TransNav
http://www.transnav.eu
the International Journal on Marine Navigation and Safety of Sea Transportation
Volume 13
Number 1
March 2019
DOI: 10.12716/1001.13.01.11

DSSA$^+$: Distributed Collision Avoidance Algorithm in an Environment where Both Course and Speed Changes are Allowed

K. Hirayama, K. Miyake, T. Shiota & T. Okimoto
Kobe University, Kobe, Japan

ABSTRACT: Distributed Stochastic Search Algorithm (DSSA) is one of state-of-the-art distributed algorithms for the ship collision avoidance problem. In DSSA, whenever a ship encounters with any number of other ships (neighboring ships), she will select her course with a minimum cost after coordinating their decisions with her neighboring ships. The original DSSA assumes that ships can change only their courses while keeping their speed considering kinematic properties of ships in general. However, considering future possibilities to address more complex situations that may cause ship collision or to deal with collision of other vehicles (such as mobile robots or drones), the options of speed changes are necessary for DSSA to make itself more flexible and extensive. In this paper, we present DSSA^+, as a generalization of DSSA, in which speed change are naturally incorporated as decision variables in the original DSSA. Experimental evaluations are provided to show how powerful this generalization is.

1 INTRODUCTION

According to statistics of ship accidents released by the Japan Transport Safety Board of the Ministry of Land, Infrastructure, Transport and Tourism, there are more than 200 ship collision accidents each year in Japanese waters. Since human, economic and environmental damage caused by a ship collision accident will be enormous, several legal or technical measures have been taken at present.

One measure is the COLREG rules, which are international operation rules to prevent ship collisions enacted in 1972 and partial revision has continued thereafter. However, since these rules basically set action guidelines in the so-called one-to-one situation where one ship faces another ship, they do not work well in a narrow area nearby a large port, where many ships are likely to get congested.

As another measure to prevent collisions, vessel traffic service (VTS) centers have been established on the coast of narrow and congested sea area, who provide each ship with instructions on the safe and efficient movement. However, in order to deal with dangerous and emergent situations, quick and proper instructions must be made, which is very challenging even for experienced marine traffic controllers. Furthermore, the cost of maintaining VTS centers is tremendously high.

In order to support decision making by ship officers, communication systems called automatic radar plotting aid (ARPA) and automatic identification system (AIS) are already working on board. ARPA is the system to plot the positions of neighboring ships, which are obtained through radar, on screen, while AIS is the system that enables ships to exchange various information such as id, types, positions, speeds, destinations, with their neighboring ships. For ships that meet certain criteria, installation of AIS is internationally obligatory.

Kim and others have proposed the Distributed Stochastic Search Algorithm (DSSA) with the aim of realizing complete automatic ship collision avoidance (Kim et.al. 2017). In DSSA , ships having communication function such as AIS exchange their intentions (future plans of actions) of each other, and automatically decide their courses to avoid collision without relying on any central control like a VTS center.

Since there is no brake on the ship, the speed cannot be changed rapidly in principle. Therefore, in DSSA , it was assumed that the ship would only change the course while keeping the speed to avoid collision. However, in consideration of the progress of vessel maneuvering technology by using controllable pitch propellers and the necessity to more effectively avoid collision, we extend DSSA in this paper so that ships can change both course and speed.

The remaining part of this paper is structured as follows. Section 2 mentions some related work on computational approaches to ship collision avoidance. Section 3 gives our general framework on distributed ship collision avoidance and one of its latest algorithms, Distributed Stochastic Search Algorithm (DSSA). Section 4 presents DSSA$^+$, as a generalization of DSSA , and followed by experimental evaluation showing its effectiveness in Section 5. Finally, Section 6 concludes this work.

2 RELATED WORK

Several computational methodologies for realizing automatic ship collision avoidance have been proposed. Most of them deal with one-on-one or one-to-many situations (Lamb and Hunt 1995; Lamb and Hunt 2000; Lee et.al. 2004; Hu et al. 2008; Tsou & Hsueh 2010; Tsou, et.al. 2010; Lazarowska 2015), and very few studies have explicitly dealt with many-to-many situations, where they simultaneously control multiple ships that encountered with each other. Although notable exceptions are the work on evolutionary algorithm for computing multi-ship trajectories avoiding collisions (Szlapczynski 2011; Szlapczynski 2015), it is a centralized algorithm that we believe become low performance if the number of encountering ships increases.

On the other hand, by modeling ships as autonomous agents, a series of distributed ship collision avoidance algorithms are recently provided (Kim, et.al. 2014; Kim et.al. 2015; Kim et.al. 2017). The latest version of it, called DSSA, runs the distributed stochastic search algorithm (Zhang, et.al. 2005) in order to change courses of ships (while keeping their individual current speeds).

