The maritime surveillance problem

Philip Kilby
NICTA and Australian National University

Patrick Tobin
Swinburne University of Technology

Ruth Luscombe
University of Melbourne

Steven I. Barry and Roslyn Hickson
University of New South Wales @ ADFA

1 Introduction

The problem examined here is that of using aircraft to find and classify ships at sea. Identifying the position and type of ships within, or close to, Australian sea borders is an important part of Australia's national security activities. Conducting the surveillance within Australia's sea borders is a mandated role for the Australian Defence Forces. Currently it is carried out by the Royal Australian Air Force. The Defence Science and Technology Organisation (DSTO) is looking at how these missions can be flown with the maximum efficiency.

There are two important considerations in running these surveillance missions. The first is to reduce as much as possible the number of ships that are "missed" in the survey. The second is to make the flying time as small as possible. In this study we will describe the problem in detail, and give some initial results on how well these two objectives can be met under different scenarios. There are three questions the DSTO are particularly interested in:

- What is the effect of treating the targets as stationary?
- Can flying time be improved?
- Is execution time of methods an issue?

The aim of the study was to find answers to these questions, or at least determine how they might be answered.

In Section 2 we describe the Maritime Surveillance Problem as presented to the MISG, and in Section 3 the problem is formulated in terms of related problems, particularly from the routing literature. In Section 4 the simulation system that was developed during the week is described. This system was able to go a long way towards answering DSTO's original questions. The results are presented in Section 5. On the way to answering these questions, a number of related questions were also examined. The results are presented here, including

- determining the heading to intercept a moving ship (Section 6),
- estimating tour length (Section 7), and
- finding what proportion of ships are detected? (Section 8).

2 The maritime surveillance problem

The basic objective of maritime surveillance is to monitor the ocean regions around Australia's coastline. A particular region of ocean may be identified, and some part of that region assigned to a single surveillance aircraft (called a *platform*). The surveillance problem is therefore partitioned into smaller problems, one for each 'Area of Interest' (AI). In this study we consider the problem for a single AI. Typically, an AI is a square or rectangle with sides in the range of 200-300 n mile[1].

Various types of aircraft are used for surveillance. Typically, it is an AP-3C Orion aircraft, but may be a helicopter. The flight speed of the platform is an important variable in determining the flight path. Values range from 100-350 kn.

Each AI has a default flight path, defined by a list of *way points*, known at the start of the mission. It is a requirement that all way points be visited, and that they be visited in the order specified in the default flight path. The way points are selected so that if the platform flies the default path, every point within the AI comes within surveillance range. An example set of way points is given in Figure 1. If no ship is seen, the platform would simply fly the route defined by default flight path.

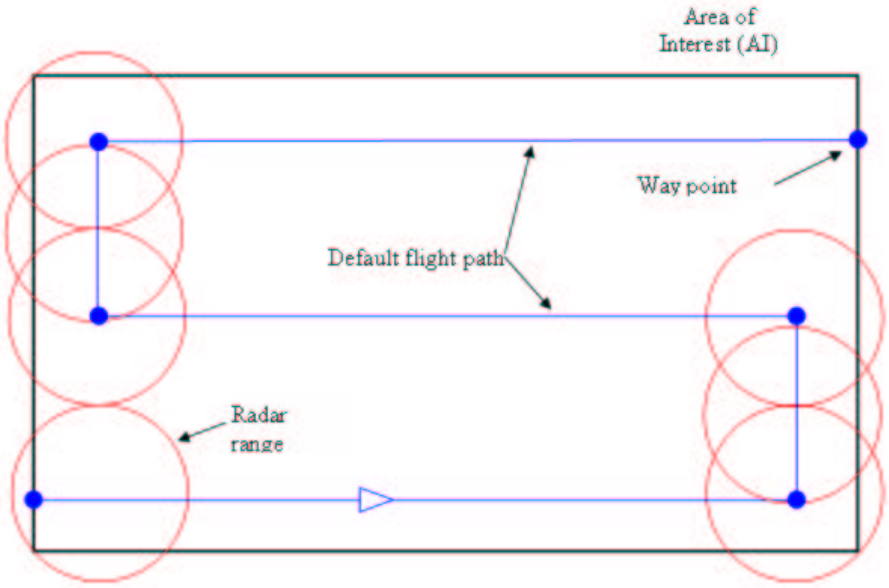

Figure 1: Way-points - blue dots - defining the default flight path.

As the platform proceeds, ships will come within range of the radar on board. The distance at which a ship can be detected is variable, and depends upon factors such as the weather, the flight altitude, and the type of equipment used on the platform. For the purposes of this study, the *detection distance* is treated as an input variable. Typical values for detection distance are 20-100 n mile. We make the simplifying assumption that detection is binary – either a

[1]The Nautical Mile (n mile) is the standard unit of distance at sea. It is equivalent to one minute of latitude along any line of longitude, and is about 1852m. Its associated speed unit is the knot (kn) $=_{defn}$ 1 n mile/hour.

ship is detected or not. In reality, there is a drop-off in the sensitivity near the boundaries of the radar where detection is fuzzy, but for this exercise we are ignoring these effects. We are also assuming that when a ship is detected, we can tell the speed and direction of travel. This is also a simplification, but one that is fairly reasonable, as updates to these estimates can be accommodated as the platform flies closer to the target. Another simplification made here is that ships do not change their speed or direction. This is an oversimplification that a practical system would not be able to assume. However for the questions examined here, it is a reasonable assumption.

When the platform detects a target, it flies to a point that is close enough to be able to classify the ship. "Classifying" a target involves seeing it well enough to determine the type of the ship. Once again, the distance at which this classification can take place is variable depending on factors such as the weather. We use an input variable *classification distance* to determine what is "close enough". Typical values are 0-20 n mile.

One restriction that is placed on routes in this study is that the platform is not permitted to leave the Area of Interest. That is, if a ship is detected close to the border of an AI, it may leave the AI before the platform can reach it. If this is the case, the platform is not permitted to "chase" it outside the AI. Similarly, a ship detected outside the AI must enter the AI before the platform can classify it. Note that it is also possible for a ship to pass through an AI without being detected, depending on the timing.

A second restriction is that the flight time is limited. Since all way-points should be visited, this places a hard deadline on the last way-point. It is a hard deadline as it relates to the ability of the platform to return to base without running out of fuel! This means that some targets may not be able to be classified if visiting them means the platform cannot reach the final way point in time.

As the platform is flying toward one target (or way point) other targets may be detected. During the course of a mission, more than 100 targets may be detected. At any one time, we may have tens of ships detected but not yet classified. The problem we address is to find a route which visits all these unclassified targets, plus the remaining way points, in as short a time as possible.

At all times the platform maintains a plan of the flight path. This flight path evolves as new detections are made. The route will visit all remaining way points, plus as many unclassified targets as possible. It will obey all of the constraints discussed above. Calculation of a new route is triggered whenever a new target is detected. It is also triggered when a previously detected target enters the AI. When the current target is classified, the platform selects the next target or way-point in the current route as the next point to be visited. In a practical system, other events such as a ship adopting a new speed or heading, or loss of contact with a target, may also trigger a re-calculation. However these complications are out of the scope of the current study.

2.1 Current solution method

DSTO currently model the maritime surveillance problem within a larger simulation model. This larger model is used in undertaking operational analysis in such areas as tactics development and capability assessment. An algorithm has been developed to solve the maritime surveillance sub-problem within the context of this larger simulation system. The algorithm employed is a simple single-swap crossover genetic algorithm. A population size of two is usually used, with no mutations. A fixed number of generations is used. This method is used

to optimise the route each time a trigger event occurs, such as a new ship being detected. This method makes the assumption that ships are stationary, and so must also be invoked periodically in order to re-calculate a route with updated ship positions.

3 Problem formulation

This problem can be seen as having aspects of many well-studied problems, but drawn together in a unique way. First, it has a strong connection with the Travelling Salesman Problem (TSP) [8]. In the TSP, a salesman wishes to visit a set of cities. He knows the cost of travel between each pair. What is the shortest route that visits each city exactly once, and then returns to the starting city? In the Open TSP, the requirement to return home is removed.

Our problem is much like an Open TSP (with the last city defined by the last way point). However, unlike the standard TSP, our cities – the ships – can move. The moving target TSP has received some attention in the literature [6, 9, 10, 13, 14, 17], but none of these solutions are immediately applicable to the problem at hand.

One important feature of the moving target problem is that, having decided to visit a particular target, an intercept point must be calculated. This point must minimise the flight time from the platform's current position to a point that intercepts the direction of travel of the target ship. These calculations are discussed in more detail in Section 6.

