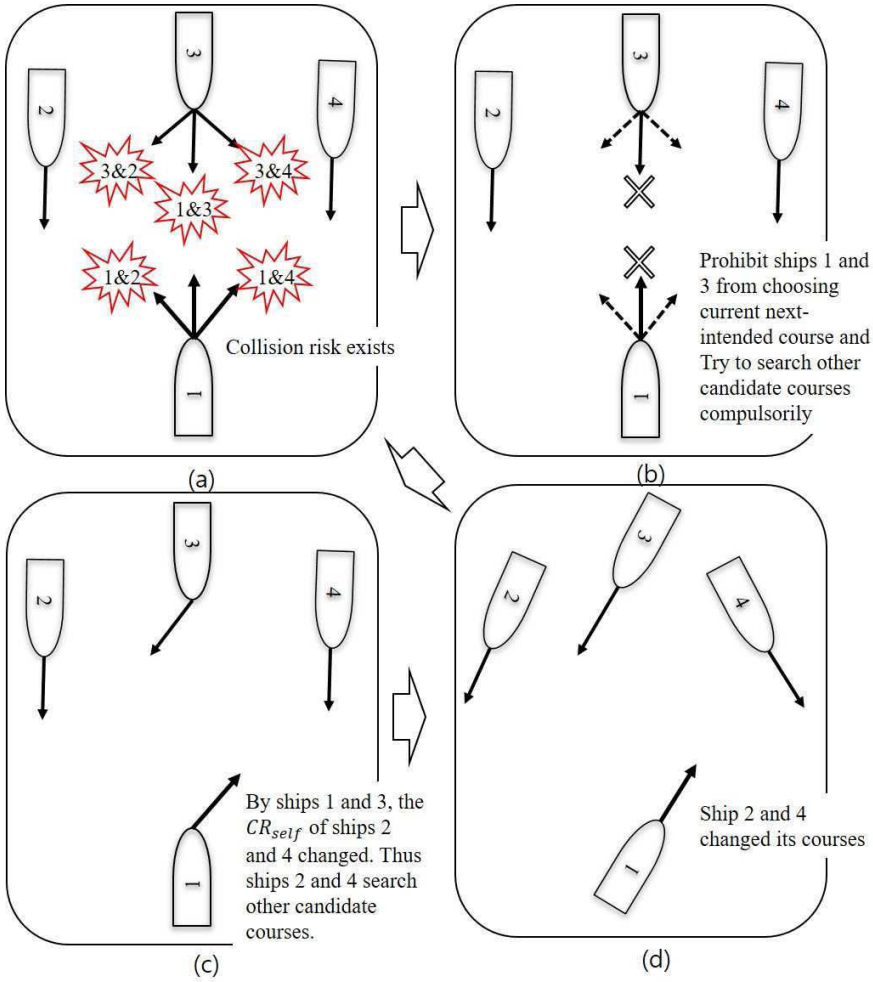



FIGURE 5.2: The process of Distributed Tabu Search Algorithm.

stored in *tabu list*. Then she sends *ok?* messages to the ships that registered in neighboring ship's list and receives *ok?* messages from neighboring ships (step 53-54). Based on the *ok?* message, she computes *cost* and *improvement* (step 56). She sends her *improvement* message to neighboring ships and receives *improvement* messages from neighboring ships (step 58-59). From step 61 to 69, it is for the process to choose a ship which has biggest improvement. And the ship can select *next-intended course*. If there are more than two ships that has same *improvement*, the tie will be broken by the ID (step 66-68). After deciding *next-intended course*, a ship $i$ proceeds to next position (step 38).

**Algorithm 5.5** Distributed Tabu Search Algorithm for Ship Collision Avoidance

```
1: improve_i ≡ maximum improvement of i
2: neighs_i ≡ neighbors of i
3: estPsn_i ≡ estimated position of i
4: ID_i ≡ identification of i
5: if i arrives at destination then
6:     arrived_i=TRUE
7: else
8:     arrived_i=FALSE
9: end if
10: Search a vicinity with Detection Range
11: if neighs_i exist then
12:     Add neighs_i to NeighsList_i
13: end if
14: while arrived_i = FALSE do
15:     while computation time < limited time do
16:         for neighs_i in neighsList_i do
17:             send ok?(estPsn_i) to neighs_i
18:             add ok?(estPsn_j) to ShipView_i
19:         end for
20:         compute cost_i and improve_i
21:         for neighs_i in neighsList_i do
22:             send improve_i to neighs_i
23:             add improve_j to ShipImprove_i
24:         end for
25:         if improve_i < max(ShipImprove_i) then
26:             i keeps current next_intended_course
27:         else if improve_i > max(ShipImprove_i) then
28:             i chooses new next_intended_course
29:         else
30:             if ID_i > ID_j of NeighsList_i then
31:                 i chooses new next_intended_course
32:             end if
33:         end if
34:     end while
35:     if QLM then
36:         Quasi-Local Minimum
37:     end if
38:     i proceeds to next position
39:     if i arrives at destination then
40:         arrived_i=TRUE
41:     else
42:         NeighsList_i = empty
43:         search a vicinity with detection range
44:         if neighs_i exist then
45:             add neighs_i to NeighsList_i
46:         end if
47:     end if
48: end while
```

**Algorithm 5.5** Distributed Tabu Search Algorithm for Ship Collision Avoidance(continued)

**procedure** QUASI-LOCAL MINIMUM
    add current $next\_intended\_course_i$ to $TabuList_i$
    $i$ chooses a candidate course randomly except $TabuList_i$
    **for** $neighs_i$ in $neighsList_i$ **do**
        send $ok?(estPsn_i)$ to $neighs_i$
        add $ok?(estPsn_j)$ to $ShipView_i$
    **end for**
    compute $cost_i$ and $improve_i$
    **for** $neighs_i$ in $neighsList_i$ **do**
        send $improve_i$ to $neighs_i$
        add $improve_j$ to $ShipImprove_i$
    **end for**
    **if** $improve_i < \max(ShipImprove_i)$ **then**
        $i$ keeps current next_intended_course
    **else if** $improve_i > \max(ShipImprove_i)$ **then**
        $i$ chooses new next_intended_course
    **else**
        **if** $ID_i > ID_j$ of $NeighsList_i$ **then**
            $i$ chooses new next_intended_course
        **end if**
    **end if**
**end procedure**

## 5.4 Experiments

The experiments are done with five different situations depending on the number of ships and various origins and destinations to test the performance of DTSA compared to DLSA. A ship has the following given values: *Safety domain* = {0.5, 1} nautical miles, *Detection range* = {10, 20} nautical miles, and *Speed* = {12} knots. The minus and plus signs indicate the port and starboard, respectively. To evaluate the performance, the success percentage and average distance are computed. Table 5.2 shows the meaning of the index used in the experimental results. All experiments are done by Matlab R2013b and a PC (Core i7-4790K, 4 cores, 8 threads, 16 GB memory and Windows 10 Professional).

### 5.4.1 1st experiment

I experimented with six ships with four variables. Figure 5.3 illustrates the situation for experiment 1. Table 5.3 shows the neighboring ship list. Each ship records

TABLE 5.2: Candidate course by Index used in experiments.

| Index | Candidate course |
|---|---|
| 15 | {−15°, 0°, +15°} |
| 30 | {−30°, 0°, +30°} |
| 45 | {−45°, 0°, +45°} |
| ALL | {−45°, −30°, −15°, 0°,+15°, +30°, +45°} |

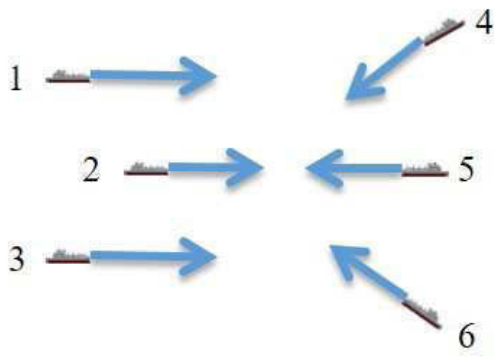



FIGURE 5.3: Situation for 1st experiment.

TABLE 5.3: List of neighboring ships for experiment 1.

| Number of ships | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|
| 1 | x | o | o | x | x | x |
| 2 | o | x | o | x | o | x |
| 3 | o | o | x | x | x | x |
| 4 | x | x | x | x | o | o |
| 5 | x | o | x | o | x | o |
| 6 | x | x | x | o | o | x |

its neighboring ships in the list. That is, ship 1 recognizes ships 2 and 3. Ship 2 recognizes ships 1, 3, and 5. There is no collision risk for ships 1, 3, 4, and 6, but ships 2 and 5 are at risk of collision. All variables are used by changing their values in one situation. In total, sixteen experiments were conducted.

Figure 5.4 shows the result for experiment 1. Compared with DLSA, DTSA has a better result. In case of 15, DTSA recorded higher success percentage than DLSA. The cases of ALL DTSA showed the best results, which were no failures and low average distance.