3 DISTRIBUTED SCA

3.1 *Framework*

As shown in Figure 1, distributed ship collision avoidance consists of two phases, which we call the *control* phase and the *search* phase. We assume that all ships iterate these two phases simultaneously. One iteration of these phases is called one *time step*.

In the control phase, a ship, who does not reach her goal, moves directly to the next position if she finds no ship in her *detection range*. If she finds some other ships in her detection range, they will shift into the search phase.

In the search phase, several ships try to avoid collisions by running a distributed algorithm. If all ships find collision-free courses by that algorithm, or if computation time exceeds a certain time limit, they update the next positions based on the courses they found, and will move there for the next time step.

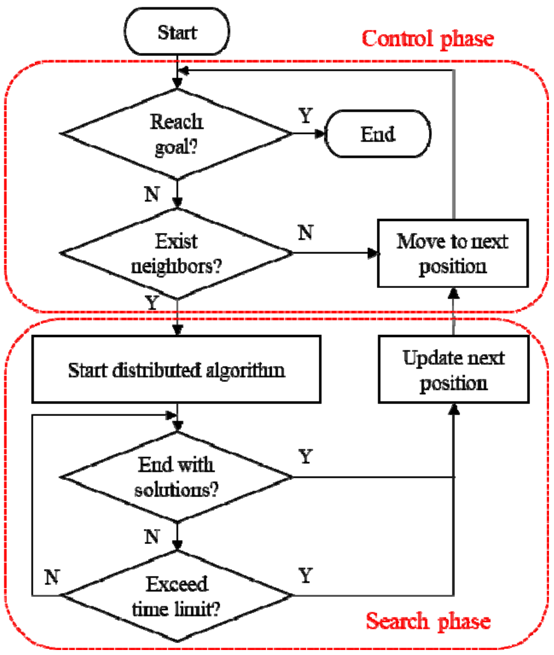

Figure 1. Framework of Distributed SCA

3.2 *Cost and Improvement*

In a distributed algorithm, we assume that ships can exchange *intentions* using a communication system such as AIS. An *intention* of a ship in this context is the course which will be selected by the ship at the time point after one time step.

When receiving current positions, courses (headings), speeds, and intentions of neighboring ships, one ship (called *self* hereafter) will compute the costs of current and every possible courses by using the following formula:

$$Cost_{self}\left(crs\right) \equiv \sum_{j \in Neighbors} CR_{self}\left(crs, j\right) + EF_{self}\left(crs\right),$$

where

$$CR_{self}\left(crs, j\right) \equiv \begin{cases} \dfrac{TimeWindow}{TCPA\left(crs, j\right)} & \text{if } self \text{ will collide with } j, \\ 0 & otherwise, \end{cases}$$

$$EF_{self}\left(crs\right) \equiv \alpha \cdot \frac{\left|\theta_{dest} - \theta\left(crs\right)\right|}{180^{\circ}},$$

$$crs \in Dom = \left\{-45^{\circ}, -40^{\circ}, \ldots, -5^{\circ}, 0^{\circ}, +5^{\circ}, \ldots, +40^{\circ}, +45^{\circ}, crs_{dest}\right\}$$

The variable *crs* indicates a relative angle to current ship heading, which can take not only any angle from -45 degree (the leftmost) to $+45$ degree (the rightmost) at a step of 5 degree, but also a special relative angle crs_{dest} that leads directly to the destination of *self* only if it is within -45 degree and $+45$ degree. Note that 0 degree of *crs* indicates a current heading of *self*.

$CR_{self}(crs, j)$ computes a "collision risk" against ship *j* when *self* changes her heading to *crs*. *TimeWindow* is a constant length of time step for each ship to predict future positions based on the intentions of herself and her neighboring ships. $TCPA(crs, j)$ is the *time to closest point of approach* against ship *j* when *self* changes her heading to *crs* and ship *j* assumes to follow the received tentative intention. Intuitively, the closer the time when *self* will collide with ship *j*, the more the value of $CR_{self}(crs, j)$ becomes.

On the other hand, $EF_{self}(crs)$ computes "inefficiency" when *self* changes her heading to *crs*. θ_{dest} is an absolute angle for the destination of *self* and $\theta(crs)$ is an absolute angle for *crs*. Intuitively, the closer the heading taken by *self* is to her destination course, the smaller the value of $EF_{self}(crs)$ becomes. Note that a balance on the impact to the cost between collision risk and inefficiency can be controlled by a value of weighting factor α.

