Proceedings of the 2015 Winter Simulation Conference
L. Yilmaz, W. K. V. Chan, I. Moon, T. M. K. Roeder, C. Macal, and M. Rosetti, eds.

AGENT-BASED MODEL OF MARITIME SEARCH OPERATIONS: A VALIDATION USING TEST-DRIVEN SIMULATION MODELLING

Bhakti Stephan Onggo

Department of Management Science
Lancaster University Management School
Lancaster University
Lancaster, LA1 4YX, UNITED KINGDOM

Mumtaz Karatas

Department of Industrial Engineering
Turkish Naval Academy
Tuzla, Istanbul, 34942, TURKEY

ABSTRACT

Maritime search operations (and search operations in general) are one of the classic applications of Operational Research (OR). This paper presents a generic agent-based model for maritime search operations which can be used to analyse operations such as search and rescue and patrol. Agent-based simulation (ABS) is a relatively new addition to existing OR techniques. The key elements of an ABS model are agents, their behaviours and their interactions with other agents and the environment. A search operation involves at least two types of agent: a searcher and a target. The unique characteristic of ABS is that we model agents' behaviours and their interactions at the individual level. Hence, ABS offers an alternative modelling approach to analyse search operations. The second objective of our work is to show how test-driven simulation modelling (TDSM) can be used to validate the agent-based maritime search-operation model.

1 INTRODUCTION

Agent-Based Simulation (ABS) has become one of the commonly used tools to model and understand complex and nonlinear systems (Ferber 1999). ABS provides a controlled environment for systematic experimentation using a simulation model that is formed by a set of interacting agents. There is no consensus on the definition of an agent in the ABS literature (Macal and North 2010). Instead, we have observed a spectrum of complexity in the definition of an agent. At one extreme, an ABS model is formed by a set of agents with a set of simple attributes (such as speed and detection radius) and simple behaviours (such as move and rescue). At the other extreme, an ABS model can be composed of a set of agents with complex attributes (such as memory and bounded rationality) and complex abilities (such as planning and learning). However, most researchers agree that an agent is an autonomous entity (i.e. it makes independent decisions without any central control), has a set of objectives and interacts with other agents and its environment.

In this paper, we develop an ABS model to help us analyse strategies for maritime search operations. These can be analysed using analytical models from the vast literature on search theory. Search theory is a classic part of Operational Research (OR) and has a long history commencing with military operations during the Second World War. Since then, the applications and tools have continued to be mostly nautical, e.g. military assets looking for enemy submarines or maritime pirates, coast guards conducting search-and-rescue (SAR) operations, and patrol boats protecting ports or high-value assets. However, the uses of ABS in the maritime search domain and sea-patrol operations are surprisingly scarce, as confirmed by Davidsson et al. (2005) and Vaněk et al. (2013). One of the main reasons is the difficulty in validating the ABS model.

Model validation is one of the most important activities in simulation modelling. There are a number of techniques that have been proposed and used to validate a simulation model. Law (2014, chapter 5) and Banks et al. (2010, chapter 10) discusses various techniques, such as increasing the face validity of a model (e.g. by involving domain experts and model users), checking the validity of model assumptions, validating the components of a model, comparing the behaviour of a model with a real system (or something that can represent a real system if a system does not exist). Pidd (2004, pp. 233–246) divides validation techniques into white-box and black-box validation. White-box validation techniques aim to ensure that the internal working of a simulation model can be justified. Black-box validation techniques compare the output of a simulation model with the output of a benchmark (such as a real system being modelled, a real-world system similar to the system being modelled or an analytic model).

Model validation in ABS is especially challenging. First, we often need to represent the behaviours of agents and the interactions between agents using a set of logical rules. It is challenging to extract this information from real-world agents, especially if the agents do not want to be exposed (such as pirates or human traffickers). Furthermore, real-world agents are often heterogeneous. Hence, it is challenging to validate whether the rules used in an ABS model represent the rules used by most real-world agents and whether we have represented the heterogeneity of real-world agents correctly. Secondly, there is a need to validate ABS models at the agent and system levels. It is challenging to validate behaviour at the system level based solely on the knowledge of behaviours of individual agents. Finally, the ABS model often requires high-fidelity data. Although the collection of high-fidelity data has become very common, qualitative behavioural data from heterogeneous agents in a population are rarely available. Hence, empirical validation may not be possible. The difficulty in validating an ABS model is reflected somewhat in the survey done by Heath et al. (2009). They surveyed 279 research articles and found that only 35 per cent of the models were validated conceptually (white box) and operationally (black box).

The first objective of this paper is to show how ABS can provide an alternative modelling approach to analyse maritime search operations. The second objective is to demonstrate how a technique called Test-Driven Simulation Modelling (TDSM) (Onggo et al. 2014) can be used to validate an ABS model. The TDSM's basic principle is that a test case for a simulation model has to be specified before the simulation model is implemented. Hence, one of the main advantages of TDSM is that the validation process is explicitly embedded in the simulation-model development process. Modellers are forced to think about how their model is going to be validated, even before they start to develop the model. TDSM is explained in detail in Onggo et al. (2014).

This paper is organized as follows. In Section 2, we review related work in TDSM and the application of ABS in marine search operations. We explain our generic tool for marine search operations, called MASSIM (Maritime Search SIMulation), in Section 3. Section 4 discusses how TDSM is used to validate a number of scenarios in MASSIM. Finally, we present our conclusions and recommendations for future work in Section 5.