FIGURE 5.4: Result for 1st experiment.

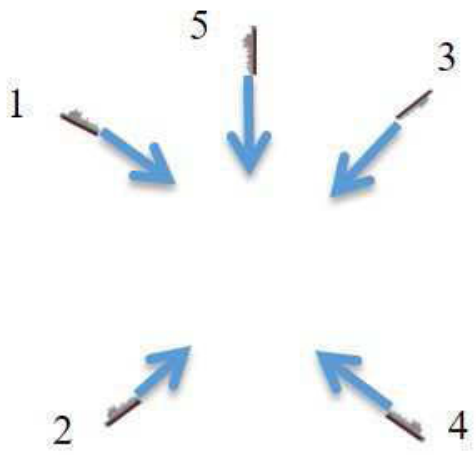



FIGURE 5.5: Situation for 2nd experiment.

### 5.4.2 2nd experiment

In experiment 2, I experimented with five ships that individual ships encounter, as shown in Figure 5.5. The tracks of ships 1, 2, 3, and 4 produced an X shape. Ship 5 cuts across the space simultaneously. Figure 5.6 shows the result for experiment 2. In the experimental result, except for 15 DLSA, the average distance showed similar figures. The cases of 30, 45 and ALL DTSA recorded no failures and low average distance. 15 DLSA had the drawback in terms of success percentage.

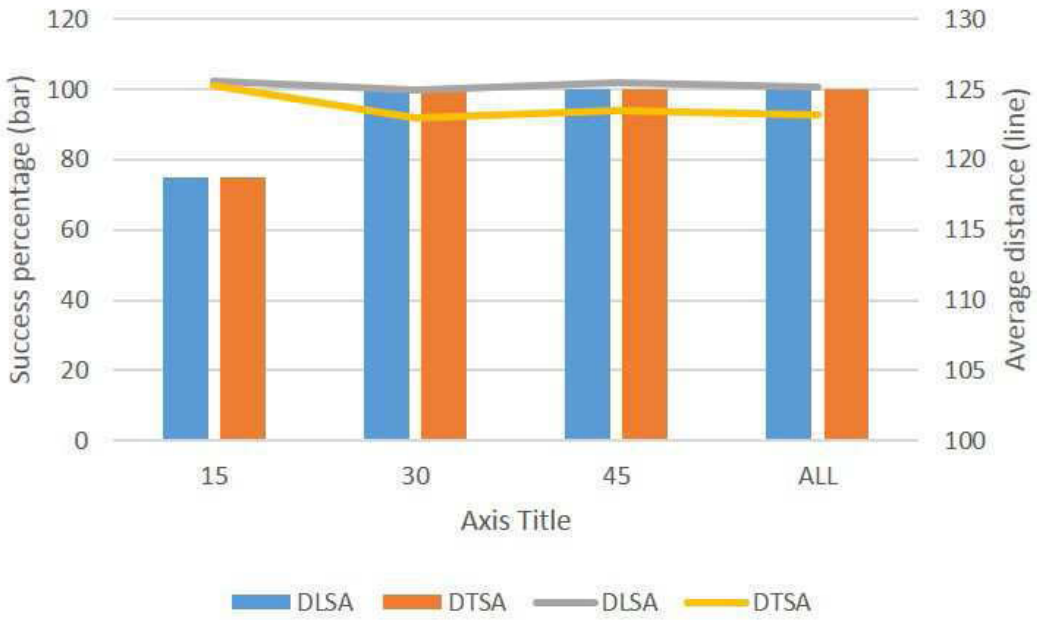



FIGURE 5.6: Result for 2nd experiment.

### 5.4.3 3rd experiment

I experimented with ten ships traveling in the same direction toward the destination from left to right, as shown in Figure 5.7. Figure 5.8 shows the result for experiment 3. Compared with DLSA, DTSA demonstrated better performance overall. All DTSA showed low and uniform average distance.

### 5.4.4 4th experiment

I experimented with twenty ships traveling in the same direction toward the destination from left to right, as shown in Figure 5.9. Figure 5.10 shows the result for experiment 4. The case of 15 for DTSA and DLSA recorded the lowest success percentage. The case of 15 for DLSA showed the highest average distance. ALL DTSA performed best in regard to the average distance. Only 15 DLSA and DTSA recorded any failures.

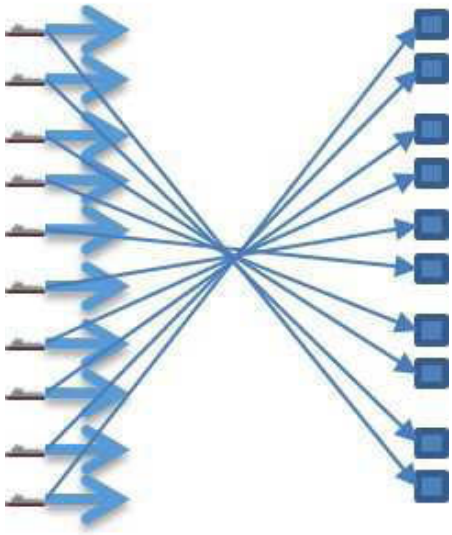


FIGURE 5.7: Situation for 3rd experiment.

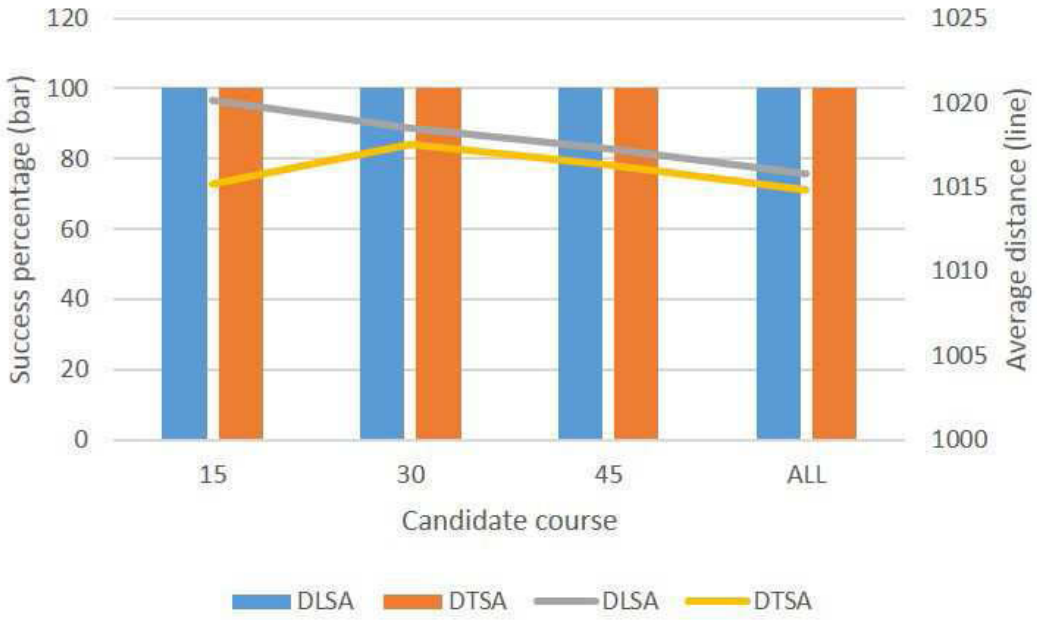



FIGURE 5.8: Result for 3rd experiment.

### 5.4.5 5th experiment

I used one hundred ships in experiment 5, as shown in Figure 5.11. The ship positions and headings were initialized randomly. The red and blue circles indicate the origin and destination for the individual ships. Figure 5.12 shows the result for experiment 5. ALL DTSA had no failure and the lowest average distance.

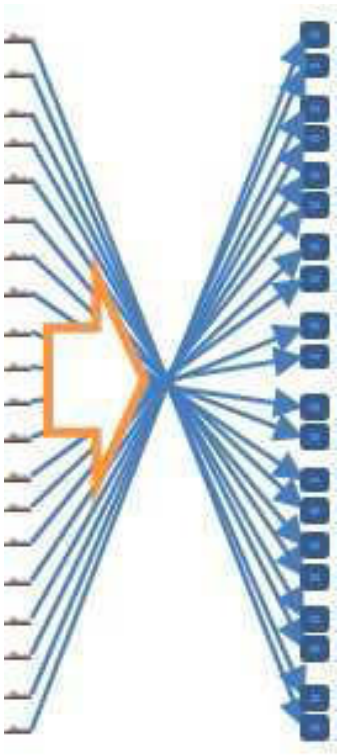


FIGURE 5.9: Situation for 4th experiment.