In a distributed algorithm, the ships try to decide their own intentions (the course which will be selected at next time step) while exchanging their tentative intentions. As her tentative intention, *self* will try to select the course that achieves the maximum cost reduction all the time. We thus define the following variables:

$$improvement_{self} \equiv \max_{crs \in Dom} \{Cost_{self}(0) - Cost_{self}(crs)\}$$

$$intention_{self} \equiv \operatorname*{argmax}_{crs \in Dom} \{Cost_{self}(0) - Cost_{self}(crs)\}$$

$Cost_{self}(0)$ is the cost of a current heading of *self* and $Cost_{self}(crs)$ is the cost when *self* changes her heading to *crs*. Therefore, $improvement_{self}$ is nothing other than the maximum cost reduction while $intention_{self}$ is the course that achieves that maximum cost reduction. We need to emphasize that the values of these variables depend on tentative intentions of neighboring ships. This is because a value of $TCPA(crs, j)$ in computing collision risk against ship *j* may vary depending on the tentative intention of ship *j*.

By allowing the ships to exchange these tentative intentions through a communication system such as AIS, we have been considering that we can construct various concrete distributed ship collision avoidance algorithms. Among those presented so far, its latest version is called Distributed Stochastic Search Algorithm (DSSA).

3.3 *DSSA*

DSSA is one instantiation of distributed stochastic search (Zhang, et.al. 2005) in the context of ship collision avoidance. The technique of distributed stochastic search was originally proposed to solve the Distributed Constraint Optimization Problem (DCOP), which is a general model for distributed problem solving. This technique is quite simple but powerful, and has been applied to solve various distributed problem, such as the scheduling problem on distributed sensor networks (Zhang, et.al. 2005).

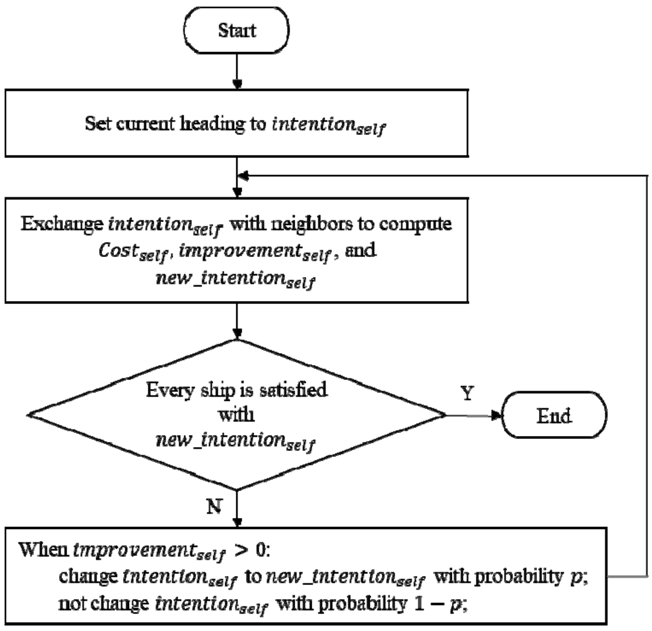
Start
Set current heading to $intention_{self}$
Exchange $intention_{self}$ with neighbors to compute $Cost_{self}$, $improvement_{self}$, and $new\_intention_{self}$
Every ship is satisfied with $new\_intention_{self}$
Y
End
N
When $improvement_{self} > 0$:
change $intention_{self}$ to $new\_intention_{self}$ with probability p;
not change $intention_{self}$ with probability $1-p$;

Figure 2. Flowchart of DSSA from a global viewpoint

Following the scheme of distributed stochastic search, we have built DSSA for ship collision avoidance, whose flowchart from a "global" viewpoint is shown in Figure 2. In DSSA, every ship (*self*) first sets her current heading to her own intention. After exchanging their intentions with neighboring ships, *self* computes $Cost_{self}$, $improvement_{self}$, and $new\_intention_{self}$ based on those exchanged intentions. We should note here that $intention_{self}$ is not yet overwritten by $new\_intention_{self}$ just computed now. *self* is satisfied with her current $intention_{self}$ if the maximum cost reduction ($improvement_{self}$) is equal to 0, and furthermore, the system naturally reaches to a quiescence if all of the ships are satisfied with their own intentions. On the other hand, if any of the ships has a positive value for $improvement_{self}$, she will change her intention probabilistically. More specifically, she will overwrite her $intention_{self}$ with $new\_intention_{self}$ with probability p and will not do with probability $1-p$. With those updated intentions, the ships proceed to the next round of exchanging intentions. They repeat this process until a quiescence has been reached or a predetermined time limit has come.