Another complicating factor is the *online* nature of the problem. We do not know all of the tasks that are to be performed at the start of the mission – they are revealed only as we proceed. There has been a growing interest in online, or *dynamic*, routing and scheduling problems recently – [2, 3, 5, 11, 12] to cite just a few.

The crux of the online problem can be expressed in the following question: *Do I spend x minutes visiting a target now, or will I perhaps be able to visit two as yet unseen targets in x minutes later in the route*? The answer depends on a host of variables, including expected density of targets and remaining flight time.

Each target has a potential time window for visit. In routing terminology, a time window is the period during which a visit may take place. Since we know the position, speed and direction for a target, we know when it will enter the AI (if it has not already) and when it will leave the AI. These two times define the time window for each target.

Finally, we have the requirement that as many targets as possible are visited in the available time. In the routing literature, such problems are sometime called "prize-collecting" problems [1, 4]. The literature also has reference to the "Close-Enough" TSP where, as in our problem, it is not necessary to visit the actual location of the "city", but it is sufficient to be "close enough" [7]. To draw these all together, we can define the problem as follows:

Maximise the number of classified targets; and within this maximum, minimise the flight time, subject to the following constraints:

- Flight time $\leq$ maximum flight time.
- All way-points visited in order.
- The route never leaves the AI.

In terms found in the routing literature, it is an Online Prize-collecting, Open, Close-Enough Travelling Salesman Problem with Time Windows and Precedence Constraints.

4 Simulation system

A simulation system called TPP (for Travelling Pilot Problem) was developed during MISG in order to begin answering the specific questions posed by DSTO (Section 1).

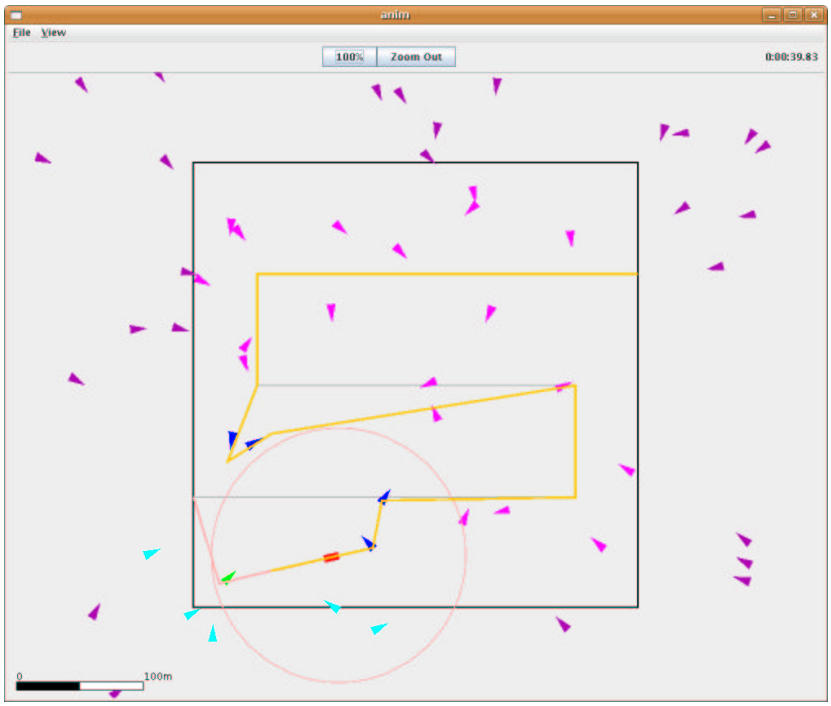

Figure 2: A screen-shot of the TPP simulation animation program.

The system had a graphical interface which allowed simulation runs to be visualised. An example screen-shot from the animation is given in Figure 2:

- The search area is boxed in grey.
- The aircraft performing the search is the red rectangle.
- The circle around the search craft is the radar detection range.
- The orange line gives the currently planned route.
- The grey line gives the default flight path.
- Targets are the triangles. Colours indicate their status:
  - Green: Classified,
  - Blue: Detected,
  - Cyan: Detected, but outside the search area (and hence ignored),
  - Pink: Undetected (within search area),
  - Purple: Undetected (outside search area).

Key	Default	Units	Description
maxTimeMin	480	minutes	The maximum flight time.
areaLenNM	350	n mile	The x-dimension of the Area of Interest.
areaWidNM	350	n mile	The y-dimension of the Area of Interest.
platformSpeedKnots	250	kn	The speed of the platform.
targetSpeedKnots	10	kn	The speed of the targets.
classifyDistNM	0	n mile	The minimum distance the platform must approach the target, in order to classify it.
detectDistNM	100	n mile	The radar range of the platform.
numTargets	30		The total number of targets.

Table 1: Configuration items.

4.1 Configuration

The system has a number of configuration parameters, reflecting the variables of interest to DSTO. These are listed in Table 1.

Values for these configuration parameters appear in a file that controls the execution of the program. Unless varied for a particular test, the value of each parameter is set at the default given in Table 1. All targets are assigned the same speed. This was done to understand the effect of target speed on solutions.

The targets created are distributed uniformly over the Area of Interest (AI) and the eight neighbouring areas of equivalent size – that is, areas of the same size are created to the North, NE, E, SE, S, SW, W and NW of the original AI. Targets are distributed uniform-randomly within the larger region. This allows targets to enter the AI during the simulation from a point outside the original boundaries.

The result of this is that not all of the targets will enter the AI. A report of the number that do is given at the end of the simulation.

4.2 Solution method

Ideas developed during MISG were implemented in the TPP system. The system uses the idea of the "current tour" – that is, the list of targets and way-points in the order in which they will be visited.

When we talk about visiting a target, we always mean choosing a flight path that will intercept the current path of the target as soon as possible. This is a relatively straightforward calculation, although practical considerations such as limits on the turning circle of platforms may influence the calculation. These considerations are discussed in Section 6. The intercept calculation used in the simulation system does not account for turning circles, and always heads directly toward the intercept point regardless of the classification distance.

We use "visit" to refer to both target ships and way-points. Thus a tour is a sequence of visits. The system operates using the procedure described in Figure 3.

If conditions 2a or 2b are met, we will always have a new "next visit". If condition 2c is

0) At the start of operations, form an initial tour considering all detected targets, plus all way-points. (If there are no detected targets, the tour will consist only of the way-points in order).
1) Move towards the first visit in the current tour.
2) Each minute, check the following conditions:
   2a) If the next visit is a ship, and it is within the classification distance, mark the target as classified and remove it from the tour.
   2b) If the next visit is a way-point and the distance to the point is sufficiently small, mark the way-point as visited and remove it from the tour.
   2c) If any new targets have been detected, attempt to create a new tour incorporating the new targets. If successful, replace the current tour with the new tour.

Figure 3: Basic TPP algorithm.

met, we may go to a new target before classifying/visiting the current target. At steps 0 and 2c, only "eligible" targets are considered for inclusion in the tour. Eligible targets are those that

- have been detected,
- have not yet been classified,
- are currently within the area of interest,
- will not cause the platform to exceed the deadline at the last waypoint (using direct flight to the target and then to the waypoint).

The method of tour construction for step 0 is a traditional construct-and-improve method, described in Sections 4.2.1 and 4.2.2. The method for extending the tour at step 2c is described in Section 4.2.3. We discuss the consequences of moving targets in Section 4.2.4.

4.2.1 Tour construction

Nearest Neighbour: One of the simplest methods for creating a route in the Travelling Salesman Problem is called "Nearest Neighbour". At each stage, the eligible targets and next way-point are considered, and the next to be visited is simply the one that is currently the closest.

Insert Heuristic: Many insert techniques have been described, for example Solomon's methods [18]. The version used here is very basic. The route is initialised with the list of way-points in order. Each eligible target is then considered. The cost of inserting the target between each pair of visits in the tour is calculated. The new intercept points of visits after the insert target are not calculated – we essentially assume all ships are stationary. The target is inserted into the position which gives the least increase in cost. If, after insertion, the route becomes infeasible due to a missed deadline at a target or the final way-point, the tour reverts to the previous configuration.

4.2.2 Tour improvement

After construction, and after any other change to the tour, a tour improvement procedure is called. This procedure runs a number of standard TSP improvement operators. Note that the operators are restricted so that changes that would alter the order of way-points are not considered.

The operators used are:

Move: Each visit is removed from its current position, and re-inserted into the (legal) position which causes the least increase in distance.

Two-opt: The two-opt operator removes two links in the tour, and replaces them with two others, effectively reversing the order of visits between the broken links. It is used often, and described in, for instance, [16].