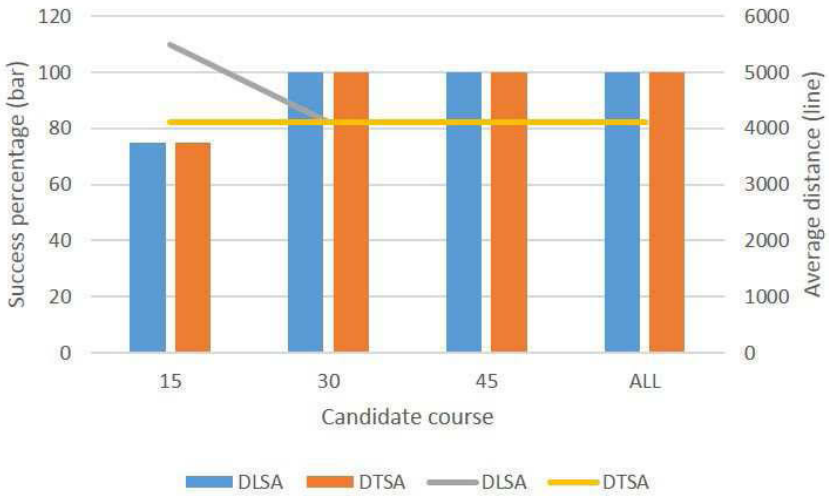



FIGURE 5.10: Result for 4th experiment.

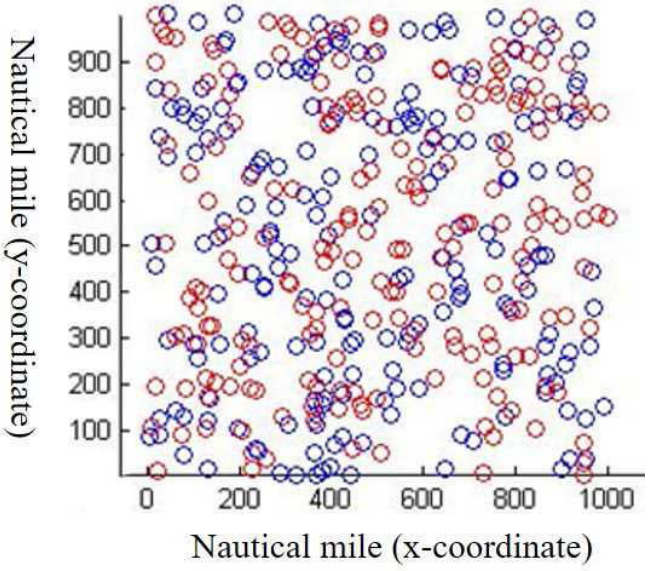



FIGURE 5.11: Situation for 5th experiment.


FIGURE 5.12: Result for 5th experiment.

The larger the domain of the candidate course, the smaller the average distance. 15 DLSA and DTSA recorded highest average distance. ALL DLSA and DTSA recorded lowest average distance.

## 5.5 Conclusion

I explained earlier that several algorithms work in specific situations, such as one-on-one situations. To avoid ship collisions in multiple-ship situations, I applied DTSA and DLSA. I used the *tabu search* algorithm to avoid the QLM problem. The experiments demonstrated how individual ships can avoid collisions in multiple-ship situations. In the experimental results, DTSA outperformed DLSA. Some experiments showed similar patterns: The more the number of candidate courses is increased, the shorter the average distance; the less the size of the degree of the candidate course, the greater the failure count. This is because a ship can bore off quickly if it drastically alters its course. ALL DTSA showed the lowest average distance in most cases. This means that the more candidate solutions, the better the performance.

# Chapter 6

# Distributed Stochastic Search Algorithm

## 6.1 Introduction

Even though navigation technology has been developing year by year, ship collision still accounts for a large percentage of maritime accidents [1]. There is no doubt that, once collisions occur, they have a negative impact on our life, economy, and the environment. To prevent ship collisions, COLREGs was adopted in 1972. It specifies navigation rules to be followed by all ships at sea to prevent collisions. However, it would be very hard to describe all possible conditions in the form of rules due to the complexity of the actual marine environment. On top of that, it would be a big burden for an officer to consider many different variables to apply to the rules in time-pressed situations. For example, in the algorithm using ship domain, the notion of the safety domain has been introduced, where a home ship prevents target ships from penetrating. Both ant colony optimization and the genetic algorithm perform searches to identify a safe course by mimicking various biological phenomena (foraging for food by ants and struggling for gene survival, respectively). Most of these methods focus on one-to-one or one-to-few situations, where a home ship decides her course by assuming that the surrounding ships will all keep sailing as they did under the previous conditions. However, since each target ship will also try to decide her course, any decision made by the home ship will inevitably affect the future decisions of the other ships, and vice versa. This might bring about chaotic behavior in the whole system, producing what we call

the "butterfly effect". In order to deal with such complex relations among multiple ships, many-to-many situations should be handled directly by modeling ships as agents who can communicate their intentions, namely *next-intended courses*, with each other to find their safest courses autonomously. For many-to-many situations, few methods have been suggested in the literature except for DLSA and DTSA in Chapter 4 and 5, respectively. Both algorithms can provide a safe course to ships in distributed system well. However, it should be taken into consideration in respect of limited range and transmission distance of frequency. In other words, the number of messages between ships needs to be reduced. In DLSA and DTSA, the mutual exclusion to prevent endless loop is one of the reasons that increases the number of messages between ships. In DLSA, each ship searches for a safer course within her own local view by exchanging intentions with neighboring ships. The DTSA enhances DLSA with the tabu search technique to escape from a QLM in which DLSA sometimes becomes trapped. One common drawback of these algorithms is that a relatively large number of messages need to be sent in order for the ships to coordinate their actions. Since message exchange accounts for the largest part of the cost of distributed algorithms, this could be fatal, especially in cases of emergency, where quick decisions should be made.

In Chapter 6, I introduce the Distributed Stochastic Search Algorithm (DSSA), where each ship changes her *next-intended course* in a stochastic manner immediately after receiving all of the intentions from the neighboring ships. In DSSA, the probability is adopted to prevent ship collision. A ship may choose new *next-intended course* with probability $p$, otherwise she will keep currently selected *next-intended course* with probability $1\text{-}p$. To know the performance of DSSA, I made experiments to compare DLSA, DTSA, and DSSA in three different settings on the number of ships, namely four, twelve, and 100 ships. As the results of experiments, in terms of sailed distance, all distributed algorithms shows similar results. However, in terms of messages that each ship exchanges with each other, DSSA spends significantly fewer messages than DLSA and DTSA. Furthermore, its stochastic nature excludes the need for a specific method to escape from QLM.

## 6.2 Motivation

The common drawback of DLSA and DTSA is that they have to send a relatively large number of messages in order for the ships to coordinate their actions. In

DLSA and DTSA, to prevent endless loop, a change of the ships among neighboring ships is prohibited, simultaneously. Since this message exchange accounts for the largest part of the cost of distributed algorithms, this could be fatal, especially in cases of emergency, where quick decisions should be made. Therefore, to reduce the number of messages is the purpose.

In the context of distributed constraint optimization, the Distributed Stochastic Algorithm has been proposed to reduce the number of messages by allowing neighboring agents to perform simultaneous changes in a stochastic manner [33, 34]. They reveal that these simultaneous changes often lead to faster convergence to a sub-optimal solution; furthermore, its stochastic nature excludes the need for a specific method to escape from QLM. The basic idea of this algorithm can be applied in the context of distributed ship collision avoidance.

## 6.3 Distirbuted stochastic algorithm

Distributed Stochastic Algorithm (DSA) had been proposed by Zhang [33]. DSA is a distributed hill-climbing algorithm to solve Distributed Constraint Optimization Problems (DCOPs) such as distributed graph coloring, distributed route planning, and resource allocation.

DCOP is composed of a tuple $(A, X, D, F)$.

- A=$\{a_1, \ldots, a_n\}$: a set of agents
- X=$\{x_1, \ldots, x_n\}$: a set of variables
- D=$\{D_1, \ldots, D_n\}$: a set of domains
- F=$\{f_1, \ldots, f_n\}$: a set of function

DSA has no ID to distinguish one another. All processes are executed synchronically. For each step, an agent sends and receives a message. They compute a cost, and then decide whether to keep current value or to change new value. If an agent chooses new value, it sends information to its neighboring agents. The principal of DSA is the followings:

**Step 0: Initialization**

- Agent $i$ chooses a value randomly.

**Step 1: Repeat until no termination is met**

- **Sending message:** Agent $i$ sends a message to neighbors, if new value is selected.
- **Receiving message:** Agent $i$ receives a message from neighbors, if any.
- **Choosing a value:** Agent $i$ chooses and assigns the next value according as table 6.1

For **Step 0**, an agent $i$ chooses a value randomly. For **Step 1**, until no termination is met, the followings are repeated. Each agent $i$ sends its current state information to its neighbors if the values of previous and current state are different. An agent $i$ receives the state information from the neighboring agents. It decides to keep a current value or change new one stochastically. The key point of DSA is how to decide next value for an agent. If there is no value that can improve its current state, an agent $i$ will not change its current value. If an agent $i$ can find new value that improves or keeps current state, new value will be chosen by the agent stochastically.