2 RELATED WORK

2.1 Application of ABS in maritime search operations

The existing applications of ABS for maritime search operations mainly focus on maritime patrol simulations to deter pirate attacks (Decraene et al. 2010, Bruzzone et al. 2011, Jakob et al. 2011, Tsilis 2011, Jakob et al. 2012, Vaněk et al. 2013), for port-protection simulation (Leathrum et al. 2009, Shieh et al. 2012), the simulation of traffic in ports and coastal waters (Hasegawa 2004), combat simulation (Champagne et al. 2003, Price 2003, Cioppa et al. 2004, Hill et al. 2004, Hill et al. 2006) and adversarial behaviour simulation (Agmon et al. 2009, Jakob et al. 2010). It is clear that most of the applications are for maritime security and military operations.

In maritime security applications, ABS is used to analyse strategies to protect merchant vessels from pirates. For example, Vaněk et al. (2013) developed an ABS model of maritime traffic that explicitly modelled pirate activities and piracy countermeasures in the Gulf of Aden. They used the model to analyse the design of a new transit-corridor system in the Indian Ocean. In another publication, they combined the ABS model with an optimization model to investigate the counter-measure configurations that yielded the best trade-off between security and cost in maritime transportation (Jakob et al. 2012). The typical agents used in maritime-transportation security applications are merchant vessels, navy vessels and pirate vessels. The software used by researchers includes AgentC (Jakob et al. 2011, Jakob et al. 2012, Vaněk et al., 2013), PANOPEA (Bruzzone et al. 2011) and MANA (Decraene et al. 2010, Tsilis 2011). PANOPEA uses discrete-event simulation but it has intelligent agent components embedded in the model.

Military applications vary. Cioppa et al. (2004) noted that there was increasing interest in the use of ABS in the Department of Defense. They highlighted three examples of ABS military applications. One of these was to study how unmanned surface vehicles could be used in force-protection missions. The simulation of the Bay of Biscay U-boat campaign was reported a few times (e.g. Champagne et al. 2003, Hill et al. 2004). Champagne et al. (2003) studied the emergent behaviours of combatants and the effectiveness of search patterns during the campaign. Hill et al. (2004) incorporated game theory into the behaviours of agents in the model. The US Navy also utilized ABS for a Naval Simulation System which was developed to support network-centric fleet battle exercises (Metron Incorporated 2015). DeStefano (2004) utilized ABS to create an executable model of a weapons system built in an agent-based combat model “System Effectiveness Analysis Simulation (SEAS)”.

Some of the models were calibrated using real-world data. For example, trajectory data for vessels (from satellite or self-reporting systems) was used to calibrate the paths taken by vessels (Vaněk et al. 2013). In the case of the Bay of Biscay U-boat campaign simulation, the authors used historical data from the war. However, the behaviour of hostile or non-cooperative agents such as pirate vessels is difficult to find, and in many cases unavailable. Hence, the calibration of such agents using empirical data is limited. Researchers can use an analytical approach to model the behaviour of such agents so that the behaviour shown by the simulation model should match the underlying analytical model. However, consistent with the findings by Heath et al. (2009), model validation is hardly discussed in the literature. Validation has mainly been done using a qualitative approach, such as face validity and an expert’s opinion (e.g. Vaněk et al. 2013, DeStefano 2004). Hence, it is clear that more research is needed for the validation of agent-based maritime search-operation models.

2.2 Test-driven simulation modelling

The idea behind Test-Driven Simulation Modelling (TDSM) comes from Test-Driven Software Development, known simply as Test-Driven Development (TDD) in software engineering. The main principle is that a test case for computer code must be created before the computer code is developed (Beck 2003). The test case is implemented as a unit test. Unit testing is a method in software engineering that is used to test an individual unit of computer code. The individual unit can be a function, a procedure or a class. Because the code does not exist when the unit test is created, the first test will always fail. A programmer will then refine the code until it passes the unit test. Subsequently, a new unit test is created and the process is repeated. At some point, the programmer may need to refactor the code. These steps are commonly known as Red-Green-Refactor, which signifies the cycle of creating unit tests that fail, writing code that passes the tests and refactoring the code.

A simulation model is a software artefact. Hence, TDD is applicable to simulation modelling as demonstrated in Collier and Ozik (2013), Asta et al. (2014) and Onggo et al. (2014). Collier and Ozik (2013) conducted the first exploratory study to investigate how unit testing could be used to verify an ABS model written in the Repast simulation library. They argued that by focusing on writing small unit tests, complex simulation code could be decomposed into smaller and more manageable components.

Asta et al. (2014) investigated the same approach and applied it to a discrete-event simulation (DES) model written using AnyLogic. Unlike Repast, AnyLogic is visual interactive modelling software. Hence, the unit test code was not written directly to the simulation source code. Instead, they wrote each unit test using a user-defined function in AnyLogic to verify a component in the model. Onggo et al. (2014) proposed TDSM as an approach to simulation-model verification and validation. TDSM can be applied to both DES and ABS. TDSM makes use of two unit-test suites, a verification suite and a validation suite. The verification suite contains a set of tests to check the correctness of a simulation model against its conceptual model. Each test represents one scenario. For example, one scenario may test the correctness of the distance travelled by a ship, i.e. at a given speed v and a travel time Δt, the distance between the original location of a ship and its new location is $v.\Delta t$. The validation suite contains a set of unit tests to check the validity of a simulation model. This is closely linked to black-box validation. Each test represents a scenario that compares the output of a simulation model against what is expected to be correct output. In this paper, we use an analytical model to provide estimations of or bounds on certain simulation outputs. These numbers are then used in the validation suite tests. This paper does not discuss the verification suite because it has been discussed in Collier and Ozik (2013), Asta et al. (2014) and Onggo et al. (2014).