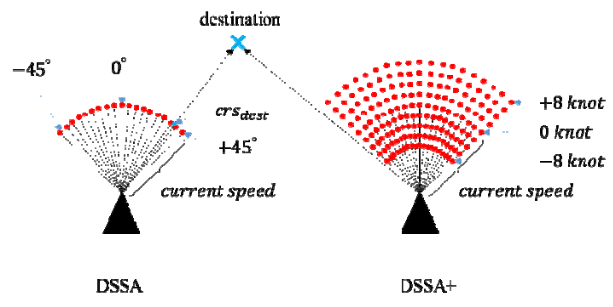

Figure 3. Domain of intentions for DSSA and DSSA+.

4 NEW ALGORITHM: DSSA+

In order to realize more sophisticated way of avoiding collisions, we extend the previous DSSA such that it can handle situations where autonomous ships can change both course and speed. We call this extended algorithm DSSA^+. The overall framework of DSSA^+ is almost the same as DSSA, but a new part is the definitions of intention and cost function.

An intention of a ship in DSSA^+ is represented by a pair of course (denoted by *crs*) and speed change (denoted by *spc*), each of which will be selected by the ship at the time point after one time step. The course is the same as that of DSSA, but the speed change is the speed change relative to the current speed, and supposed to take a value from -8 knot to $+8$ knot in a step of 2 knot. A domain of values that can be selected by the ship in DSSA and DSSA^+ as her intention is conceptually illustrated in the left and right of Figure 3, respectively. Each dot represents a possible value that a ship can take as her intention. Note that the number of possible intentions for DSSA is at most 20, but that for DSSA^+ may increase up to 180.

The cost of any possible pair (crs, spc) of course and speed change can be computed by the following formula:

$$Cost_{self}(crs, spc) \equiv \sum_{j \in Neighbors} CR_{self}(crs, spc, j) + EF_{self}(crs, spc),$$

where

$$CR_{self}(crs, spc, j) \equiv \begin{cases} \dfrac{TimeWindow}{TCPA(crs, spc, j)} & \text{if } self \text{ will collide with } j, \\ 0 & otherwise, \end{cases}$$

$$EF_{self}(crs, spc) \equiv \alpha \cdot \frac{|\theta_{dest} - \theta(crs)|}{180^\circ} + \beta \cdot \frac{|\min\{\max\{CurSPD + spc, MinSPD\}, MaxSPD\} - RefSPD|}{MaxSPD},$$

$$(crs, spc) \in Dom = \{(-45^\circ, -8kt), \ldots, (-45^\circ, +8kt), (+45^\circ, -8kt), \ldots, (+45^\circ, +8kt), (crs_{dest}, -8kt), \ldots, (crs_{dest}, +8kt)\}.$$

$CR_{self}(crs, spc, j)$ computes a "collision risk" against ship *j* when *self* changes her heading to *crs* and her speed by *spc*. $TCPA(crs, spc, j)$ is the *time to closest point of approach* against ship *j* when *self* changes her heading to *crs* and her speed by spc while ship *j* assumes to follow the received tentative intention (see Appendix). *TimeWindow* is the same as that of DSSA. As mentioned earlier, intuitively, the closer the time when *self* will collide with ship *j*, the more the value of $CR_{self}(crs, spc, j)$ becomes.

On the other hand, $EF_{self}(crs, spc)$ computes "inefficiency" when *self* changes her heading to *crs* and her speed by *spc*. Unlike DSSA, it consists of two terms. The first term calculates inefficiency about the course, which is actually the same as that of DSSA. The second term, which is newly introduced in DSSA^+, calculates inefficiency about the speed. The value of this second term becomes smaller as the speed of *self* after a given speed change *spc*, is closer to the reference speed of *self*. The reference speed, denoted as *RefSPD*, is the most desirable speed for *self*, which may depend on ship types in general. Also, *MaxSPD* and *MinSPD* are the maximum and minimum speed of *self*, respectively, which may again depend on ship types. These are introduced to ensure that the current speed of *self*, denoted as *CurSPD*, becomes not less than *MinSPD* and not more than *MaxSPD* after speed change *spc* is applied.

The values of these first and second terms in $EF_{self}(crs, spc)$ are weighted by the parameters α and β, respectively. When α is larger than β, *self* gets more likely to select relatively the speed change rather than the course change in order to avoid collisions. Conversely, when α is smaller than β, *self* gets more likely to select the course change relatively rather than the speed change to avoid collisions.

The flowchart of DSSA^+ is the same as that of DSSA illustrated in Figure 2, except that $intention_{self}$ is instantiated with a pair of course and speed change not with just course.