Table 6.1 shows five possible strategies for DSAs. The $\Delta$ means best improvement which is a number indicating how much the cost can be reduced between previous and current state. - means no value change. The $v$ and $p$ mean the value giving $\Delta$ and the probability.

In the case of DSA-A, for example, an agent changes its value $v$ when $\Delta > 0$ with the probability $p$. In other words, an agent can consider whether to change new value or keep current value when the $v$ brings better state than current one. Therefore, the agent may change to new value that gives best improvement with probability $p$.

TABLE 6.1: Five possible strategies for DSAs

| Algorithm | $\Delta > 0$ | Conflict, $\Delta = 0$ | No conflict, $\Delta = 0$ |
|---|---|---|---|
| DSA-A | $v$ with $p$ | - | - |
| DSA-B | $v$ with $p$ | $v$ with $p$ | - |
| DSA-C | $v$ with $p$ | $v$ with $p$ | $v$ with $p$ |
| DSA-D | $v$ | $v$ with $p$ | - |
| DSA-E | $v$ | $v$ with $p$ | $v$ with $p$ |

In the case of DSA-B, the manner is the same as DSA-A. However, a constraint is added. An agent may also change its value even though there exists a conflict and the best improvement $\Delta$ equals zero. Because the currently violated constraint may be satisfied with next step. And it may give better improvement to an agent. Consequently, DSA-B will change a value more than DSA-A.

In the case of DSA-C, an agent changes its value more often compared with DSA-B. The agent may change a value even if there is no conflict and the best improvement $\Delta$ equals zero.

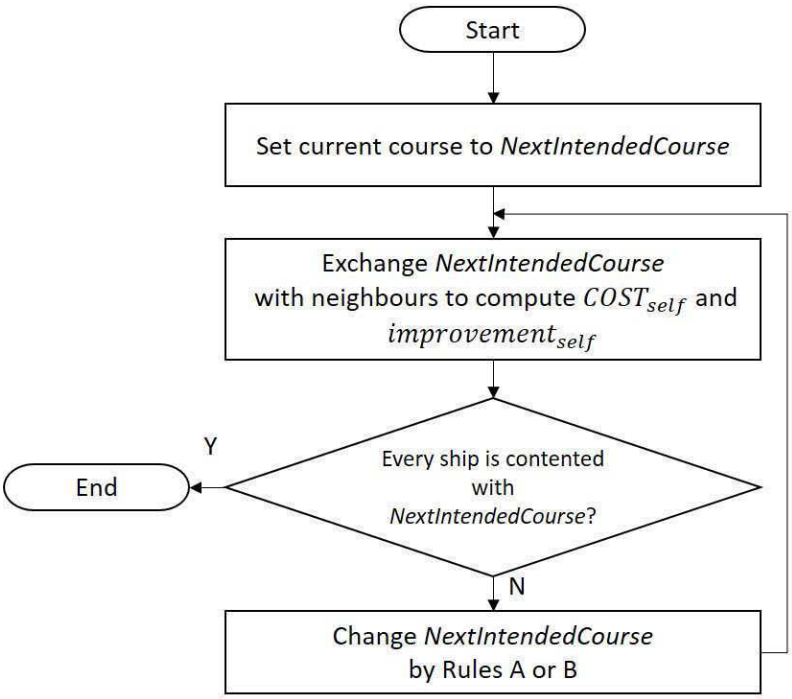
DSA-D and DSA-E are the extended models of DSA-B and DSA-C, respectively. In the case of DSA-D and E, an agent $i$ chooses the value $v$ giving $\Delta$, deterministically. This manner is greedy method. If there is any good value, an agent will change it immediately. The frequency to change a value for each agent is as followings:

- DSA-A $<$ DSA-B $<$ DSA-C
- DSA-D $<$ DSA-E

From DSA-A, to DSA-B and DSA-C, each agent changes a value frequently. DSA-E may change a value often than DSA-D. Based on the type of DSAs and the variation of probability $p$, the degree of parallel executions can be changed.

## 6.4 Detail

Figure 6.1 shows the procedure for the DSSA for ship collision avoidance. First, a ship selects her current course as the *next-intended course*. After exchanging *next-intended courses* with neighboring ships, an agent computes $COST_{self}$ and $improvement_{self}$. If some ships are not contented with the *next-intended course*, she changes her course by following rules $A$ or $B$, which are described below. This process is repeated until all ships are satisfied with their current *next-intended courses*. The *next-intended course* is chosen stochastically as follows. A certain ship, which depends on rule $A$ or $B$, chooses the course giving $improvement_{self}$ with probability $p$, but does not change with probability $1-p$. In DSSA with rule $A$ (denoted by DSSA-A), only the ships with positive $improvement_{self}$ can change




Rule A:

- When $improvement_{self} > 0$,
  change to $v$ with probability $p$;
  does not change with probability $1 - p$.

Rule B:

- When $improvement_{self} > 0$, change to $v$ with probability $p$;
  does not change with probability $1 - p$.
- When $improvement_{self} = 0$ and the cost of *NextIntendedCourse*
  is positive, change to $v$ with probability $p$;
  does not change with probability $1 - p$.

Note:

- $v$: course giving $improvement_{self}$

FIGURE 6.1: Procedure for DSSA

the *next-intended courses* stochastically. On the other hand, in DSSA with rule $B$ (denoted by DSSA-B), the ships with zero $imprvement_{self}$ can also change the *next-intended courses* if they have positive costs. This is because the change in *next-intended course* of a ship may produce better results at the next step, even if it does not reduce the cost presently. Therefore, the new *next-intended course* may be chosen with the probability $p$.

Algorithm 6.6 shows the pseudocode for DSSA. The most parts are essentially the same as DLSA; only the procedure for DSSA-A and B are added (step 20). In DSSA-A, a ship $i$ checks whether *improvement* is greater than zero (step 34).

The *next_intended_course* is replaced with the candidate course that has biggest improvement (step 35). If the probability $p$ is greater than a criterion, she will choose new *next_intended_course* (step 37). Otherwise, she will keep current *next_intended_course* (step 39). In DSSA-B, the fundamental rule is the same as DSSA-A. The condition that *improvement* is greater than or equal to zero is added (step 44).

**Algorithm 6.6** Distributed Stochastic Search Algorithm for Ship Collision Avoidance

1: $improve_i \equiv$ maximum improvement of $i$
2: $neighs_i \equiv$ neighbors of $i$
3: $estPsn_i \equiv$ estimated position of $i$
4: **if** $i$ arrives at destination **then**
5: $arrived_i$=TRUE
6: **else**
7: $arrived_i$=FALSE
8: **end if**
9: Search a vicinity with Detection Range
10: **if** $neighs_i$ exist **then**
11: Add $neighs_i$ to $NeighsList_i$
12: **end if**
13: **while** $arrived_i$ = FALSE **do**
14: **while** computation time < limited time **do**
15: **for** $neighs_i$ in $neighsList_i$ **do**
16: send $ok?(estPsn_i)$ to $neighs_i$
17: add $ok?(estPsn_j)$ to $ShipView_i$
18: **end for**
19: compute $cost_i$ and $improve_i$
20: call procedure for DSSA-A or B
21: $i$ proceeds to next position
22: **if** $i$ arrives at destination **then**
23: $arrived_i$=TRUE
24: **else**
25: $NeighsList_i$ = empty
26: search a vicinity with detection range
27: **if** $neighs_i$ exist **then**
28: add $neighs_i$ to $NeighsList_i$
29: **end if**
30: **end if**
31: **end while**
32: **end while**

**Algorithm 6.6** Distributed Stochastic Search Algorithm for Ship Collision Avoidance(continued)

```
33: procedure DSSA-A
34:     if improve_i > 0 then
35:         new next_intended_course = candidate course with improve_i
36:         if rand(0,1) > criterion then
37:             i chooses new next_intended_course
38:         else
39:             i keeps current next_intended_course
40:         end if
41:     end if
42: end procedure
43: procedure DSSA-B
44:     if improve_i ≥ 0 then
45:         new next_intended_course = candidate course with improve_i
46:         if rand(0,1) > criterion then
47:             i chooses new next_intended_course
48:         else
49:             i keeps current next_intended_course
50:         end if
51:     end if
52: end procedure
```

TABLE 6.2: Comparison of DLSA, DTSA, and DSSA.

| | DLSA(Kim, 2014) | DTSA(Kim, 2015) | DSSA |
|---|---|---|---|
| Solution for QLM | None | Tabu | Stochasticity |
| Solution for endless loop | Mutual exclusion with neighbors | | Stochasticity |
| #opportunities of message exchange per round | Twice | | Once |

## 6.5 Comparison DSSA with DLSA and DTSA

Figure 6.2 and 6.3 show the different communication method among DLSA, DTSA and DSSA. Let me suppose three ships are encountered with each other. If all ships proceed to current course, a collision will happen at the center. To prevent the collision, ships exchange messages with neighbors such as *ok?* message as shown in Figure 6.2(1). They exchange messages again such as *improvement* message as shown in Figure 6.2(2). If ship A has highest improvement, than she alters *next_intended_course* as shown in Figure 6.2(3). While ships B and C do nothing, because of the prevention for endless loop. Thus the collision between ships B and C still remains, they exchange messages with neighbors as shown in Figure 6.2(4, 5). Finally, ships A and B alter their courses. And the collision disappeared. At that time, the total number of messages are 24 times.