3 MARITIME SEARCH SIMULATION (MASSIM)

Since a search operation consumes considerable amounts of time and effort, it needs to be planned and conducted efficiently. A tool that can help us analyse the expected performance of a search-operation strategy is very useful. This section explains MASSIM, a generic ABS tool for maritime search operations that we have developed using Repast (North et al. 2013).

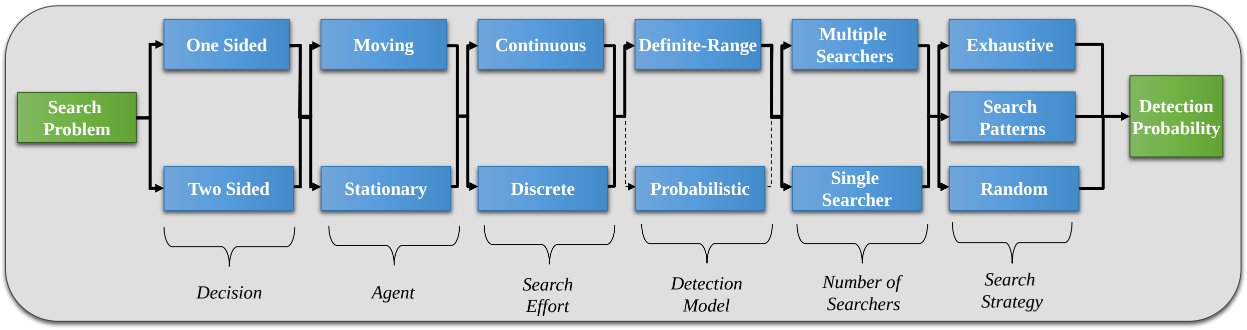
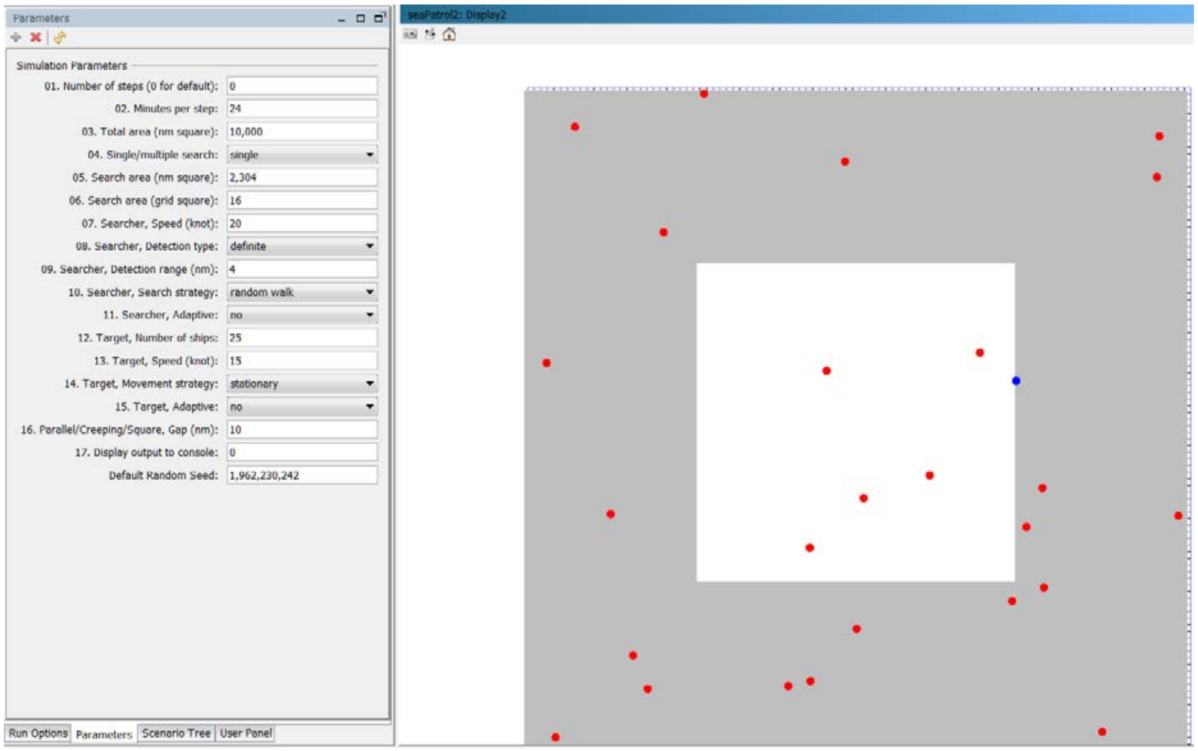


Figure 1. Search Problem Classification (From Karatas (2012))

