Multi-agent Planning for Ship Collision Avoidance

Yuhong Liu
Merchant Marine College
Shanghai Maritime University
Pudong Avenue 1500, Shanghai, China
yhliu@cen.shmtu.edu.cn

Chunsheng Yang
Institute for Information Technology
National Research Council
Ottawa, ON K1A 0R6,Canada
Chunsheng.Yang@nrc-cnrc.gc.ca

Xuanmin Du
Shanghai Marine Electronic
Equipment Research Institute
Jindu Road 1500, Shanghai, China
duxuanmin@21cn.com

***Abstract*—Multi-agent based planning techniques have been widely applied to various decision-making support applications. In this paper, we investigate how to apply multi-agent planning to collision avoidance in ship navigation. We have developed three multi-agent-based planning algorithms: the independent planning for self-benefit purpose, the centralized planning for union-benefit purpose and the negotiation-based planning for mutual-benefit purpose. Having introduced collision avoidance planning, we present the developed planning algorithm in detail. We also report the experiments and some results. The experimental results illustrate the feasibility and validity of the multi-agent planning for collision avoidance.**

***Keywords*—multi-agent planning, collision avoidance, coordination, negotiation**

I. INTRODUCTION

To improve the safety of ship navigation, many research efforts have been focused on developing intelligent collision avoidance systems such as expert systems and decision-making support systems [1, 2, 3, 4, 5, 6, 7]. However, few of such systems can be applied to a real navigation. The main reason is that those systems were developed for making decision on collision avoidance for a single ship or own ship other than a group of ships. In the real navigation, an effective solution of collision avoidance involves the coordination, cooperation, and interaction among ships and human behaviors. One way to incorporate such coordination or interaction among multiple ships into decision-making processes is to apply multi-agent techniques to collision avoidance. Multi-agent system (MAS) is emerging as an important software model for next generation computing problems that deal with distributed tasks in dynamic and heterogeneous environments and interested in the behavior of a set of autonomous agents solving a given problem. Within a multi-agent system, agents are able to solve problems either on their own requirements or on cooperation among ships. A ship navigating at sea can be looked as a rational and intelligent agent in MAS-based decision-making support systems, for a ship has the characteristics of personality, reactivity, adaptability, and autonomy that an agent might have. We have developed an MAS-based collision avoidance system for ship navigation using MAS techniques [8]. We focused on the design and implementation of the system architecture, agent BDI structure, and agent communication mechanism. In MAS-based collision avoidance systems, collision avoidance planning is a principle component.

Planning in collision avoidance is to establish an effective and safe ship-handling procedure or plan that is a sequence of actions for achieving a collision avoidance goal. Multi-agent planning is a useful approach for collision avoidance planning. Several techniques are available, including distributed planning, centralized planning, plan reconciliation, and organizational analysis [9, 10, 11, 12]. We have developed three planning algorithms: the independent planning for self-benefit purpose, the centralized planning for union-benefit purpose and the negotiation-based planning for mutual-benefit purpose. These algorithms are developed for either a single ship's planning, or a group of ships' planning on collision avoidance. In this paper, focusing on the multi-agent planning for collision avoidance, we present the developed algorithms and experimental results.

In Section 2, we describe collision avoidance plans; in Section 3, we present the developed multi-agent-based planning algorithms and give a brief overview of a multi-agent system; in Section 4, we report the experiments and results; and the final Section concludes the paper.

II. COLLISION AVOIDANCE PLAN

A. *The representation of a collision avoidance plan*

We describe collision avoidance plan as an actual ship-handling procedure for collision avoidance. Figure 1 shows a typical procedure between two ships (noted as Ship_AgentX). This procedure consists of four operation phases: holding, collision avoidance acting, returning to original course, and returning to scheduled route. In this work, we are focusing on the former three phases. To represent such a plan, some definitions are given as follows.

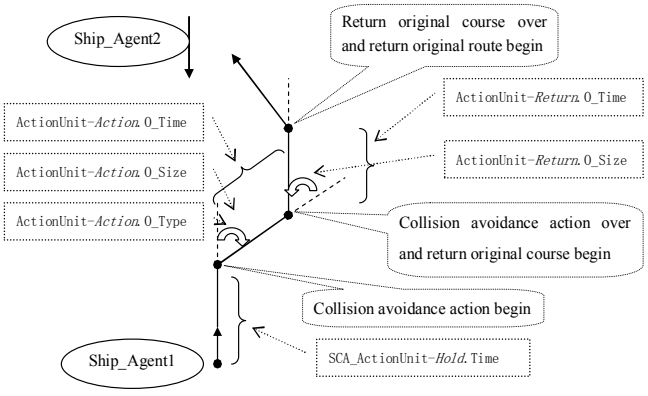


Fig.1 The typical collision avoidance process

Definition 1, Operation type (noted as O_Type) is defined as an operation for collision avoidance. Given an

encounter situation which is described using the number of encounter Ship_Agents (noted as SA_Num), encounter stage (noted as E_Stage) and encounter situation (noted as E_Situation), an operation type set (noted as Type_Set) for collision avoidance, can be denoted as.