For DSSA, the total number of messages are 6 times. Because all ships send messages to neighbors. And each ship changes next intended course by probability $p$. By the stochastic nature, there is no need to wait for the decision of neighbors or send *improvement* message.

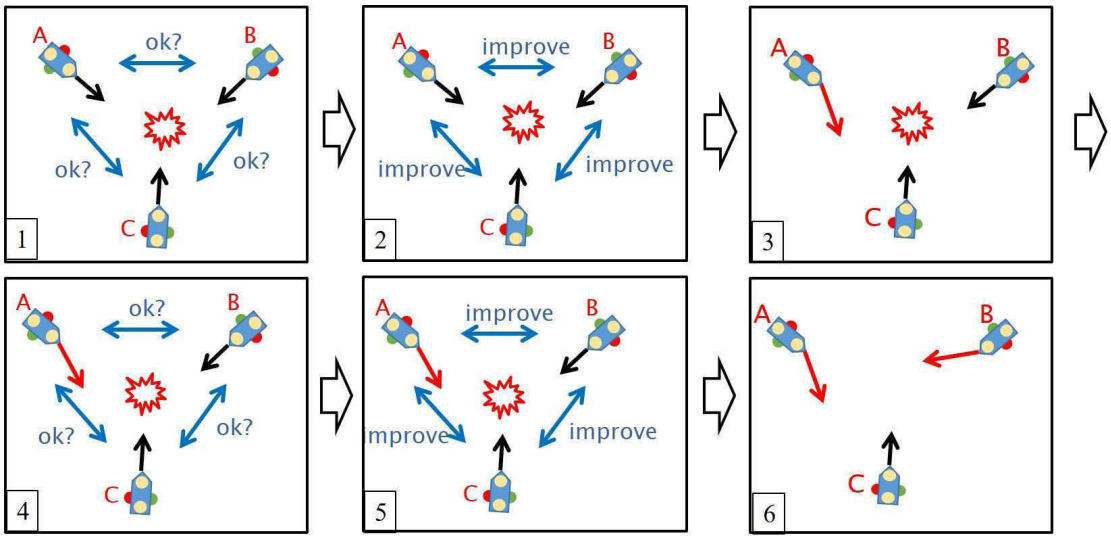



FIGURE 6.2: How to exchange messages for DLSA and DTSA

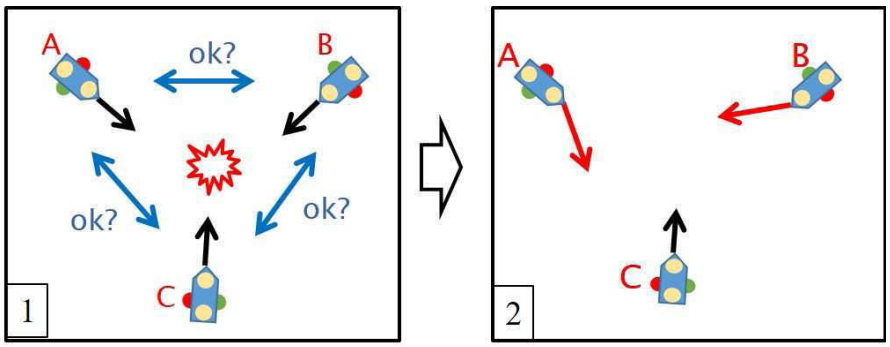



FIGURE 6.3: How to exchange messages for DSSA

## 6.6 Experiments

DSSA is compared with DLSA and DTSA to evaluate the performance. As a result of the preliminary experiments, DSSA-C, D and E are excluded. Figures 6.4, 6.5, and 6.6 show the results of them. As mentioned above, DSSA-C, D, and E are the modified models of DSSA-A and B. In DSSA-C, even though a ship satisfies a current course, she may change her course when the probability $p$ is greater than a criterion. DSSA-D and E are the greedy methods that all ships take a candidate course with best improvement at the same time. These methods, namely DSSA-C, D and E, can complicate the situation.

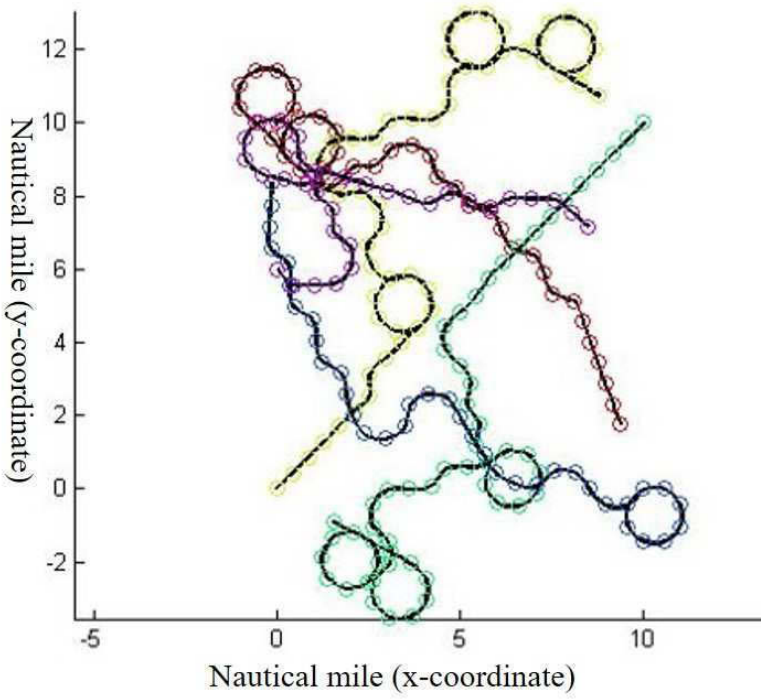



FIGURE 6.4: Simulated trajectories by DSSA-C.

The experiments are done by changing the number of ships. For 1st experiment, total four ships are used. For 2nd experiment, total twelve ships are used. For 3rd experiment, total 100 ships are used. The heading, origin and destination for all ships were generated randomly. All experiments are done by Matlab R2013b and a PC (Core i7-4790K, 4 cores, 8 threads, 16 GB memory and Windows 10 Professional).

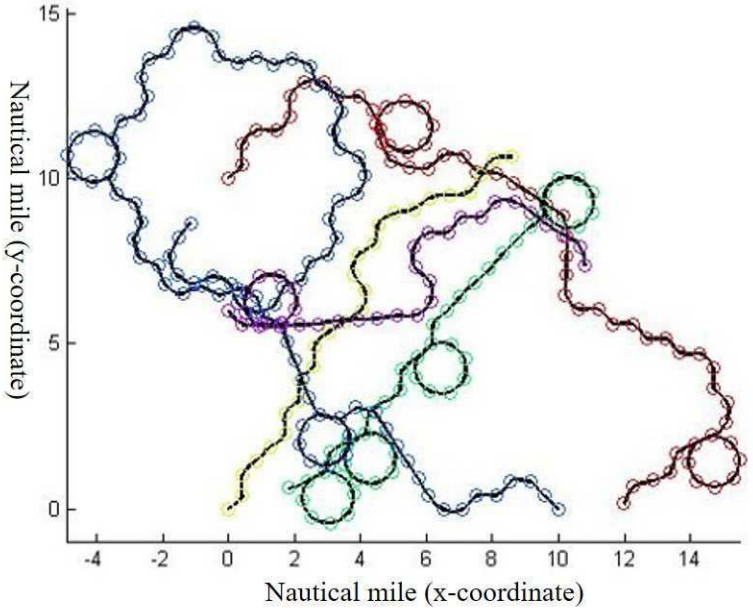



FIGURE 6.5: Simulated trajectories by DSSA-D.