First, we will explain how we abstract maritime search operations in MASSIM. From the ABS perspective, all search problems have two agents in common: a target (in a broad sense, something being searched for), and a searcher (Nunn 1981). Figure 1 shows the classification of a maritime search-operation problem. An operation can be classified as a two-sided problem when both target(s) and searcher(s) are active and behave in an intelligent way or a one-sided problem when either target(s) or searcher(s) is active and intelligent. The level of intelligence can be as simple as conducting a specific search pattern or as complex as learning from past experience. The agents can be stationary or moving. The search efforts in a discrete search operation are made at a number of discrete locations (e.g. a helicopter dips sonar at a location, pulls it and then flies to another location, and repeats the process for a number of discrete locations). In a continuous search operation, search efforts are carried out continuously along a certain path (e.g. a search conducted by a ship that carries hull-mounted sonar as it moves from one location to another). The detection function used during an operation is called definite range (also known as a "cookie cutter") when a target is detected whenever it is inside the detection range. In a probabilistic detection function (such as polynomial, exponential and cubic attenuation models), the detection probability is a function of the distance between searcher and target. The analysis of a search operation can consider a single-searcher strategy or a multiple-searchers strategy. The main objective is to analyse the effectiveness of a search strategy in a number of settings. Multiple-searcher operations are especially useful in the analysis of the effectiveness of a collaborative search strategy where a large area of interest is divided into sectors or sub-responsibility areas for each searcher. Finally, we can evaluate various search algorithms that can be used in a search operation, such as exhaustive and random.

MASSIM has been designed to support the analysis of a number of search operation settings, as shown in Figure 1. At the time of writing, MASSIM can support one- or two-sided problems, stationary/ moving agents, discrete/ continuous search efforts, a definite-range detection model, single/ multiple searchers, and a number of search patterns. The ABS model in MASSIM is formed by two agent types: searcher and target. A searcher's main objective is to detect a target. The behaviour of a target can be classified into three groups: a cooperative target that wishes to be detected by a searcher (e.g. the victims in a SAR operation), a non-cooperative or evading target that wishes to hide or escape from the searcher (e.g. a refugee trying to reach his/her destination without being detected by the coast guard), and a non-cooperative target that wishes to be as close as possible to a searcher without being detected (e.g. a hostile submarine trying to approach surface ships as close as its effective torpedo range).

The area of interest in MASSIM is modelled as a 2-dimensional plane (we refer to this as the total area). In a single-searcher operation, we define a sub-area inside the total area and refer to it as a search area (the white square in Figure 2). A searcher is denoted by a blue disk and a target is denoted by a red disk. In a multiple-searchers operation, the total area is divided into a number of equal-size grids where each grid represents a search area.



Simulation Parameters

01. Number of steps (0 for default):	0
02. Minutes per step:	24
03. Total area (nm square):	10,000
04. Single/multiple search:	single
05. Search area (nm square):	2,304
06. Search area (grid square):	16
07. Searcher, Speed (knot):	20
08. Searcher, Detection type:	definite
09. Searcher, Detection range (nm):	4
10. Searcher, Search strategy:	random walk
11. Searcher, Adaptive:	no
12. Target, Number of ships:	25
13. Target, Speed (knot):	15
14. Target, Movement strategy:	stationary
15. Target, Adaptive:	no
16. Parallel/Creeping/Square, Gap (nm):	10
17. Display output to console:	0
Default Random Seed:	1,962,230,242

Figure 2: Single-searcher operation

The simulation parameters can be controlled from the left panel. The first two parameters in the left panel are the number of time steps and the duration per time step (in minutes), respectively. A smaller time-step duration leads to a more accurate result in a continuous search operation. The third parameter defines the total area in nm^2 (nautical miles square). The area is assumed to be a square to make the simulation visualization easier. The fourth parameter is used to select a single-searcher operation or a multiple-searchers operation. For a single-searcher operation, we need to specify the size of the search area in nm^2 (parameter 5). For a multiple-searchers operation, we need to specify the number of search areas (or grids) in parameter 6. A searcher has a number of attributes (parameters 7 to 11): speed in knots, detection type (such as definite range, polynomial, exponential or cubic), detection range in nautical miles, search strategy (such as discrete random search, continuous random search, exhaustive search or parallel search), and whether the searcher has the ability to learn. Parameter 12 controls the number of targets in the total area. Each target has the following attributes (parameters 13 to 15): speed in knots, movement strategy (such as stationary or random walk) and whether it has the ability to learn. Parameter 16 is the distance between each track in a parallel/ creeping/ square search in nm.

4 VALIDATION USING TEST-DRIVEN SIMULATION MODELLING

In this paper, we validate our ABS model by comparing it with solutions based on analytic models from the search-theory literature. The use of analytic models is useful when empirical validation using complete real data sets is virtually impossible (e.g. we may know the location and number of refugee boats detected annually but not the number of undetected refugee boats that manage to land in a protected coastal region). The notations used in this section are listed in Table 1.

Table 1: List of notation items

Notation	Description
A	The total area or the size of the total area in nm^2 (depending on the context)
A_S	The search area or the size of the search area in nm^2 (depending on the context)
$P(S)$	Probability that a target is inside A_S
$P^i(D)$	Probability that a target is detected using search strategy i
$P^i(D\|S)$	Probability that a target is detected using search strategy i given that the target is inside A_S
(S_x, S_y)	Location of a searcher
(T_x, T_y)	Location of a target
N_T	Number of targets
R	Detection radius in nm
Δt	Duration of a time step in minutes
v_S	The speed of the searcher ship in knots
v_T	The speed of a target ship in knots
w	Sweep width (or 2R) in nm
G	The distance between each track in a parallel search (nm)

To make the explanation easier, in this paper we focus on a single-searcher operation and a "definite range" detection model. The total area and search area are squares with sides $\sqrt{A}$ and $\sqrt{A_S}$, respectively, and $A_S \leq A$. The position of a stationary target is assumed to be uniformly distributed over the total area. In the definite-range detection model, detection occurs when the distance between target and searcher is smaller than the detection range R. The detection probability of a target is defined by function P:

$$P = \begin{cases} 1, & \sqrt{(S_x - T_x)^2 + (S_y - T_y)^2} \leq R \\ 0, & \text{otherwise} \end{cases} \tag{1}$$

where (S_x, S_y) and (T_x, T_y) are the locations of the searcher and target, respectively.