$$O\_Type \subseteq Type\_Set := < SA\_Num, E\_Stage, E\_Situation > \quad (1)$$

Where, *Type_Set = {STANDON, TURN, TURNSTBD, TURNPORT, INCSPEED, DECSPEED, STOP}*.

Definition 2, Operation size (noted as O_Size) is defined as a range of the course changing or speed changing for a Ship_Agent.

Definition 3, Operation time (noted as O_Time) is defined as a period time for taking action for avoiding the collision given a risk threshold and the values of O_Type and O_Size. We note it as follows:

$$O\_Time := < O\_Type, O\_Size, \text{Time satisfied (Risk } | \text{ }_{<logic \in Logic\_Set>} \text{ Threshold)} > \quad (2)$$

Where Logic_Set = {>, <}.

Definition 4, Action Unit (noted as ActionUnit) is defined as a basic operation structure, consisting of O_Type, O_Size and O_Time. It is denoted as

$$ActionUnit: = < O\_Type, O\_Size, O\_Time> \quad (3)$$

Definition 5**,** Action Plan (noted as ActionPlan) is defined as an operation process which contains a series of ActionUnits, i.e.

$$ActionPlan := < ActionUnit_1 \mid ActionUnit_2 \mid \ldots \mid ActionUnit_N> \quad (4)$$

The N is the number of phase in a collision avoidance plan. *ActionUnit1* is the phase of ship holding (ActionUnit-*Hold*). During this phase, a Ship_Agent will keep its original course and speed. So, O_Type is set to STANDON; O_Size is set to "zero"; and O_Time is set to a unknown numeric value. *ActionUnit2* is the phase of collision avoidance acting (ActionUnit-*Action*). During this phase, a Ship_Agent will change its course or speed to a new one according to O_Type and O_Size. *ActionUnit3* is the phase of returning original course (ActionUnit-*Return*). During this phase, a Ship_Agent will change its course or speed back to the original one. The ActionPlan can be represented as Table I.

TABLE I
REPRESENTATION OF A COLLISION AVOIDANCE PLAN

ActionPlan := <
ActionUnit-*Hold*.O_Type = STANDON
ActionUnit-*Hold*.Size = 0
ActionUnit-*Hold*.Time = Time satisfied (Risk > Threshold -*Hold*)
ActionUnit-*Action*.O_Type = { TURNSTBD, TURNPORT, INCSPEED, DECSPEED, STOP}
ActionUnit-*Action*.O_Size = *x*
ActionUnit-*Action*.O_Time = Time satisfied (Risk < Threshold-*Action*)
ActionUnit-*Return*.O_Type = Reversed (ActionUnit-*Action*. O_Type)
ActionUnit-*Return*.O_Size = ActionUnit-*Action*.O_Size
ActionUnit-*Return*.O_Time = Time satisfied (Risk < Threshold-*Return*)
>

B. The solution space

A solution space (Ω_p) for a plan is three dimensional. The solution space of O_Type ($\Omega_{p\text{-}Type}$) consists of discrete values, and the solution space of O_Size ($\Omega_{p\text{-}Size}$) and O_Time ($\Omega_{p\text{-}Time}$) are continuous values within [*x*, *y*] and [0, *z*] respectively. Where *x*, *y* and *z* are numeric values for different O_Types (only course changing operation types are considered) and operation phases. They are shown in Table II and III.

TABLE II
DIFFERENT X AND Y FOR DIFFERENT O_TYPES.

O_Type	O_Size (°)	
	x	*y*
STANDON	0	0
TURN	-40	40
TURNSTBD	0	40
TURNPORT	-40	0

TABLE III
DIFFERENT Z FOR DIFFERENT OPERATION PHASES.