5 EXPERIMENT

We compare the performance of DSSA and DSSA^+ by simulation in a 800 × 800 two-dimensional space. Throughout these experiments, we set the probability p in Figure 2 to be 0.8 both in DSSA and DSSA^+. The radius of *detection range* of each ship was set to 500, in which the ship is assumed to be able to exchange intentions with other ships. *TimeWindow*, a constant length of time step for each ship to predict future positions, was fixed to 20. In this experiment, the following three versions of DSSA^+ were tried.

Table 1. Experimental results for two-ship instance

	DSSA	$\mathrm{DSSA}^{+}_{s>c}$	$\mathrm{DSSA}^{+}_{s<c}$	$\mathrm{DSSA}^{+}_{s=c}$
Total length of paths	1150.0	**985.5**	1154.0	1154.0
Travel time of each ship	23.0	**22.7**	25.0	25.0
Detour rate of each ship	1.17	**1.00**	1.17	1.17
Success rate	**20/20**	**20/20**	**20/20**	**20/20**

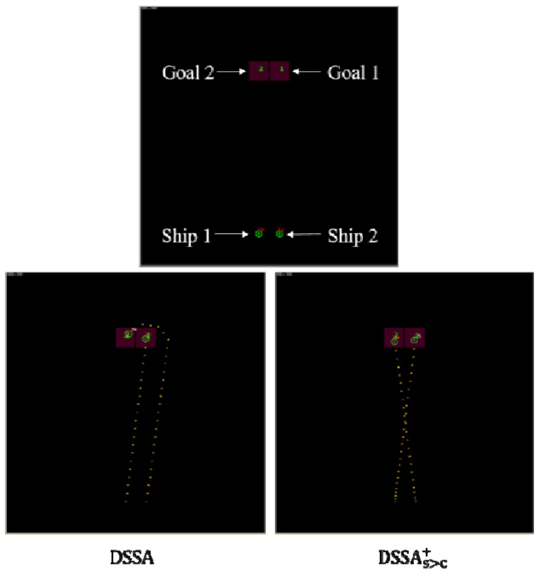

Figure 4. Trajectories for two ships pushing each other

$\mathbf{DSSA}^{+}_{\mathbf{s>c}}$: the version where, for all of the ships, the value of α in $EF_{self}\left(crs, spc\right)$ was set to be 10 times higher than the value of β, by doing such, the speed change is more likely to be selected than the course change.

$\mathbf{DSSA}^{+}_{\mathbf{s<c}}$: the version where, for all of the ships, the value of α in $EF_{self}\left(crs, spc\right)$ was set to be 10 times lower than the value of β, by doing such, the course change is more likely to be selected than the speed change.

$\mathbf{DSSA}^{+}_{\mathbf{s=c}}$: the version where, for all of the ship, the value of α in $EF_{self}\left(crs, spc\right)$ was set to be the same as the value of β.

5.1 *Two Ships Pushing Each Other*

In order to provide an example clearly showing the merit to consider speed change to avoid collisions, we made a simple experiment on two ships going in parallel and heading towards the destinations on the other side to each other. Typical trajectories of these two ships generated by DSSA and $\mathrm{DSSA}^{+}_{s>c}$ are shown in Figure 4. Note that in $\mathrm{DSSA}^{+}_{s>c}$ the reference speeds of two ships were the same as 25kt and their speeds are changeable, but in DSSA the speeds of two ships were exactly the same as 25kt at all times.

Obviously, $\mathrm{DSSA}^{+}_{s>c}$ generates a more natural trajectory. In $\mathrm{DSSA}^{+}_{s>c}$, one of the ships autonomously slows down her speed and lets the other ship go first before heading to her own destination. On the other hand, in DSSA, as a result of both ships attempting to avoid collisions by only changing the course without slowing down their speeds, one ship is pushed out in the direction opposite to her own destination, resulting in a large detour (see the trajectory of Ship 2 in the left-bottom of Figure 4).

Since DSSA and DSSA^{+} are stochastic algorithms, we may get totally different results even when we try each of them on the same problem instance. Table 1 shows some statistical data on the average performance of these DSSA^{+} and DSSA over 20 trials on this two-ship example. The detour rate is computed for each ship by dividing the distance actually traveled from her initial position to goal position by the shortest distance therebetween. As can be seen from Table 1, in $\mathrm{DSSA}^{+}_{s>c}$, each ship never detours, and both total length of paths and travel time of each ship are minimized. However, for DSSA^{+} with different weight values, it is somewhat worse than the conventional DSSA in terms of both total length of paths and travel time of each ship.