Or-opt: First described by Or [15] and also described in [16], this procedure removes a chain of k consecutive visits, and tries to re-insert it between each pair of visits in the remaining tour. If no improvement can be found, the procedure is repeated with chains of length $k-1$, and on down to length 2. A length 1 Or-opt is exactly equivalent to the *Move* operator. In TPP, chains of length (k) up to 5 were considered.

These improvements procedures are called in the order given. Improvements are implemented as they are found (i.e. first-found rather than best-first). When all three complete without improving the solution, improvement is finished. One important consideration is the procedure for 0-cost changes (i.e. changes where the costs before and after are equal). 0-cost changes are important as they can move the solution to a new part of the solution space from which improvements can be found. However, accepting all 0-cost changes results in cycling behaviour. In the method implemented in TPP, 0-cost improvements are accepted with probability 0.5.

4.2.3 Tour extension

At step 2c of the algorithm in Figure 3, newly detected targets are incorporated into the existing tour. In TPP, this is handled as a tour modification procedure in the following way.

1) First, an attempt is made to include the new target in the current tour. This is done by adding the new target, and running the tour improvement procedure until completion. If the resulting route is legal, it is accepted and we exit – we have a new tour with more targets. Otherwise, continue to step 2).
2) The illegal tour is examined. The target (other than the new target) that causes the greatest deviation is removed from the tour. The improvement procedure is re-run. If the resulting tour is legal, the cost of the new tour is compared to the original tour. If the original tour is shorter, then it is kept, and the addition of the new target has failed. Otherwise, we move on to step 3).
3) The new target has been added, and another target deleted, with a shorter route resulting. With the extra time now available, it may be possible to include a target that had previously been skipped. If there are any eligible targets not currently in the tour, the target that is closest to the new tour (i.e. that causes the least deviation) is identified. We return to step 1 with the identified target as the target to be inserted, and the new tour treated as the "original" tour.

The result of this procedure is either the original tour, a new tour with more targets but greater distance, or a new tour with the same number of targets but smaller distance. These outcomes are consistent with the objectives of the problem.

4.2.4 Moving target considerations

Until now, algorithms have been described in terms which largely ignore the movement of targets. We now consider how this target movement affects the algorithms. We have implemented three variants of the algorithms described above.

Stationary ships

This is the method currently used by DSTO in their maritime surveillance search model. The ships are assumed to be stationary. The current intercept point for all ships is calculated each time a tour is to be created, and this position is used for all calculations.

The change in costs for the improvement operators described in the improvement are easy to calculate in this case. Consider a tour a, b, c, d, e, and $D(x, y)$ gives the distance between current intercept points for visits x and y. The cost of moving d to follow a is

$$D(c, e) - D(c, d) - D(d, e) + D(a, d) + D(d, b) - D(a, b).$$

The same formula holds true for any pair a and b on the tour, regardless of whether they are before or after d. In fact the expression $D(c, e) - D(c, d) - D(d, e)$ can be pre-calculated and used for all pairs on the route.

Thus the test for acceptance of an improvement is $O(1)$ (i.e. constant-time) operation. If we have n visits in the tour, then we have $O(n)$ insert positions, and $O(n)$ candidate visits to move – hence all *Move* type improvements can be tested in $O(n^2)$

Moving ships

This is a fully-dynamic version of the algorithm. In this variant, each time a tour is modified, new intercepts for all targets are calculated. This has a large effect on the testing of improvement operations. Each improvement – such as the example move discussed above – requires essentially the whole tour to be reconstructed in order to find the new intercept points. Testing of acceptance of a move improvement is now $O(n)$, and testing all *Moves* is now $O(n^3)$.

Jumpy ships

There is a mid-point between these two variants. Jumpy ships works as follows. During a particular iteration, the position of targets is treated as fixed, so improvements can be tested quickly using the same techniques as the "Stationary Ships" variant. However, once a new tour has been calculated, the position of each ship is updated. The new intercept point based on the new route is calculated. But rather than simply updating the ship's position, it is moved to the mid-point between the old and new points. The improvement phase is re-run using the new fixed positions. Improvement and update is iterated until the position of targets converges sufficiently. If the positions do not converge sufficiently within 20 iterations, the procedure is terminated and the position updated using the order of visits in the last tour.

The advantage of not updating the position of the target in a single step is that occasionally a better route can be found that uses the fact that the target will be in a different area at a different time. It is a way of preventing the solution converging to a given point too quickly, and allows a neighbourhood of that solution to be explored.

4.2.5 Other heuristics

Pro-rata deadlines

It was seen that when there are many potential targets, the algorithms would tend to make the platform spend disproportionate amounts of time in the first part of the AI, and

then have no time to explore the rest of the space. In order to overcome this, an artificial deadline is assigned to each way-point. The deadline is calculated pro-rata, based on how far along the default flight path the way-point is located. So, given an 8-hour deadline, if the way-point is $\frac{1}{4}$ of the way along the default flight path, it would have a deadline of $\frac{1}{4} * 8 = 2$ hours.

5 Results from the simulation system

5.1 Stationary ships assumption

The first question we looked at was the validity of the "stationary ships assumption" – that is, what is the effect of target speed on the measures of effectiveness (MOEs).

The most important MOE is the classification rate – how many of the targets that are in the area are classified? We consider "number in area" to mean the number of targets that are ever in the area. This can mean targets that are close to the border when the mission starts, and leave shortly afterwards, plus those that enter the area while the mission proceeds.

This definition unfortunately introduces a bias into the statistics. An efficient tour will mean the mission is finished early, and hence targets that move into the AI after a short while later are not counted. Unfortunately all other methods of direct comparison considered made a larger bias. For instance taking the "number in area" count over any defined period disadvantages an efficient tour.

An experiment was run using 100 scenarios. Target speeds of 0, 5, 10, 15 and 30 kn were examined. Speeds up to 15 kn are common in shipping, and the upper limit of 30 kn was also tested.

In order to look at a wide variety of scenarios, scenarios with 25, 50, 100, 200, 500, and 750 targets were run. As described previously, these counts are for the larger area, and hence different numbers of targets will enter the AI over the course of the mission. In general, the "number in area" count also goes up with target speed, as targets that start further away are able to reach the AI.

The solution method described in Section 4.2 was tested. Each of the 3 methods of updating tours – Stationary Ships, Jumpy Ships and Moving Ships (described in Section 4.2.4) – was tested. As the improvement methods used are stochastic (in the acceptance of 0-cost moves) each was run 5 times to arrive at a reasonable average. Average (arithmetic mean) performance is reported here. A maximum elapsed time of 5 seconds was placed on calculations at each time point. This equates to a restriction that a pilot not wait more than 5 seconds for a new tour to be calculated after updated positions were supplied.

The results are shown in Figures 4 to 8. The graphs show the percentage of targets successfully classified as a function of the number of targets in the area. Figure 4 shows that if the targets are indeed stationary, all three methods perform almost exactly as well. The three graphs are coincident for most values. In Figure 5 (target speed 5 kn) we see that some improvement is possible, with the effect growing with the number of targets. However the effect is fairly marginal – *Moving Ships* method producing about 1.4% more classifications than *Stationary Ships* (76.5% compared to 75.1%) for scenarios with 50 or more ships in area.

A similar story is told in Figures 5 and 6, with the difference growing. For target speed 10 kn, the comparison is 71.4% vs 68.4% - a 3% improvement. For 15 kn, the improvement is 4%. At the extreme (target speed 30 kn), moving ships performs 5% better on the 50+ scenarios. In each case, the *Jumpy Ships* method produced an inferior value. For scenarios with less

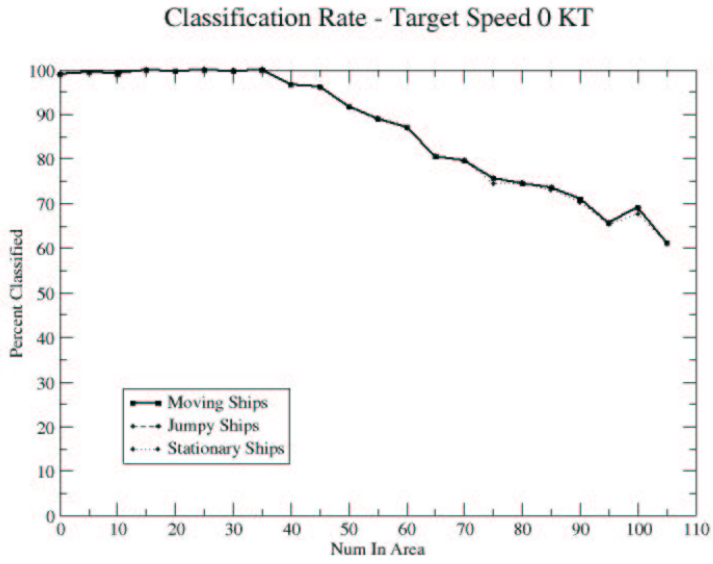

Figure 4: Classification Rate – target speed 0 kn.