Operation phase	*z* (minute)
ActionUnit-*Hold*	10
ActionUnit-*Action*	20
ActionUnit-*Return*	30

Generally, we select the sampled discrete value for O_Size and O_Type instead of a continuous value in planning. For example, when O_Type is TURNSTBD, value domain [0, 40] of O_Size can be sampled as {5, 10, 15, 20, 25, 30, 35, 40} and the value domain [0, 20] of O_Time can be dispersed as {2, 4, …,20} with sampling every 2 minutes. For fase searching purpose, we describe a solution space as a tree shown in Figure 2. Once SA_Num, E_Stage and E_Situation are determined, collision avoidance planning will search the solution space using the deep-first searching strategy, i.e., with O_Type→O_Size→O_Time order. For a given O_Type, such as TURN, the solution space sizes of ActionUnit-*Hold*, ActionUnit-*Action* and ActionUnit-*Return* are 5, 160 and 15 respectively. The maximum solution space size will be 5×160×15=12000.

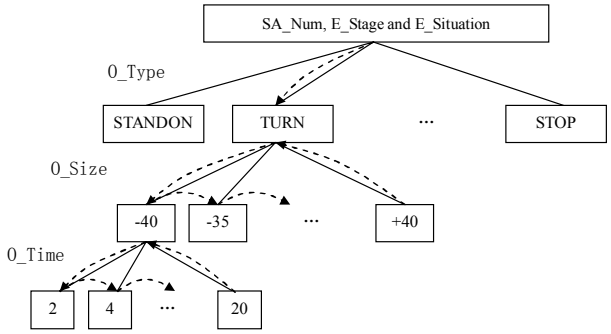


Fig. 2 The solution tree and its searching order.

C. Plan Evaluation

Plan evaluation plays an important role in collision avoidance planning. In this work, we proposed a preference function for evaluating plans. The function is used to compute a preference value for each plan based on three preference elements: an action scene preference (noted as *Scene_Pref*), an action cost preference (noted as *Cost_Pref*) and an action risk preference (noted as *Risk_Pref*).

Scene_Pref is given by navigation experts. With different encounter situations, action distances and pass types between two ships, preference values are different. For example, if two

ships are heading on with a long action distance and left side pass type, then *Scene_Pref* is given as 8.5. But if the action distance becomes short, *Scene_Pref* is given as 5.5.

Cost_Pref is a sum of the action time cost (*Time_Cost*), course changing cost (*Course_Cost*) and deviating distance cost (*Dist_Cost*) from its original route. *Time_Cost* is decided by a total action time of the three operation phases and can be computed by Equation (5). *Course_Cost* is decided by an actual course change size and an optimal course change value ($\Gamma Course$) which is given out by navigation experts. The *Course_Cost* is calculated by Equation (6) and where $\lambda = 0.7$ and $\lambda = 0.5$ for right and left turn respectively. *Dist_Cost* is determined by an actual deviating distance (*Dist*) and a deviating distance threshold ($\Gamma Dist$) which is also given out by navigation experts. It is calculated by Equation (7).

$$Time\_Cost = (\text{ActionUnit-}Hold.\text{O\_Time} + \text{ActionUnit-}Action.\text{O\_Time} + \text{ActionUnit-}Return.\text{O\_Time}) / 60 \quad (5)$$

$$Course\_Cost = 1.0 - e^{-\frac{(ActionUnit-Action.O\_Size-\Gamma Course)^2}{\lambda \times \Gamma Course^2}} \quad (6)$$

$$Dist\_Cost = \begin{cases} 0 & If\ Dist \geq \Gamma Dist \\ Dist/\Gamma Dist & if\ Dist < \Gamma Dist \end{cases} \quad (7)$$

Risk_Pref is used to measure the efficiency of the action and determined by a sum of the maximum risk (*Max_Risk*) and average risk (*Ave_Risk*) among ships while a collision avoidance action is taken.

The preference function is given in Equation (8). The larger the preference value is, the better the performance of a collision avoidance plan is. The plan with the maximal preference value is selected as an optimal plan in collision avoidance planning.

$$Preference = Scene\_Pref - Cost\_Pref - Risk\_Pref \quad (8)$$

III. Multi-agent Planning Algorithms

Before presenting the algorithms, we introduce briefly a multi-agent-based decision-making developed for ship collision avoidance.

A. Multi-agent-based systems for collision avoidance

A multi-agent-based system consists of System_Agent, Union_Agent and Ship_Agent [8, 13, 14, 15]. A ship is modeled as a Ship_Agent with agent characteristics such as perception, memory, communication, emotion, action, learning, and thinking. A System_Agent controls systems and manages the Ship_Agents such as Ship_Agent life cycle management. When risk is detected, several Ship_Agents are involved automatically to form a "dynamic society", which is called a union in our system, for working together. A Union_Agent assigned to a union manages the union data and makes union plans. In practice, a Union_Agent may be taken on by VTS (Vessel Traffic System). The basic architecture of the system is shown in Figure 3.