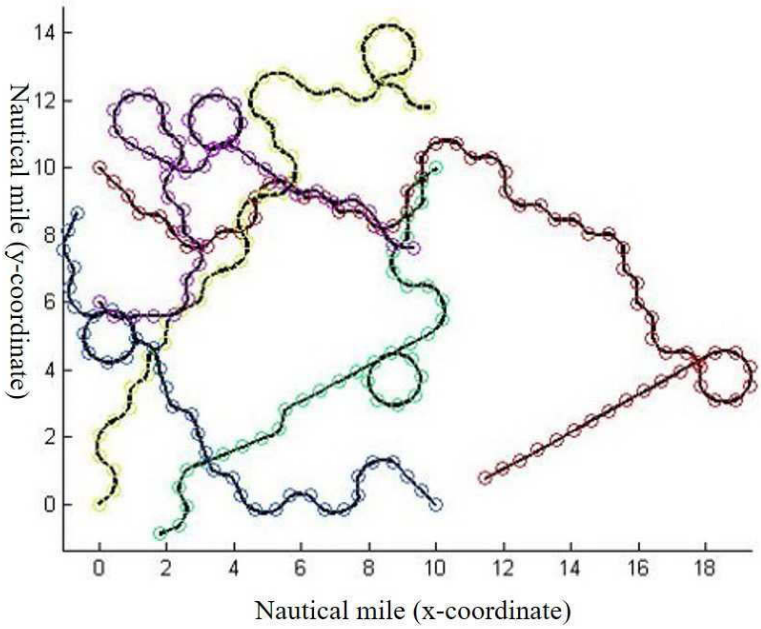



FIGURE 6.6: Simulated trajectories by DSSA-E.

### 6.6.1 Four-ship Encounter

In distributed ship collision avoidance, individual ships all cooperate with each other. To clarify the importance of such cooperation, I show a simple result before conducting comprehensive experiments using DLSA, DTSA, and DSSA.

#### 6.6.1.1 Cooperative and non-cooperative situation

Figure 6.7 shows the simulated trajectories of (a) Non-Cooperative and (b) Cooperative ships for a four-ship encounter instance. In this instance, four ships, each sailing in a diagonal direction, are arranged so that they intersect with each other at the center. Non-Cooperative ships means that they make decisions on the basis of their own gathered information such as from radar or AIS without any exchange of information. Most previous studies have been conducted along the same line. As shown in Figure 6.7(a), each ship's trajectory forms a jagged shape. This indicates that each ship has to suddenly and significantly change her course. I observed that these behaviors sometimes lead to collisions. On the other hand, Cooperative ships can exchange information with any relevant target ship. A ship can predict the next movement of a target ship, thereby enabling each ship to figure out whether a collision risk exists or not beforehand. As shown in Figure 6.7(b), all trajectories are smooth, and finally every ship arrives at her destination without any collision. Note also that in terms of average distance each ship has to travel, Cooperative ships give shorter ones than Non-Cooperative ships.

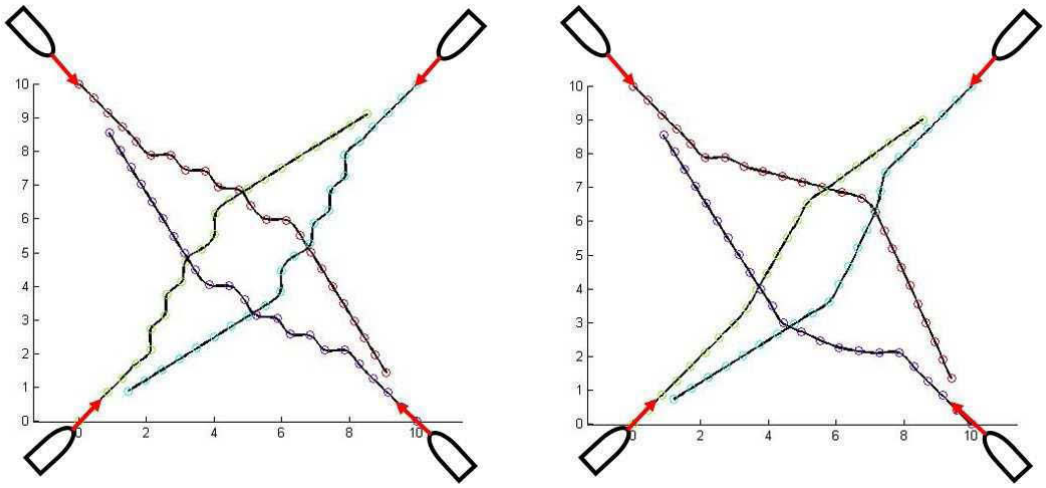



FIGURE 6.7: Simulated trajectories of four ships by Non-Cooperative (a, left) and Cooperative (b, right).

Then, I compare the performance of DLSA, DTSA, and DSSA for the four-ship encounter instance while varying time limit over three to ten seconds. The results of this simulation are shown in Figure 6.8. The bar indicates the average distance over the ships and the line indicates the number of messages exchanged among the ships. The number in parenthesis of DSSA is probability $p$. Compared to DLSA and DTSA, DSSA-A and B showed better performances. In the case of

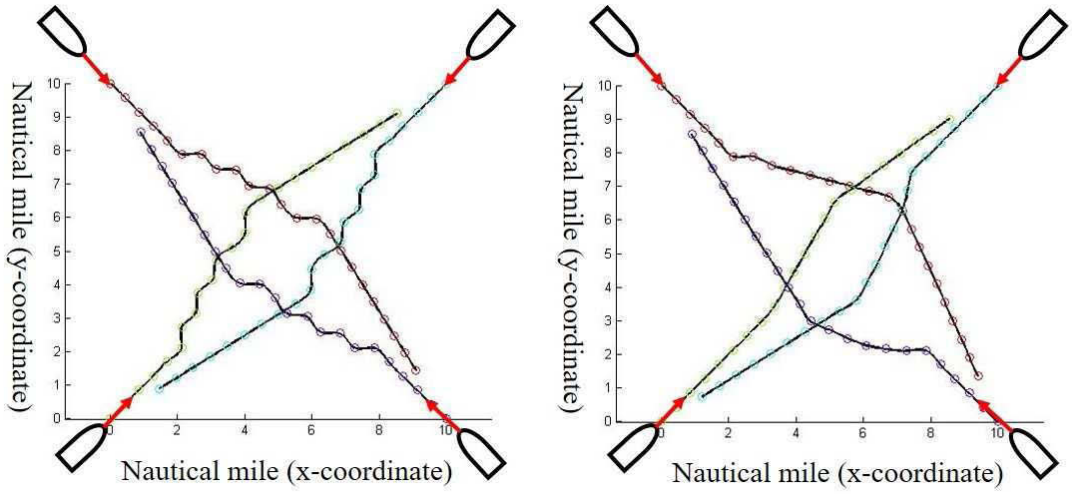



FIGURE 6.8: Result of simulation for Four-ship Encounter.

average distance, DLSA and DTSA showed almost the same results. The average distance for DSSA-A and B recorded lower than DLSA and DTSA. In the case of the number of messages for DLSA and DTSA, the longer the time limit, the greater the number of messages. Compared to DLSA and DTSA, DSSA spent much fewer messages than that, regardless of time limit. This implies that DSSA did not exceed any time limit to find optimal courses for all ships, while DLSA and DTSA often did so. DSSA enables multiple ships to alter their *next-intended courses* simultaneously, leading to fast convergence to the optimum within one second.

### 6.6.2 Twelve-ship Encounter

Total twelve ships are used, as shown in Figure 6.9. This is one of the simulated trajectories of twelve ships by DSSA-A. All ships arrived at their destinations without collision. It also demonstrates how much the home ship's decision is affected by the target ships. The ships in the middle that are surrounded by many target ships altered their courses significantly while other ships altered their courses only a little. Figure 6.10 shows the average distance and the number of messages for the twelve-ship encounter instance. In terms of average distance, all algorithms showed a similar result. In terms of the number of messages, DSSA had much fewer than DLSA and DTSA.

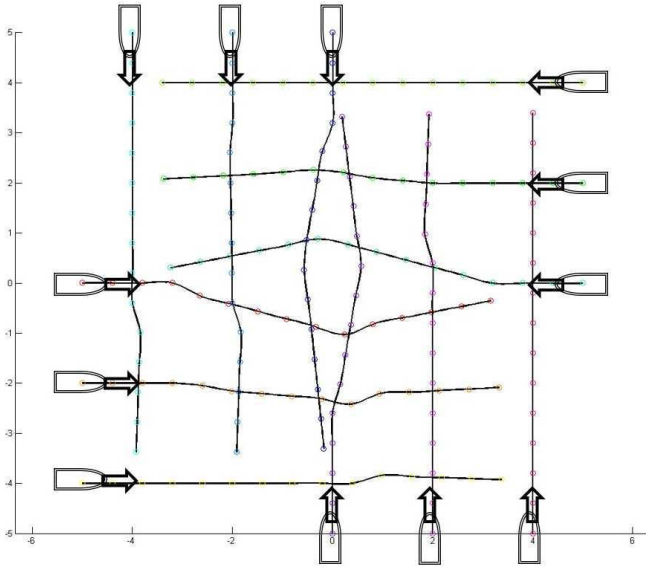


FIGURE 6.9: Simulated trajectories of twelve-ship by DSSA-A.