In TDSM, model validation is done using a validation suite that contains a number of unit tests. Each unit test represents one validation case. A detailed method of how to add a unit test in Repast is described in Collier and Ozik (2013). The first validation case is used to compare the performance of an exhaustive search between the simulation model and the analytical model. In the second case, we repeat the same case but for a random search. The two cases are based on two hypothetical search strategies commonly used to provide bounds for the performance of a real-world search strategy. In the third validation case, we validate a real-world search called a parallel search against the lower and upper bounds obtained from analytical models. The simulation model uses the same assumptions as in the analytic models (e.g. stationary targets).

4.1 Exhaustive search

For a given speed v_S and sweep width w, within a duration of Δt, a searcher can cover an area of $v_S.\Delta t.w$ nm^2 (we refer to this area as the coverage area). An exhaustive search assumes that there is no overlap in the coverage areas in each search effort, and that no search effort is placed outside the search area. Hence, an exhaustive search can be thought of as carefully placing a number of non-overlapping coverage areas inside the search area, as shown in Figure 4 (left). Hence, an exhaustive search provides an upper bound on the search performance (Washburn 2002).

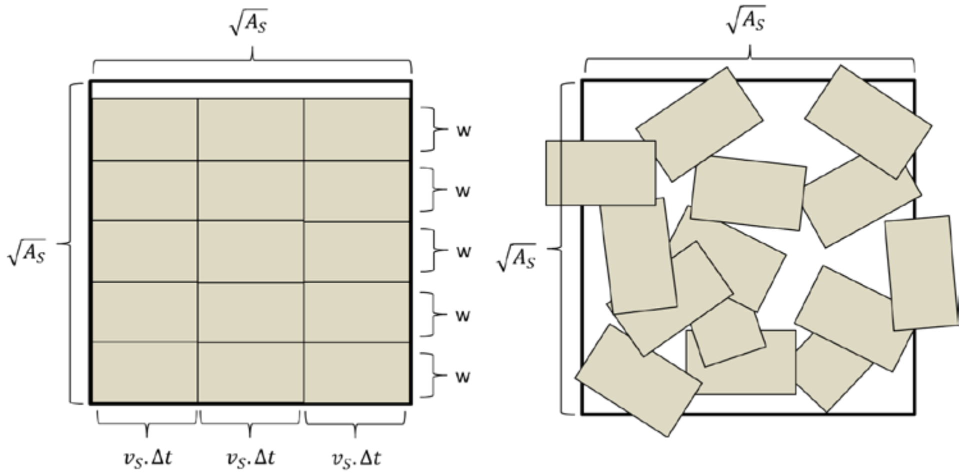


Figure 4: An illustration of Exhaustive search (left); Discrete Random Search (right)

The detection probability of a stationary target depends on the total area covered, which is a linear function of time. Once the search area is completely covered, the detection probability reaches its maximum value and remains constant. The probability of detecting a target by time t given that a target is present inside the search area can be written as equation 2 (Washburn 2002).

$$P_t^{exhaustive}(D|S) = \min\left\{\frac{v_S w t}{A_S}, 1\right\} \tag{2}$$

Figure 5 shows the performance ($P(D|S)$) of an exhaustive search, over time, obtained from the simulation model and the analytic model (equation 2), where A_S = 2,304 nm^2, v_S = 20 knots, Δt = 6 minutes and w = 8 nm. All simulation results shown in this paper are from an average of 20 replications. The figure shows a good match between the output of the simulation model and the analytical model. We used equation 2 in our unit test to validate the exhaustive search implementation in our model. For example, under the settings in Figure 5, we expect that $P(D|S)$ will reach 1 after searching for 864 minutes (i.e. 144 time steps). Our model passes this test, as shown in Figure 5.