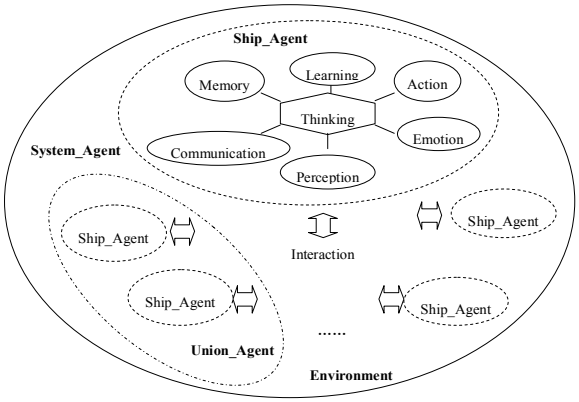


Fig.3 The architecture of multi-agent-based systems

Based on the architecture of multi-agent-based systems and the characteristics of a Ship_Agent, we developed three algorithms using multi-agent planning techniques [16].

B. Algorithm 1: independent planning with self-benefit purpose

Independent planning for self-benefit purpose is developed for each individual ship, which plans its own collision avoidance plan and executes the plan without any communication and coordination with other ships. From the viewpoint of multi-agent planning, this algorithm is fully distributed. In this algorithm, a Ship_Agent searches its own solution space for feasible plans, and sort them with their preference value. The algorithm is shown in Table IV.

TABLE IV
ALGORITHM 1: INDEPENDENT PLANNING FOR SELF-BENEFIT PURPOSE

Input: the basic information of each Ship_Agent, solution space (*S*)
Output: the plans in the solution queue S_p.

```
For each Ship_Agent do
    Sp ← Φ;
    ActionUnit-Hold.O_Time =SearchUnitHold( START_RISK_LEV, S);
    If !(∃ ActionUnit-Hold.O_Time∈ Ωp-Time), then end this planning.
    Else (∃ ActionUnit-Hold.O_Time∈ Ωp-Time)
        ActionUnit-Action.O_Type ⇐<SA_Num, E_Stage,
E_Situation>;
        While Ωp-Size != Φ do
            ActionUnit-Action.O_Size ← v_Size∈ Ωp-Size ;
            Delete v_Size from Ωp-Size ;
            ActionUnit-Action.O_Time = SearchUnitAction
              ( MIN_RISK_LEV,S);
            If ∃ ActionUnit-Action.O_Time∈ Ωp-Time then
              ActionUnit-Return.O_Time = Search UnitReturn
              (SED_RISK_LEV, S);
                Sp ← ActionPlan;
            End
        End
    End
    PlanEvaluation(Sp); // calculating preference value for each plan and
                        sorting plans
End
Return Sp.
```

The algorithm involves the following steps. First, each Ship_Agent searches the ActionUnit-*Hold*.O_Time. Second, the algorithm determines ActionUnit-*Action*.O_Type. For each ActionUnit-*Action*.O_Size value, the algorithm only keeps the least ActionUnit-*Action*.O_Time value that satisfied the

threshold. If all three parameters of ActionUnit-*Action* are found, then the ActionUnit-*Return*.O_Time will be searched. After that, a potential collision avoidance plan is created and stored in the solution queue. Finally, the plans in the solution queue are evaluated and sorted according to their preference values. The plan generated for a Ship_Agent may be conflict or inconsistent because there is no communication and conflict checking among Ship_Agents.

C. Algorithm 2: centralized planning with union-benefit purpose

With the development of communication techniques, many ships can communicate each other or with VTS. For such practice, a centralized-based algorithm for collision avoidance planning is more efficient. Such centralized planning algorithm could benefit a group of ship (so-called union). In a union, a Union-Agent makes collision avoidance plan and distributes it to each Ship_Agent in the union. The pans are executed parallel among Ship_Agents. Some communication and coordination among Ship_Agents are necessary for plan execution and distribution. Table V shows the details of Algorithm 2.