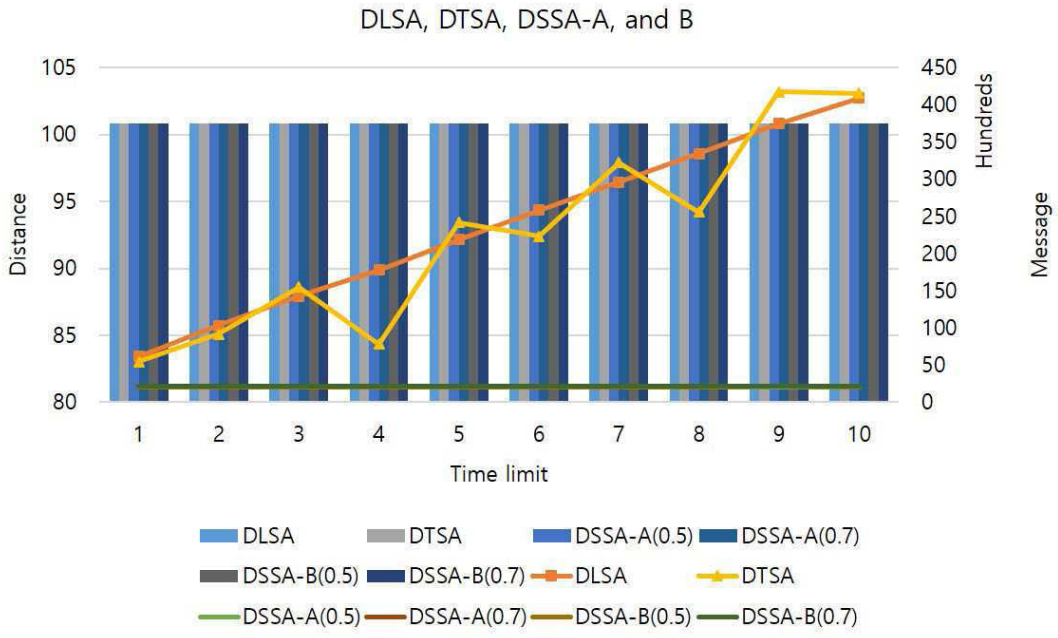



FIGURE 6.10: Result of simulation for Twelve-ship Encounter.

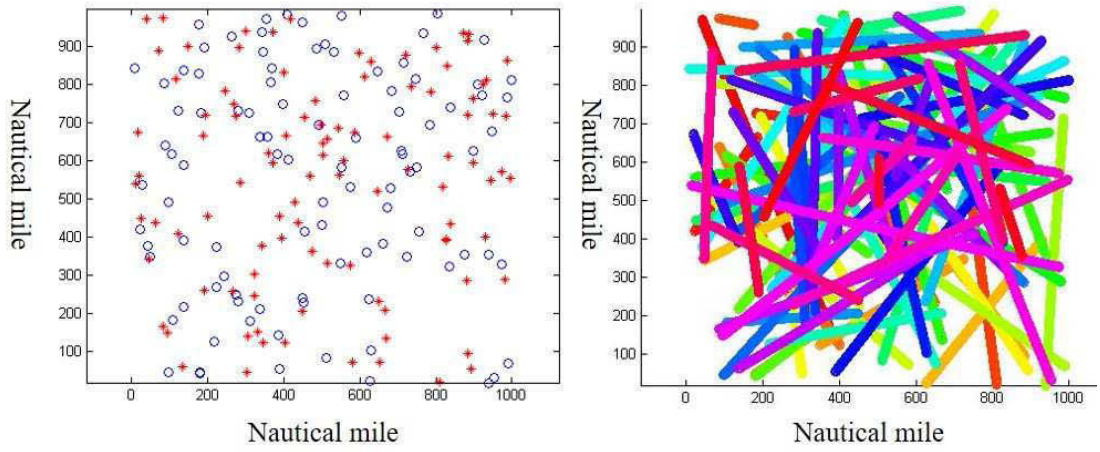



FIGURE 6.11: Initialization (a, left) and trajectories (b, right) for 100-ship encounter instance.

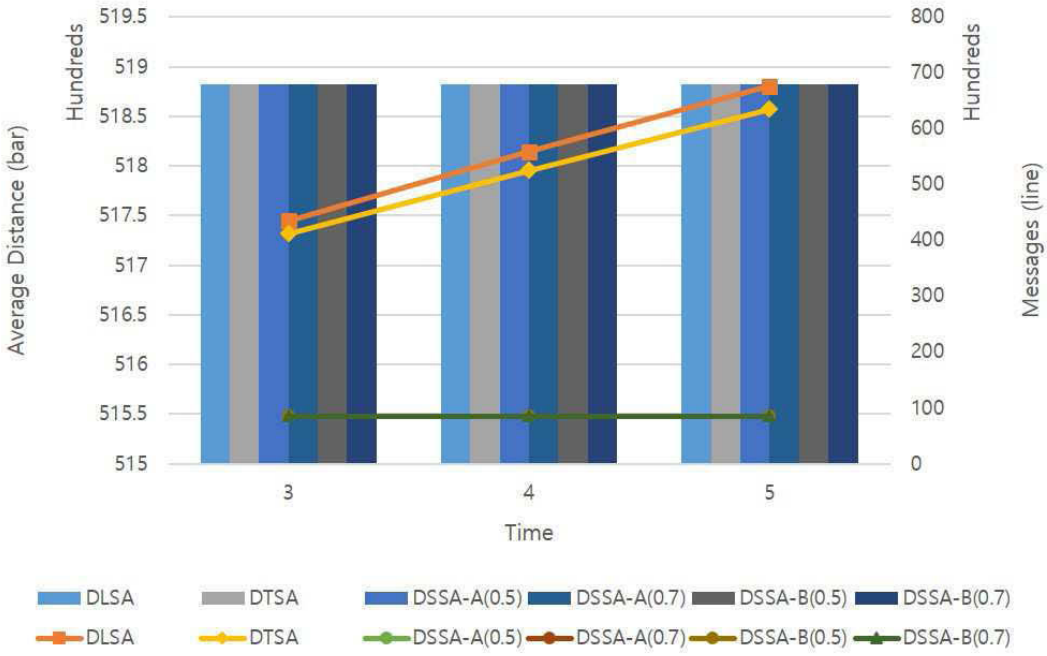



FIGURE 6.12: Result of simulation for One hundred-ship Encounter.

### 6.6.3 One hundred-ship Encounter

I used 100 ships in this experiment. The heading, current position and destination of each ship were generated randomly. Figure 6.11(a) shows their origins and destinations in blue circles and red stars, respectively. Figure 6.11(b) shows the trajectories computed by DSSA-A for this problem instance. Figure 6.12 indicates both the average distance of trajectories and the number of messages exchanged

by DLSA, DTSA and DSSA when time limit varies from three to five in order to finish simulations within a reasonable amount of time. Note that no collision occurs in this experiment. Again, I can see that while all algorithms showed a similar result in average distance, DSSA performed much better than DLSA and DTSA in terms of the number of messages.

## 6.7 Conclusion

In Chapter 6, I proposed Distributed Stochastic Search Algorithm for ship collision avoidance. There are five types of DSSA depending on the probability and the improvement. Depending on the probability, a ship chooses a *next-intended course*. This manner can solve a QLM problem and reduce the number of messages. This simplified the algorithm. Also I showed the pseudocode of DSSA-A and B. I explained how to exchange messages among ships as shown in Figure 6.2 and 6.3. To verify the importance of such cooperation, I showed the experiments when Non-Cooperative and Cooperative situations by DLSA, DTSA, and DSSA.

In experimental results, compared to DLSA and DTSA, DSSA-A and B recorded lowest average distance and messages. Figure 6.9 shows that how much a ship's decision is affected by neighboring ships. Until now, there is no large scale experiment, such as one hundred-ship encounter. I demonstrated that Distributed Algorithms can be applied to multiple-ship situations. Note that all experiments are recorded no collision.

For future work, It needs to consider the characteristics of ships. Some of variables, such as *detection range, safety domain* may be sensitive to a distributed algorithm for ship collision avoidance. It needs to be able to cope with various situations.

# Chapter 7

# Conclusions and Future work

This doctoral dissertation has described how to avoid a collision between ships by applying Distributed Algorithms. In Chapter 7, I summarize the research.

## 7.1 Research summary

In Chapter 1, I introduced the definition of the ship collision and presented the characteristics of ship and the statics of marine accidents.

In Chapter 2, I presented the background and related work in the field of ship collision. To do that, I explained precedent methods, such as COLREGs, ship domain, fuzzy theory, and genetic algorithm. From these related works, I recognized the problem of centralized system for ship collision avoidance.