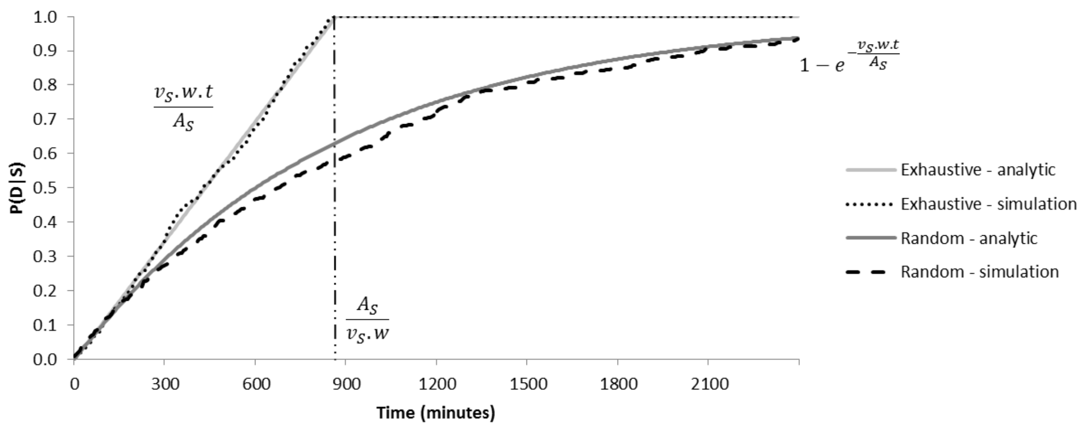


Figure 5: Performance of exhaustive search and discrete random search

The unit test to represent this validation case is shown below.

```
@Test
public void testExhaustiveMaxPerformance() {
    double prob_detected_when_inside = 0.0;
    ISchedule schedule = RunEnvironment.getInstance().getCurrentSchedule();
    for (int i=0; i<144; i++) schedule.execute();
    ContinuousSpace<Object> space =
        (ContinuousSpace<Object>)context.getProjection("space");
    Iterable<Object> listObjects = space.getObjects();
    for (Object obj : listObjects) {
        if (obj instanceof Observer) {
            Observer obs = (Observer) obj;
            prob_detected_when_inside = obs.GetProbDetectedWhenInside();
            break;
        }
    }
    assertTrue(prob_detected_when_inside > 0.9999);
}
```

4.2 Random search

Unlike an exhaustive search, a random search places the coverage areas at random inside the search area. The size of a coverage area is the same as in exhaustive search, i.e. $v_S.\Delta t.w$ nm^2. Random placement results in wasted effort because some coverage areas may overlap and some coverage areas may also cover an area outside the search area (see Figure 4, right). Hence, it performs worse than an exhaustive search and provides a lower bound on the performance of any sensible search strategy. The probability of detecting a stationary target by time *t* given that a target is present inside the search area is shown in equation 3. The proof can be found in Koopman (1946).

$$P_t^{random}(D|S) = 1 - e^{-\frac{v_S w t}{A_S}} \tag{3}$$

The performance ($P(D|S)$) of a discrete random search over time is shown in Figure 5 (using the same parameters as in the exhaustive search). The output of the simulation model matches the output of the analytical model. Equation 3 was used in our unit test to validate the implementation of a discrete random search in our model. For example, the following unit test checks whether the analytical result (i.e. 0.93) is within the confidence interval of the simulation result at time 2,298 minutes (i.e. 383 time steps).

```
@Test
public void testRandomAtTime2298() {
   double prob_detected_when_inside = 0.0;
   double sum = 0.0;
   double sumsq = 0.0;
   double meanPDS, halfCI
   for (int run=0; run<NRuns; run++) {
      ResetModel();
      ISchedule schedule = RunEnvironment.getInstance().getCurrentSchedule();
      for (int i=0; i<383; i++) schedule.execute();
      ContinuousSpace<Object> space =
         (ContinuousSpace<Object>)context.getProjection("space");
      Iterable<Object> listObjects = space.getObjects();
      for (Object obj : listObjects) {
         if (obj instanceof Observer) {
            Observer obs = (Observer) obj;
            prob_detected_when_inside = obs.GetProbDetectedWhenInside();
            sum += prob_detected_when_inside;
            sumsq += prob_detected_when_inside * prob_detected_when_inside;
            break;
         }
      }
   }
   meanPDS = prob_detected_when_inside/NRuns;
   halfCI = CalculateHalfCI(sumsq, NRuns, 0.1);
   assertTrue(((meanPDS–halfCI)<0.93) && ((meanPDS+halfCI)>0.93));
}
```

4.3 Bounds on parallel search

A parallel search is a real-world continuous search strategy. The search is conducted by moving along parallel tracks with a separation distance of *G*, as shown in Figure 6 (left). The performance of parallel search depends on *G* and, as expected, setting *G*=*w* provides the highest detection probability. For A_S = 2,304 nm^2, v_S = 20 knots, Δt = 6 minutes and *w* = 8 nm, the performance of a parallel search with *G* = 8, 10 and 12 is shown in Figure 6 (right). If we stop the simulation at time 864 minutes (i.e. the time when an exhaustive search reaches its maximum performance), we can see that the performance of the parallel search is bounded by the performance of the exhaustive search (upper bound) and the discrete random search (lower bound). When *G*=*w* (i.e. *G*=8), the parallel search's performance is close to the exhaustive search's performance. The difference is caused by the overlapping coverages when the searcher makes a turn. If *G*>*w*, the performance will be lower because of the areas not covered between each pair of parallel tracks. Since the performance of any sensible real-world search is bounded by the exhaustive search and the discrete random search, we can use this in our unit test to validate the performance of any sensible real-world search, as shown in the unit test below (0.632 and 1.0 are the performance of a discrete random search and a parallel search at time 864 minutes or 144 time steps, obtained using analytical models, respectively). Of course, we can always find a non-sensible search that would perform worse than a random search (e.g. if we set *G* to 20 in our example).