TABLE V
ALGORITHM 2: CENTRALIZED PLANNING FOR UNION-BENEFIT PURPOSE

Input: basic information of each Ship_Agents in the union, solution space (*S*)
Output: the plans in the solution queue S_p

For each Union_Agent **do**
  $S_p \leftarrow \Phi$;
  DetermineUnionSolutionSspace ($\Omega_{p\text{-}TypeSzie}$, O_Type, O_Size);
  While $\Omega_{p\text{-}TypeSize} != \Phi$ **do**
    For $\forall$ Ship_Agent $\in$ union **do**
      <ActionUnit-*Action*.O_Type, ActionUnit-*Action*.O_Size> $\leftarrow$ <*v_Type, v_Size*>$\in \Omega_{p\text{-}TypeSzie}$
    End
  ActionUnit-*Action*.O_Time = Search UnitAction (MIN_RISK_LEV);
  If $\exists$ ActionUnit-*Action*.O_Time$\in \Omega_{p\text{-}Time}$ **then** $S_p \leftarrow$ ActionPlan **End**
  End
  PlanEvaluationAndDistribution(S_p); // calculating preference value and distributing plans
End
Return S_p.

The algorithm 2 consists of the several steps. First step is to create a union solution spaces by combining all solution space from each Ship_agent. Second step is to search potential solution from the ActionUnit-*Action* operation phase. For efficiency, we only search spaces of O_TYPE and O_TIME. For all possible O_TYPE and O_TIME, we create an action plan and add it to the queue. The last step is to evaluate each plan by computing their preference. A plan with maximal preference is selected and assigned to each Ship_Agent as the collision avoidance plan.

D. Algorithm 3: negotiation-based planning with mutual-benefit purpose

To overcome the limitation of Algorithm 1, we developed a negotiation-based planning algorithm with mutual-benefit purpose. Relying on the communication among ships, negotiation is to form a joint decision and make each ship reach an individual goal. Generally, the negotiation process consists of several negotiation cycles. Each cycle has an initiator and one or more responders, the former will propose its opinion (a selected plan) while the latter will respond on the proposals, accepting or rejecting the plan. There are two kinds of management strategies: an initiator selects and sends the next plan in its solution queue (strategy 1), and one of the responders is chosen as a new initiator for sending its proposed plan (strategy 2). The algorithm with these two strategies is described in Table VI.

TABLE VI
ALGORITHM 3: NEGOTIATION-BASED PLANNING FOR MUTUAL-BENEFIT PURPOSE

Input: S_p from algorithm 1.
Output: negotiation true (with the consistent multiagent plan P_c) or negotiation false.

Initiator $\leftarrow$ Ship_Agent$|_{\text{max risk}} \in$ negotiation , responder$\leftarrow$ other Ship_Agents $\in$ negotiation;
For $\forall$ Ship_Agent $\in$ negotiation **do**
  If Ship_Agent = = initiator **then**
    If $\exists P_i \in S_p$, **then**
      Im_i (informing message) $\leftarrow P_i$ and delete P_i from S_p;
      S_q (sending queue) $\leftarrow Im_i$;
    Else strategy 1 **return** *false*;
      strategy 2 Initiator $\leftarrow$ Ship_Agent$|_{\text{!= initiator}}$, responder $\leftarrow$ Ship_Agent$|_{\text{= = initiator}}$
    End
    If $\exists Im_i \in S_q$ **then**
      R_q (receiving queue) of other Ship_Agents $\leftarrow Im_i$ and delete Im_i from S_q;
    End
    If $\exists Rm_i$ (responding message) $\in R_q$ **then**
      If $\forall Rm_i$ = = "Accept", **then return** *true* and $P_c \leftarrow P_i$;
      Else if *strategy*2 **then** Initiator $\leftarrow$Ship_Agent$|_{\text{!= initiator}}$, responder $\leftarrow$ Ship_Agent$|_{\text{= = initiator}}$
        End
      End
    End
  End
  If Ship_Agent = = responder **then**
    If $\exists Rm_i \in S_q$ **then**
      R_q of the initiator $\leftarrow Rm_i$ and delete Rm_i from S_q;
    End
    If $\exists Im_i \in R_q$ **then**
      $P_i \leftarrow Im_i$; Response R_i ("Accept" or "Reject") $\Leftarrow P_i$;
      $Rm_i \leftarrow R_i$; $S_q \leftarrow Rm_i$;
    End
  End
  If *strategy* 2 **then**
    If !($\exists P_i \in S_p$) for $\forall$ Ship_Agent, **then** return *false* **End**
  End
End

Algorithm 3 relies on Algorithm 1. We assume that each Ship_Agent can run Algorithm 1 to generate its own plans first. Then, Algorithm 3 will adjust these plans based on the negotiation among agents. In the end, a feasible and reasonable plan is established to benefit each ship and to avoid the inconsistence of the collision avoidance plans.