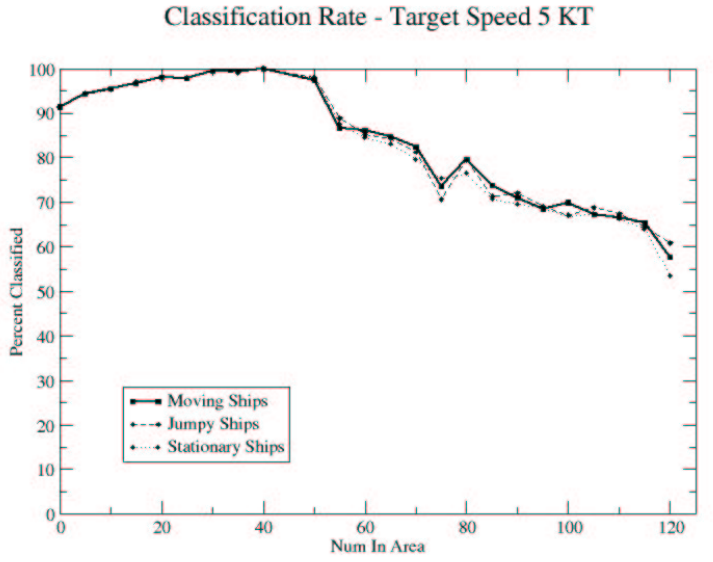

Figure 5: Classification Rate – target speed 5 kn.

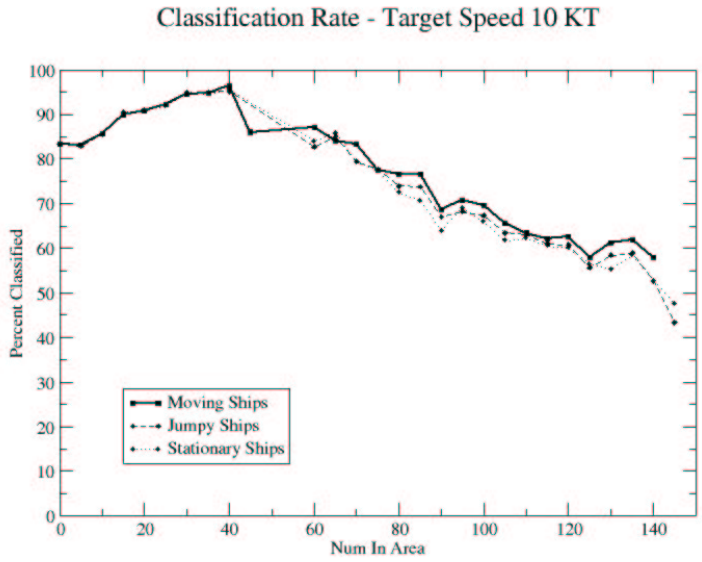

Figure 6: Classification Rate – target speed 10 kn.

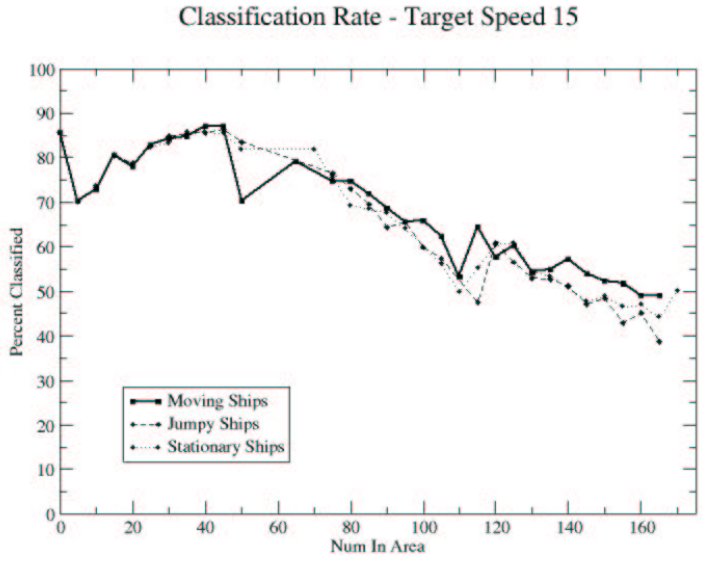

Figure 7: Classification Rate – target speed 15 kn.

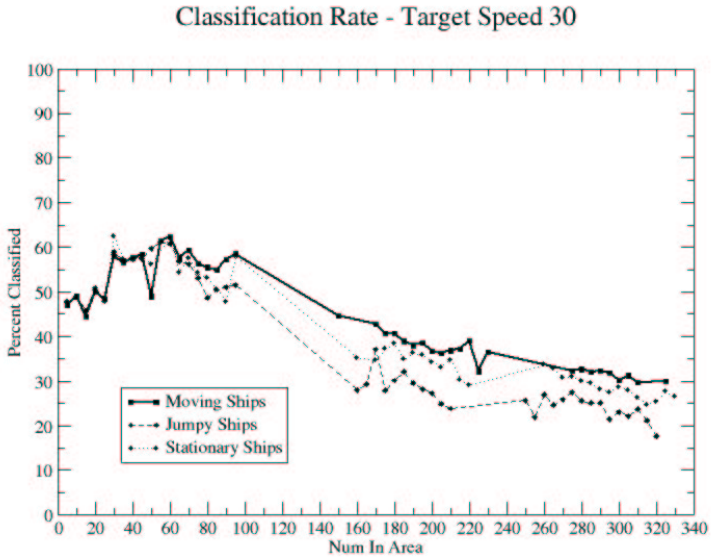

Figure 8: Classification Rate – target speed 30 kn.

than 50 ships, the *Moving Ships* and *Stationary Ships* methods are almost indistinguishable.

The conclusions from this are pretty clear. For the simple methods considered thus far, the stationary ships assumption is reasonable for scenarios with fewer than 50 ships. If there are more than 50 ships, however, the stationary ships assumption yields inferior results. Up to 5% more ships might be classified for targets travelling at 15 kn. At the extreme, 12% more classifications were achieved using the *Moving Ships* method. The *Jumpy Ships* method should not be used.

5.2 Time to complete mission

Next we examined the time to complete mission, that is, how long it took to reach the final way-point. We looked only at a "typical" case of target speed 10 kn. The results from the experiments reported in Section 5.1 for target speed 10 kn were used. The results are shown in Figure 9. The graph clearly has two parts. In the first part – up to about 40 targets – there is sufficient time to visit all targets. In the second part, the flight time is limited by the 8-hour maximum.

This divergence at 40 targets is also interesting in the context of Section 5.1, as it partly explains why the results there seemed to have two distinct parts. The differences between the algorithms are very small – of the order of only 1 minute in 8 hours for scenarios with less than 40 targets. The conclusion is that there is not much difference between methods on the time-to-complete MOE. Section 7 also gives some results on estimating tour length.

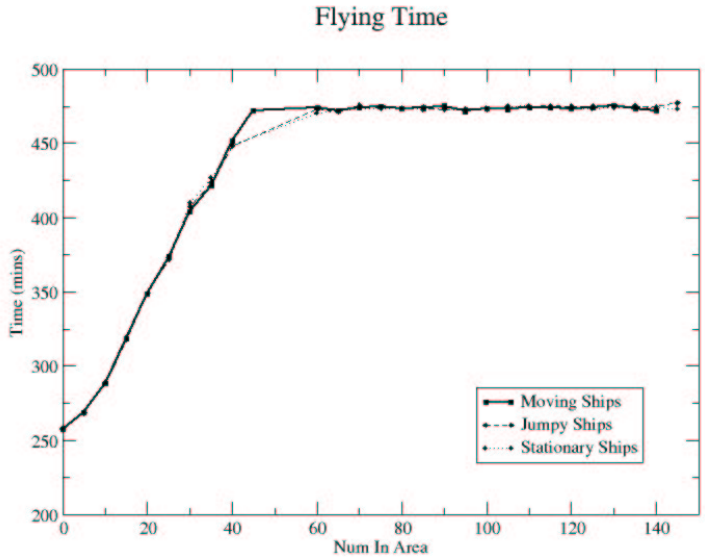

Figure 9: Total flying time – target speed 10 kn.

5.3 Execution time

The final question from DSTO regarded the execution time of the methods. The methods were tested on a Linux system with 2 Intel Pentium III-based CPUs, each running at 2.13 GHz and 2048 Mb cache. Figure 10 shows the average time to calculate a new route when a new target is detected. Note that a 5-second maximum time was in force during execution.