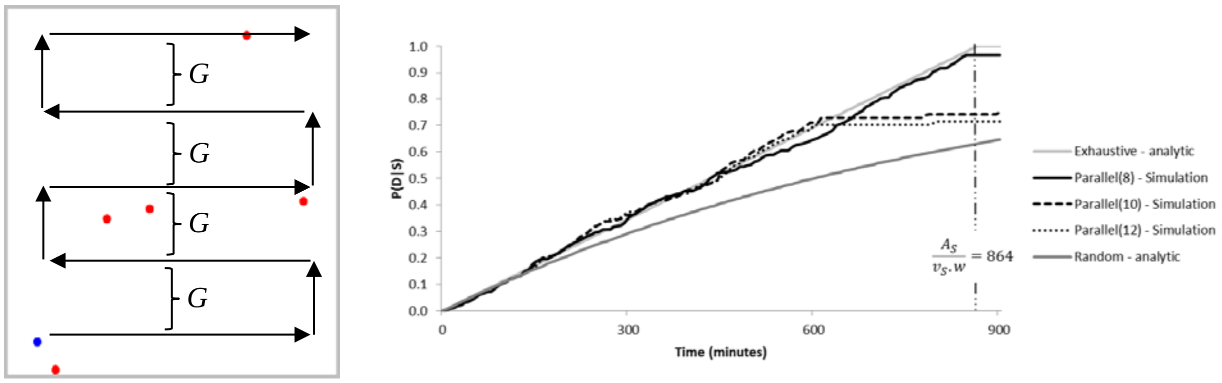


Figure 6: Parallel Search (left: pattern, right: performance with varying *G*)

The unit test to represent this validation case is shown below.

```
@Test
public void testExhaustiveMaxPerformance() {
    double prob_detected_when_inside = 0.0;
    ISchedule schedule = RunEnvironment.getInstance().getCurrentSchedule();
    for (int i=0; i<144; i++) schedule.execute();
    ContinuousSpace<Object> space =
        (ContinuousSpace<Object>)context.getProjection("space");
    Iterable<Object> listObjects = space.getObjects();
    for (Object obj : listObjects) {
        if (obj instanceof Observer) {
            Observer obs = (Observer) obj;
            prob_detected_when_inside = obs.GetProbDetectedWhenInside();
            break;
        }
    }
    assertTrue((prob_detected_when_inside >= 0.632)
        && (prob_detected_when_inside <= 1.0));
}
```

5 CONCLUSION

We have discussed how agent-based simulation can be useful in the analysis of maritime search operations and how the simulation model is validated against the analytical model using Test-Driven Simulation Modeling (TDSM). Analytical models have been shown to be useful when it is virtually impossible to validate a simulation model against empirical data. We can use analytic models to calculate the performance of simple search strategies. Hence, strictly speaking a simulation model is not needed for the analysis of simple search strategies. For a more complex search strategy where an analytic model becomes intractable, we need a simulation model. An analytic model is still useful in the analysis of a more complex simulation model because it can provide us with the bounds on performance. These can be used in a unit test, as we have demonstrated in this paper. More work is needed to combine TDSM and analytical models that can provide tighter bounds on some of the more complex search strategies, which include moving targets, probabilistic detection functions and search strategies involving adaptive agents.

REFERENCES

Agmon, N., S. Kraus, G. A. Kaminka, and V. Sadov. 2009. "Adversarial uncertainty in multi-robot patrol." In *International Joint Conference on Artificial Intelligence*, 1811–1817.

Asta, S., E. Özcan, and P.-O. Siebers. 2014. "An Investigation on Test Driven Discrete Event Simulation." In *Proceedings of the Operational Research Society Simulation Workshop*, 35–45.

Banks, J., J. S. Carson, B. L. Nelson, and D. M. Nicol. 2010. *Discrete-Event System Simulation*. 5th ed. Upper Saddle River, New Jersey: Pearson Education.

Beck, K. 2003. *Test-Driven Development by Example*. Boston, MA: Addison Wesley.

Bruzzone, A., M. Massei, F. Madeo, F. Tarone, and M. Gunal. 2011. “Simulating marine asymmetric scenarios for testing different C2 maturity levels.” In *Proceedings of the 16th International Command and Control Research and Technology Symposium*, 12–23.

Champagne, L., R. G. Carl, and R. Hill. 2003. “Agent models II: search theory, agent-based simulation, and U-boats in the Bay of Biscay.” In *Proceedings of the 2003 Winter Simulation Conference*, 991–998.

Cioppa, T. M., T. W. Lucas, and S. M. Sanchez. 2004. “Military applications of agent-based simulations.” In *Proceedings of the 2004 Winter Simulation Conference*.

Collier, N., and J. Ozik. 2013. “Test-Driven Agent-Based Simulation Development.” In *Proceedings of the 2013 Winter Simulation Conference*, 1551–1559.

Davidsson, P., L. Henesey, L. Ramstedt, J. Törnquist, and F. Wernstedt. 2005. “An analysis of agent-based approaches to transport logistics.” *Transportation Research Part C: Emerging Technologies*, 13(4): 255–271.

Decraene, J., M. Anderson, and M. Y. H. Low. 2010. “Maritime counter-piracy study using agent-based simulations.” In *Proceedings of the 2010 Spring Simulation Multiconference*, p.165.

DeStefano, G. V. 2004. “Agent based simulation SEAS evaluation of DoDAF architecture.” Master’s thesis, Department of the Air Force, Air University, Air Force Institute of Technology, Wright-Patterson Air Force Base, OH, No. AFIT/GOR/ENS/04-05.

Ferber, J. 1999. *Multi-Agent Systems: An Introduction to Distributed Artificial Intelligence.* Addison-Wesley, Harlow, England.