IV. EXPERIMENTS AND RESULTS

We implemented the proposed algorithms in a multi-agent-based simulation system for collision avoidance. We conducted some experiments for validating the applicability

and usefulness of these algorithms. In this section, we report some experimental results, which demonstrate how the developed algorithms work through a typical encounter situations between two ships.

Figure 4 illustrates how Algorithm 1 works for a heading on encounter situation. Each line stands for a plan for each Ship_Agent and the red one is the optimal plan. In (*a*), the planning results of each Ship_Agent are turn to right with different size and time. For Ship_Agent1, the preference of each plan is {4.001244, 4.207060, 4.376547, 4.417589, 4.289188, 4.033006, 3.741820, 3.496831} corresponding course changing degrees which vary from 5° to 40° every 5°. One plan, which has the following attributes: ActionUnit-*Hold*.O_Time=2, ActionUnit-*Action*.O_Type=TURNSTBD, ActionUnit-*Action*.O_Size=20°, ActionUnit-*Action*.O_Time=8, ActionUnit-*Return*.O_Time=2, is selected because it has the highest preference value among the plans. When two ships are closer to each other, the plans will be different as shown in (*b*). In other words, the ActionUnit-*Action*.O_Type turns out to TURN from TURNSTBD. Meanwhile, each Ship_Agent can also turn left to avoid collision. In the example (*c*), if Ship_Agent1 turns left while Ship_Agent2 turns to right at that moment, the situation will become much worse. To avoid such situation, the developed Algorithm 1 has to obey the navigation rules and traffic law at sea.

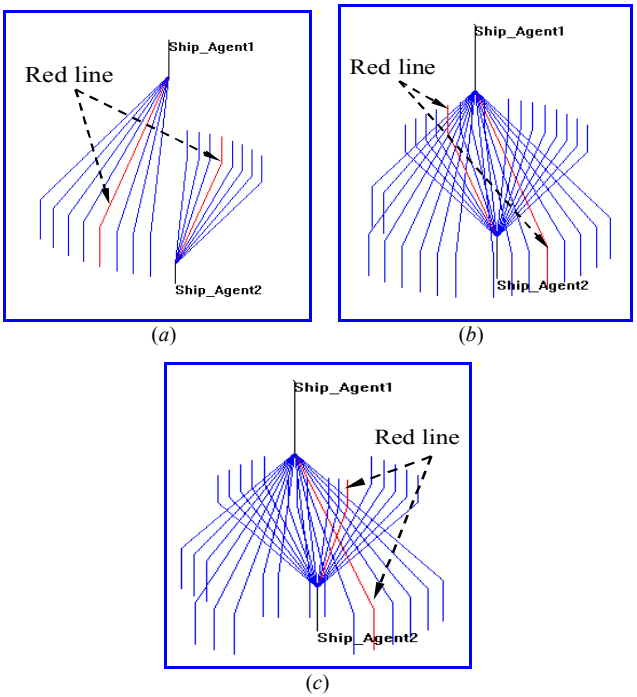


Fig. 4 The examples of collision avoidance with Algorithm 1.

Figure 5 illustrates how Algorithm 2 works in planning. Figure 5's (*a*) and (*b*)show the encounter scenario of two ships with head on and crossing. Since the combined solution space is much larger than one of Algorithm 1, and the number of potential plans is lager as well. The optimal plans are displayed in the form of green line. In (*a*), the optimal plan (Ship_Agent1 turns to right 40° and sails 8 minutes, Ship_Agent2 turns to right 10° and sails 8 minutes) has the preference value 13.673263, which is larger than 12.395982 (the sum of 4.417589 and 7.978393 of two Ship_Agents' maximal preference). Comparing with Algorithm 1, the union plan is more rational, economic and safe although it takes a longer time to make plan. In (*b*), Ship_Agent4 is planed to turn to left 40°, while Ship_Agent3 keeps its own course and speed. This decision is reasonable because Ship_Agent4 is faster than Ship_Agent3 (14 kts to 10 kts).