All the methods run fairly quickly, and the 5-second maximum was seldom reached. The full moving ships code runs faster than jumpy ships. This means that the time it takes for a jumpy-ships solution to converge is more than the time saved by the faster check. Hence (again), jumpy ships is not recommended. The full moving-ships code is still quite fast enough - even for the larger problems. An answer is usually returned in less than a second. These results indicate that execution time may not be important in a standalone system. However, the effect of the execution times will need to be tested in the context of repeated runs within the larger DSTO maritime surveillance model.

6 Determining the heading to intercept a moving ship

In this section we consider the problem of which heading a plane should take in order to investigate a moving ship. While easy to solve in its simplest form, the solution becomes complicated when the physical restrictions of a finite turning circle are included and when the plane only needs to get within a certain radius of the ship in order to investigate it.

For clarity we initially consider the simple problem of the straight line intersection of the plane and ship trajectory, since this provides a gentler introduction to the methodology. We then generalise this to when the plane only needs to come within a certain distance r of the ship (the classification distance) and then to the case where the plane is only able to

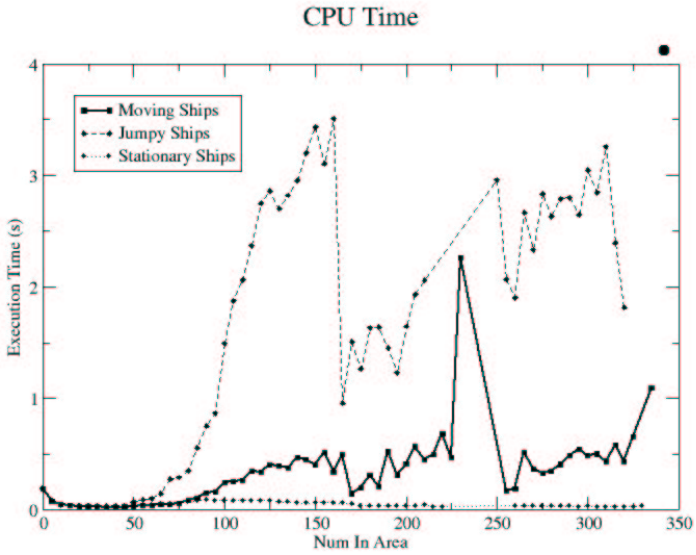

Figure 10: Execution time.

undertake a finite turning circle with radius r_c. The plane is assumed to have a constant speed v, and initial heading ϕ_i, with the ship moving with constant speed w, fixed heading θ and initial position (x_0, y_0). The aim is to find the optimal heading of the plane ϕ, and the resultant time to intersection, t_f. We assume the plane is at $(0,0)$ when $t = 0$, the time when it decides to intercept. This is illustrated in Figure 11. The angles ξ_0 and ξ_1 are the parameter angles representing position around the turning circle at the beginning and end of the turn. The angle η is the parameter representing position around the detection circle.

6.1 Straight interception

Here we assume the plane has no turning circle and that it needs to intercept the path of the ship exactly. The plane is assumed to initially be at $(0,0)$. The interception point of the plane and ship is when

$$\begin{aligned} x_0 + wt\cos\theta &= vt\cos\phi \\ y_0 + wt\sin\theta &= vt\sin\phi \end{aligned} \tag{1}$$

which we have to solve for t and ϕ. Squaring and adding both equations gives

$$(x_0 + wt\cos\theta)^2 + (y_0 + wt\sin\theta)^2 = v^2t^2 \tag{2}$$

which, in anticipation of later solutions, is written as

$$\begin{aligned} &\alpha t^2 + \beta t + \gamma = 0, \quad \text{where} \\ &\alpha = w^2 - v^2, \quad \beta = 2x_0 w\cos\theta + 2y_0 w\sin\theta, \quad \gamma = x_0^2 + y_0^2, \end{aligned} \tag{3}$$

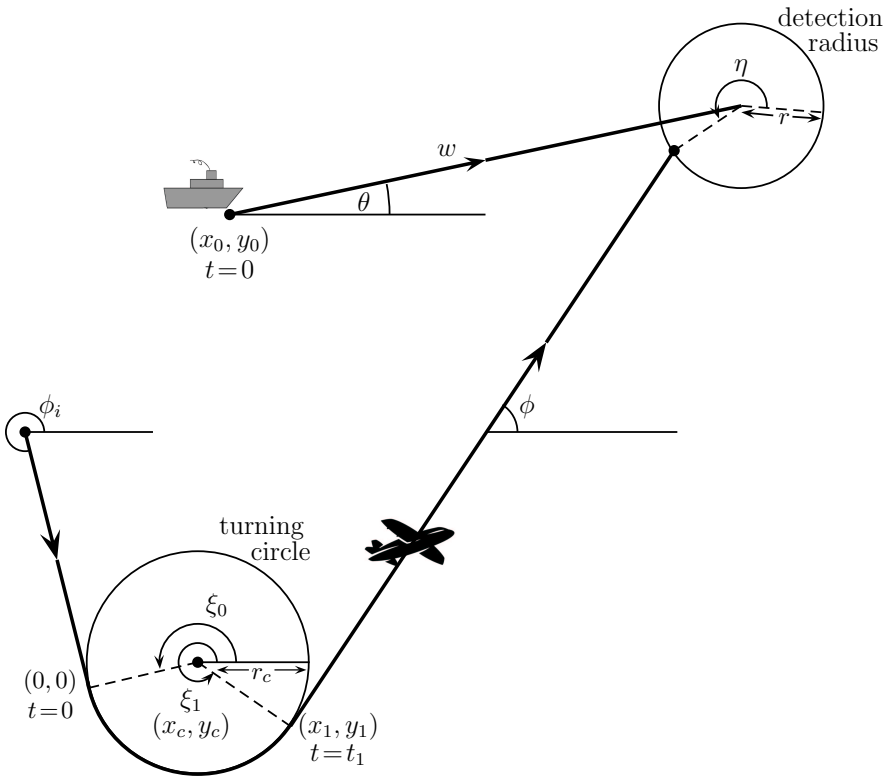

Figure 11: Illustration of a plane turning to intercept a ship. The plane is initially on a heading ϕ_i, moving with velocity v, and begins turning at (0,0) at $t = 0$ with a turning circle of radius r_c which has centre (x_c, y_c). It ends the turn at $t = t_1$ at (x_1, y_1) and moves along a heading ϕ until it detects the ship at the end of its detection radius r. The ship moves with velocity w along a heading θ.

giving the obvious solutions

$$t = \frac{-\beta \pm \sqrt{\beta^2 - 2\alpha\gamma}}{2\alpha}, \quad \phi = \cos^{-1}\left(\frac{x_0 + wt\cos\theta}{tv}\right). \tag{4}$$

In equation (4), $\alpha < 0$ since the plane is faster than the ship, and $\gamma \geq 0$, so the equation always has real roots. In addition, $\sqrt{\beta^2 - 2\alpha\gamma} \geq \text{abs}(\beta)$, so there is at exactly one positive solution for t.

6.2 Interception including detection zone and turning circle

In order to consider the effect of a non-zero classification distance, a simulation system was written in MATLAB that looked at flying between two way-points, with a variable number of targets to be visited along the way. A MATLAB library routine using a genetic algorithm was used to solve the routing problem.

For around 10 intermediate visits, the route length was of the order of 2.5 to 3 times the length of the straight line path linking the way-points (i.e. the default flight path). Further simulation runs suggested that this path may be reduced by up to 25% in length if the classification range was extended to 20 nautical miles.

This emphasises that a large classification range can have a very beneficial effect on flight distances if it is fully exploited.

Since the plane only has to come within a distance r of the ship in order to investigate it, the plane path $x(t), y(t)$ must intersect the circle around the ship defined by

$$\begin{aligned} x(t) &= x_0 + wt\cos\theta + r\cos\eta \\ y(t) &= y_0 + wt\sin\theta + r\sin\eta \end{aligned} \tag{5}$$

where η is the angle parameter defining the detection circle as shown in Figure 11.

The position of the plane is governed by three equations:

- for $t \leq 0$, prior to the plane beginning to turn,

$$x = vt\cos\phi, \quad y = vt\sin\phi; \tag{6}$$

- for $0 \leq t \leq t_1$, when the plane is turning,

$$x = x_c + r_c\cos\left(\frac{vt}{r_c} + \xi_0\right), \quad y = y_c + r_c\sin\left(\frac{vt}{r_c} + \xi_0\right) \tag{7}$$

  where (x_c, y_c) is the centre of the turning circle, ξ_0 is the parameter angle at the start of the turning circle when $t = 0$;

- for $t > t_1$, when the plane is on a heading to the ship,

$$x = x_1 + v(t - t_1)\cos\phi, \quad y = y_1 + v(t - t_1)\sin\phi. \tag{8}$$

  where (x_1, y_1) are the coordinates around the circle when the plane has finished turning.