Hasegawa, K., K. Hata, M. Shioji, K. Niwa, S. Mori, and H. Fukuda. 2004. “Maritime traffic simulation in congested waterways and its applications.” In *4th Conference for New Ship and Marine Technology*, 195–199.

Heath, B., R. Hill, and F. Ciarallo. 2009. A survey of agent-based modeling practices (January 1998 to July 2008). *Journal of Artificial Societies and Social Simulation*, 12, 9.

Hill, R. R., L. E. Champagne, and J. C. Price. 2004. “Using agent-based simulation and game theory to examine the WWII Bay of Biscay U-boat campaign.” *The Journal of Defense Modeling and Simulation: Applications, Methodology, Technology*, 1(2): 99–109.

Hill, R. R., R. G. Carl, and L. E. Champagne. 2006. “Using agent-based simulation to empirically examine search theory using a historical case study.” *Journal of Simulation*, 1(1): 29–38.

Jakob, M., O. Vaněk, Š. Urban, P. Benda, and M. Pěchouček. 2010. “AgentC: Agent-based testbed for adversarial modeling and reasoning in the maritime domain.” In *Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems*: Volume 1, 1641–1642.

Jakob, M., O. Vaněk, and M. Pechoucek. 2011. “Using agents to improve international maritime transport security.” *Intelligent Systems*, 26(1): 90–96.

Jakob, M., O. Vaněk, O. Hrstka, and M. Pěchouček. 2012. “Agents vs. pirates: multi-agent simulation and optimization to fight maritime piracy.” In *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems*-Volume 1, 37–44.

Karataş, M. 2012. "An analytical comparison of random and exhaustive search of an expanding area with binary sensors." Engineer & the Machinery Magazine, 23(4): 2–13.

Koopman, B. O. 1946. “Search and Screening.” OEG Report No. 56.

Law, A. M. 2014. *Simulation Modeling and Analysis*. 5th ed. Boston, MA: McGraw-Hill.

Leathrum, J. F., R. Mathew, and T. W. Mastaglio. 2009. “Modeling and simulation techniques for maritime security.” In *Conference on Technologies for Homeland Security*, 636–642.

Macal, C. M., and M. J. North. (2010). “Tutorial on agent-based modelling and simulation”. *Journal of Simulation*, 4: 151–162.

Metron Incorporated. 2015. Naval Simulation System (NSS). Accessed 16 June 2015. http://www.metsci.com/Division/ORCA/Naval-Simulation-System-NSS-2.

North, M. J., N. T. Collier, J. Ozik, E. R. Tatara, C. M. Macal, M. Bragen, and P. Sydelko (2013). "Complex adaptive systems modeling with Repast Simphony". *Complex Adaptive Systems Modeling*, 1(1): 3. doi:10.1186/2194-3206-1-3

Nunn, L. H. 1981. "An introduction to the literature of search theory." No. CNA-PP-305. Center for Naval Analyses Alexandria VA, Operations Evaluation Group.

Onggo, B. S. S., C. Indriany, and M. Gunal. 2014. "Test-Driven Simulation Modelling", In *Proceedings of the 7th International Workshop on Applied Modeling and Simulation (WAMS14)*, Istanbul, Turkey, edited by Bruzzone, Çayırcı, Günal, Kaylan, Massei, Zanni-Merk. 43–48.

Pidd, M. 2004. *Computer simulation in management science*. 5th ed. Chichester, England: John Wiley & Sons.

Price, J. C. 2003. "Game theory and U-boats in the Bay of Biscay." Master's thesis, Department of the Air Force, Air University, Air Force Institute of Technology, Wright-Patterson Air Force Base, OH, No. AFIT/GOR/ENS/03-18.

Shieh, E., B. An, R. Yang, M. Tambe, C. Baldwin, J. DiRenzo, B. Maule, and G. Meyer. 2012. "PROTECT: A deployed game theoretic system to protect the ports of the United States." In *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems,* Volume 1, 13–20.

Tsilis, T., 2011. "Counter-Piracy Escort Operations in the Gulf of Aden." Master's thesis, Naval Postgraduate School, Monterey, CA.

Vaněk, O., M. Jakob, O. Hrstka, and M. Pěchouček. 2013. "Agent-based model of maritime traffic in piracy-affected waters." *Transportation research part C: emerging technologies,* 36: 157–176.

Washburn, A. R. 2002. *Search and detection*. INFORMS.

AUTHOR BIOGRAPHIES

BHAKTI STEPHAN ONGGO is Lecturer (Assistant Professor) in the Department of Management Science at the Lancaster University Management School, Lancaster, United Kingdom. He completed his PhD in Computer Science from the National University of Singapore and his MSc in Management Science is from Lancaster University. His research interests are in the areas of simulation methodology (conceptual modelling, discrete-event simulation, system dynamics and agent-based simulation) and simulation applications in business and management. His email address is s.onggo@lancaster.ac.uk.

MUMTAZ KARATAS graduated from the Turkish Naval Academy in 2001. He received his MS degree in Industrial & Operations Engineering from the University of Michigan and his PhD in Industrial Engineering from Kocaeli University, Turkey. He spent two years at the Naval Postgraduate School as a visiting researcher and postdoctoral fellow between 2011 and 2013. He is currently an Assistant Professor in the Industrial Engineering Department at the Turkish Naval Academy. His current research areas include optimization and military operations research. His email address is mkaratas@dho.edu.tr.