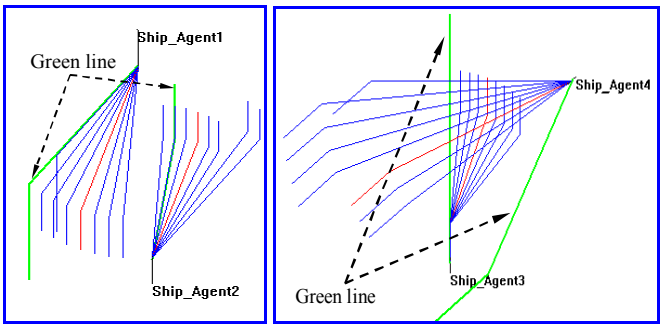
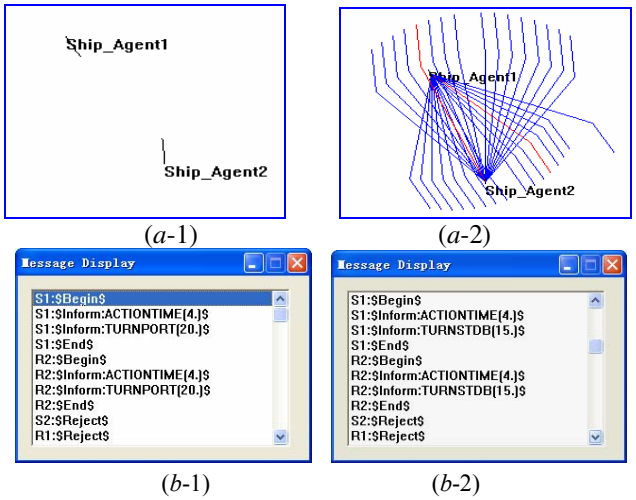


Fig. 5 The examples of collision avoidance with Algorithm 2.

Figure 6 shows the results for Algorithm 3. The (*a*) shows the encounter situation of two ships and the potential plans generated by running Algorithm 1. Figure 6's (*b*-1) to (*b*-5) and Figure 6's (*c*-1) to (*c*-5) demonstrate two kinds of negotiation processes. The difference is that each process has different initiator. In (*b*-1), Ship_Agent1 is an initiator and sends its plan (turn to left 20° in 4 minutes) to a responder Ship_Agent2. Ship_Agent2 receives the plan and makes decision on rejection or acceptance of the plan. If the plan is rejected, the initiator sends next potential plan in next cycle, until a plan is accepted by a responder. Figure 6's (*b*-2) to (*b*-4) show these cycles. Finally, (*b*-5) shows that the plan (turn to right 30° in 4 minutes) is accepted by Ship_Agent2 and the negotiation process ends successfully. Figure 6's (*c*-1) to (*c*-5) show another negotiation process in which Ship_Agent2 is an initiator and Ship_Agent1 is a responder. After several negotiation cycles as shown in (*c*-1), (*c*-2), (*c*-3) and (*c*-4), a plan (turn to right 25° in 6 minutes) is accepted by Ship_Agent1 as shown in (*c*-5).



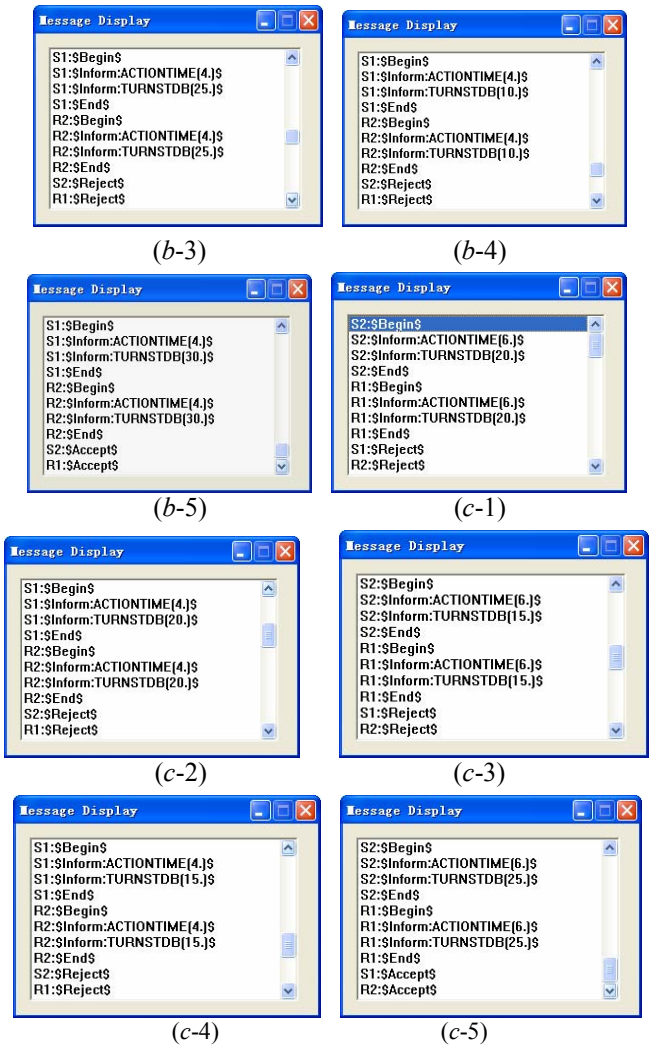


Fig. 6 The examples of negotiation-based planning (Algorithm 3).