Some of these variables are easily determined. For example, setting $t = 0$ in equation (8) gives the centre of the turning circle as

$$x_c = -r_c\cos\xi_0, \qquad y_c = -r_c\sin\xi_0. \tag{9}$$

The angle around the turning circle when the plane begins to turn is ξ_0 and ξ_1 when it finishes its turn, so by geometry

$$\xi_0 = \phi_i - \frac{\pi}{2}, \qquad \xi_1 = \frac{3\pi}{2} + \phi. \tag{10}$$

We note that these equations assume a counter-clockwise turning circle in the geometry given in Figure 11 and code can easily be implemented to account for a variety of geometries although care must be taken to program all geometries carefully. Hence, similar equations can be derived for the clockwise turning situation. The time t_1 that the plane finishes turning is

$$t_1 = (2\pi + \phi - \phi_i)\frac{r_c}{v} \tag{11}$$

at position (x_1, y_1) given by equations (7) with $t = t_1$. Note again that for some geometries the additional 2π is not necessary.

Thus equating (8) and (5) and using equations (7) and (11) gives

$$x_0 + r\cos\eta + wt\cos\theta = x_c + r_c\sin\phi + v(t - t_1)\cos\phi \tag{12}$$

$$y_0 + r\sin\eta + wt\sin\theta = y_c - r_c\cos\phi + v(t - t_1)\sin\phi \tag{13}$$

where our unknowns are the plane heading ϕ, the intersection time t, and the detection circle angle η with $t_1(\phi)$ given in equation (11) and $x_c, y_c, \theta, r, r_c, v, w$ all known quantities. This system is solved for $t(\eta), \phi(\eta)$ by squaring and adding the equations, in much the same way as equation (1), and then η is determined by minimising $t(\eta)$.

Hence, the equation analogous to (2) is,

$$(x_0 + r\cos\eta - x_c + wt\cos\theta)^2 + (y_0 + r\sin\eta - y_c + wt\sin\theta)^2 = r_c^2 + v^2(t - t_1)^2 \tag{14}$$

which is a quadratic in t, as in equation (3), with

$$\alpha = w^2 - v^2, \tag{15}$$

$$\beta = 2(x_0 + r\cos\eta - x_c)w\cos\theta + 2(y_0 + r\sin\eta - y_c)w\sin\theta + 2v^2 t_1 \tag{16}$$

$$\gamma = (x_0 + r\cos\eta - x_c)^2 + (y_0 + r\sin\eta - y_c)^2 - r_c^2 - t_1^2 v^2. \tag{17}$$

The resultant solution $t(\phi)$ is substituted into equation (12) to give an implicit equation for ϕ which must be solved numerically. Because η is still not determined for this most general case, a numerical solution involves iterating over all $\eta \in [0, 2\pi]$ and solving the resultant ϕ numerically for each η before choosing the value of η which minimises t. Whilst this is relatively easy to program, a simpler approximate solution would be desirable.

6.3 Approximate solutions when $y_0 >> r, r_c$

A simpler approximate solution is possible when the distances between the ship and plane are all large relative to the detection radius and the turning circle. In the limit when $r, r_c \to 0$ we naturally recover the simple straight intersection solution outlined earlier. Without loss of generality the problem can be rescaled with $x_0 = 0$ and $y_0 = 1$ hence allowing us to write $r = \epsilon$ where ϵ is a small parameter. Once the perturbation solution is found it is a simple matter to program the re-scaled solution. Thus we assume a regular perturbation system with

$$\begin{aligned} \phi &= \phi^{*0} + \epsilon\phi^{*1} + \cdots, \quad t = t^{*0} + \epsilon t^{*1} + \cdots, \\ r_c &= r_c^*\epsilon, \quad x_c = x_c^*\epsilon, \quad y_c = y_c^*\epsilon, \quad t_1 = t_1^*\epsilon, \end{aligned} \tag{18}$$

since we expect r_c, x_c, y_c, t_1 all to be of similar small order.

The zeroth order solution to t^{*0} and ϕ^{*0} is the straight intersection solution given earlier. The $O(\epsilon^1)$ solution is

$$\begin{aligned} &\begin{bmatrix} w\cos\theta - v\cos\phi^{*0} & t^{*0}v\sin\phi^{*0} \\ w\sin\theta - v\sin\phi^{*0} & -t^{*0}v\cos\phi^{*0} \end{bmatrix} \begin{bmatrix} t^{*1} \\ \phi^{*1} \end{bmatrix} \\ &\qquad = \begin{bmatrix} x_c^* + r_c^*\sin\phi^{*0} - t_1^* v\cos\phi^{*0} - \cos\eta \\ y_c^* - r_c^*\cos\phi^{*0} - t_1^* v\sin\phi^{*0} - \sin\eta \end{bmatrix} \end{aligned} \tag{19}$$

which can be inverted to find t^{*1} and ϕ^{*1}. However, the value of η still needs to be evaluated. This can be done by minimising t as outlined earlier, but this has to be done numerically. As we are only concerned with a simple approximation we take $\eta \approx \pi + \phi^{0*}$ which is the case for slow moving ships.

Figure 12 shows this approximation along with the exact solution for a specific case where the ship initial position is $(x_0, y_0) = (0.5, 1)$, ship heading is $\theta = \pi/8$ with speed $w = 2$, the plane velocity is $v = 4$ with turning circle $r_c = 0.1$, and detection radius $r = 0.25$, all in non-dimensional units. Initially the plane is on a heading $\phi_i = 3\pi/2 + \pi/6$. Even for this relatively large detection radius the perturbation solution is a good approximation.

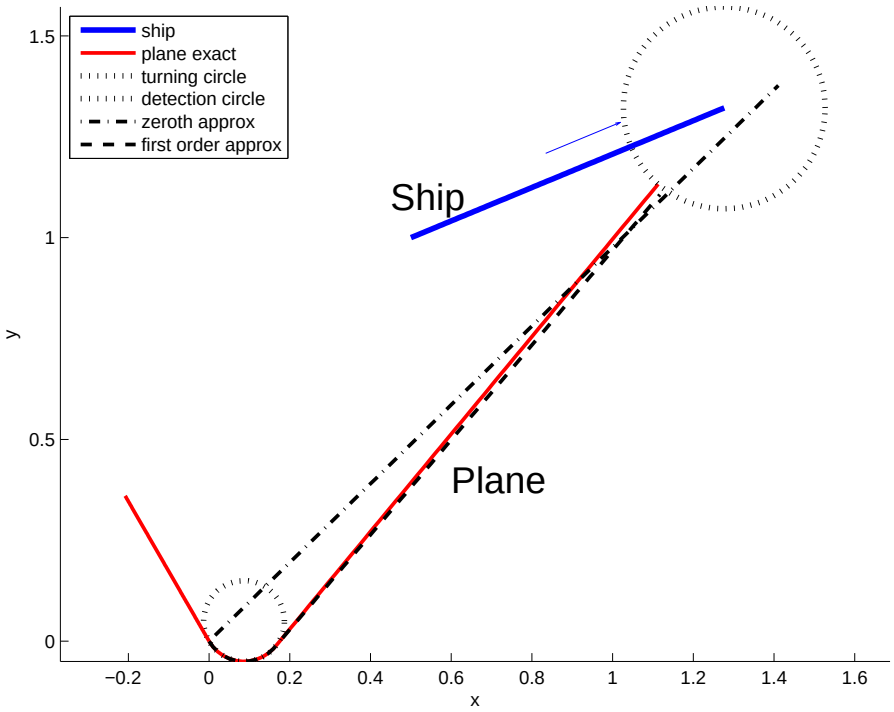

Figure 12: Illustration of the exact and perturbation solutions for a plane intersecting a ship with parameters $(x_0, y_0) = (0.5, 1)$, $\theta = \pi/8$, $w = 2$, $v = 4$, $r_c = 0.1$, $r = 0.25$ and $\phi_i = 3\pi/2 + \pi/6$.

7 Estimating tour length

There is a theoretical result [19] that gives bounds on the shortest travelling salesman tour length of a standard TSP on a unit square with n cities, in the worst case the tour length is

$$\alpha^* \sqrt{n} + o(\sqrt{n}) \text{ where } 1.075 \leq \alpha^* \leq 1.414 \tag{20}$$

We are motivated by the above result to generate empirical estimates for the tour length under perfect and incomplete information. Under perfect information the problem is simply to generate a tour on the set of known targets. With incomplete information, a tour is generated on the current set of detected targets and is updated each time new targets are detected.