- Most of the precedent methods need to make up for their weak points.
- It is important to know the intention of neighboring ships.
- Small action of ship may have a major influence to the decision of neighboring ships. And it may bring about the 'butterfly effect' of the whole system.
- A centralized system like VTS is not enough to prevent collision between ships.

In Chapter 3, the common parts for the distributed algorithms are explained. This chapter presented framework, terminology, and new cost function. The framework

is made up of two procedures: control and search. For the control procedure, a ship decides whether to proceed to the next position. for the search procedure, a ship tries to avoid a collision by running a distributed algorithm when she confirms that there is a collision risk. The ships alternate the search and control procedures until they arrive at their destinations. For terminology, I defined the meaning of terms for distributed collision avoidance. To compute the collision risk, I suggested the new cost function and demonstrated an example how to compute it. The cost function is comprised of two parts, such as the *collision risk* against a neighboring ship and the *relative angle* between a candidate course and destination.

In Chapter 4, I described Distributed Local Search Algorithm. This method is first trial in the field of ship collision especially when many ships are encountered, e.g. 100 ships. I presented the process of DLSA and showed how to exchange messages with neighboring ships.

In Chapter 5, I proposed Distributed Tabu Search Algorithm. To solve the problem of DLSA, tabu search is applied. DLSA is sometimes trapped in QLM that prevents a ship from changing course. DTSA enables a ship to search for other course compulsorily when trapped in QLM. The framework of DTSA is the same as DLSA, essentially. The QLM procedure is added. I described the process of DTSA. I made total five experiments by changing the number of ships and the variables.

In Chapter 6, I proposed Distributed Stochastic Search Algorithm (DSSA). I applied Distributed Stochastic Algorithm that proposed by Zhang to ship collision avoidance. In DSSA, ships have no ID to distinguish one another. All processes are done synchronically. To choose *next-intended course*, a probability is applied. This stochastic manner can reduce the number of messages and solve QLM. Furthermore, I simulated two kinds of situations, i.e. cooperative and non-cooperative situations. In non-cooperative situation, a ship's trajectory forms a jagged shaped. On the other hand, all trajectories of cooperative ships are smooth. It signified how much it is important to know the intention of neighboring ships. As the results of experiments, the number of messages for DSSA showed much fewer than DLSA and DTSA.

## 7.2 Future work

- The multiple destinations or waypoints for ships need to be considered. In this doctoral dissertation, all ships have only one destination. In real situation, however, there are many waypoints on the way to the destination from origin.
- All ships have different maneuvering characteristics. Considering that, the parameters, e.g. *detection range, safety domain, timestep* and the weight factor $\alpha$ have to be adjusted.
- I assumed that all ships can exchange messages by AIS. It is necessary to research how to exchange messages in practice.
- The detailed study for cost function is needed whether any variable requires.
- The obstructions, e.g. island, breakwater, and shallow water, need to be considered for the many diverse situations.
- It requires the path finding algorithms, such as A*, D*, and Dijkstra algorithm to make sailing distance shortened or improve efficiency of distributed algorithms.

# Bibliography


[1] Clifford C. Baker and Ah Kuan Seah. Maritime accidents and human performance: the statistical trail. *ABS Technical Papers*, 2004.

[2] Szlapczynski Rafal. A unified measure of collision risk derived from the concept of a ship domain. *Journal of Navigation*, 59(3):477–490, September 2006.

[3] Szlapczynski Rafal. Determining the optimal course alteration manoeuvre in a multi-target encounter situation for a given ship domain model. *Annual of Navigation*, 12:75–85, 2007.

[4] E.M. Goodwin. A statistical study of ship domains. *Journal of Navigation*, 28(28):328–344, January 1975.

[5] Y. Fujii and K. Tanaka. Traffic capacity. *Journal of Navigation*, 24(4):543–552, October 1971.

[6] K. Hasegawa, A. Kouzuki, T. Muramatsu, H. Komine, and Y. Watabe. Ship auto-navigation fuzzy expert system (safes). *Journal of the Society of Naval Architects of Japan*, 166(11):445–452, November 1989.

[7] N. Wang, X. Meng, Q. Xu, and Z. Wang. A unified analytical framework for ship domains. *Journal of Navigation*, 62:643–655, 2009.

[8] S. Lee, K. Kwon, and J. Joh. A fuzzy logic for autonomous navigation of marine vehicles satisfying colreg guidelines. *International Journal of Control, Automation, and Systems*, 2(2):171–181, 2004.

[9] Transport Minister of Land, Infrastructure and Tourism. Statics of marine accident, April 2016. URL http://www.mlit.go.jp/jtsb/statistics_mar.html.

[10] Wikipedia. Exxon valdez oil spill, June 2016. URL https://en.wikipedia.org/wiki/Exxon_Valdez_oil_spill#cite_note-40.

[11] Szlapczynski Rafal. Solving multi-ship encounter situations by evolutionary sets of cooperating trajectories. *International Journal on Marine Navigation and Safety of Sea Transportation*, 4(2):185–190, 2010.

[12] Szlapczynski Rafal. Evolutionary sets of safe ship trajectories: A new approach to collision avoidance. *Journal of Navigation*, 64:169–181, 2011.

[13] Szlapczynski Rafal. Evolutionary planning of safe ship tracks in restricted visibility. *Journal of Navigation*, 68:39–51, 2015.

[14] W.G.P. LAMB and J. M. HUNT. Multiple crossing encounters. *Journal of Navigation*, 48(1):105–113, January 1995.

[15] W.G.P. LAMB and J. M. HUNT. Multiple encounter avoidance manoeuvres. *Journal of Navigation*, 53(1):181–186, 2000.

[16] S. Hornauer. Decentralized collision avoidance in a semi-collaborative multi-agent system. *11th German Conference, MATES 2013*, 8076:412–415.

[17] S. Hornauer, A. Hahn, M. Blaich, and J. Reuter. Trajectory planning with negotiation for maritime collision avoidance. *International Journal on Marine Navigation and Safety of Sea Transportation*, 9(3):335–341, 2015.

[18] COLREG. *International Maritime Organization*.

[19] M. Tsou and C. Hsueh. The study of ship collision avoidance route planning by ant colony algorithm. *Journal of Marine Science and Technology*, 18(5): 746–756, 2010.

[20] L.A. Zadeh. Fuzzy sets. *Information and control*, 8(3):338–353, 1965.

[21] E. Kim, I. Kang, and Y. Kim. Collision risk decision system for collision avoidance. *Korean Institute of Intelligent systems*, 11:524–527, 2001.

[22] D. Kim, K. Hirayama, and G. Park. Collision avoidance in multiple-ship situations by distributed local search. *Journal of Advanced Computational Intelligence and Intelligent Informatics* (JACIII), 18(5):839–848, 2014.

[23] V. Arya, N. Garg, R. Khandekar, A. Meyerson, K. Munagala, and V. Pandit. Local search heuristics for $k$-median and facility location problems. *Society for Industrial and Applied Mathematics of Journal on Computing*, 33(3):544–562, 2012.

[24] Local search, 2010. URL http://artint.info/html/ArtInt_83.html.

[25] D. Bertsimas and J. Tsitsiklis. Simulated annealing. *Statistical Science*, 8(1): 10–15, 1993.

[26] John H. Holland. *Adaptation in Natural and Artificial Systems*. The MIT Press, 1975.

[27] K. Hirayama and M. Yokoo. The distributed breakout algorithms. *Aritificial Intelligence*, 161(1–2):89–115, 2005.

[28] M. Yokoo and K. Hirayama. Distributed breakout algorithm for solving distributed constraint satisfaction problems. *Second International Conference on Multiagent Systems*, (5):401–408, 1996.

[29] M. Yokoo, Edmund H. Durfee, T. Ishida, and K. Kuwabara. The distributed constraint satisfaction problem: Formalization and algorithms. *IEEE Transactions on Knowledge and Data Engineering*, 10(5):673–685, September/October 1998.

[30] Makoto Yokoo and Katsutoshi Hirayama. Algorithms for distributed constraint satisfaction: A review. *Autonomous Agents and Multi-Agent Systems*, 3(2):185–207, June 2000.

[31] D. Kim, K. Hirayama, and T. Okimoto. Ship collision avoidance by distributed tabu search. *International Journal on Marine Navigation and Safety of Sea Transportation* (TRANSNAV), 9(1):23–29, 2015.

[32] F. Glover. Tabu search-part I. *ORSA Journal on Computing*, 1(3):190–206, 1989.

[33] W. Zhang, O. Wang, and L. Wittenburg. Distributed stochastic search for constraint satisfaction and optimization: Parallelism, phase transitions and performance. *the Association for the Advancement of Artificial Intelligence*, pages 53–59, 2002.

[34] W. Zhang, G. Wang, Z. Xing, and L. Wittenburg. Distributed stochastic search and distributed breakout: properties, comparison and applications to constraint optimization problems in sensor networks. *Artificial Intelligence*, 161(1–2):55–87, 2005.