5.2 *16 Ships Crossing Each Other*

In the next experiment, we assume that 16 ships, each with a reference speed of 25kt, are approaching each other from upper, lower, left and right sides and orthogonally try to cross each other towards their individual destinations. This situation is depicted in the top of Figure 5. Typical trajectories generated from some successful runs by DSSA and $\mathrm{DSSA}^{+}_{s>c}$ are shown in the left- and right-bottom of Figure 5, respectively. As can be seen from these figures, with DSSA, some ships may detour too much and go outside the area surrounded by 16 ships to avoid collision, but such a thing never happens with $\mathrm{DSSA}^{+}_{s>c}$.

We have conducted 50 random trials for each of DSSA and DSSA^{+} on this problem instance. Table 2 shows the average of the total length of paths by all ships, the average of travel time of each ship, the average detour rate of each ship, and the success rate out of 50 trials. The average value is calculated only on the results of successful trials. Generally, with DSSA^{+} , the ships can reach their respective destinations without collisions by properly adjusting the speed while taking more time on their shorter paths.

Table 2. Experimental results for 16-ship instance

	DSSA	$\mathrm{DSSA}^{+}_{s>c}$	$\mathrm{DSSA}^{+}_{s<c}$	$\mathrm{DSSA}^{+}_{s=c}$
Total length of paths	9833.6	**8693.4**	9816.9	9096.4
Travel time of each ship	**24.6**	33.2	26.5	25.3
Detour rate of each ship	1.14	**1.01**	1.15	1.06
Success rate	32/50	**50/50**	47/50	49/50

Table 3. Experimental results for 3-ship instance

	DSSA	$\mathrm{DSSA}^{+}_{s>c}$	$\mathrm{DSSA}^{+}_{s<c}$	$\mathrm{DSSA}^{+}_{s=c}$
Total length of paths	1709.0	**1702.0**	1713.5	1705.9
Travel time of each ship	49.6	59.3	51.4	**49.5**
Detour rate of each ship	1.02	1.01	1.02	**1.01**
Success rate	**20/20**	1/20	**20/20**	17/20

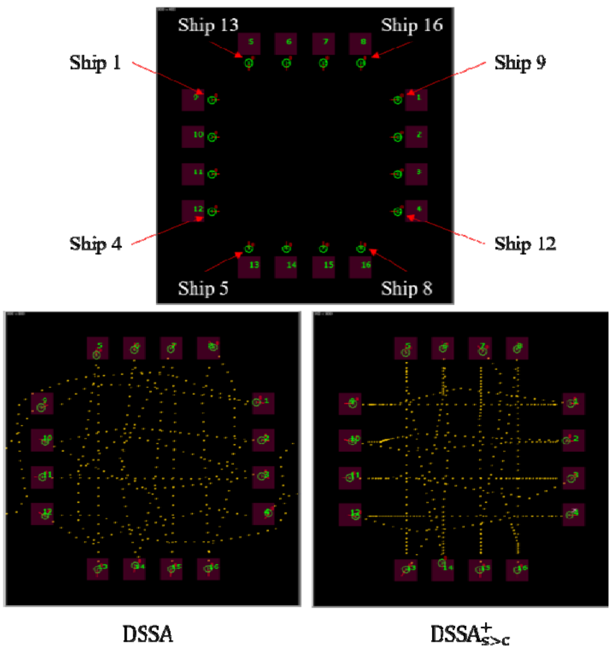

Figure 5. Trajectories for 16 ships crossing each other

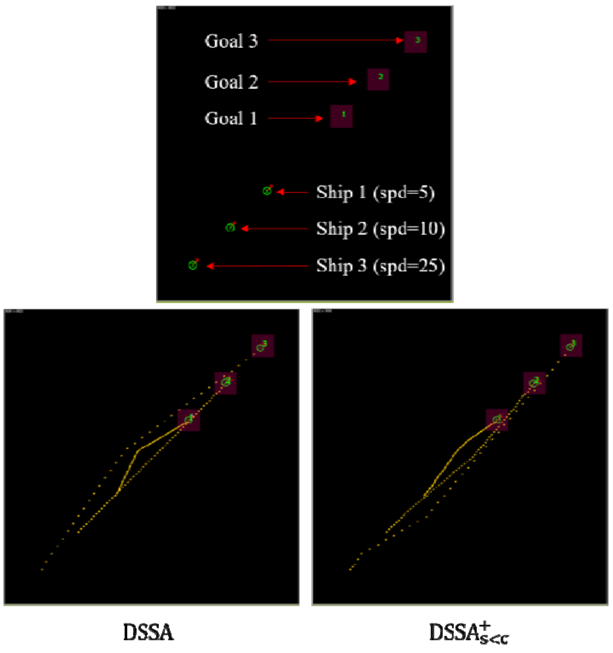

Figure 6. Trajectories for three ships overtaking or being overtaken

5.3 *Three Ships Overtaking or Being Overtaken*

So far, when DSSA^{+} is applied to some specific scenarios, it has been observed that it seems to be more effective to put more emphasis on speed change rather than course change. However, that is certainly a misleading discussion. Indeed, there is also a counterexample in which course change is more effective than speed change.