A simulation program was written for this experiment in MATLAB. The program solves a TSP problem using a genetic algorithm. The TSP algorithm was sourced from the MATLAB Central file exchange.

The problem instances for estimating tour length all have an AI with dimension 400×100 n mile. The coordinates of the two way-points are $w_1 = (50, 50)$ and $w_2 = (350, 50)$. These way-points represent the start and end points for a tour.

For the air platform with perfect information, the detect radius is ∞ and classification radius is 0. In the incomplete information case, we have detect radius 50. The classification radius is still set at 0. The vessels in each instance are all stationary and are assigned random

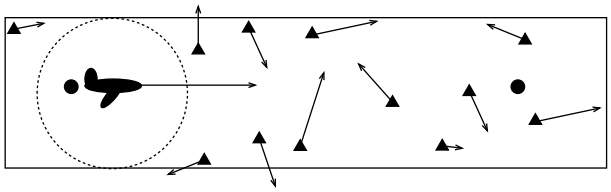

Figure 13: The solid rectangle is the AI and the way-points are shown by filled circles. The large dashed circle denotes the detection radius about the platform at its current position and the vessels are denoted by triangles with arrows representing velocity vectors.

	Perfect Info		Incomplete Info		
N ships	μ_t	σ_t	μ_t	σ_t	$\bar{N}(\%)$
5	427.03	52.36	400.76	47.85	87.8
10	557.47	60.85	510.32	66.84	90.5
15	672.81	70.19	617.20	72.73	92.6
20	777.72	78.20	734.62	81.77	95.0
25	879.27	79.07	834.60	83.54	96.4

Table 2: Results for estimating tour length

positions in the AI according to a uniform distribution. The experiment is run with number of vessels $N = 5, 10, 15, 20, 25$. The experiments were repeated for 1000 instances of each parameter combination. The results are presented in Table 2.

The entries in Table 2 show the means and standard deviations of the tour lengths in the perfect and incomplete information cases. The final column of the table shows the average proportion of the targets that are classified. The results show that the average tour length with perfect information is greater than the tour length with incomplete information. This can be attributed to some of the targets being missed in the incomplete information case. Table 2 also shows that the number of targets detected and classified increases with the target density.

8 What proportion of ships do we detect?

To gain a better perspective on how effective a search pattern is in detecting ships we consider how many ships enter the search area versus how many are detected. For example, Figure 14 shows a plane's circular detection region moving from left to right with a random distribution of ships moving in and out of the search region with a distribution of velocities. In the simulations shown here the plane moves left to right with speed 150 kn, the plane detects ships within a 50 n mile radius, and the ships were scattered over a 300 by 300 n mile square region.

If we take the ship velocities as $\mathbf{v}_i(t)$, $i = 1, \ldots, n$; the detection radius is r, and $\mathbf{p}(t)$ the plane path then the proportion of ships detected is

$$q = \frac{\sum_{i=1}^{N} H\left(\int_0^T 1 - H(|\mathbf{v}_i - \mathbf{p}| - r)dt\right)}{\sum_{i=1}^{N} H\left(\int_0^T 1 - H(|\mathbf{v}_i| \in R)dt\right)}$$

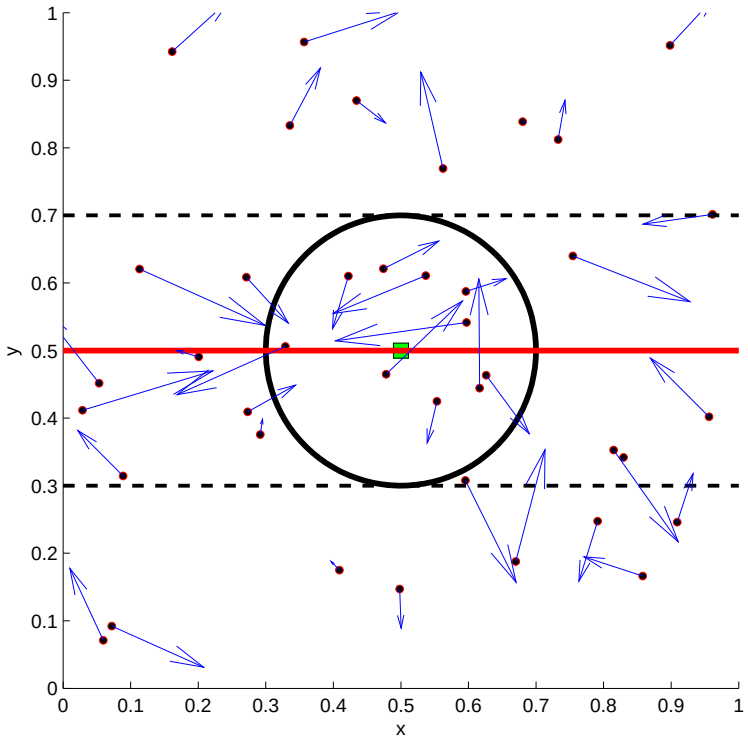

Figure 14: A plane moving left to right has a circular detection region within the rectangular search region. Various ships move in and out of the search region hence only a proportion of ships will be detected.

where R is overall search region and H is the Heaviside function, that is $H(x) = 1$ if $x \geq 0$ and $H(x) = 0$ if $x < 0$. Hence if ship i enters the detection region then $|\mathbf{v}_i - \mathbf{p}| - r < 0$ so that $1 - H(|\mathbf{v}_i - \mathbf{p}| - r) = 1$ when a ship is detected. By integrating this over all times and applying a second Heaviside function means that the numerator is the number of ships that are detected. The denominator is similar with $H(|\mathbf{v}_i| \in R) = 0$ if a ship is within the search zone. Thus q is a mathematical way of expressing a rather simple numerical calculation of counting the number of ships entering the space and the number that is counted.

In Figure 15 we show the proportion of ships detected, q, for 1000 different simulations and a histogram of this proportion. Here the ship velocities were normally distributed with mean 15 kn and variability $\pm$ 1 standard deviation. The ships were scattered in a uniform random distribution with random direction.

Figure 16 shows the mean of the proportion detected as a function of the mean ship velocity. Different ship velocity distributions are compared, with the ships having either a uniform distribution of velocities, a constant velocity or a normal distribution with two different variances. As expected when the velocity of the ships increases then less are detected – since more ships move in and out of the region without being spotted. If all the ships have velocity $\mathbf{v}_i = 0$ then all will be detected giving $q = 100$ %. If the ships have velocities with a larger standard deviation, then the proportion falls, since with more ships move out of the region before being detected.

Figure 17 shows how the proportion detected changes with the variation in the ship

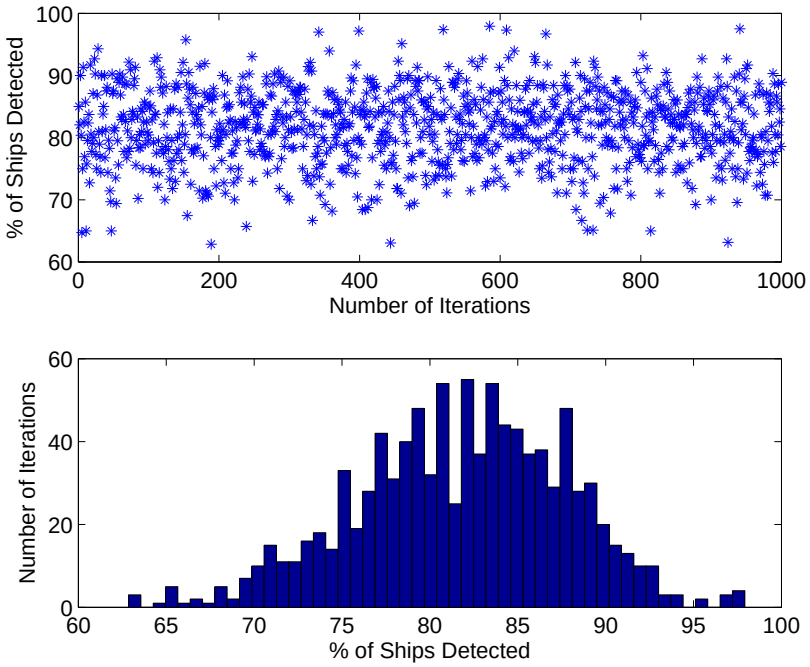

Figure 15: Proportion of ships detected for 1000 simulations. The lower plot shows the histogram for all simulations.

velocities. Shown is the mean proportion and upper and lower bounds represented by the 25 and 75 % levels on the detection distribution. In all cases the mean ship speed was 15 kn. As the variation in ship velocities increases, so the proportion detected decreases, as discussed in Figure 15. Not obvious however, is why the proportion detected has slope near zero for low variation and then changes to uniform slope for higher variations.