V. Conclusions and Future Work

In this paper, we proposed applying multi-agent planning techniques to collision avoidance planning. Considering the real navigation constrains and environments, we have developed three multi-agent-based planning algorithms: the distributed independent planning for self-benefit purpose, the centralized planning for union-benefit purpose, and the negotiation-based planning for mutual-benefit purpose. Having introduced collision avoidance plan, we presented three algorithms in details. We have implemented these algorithms on a multi-agent-based simulation system. We conducted some experiments for validating the usefulness and applicability of the proposed algorithms. The experimental results demonstrated that these algorithms are useful and applicable for making decision on collision avoidance. As future work, we will work on adding domain knowledge and intelligent methods into planning procedure for enhancing the algorithms. We also need to improve the negotiation strategies and performance in case of more complicated encounter situations.

Acknowledgment

This paper is supported by HuoYingdong educational fund grant (No.0254-05-FN580001) and shanghai educational committee project (No.06FZ016).

References

[1] Chunsheng Yang, "An Expert System for Collision Avoidance and Its Application", *Ph.D. thesis*. Hiroshima University, Japan. September 1995.

[2] Y. Sato and H. Ishii, "Study of collision-avoidance system for ships". *Control Engineering Practice*. vol. 6, pp. 1141-1149, 1998.

[3] Yuhong Liu, "A design and Study on Intelligence Collision Prevention Expert System for Navigation," *Ph.D. thesis*. Harbin Engineering University, china. July 1999.

[4] C. Hwang, "The Integrated Design of Fuzzy Collision Avoidance and H∞- Autopilots on ships," *The Journal of navigation*. vol. 55, no.1, pp.117-136. 2002.

[5] Y. Liu and W. Yang. "The structure Design of an Intelligent Decision Support System for Navigation Collision Avoidance.," *IEEE The Third International Conference on Machine Learning and Cybernetics*. vol. 1. pp 302-307, August 2004.

[6] Yuhong Liu, Xuanmin Du and Shenhua Yang. "The Design of a Fuzzy-Neural Network for Ship Collision Avoidance". ICMLC 2005. Lecture Notes in Computer Science. vol. 3930. pp. 804-812. July, 2006.

[7] Shenhua Yang, Chaojian Shi, Lina Li, and Keping Guan, "Construction of Simulation Platform for Vessel Automatic Collision Avoidance with the Technologies of MAS and Navigational Simulato"r. JOURNAL OF SYSTEM SIMULATION. Vol18, pp.686-690, 2006.

[8] Yuhong Liu, Chunsheng Yang and Xuanmin Du. "A Multiagent-Based Simulation System for Ship Collision Avoidance". Advanced Intelligent Computing Technology and Applicatin-ICIC2007, in press, August (2007).

[9] Zlatina Lubomirova Marinova. Planning in Multiagent System. MSc Thesis. Sofia University. 2002.

[10] Mathijs de Weerdt, Adriaan ter Mors, and Cees Witteveen. "Multi-agent Planning: An introduction to planning and coordination". Delft University of Technology.

[11] Mathijs de Weerdt and Roman van der Krogt. "A Method to Integrated Planning and Coordination". American Association for Artificial Intelligence. 2002.

[12] Michael Bowling, Rune Jensen, and Manuela Veloso. "A Formalization of Equilibria for Multiagent Planning". American Association for Artificial Intelligence. 2002.

[13] Mark F. Wood, Scott DeLoach. "An Overview of the Multiagent Systems Engineering Methodology". AOSE. pp. 207-222, 2000.

[14] ichael P.Georgeff, Barney Pell, Martha E. Pollack, Milind Tambe, Michael Wooldridge. "The Belief-Desire-Intention Model of Agency". Lecture Notes in Computer Science. Vol. 1555. Proceedings of the 5th International Workshop on Intelligent Agents V, Agent Theories, Architectures, and Languages. Pp. 1-10, 1998.

[15] Chia-Lin Hsu, Hwai-Jung Hsu, Da-Ly Yang and Feng-Jian Wang. "Constructing a Multiple Mobile-BDI Agent System". The 14th Workshop on OOTA. pp. 109-116., 2003.

[16] Jeffrey S. Cox and Edmund H.Durfee. "An Efficient Algorithm for Multiagent Plan Coordination". AAMAS'05. Utrecht, Netherlands. pp. 828-835., July 2005.