Suppose that three ships with different reference speeds (5, 10, and 25kt) are lined up on the straight line in ascending order of reference speed and each goes in the same direction towards her destination that is on the extension of that straight line. Also assume that the destination of the slowest ship at the head is the closest and conversely the destination of the last fastest ship is the furthest. This situation is illustrated in the top of Figure 6. Typical trajectories generated by DSSA and $\mathrm{DSSA}^{+}_{s<c}$ are shown in the left- and right-bottom of Figure 6, respectively. Note that the right-bottom of Figure 6 is not the trajectory of $\mathrm{DSSA}^{+}_{s>c}$, but the trajectory of $\mathrm{DSSA}^{+}_{s<c}$ that is more likely to change the course than the speed. As a matter of fact, using $\mathrm{DSSA}^{+}_{s>c}$ that is more likely to change the speed than the course will cause collisions in most of the trials on this example. The reason is quite simple. It is impossible to avoid collisions with speed changes alone on this example in the first place.

We have conducted 20 random trials for each of DSSA and DSSA^{+} on this problem instance to confirm this fact. Table 3 shows their results. Certainly the performance of $\mathrm{DSSA}^{+}_{s>c}$ on this example is clearly deteriorating. On the other hand, I would like to point out the performance of $\mathrm{DSSA}^{+}_{s<c}$ is still comparable to that of conventional DSSA. This fact indicates that, in applying DSSA^{+}, which of the course change or the speed change should be weighed may greatly depends on the situation. To overcome this issue technically, we need some new principle to dynamically adjust the values of α and β that appear in the "inefficiency" term $EF_{self}(crs, spc)$ of our cost function. We consider that it would belong to our future work.

As a final note regarding our experiments, in (Kim, et.al. 2017), experimental results with actual data obtained from AIS have been reported, but in this paper we have not dared to do such an experiment. This is because real data obtained from AIS is generally rather easy for both DSSA and DSSA^{+}, so it will not be a very interesting comparison.

6 CONCLUSION

Automatic ship collision avoidance is one of the key technologies in the automation of ship operation which has received lots of attention in recent years. Although several computational methodologies for realizing automatic ship collision avoidance have been proposed, most of them deal with one-on-one or one-to-many situations. We consider that these methodologies will face a difficulty in so-called many-to-many situations, where all of the ships work with that methodology. Kim and colleague has recently proposed a series of distributed ship collision avoidance algorithms, in which a group of ships are modeled by a multi-agent system where autonomous agents exchange their intentions in advance to decide and perform their next actions. In these algorithm, it is assumed that ships would only change the course while keeping the speed. In this paper, in consideration of recent progress of vessel maneuvering technology and the necessity to more effectively avoid collision, we have presented DSSA^{+} in which ships can change both course and speed.

As a result of the experiment, the merit of introducing not only the course change but also the speed change as an option of possible actions was demonstrated. On the other hand, however, there was no single answer as to whether to set priority on the course change or the speed change to effectively avoid the collision, it turned out to depend on the situation. As our future work, we are considering to design a function that, when a situation around some ship is given as input, is capable of returning appropriate weighting over the options of the course change and the speed change.

REFERENCES

Hu, Q., Yang. C., Chen. H. & Xiao, B. 2008. Planned Route Based Negotiation for Collision Avoidance between Vessels. TransNav, the International Journal on Marine Navigation and Safety of Sea Transportation 2(4): 363-368.

Kim, D., Hirayama, K., & Park, G. 2014. Collision Avoidance in Multiple-ship Situations by Distributed Local Search. Journal of Advanced Computational Intelligence and Intelligent Informatics 18(5): 839-848.

Kim, D., Hirayama, K., & Okimoto, T. 2015. Ship Collision Avoidance by Distributed Tabu Search. TransNav, the International Journal on Marine Navigation and Safety of Sea Transportation 9(1): 23-29.

Kim, D., Hirayama, K., & Okimoto, T. 2017. Distributed Stochastic Search Algorithm for Multi-ship Encounter Situations. The Journal of Navigation 70(4): 699-718.