The simple simulations shown above are illustrative only, with the aim of showing the general trends in detection rate with ship velocity. However, similar simulations can be done with realistic flight paths, to estimate how effective a plane's path is as measured by detection rate. Further work needs to be done on this to obtain mathematical results which allow prediction of these simulation results.

9 Conclusions

The group looked at a number of issues concerned with route planning within the context of maritime surveillance. The routing problem is very complex. The factors to be considered include moving targets, dynamically changing data, time windows, precedence constraints, intercept points, and demand prediction.

A simulation system was developed which went some way to answering the main questions posed by DSTO.

- For the typical scenarios faced by the RAAF, ships can be treated as stationary, but slightly better results (in terms of number of classifications and distance travelled) can

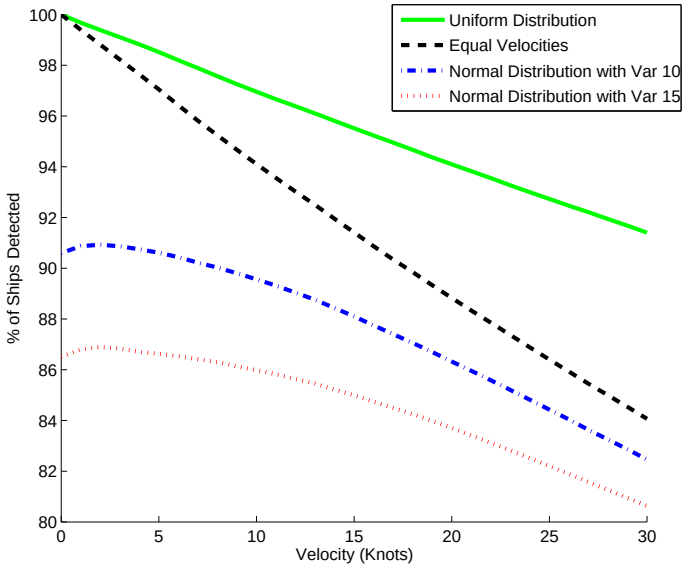

Figure 16: Mean proportion of ships detected as a function of mean ship velocity.

be obtained by more realistic modelling. However, modelling suggests that with only a few fast-moving ships, the detection rate can be badly affected. Further investigation is required to test this effect.

- Flying time can be improved with better algorithms. With standard TSP algorithms, these gains are modest; but we would expect that better treatment of the online aspects of the problem would yield better results.
- The computation times for all the methods tried were quiet modest (for the typical problem sizes used), but the effect needs to be tested in the context of the larger model within which the search model sits.

We also examined some additional areas of interest. In particular, the flight paths which account for turning circles and non-zero classification distances. The effect of large classification distances is particularly interesting, as large savings seem to be possible.

This study indicates that, while some questions have been answered, more questions have been raised. These suggest that further study in a number of areas would help produce better answers to the problems seen in maritime surveillance - particularly better handling of the online aspects of the problem, exploiting large classification distances, and investigating the effect of fast-moving ships.

Acknowledgements

Thanks very much to the industry representatives, David Marlow and Jason Looker from the Defence Science and Technology Organisation, Fishermens Bend, Victoria. This report captures the input from a large group of people who contributed during the week. The organisers and industry representatives would like to thank those who participated – including

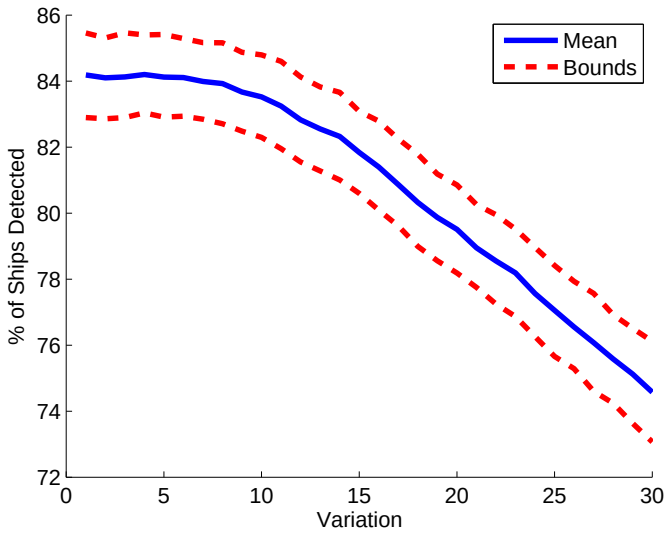

Figure 17: Proportion of ships detected as a function of variation in ship speeds. The ships have a normal distribution of speeds with mean 15 kn and variation σ shown on the x axis.

Jos Beunen	Robert McKibben	Judy Shand
Jonathan Crook	Geoff Mercer	Lauren Coulter Smith
Scott Greybill	Phil Neame	Virginia Wheway
Steve Ha	Sarah Neville	Richard White

References

[1] Balas, E. (1989) The prize-collecting traveling salesman problem, *Networks*, **19**, 621–636.

[2] Bent, R. & Van Hentenryck, P. (2005) Online stochastic optimization without distributions, In *ICAPS'05, Proc. 15th Int. Conf. Auto. Plan. & Sched.*

[3] Chen, X-L, and Xu, H. (2006) Dynamic column generation for dynamic vehicle routing with time windows, *Trans. Sc.*, **40**, 74–88.

[4] Feillet, D., Dejax, P. & Gendreau, M., (2005) Traveling salesman problems with profits, *Trans. Sc.*, **39**, 188.

[5] Gendreau, M., Guertin, F., Potvin, J-V. & Seguin, R. (2006) Neighborhood search heuristics for a dynamic vehicle dispatching problem with pick-ups and deliveries *Trans. Res. Part C: Emerg. Tech.*, **14**, 157-174.

[6] Grob, M. (2006) Routing of platforms in a maritime surface surveillance operation *Euro. J. Oper. Res.*, **170**, 631-628.

[7] Gulczynski, D., Heath, J. & Price, C. (2006) The close enough traveling salesman problem: A discussion of several heuristics, In Alt, F.B., Fu, M.C, & Golden, B.L. editors, *Pers. Oper. Res.*, **36**, 271-283.

[8] Gutin, G. & Punnen, A. editors. (2002) *The Traveling Salesman Problems and its Variations*, Kluwer Academic Publishers.

[9] Hammar, M. & Nilsson, B. (1999) Approximation results for kinetic variants of TSP. In Wiedermann, j. et al., editor, *ICALP'99 Proc. Auto. Lang. Prog.: 26th Inter. Coll.*, **1644**, 392-401.

[10] Helvig, C., Robins, G. & Zelikovsky, A. (1998) Moving-target tsp and related problems In Bilardi, G., Italiano, G., Pietracaprina, A. &, Pucci, G. editors, *Algorithms - ESA '98: 6th Ann. Euro. Symp. Proc.*, **1461**, 453-464.

[11] Hvattum, L., Løkketangen, A. & Laporte, G. (2006) Solving a dynamic and stochastic vehicle routing problem with a sample scenario hedging heuristic *Trans. Sc.*, **40**, 421-438.

[12] Jaillet, R. & Wagner, M. R., (2006) Online routing problems: Value of advanced information as improved competitive ratios *Trans. Sc.*, **40**, 200-210.

[13] Jiang, Q., Sarker, R. & Abbass, H. (2004) Tracking moving targets and the non-stationary traveling salesman problem, In *Proc. 8th Asia Pac. Symp. Intell. Evol. Sys.*

[14] Miele, A. Weeks, M. & Ciarcià, M. (2007) Optimal trajectories for spacecraft rendezvous *J. Opt. Theor. Appl.*, **132**, 353-376.

[15] Or, I. (1976) Travelling Salesman-Type Combinatorial Problems and Their Relation to the Logistics of Blood-Banking PhD thesis, Northwest Uni. USA.

[16] Savelsbergh, M. (1985) Local search in routing problems with time windows *Ann. Oper. Res.*, **4**, 285-305.

[17] Schumacher, C., Chandler, P., Pachter, M. & Pachter, L. S. (2006) Optimization of air vehicles operations using mixed-integer linear programming *J. Oper. Res. Soc.*, **58**, 516-527.

[18] Solomon, M. (1987) Algorithms for the vehicle routing and scheduling problem with time window constraints *Oper. Res.*, **35**, 254-265.

[19] Supowit, K.J., Reingold, E.M., & Plaisted, D.A. (1983) The travelling salesman problem and minimum matching in the unit square *SIAM J. Comput.*, **12**, 144-156.