Lamb, W.G.P. & Hunt, J.M. 1995. Multiple Crossing Encounters. The Journal of Navigation 48(1): 105-113.

Lamb, W.G.P. & Hunt, J.M. 2000. Multiple Encounter Avoidance Manoeuvres. The Journal of Navigation, 53(1): 181-186.

Lazarowska, A. 2015. Ship's Trajectory Planning for Collision Avoidance at Sea Based on Ant Colony Optimisation. The Journal of Navigation 68(2): 291-307.

Lee, S., Kwon, K. & Joh, J. 2004. A Fuzzy Logic for Autonomous Navigation of Marine Vehicles Satisfying COLREG Guidelines. International Journal of Control, Automation and Systems, 2(2): 171-181.

Szlapczynski, R. 2011. Evolutionary Sets of Safe Ship Trajectories: A New Approach to Collision Avoidance. The Journal of Navigation 64(1): 169-181.

Szlapczynski, R. 2015. Evolutionary Planning of Safe Ship Tracks in Restricted Visibility. The Journal of Navigation 68(1): 39-51.

Tsou, M. & Hsueh, C. 2010. The Study of Ship Collision Avoidance Route Planning by Ant Colony Algorithm. Journal of Marine Science and Technology 18(5): 746-756.

Tsou, M., Kao, S. & Su, C. 2010. Decision Support from Genetic Algorithms for Ship Collision Avoidance Route Planning and Alerts. The Journal of Navigation 63(1): 167-182.

Zhang, W., Wand G., Xing, Z. & Wittenburg, L. 2005. Distributed Stochastic Search and Distributed Breakout: Properties, Comparison and Applications to Constraint Optimization Problem in Sensor Networks. Artificial Intelligence 161(1–2): 55–87.

APPENDIX

We show the formula to compute TCPA against ship j, from which *self* already received her intention, when *self* selects its own course change crs and speed change spc. Let the current position of *self* be $\left(x_0^{self}, y_0^{self}\right)$ and let the position after advancing by *TimeWindow* with crs and spc be $\left(x_{tw}^{self}, y_{tw}^{self}\right)$. Also, let the current position of ship j be $\left(x_0^j, y_0^j\right)$ and let the position of ship j after advancing by *TimeWindow* with the recieved intention be $\left(x_{tw}^j, y_{tw}^j\right)$. In addition, we introduce the following four new variables that can calculate their values from the above coordinates:

$$X_1 \equiv x_0^{self} - x_0^j,\ X_2 \equiv \left(x_{tw}^{self} - x_0^{self}\right) - \left(x_{tw}^j - x_0^j\right),$$
$$Y_1 \equiv y_0^{self} - y_0^j,\ Y_2 \equiv \left(y_{tw}^{self} - y_0^{self}\right) - \left(y_{tw}^j - y_0^j\right).$$

A set of rules to compute TCPA between *self* and ship j is as follows.

If $X_2^2 + Y_2^2 > 0$ and $X_2^2 + Y_2^2 + X_1X_2 + Y_1Y_2 \le 0$ then $TCPA\left(crs, spc, j\right) = TimeWindow$.

If $X_2^2 + Y_2^2 > 0$ and $X_1X_2 + Y_1Y_2 \ge 0$ then $TCPA\left(crs, spc, j\right) = 0$.

If $X_2^2 + Y_2^2 > 0$ and $X_1X_2 + Y_1Y_2 < 0$ and $X_2^2 + Y_2^2 + X_1X_2 + Y_1Y_2 > 0$ then $TCPA\left(crs, spc, j\right) = -\dfrac{X_1X_2 + Y_1Y_2}{X_2^2 + Y_2^2} \times TimeWindow$.

If $X_2^2 + Y_2^2 = 0$ and $X_1X_2 + Y_1Y_2 > 0$ then $TCPA\left(crs, spc, j\right) = 0$.

If $X_2^2 + Y_2^2 = 0$ and $X_1X_2 + Y_1Y_2 \le 0$ then $TCPA\left(crs, spc, j\right) = TimeWindow$.

In order to compute DCPA, *distance at the closest point of approach*, between *self* and ship j, simply calculate it by substituting $\frac{TCPA(crs, spc, j)}{TimeWindow}$ for variable t in the following formula:

$$DCPA(t) \equiv \sqrt{X_1^2 + Y_1^2 + 2\left(X_1X_2 + Y_1Y_2\right)t + \left(X_2^2 + Y_2^2\right)t^2}.$$

When the value of this DCPA is sufficiently small, we consider that *self* and ship j will collide within *TimeWindow*.