Search and Rescue Optimal Planning System

Thomas M Kratzke
Lawrence D Stone
Metron, Inc
Reston, VA, U.S.A.
stone@metsci.com
kratzke@metsci.com

John R Frost
U.S. Coast Guard
Office of Search and Rescue
Washington, DC 20593
John.R.Frost@uscg.mil

Abstract – In 1974 the U.S. Coast Guard put into operation its first computerized search and rescue planning system CASP (Computer-Assisted Search Planning) which used a Bayesian approach implemented by a particle filter to produce probability distributions for the location of the search object. These distributions were used for planning search effort. In 2003, the Coast Guard started development of a new decision support system for managing search efforts called Search and Rescue Optimal Planning System (SAROPS). SAROPS has been operational since January, 2007 and is currently the only search planning tool that the Coast Guard uses for maritime searches. SAROPS represents a major advance in search planning technology. This paper reviews the technology behind the tool.

Keywords: search, Monte Carlo simulation, particle filters, optimization, Bayesian

1 Introduction

The U.S. Coast Guard's Search and Rescue Optimal Planning System (SAROPS) is the successor to the Computer-Assisted Search Planning (CASP) System. Both programs use a Monte-Carlo based simulator (particle filter) for developing probability distributions (maps) for the location of objects missing at sea, and both programs aid in search planning.

Based on the principles in [1], CASP [2] was developed in the early 1970s and deployed in 1974. Since CASP was developed, computer capability has improved dramatically which has allowed SAROPS to use more accurate environmental and object motion models in its simulation and to recommend operationally feasible "optimal" search plans. SAROPS provides a greatly improved user interface that presents results on geographic maps which incorporate environmental data products not available when CASP was developed. Development on SAROPS started in October, 2003, and version 1.0 was deployed in early 2007.

1.1 Basic Components of SAROPS

There are four basic components of SAROPS: The Environmental Data Server (EDS), simulator (SIM), search Planner, and Graphical User Interface (GUI).

EDS. SAROPS requires environmental estimates in order to account for possible drift of the search object and to estimate the detectability of the object by various search sensors, e.g., visual and radar. Estimates of ocean currents and winds are needed to account for drift and leeway of search objects. Wave height, cloud cover, sun, and rain all affect the detectability of search objects. Water temperature is used to estimate survival times. The EDS provides the data on appropriate spatial and temporal grids to cover the area and time period of interest.

SIM. The simulator uses information about the time and last known position of the search object and information about its intentions, to produce probability distributions (maps) for the object's location using a particle filter where each particle represents a possible path for the search object. The particle filter takes into account drift using estimates of winds and currents provided by the EDS. SIM includes information about the object's intended path, areas where trouble is likely to occur, areas where searches have already occurred, and other considerations. It incorporates information from unsuccessful searches in a Bayesian fashion. This information is used to produce a probability distribution on the particles i.e., paths. The collection of particles and their weights define the object location distributions as a function of time. The distribution for a selected time is displayed as a set of rectangular cells with the probabilities associated with each cell indicated by a color scale.

Planner. In the typical cycle, SIM produces a probability distribution for the object's location at the time of the next search. The Planner uses this distribution along with a list of assigned search assets to produce operationally feasible search plans that maximize the increase in probability of detecting the object.

After the search takes place, and if it is unsuccessful, SIM will produce a posterior probability map for object location that accounts for the unsuccessful search and the possible motion of the object. This distribution provides the basis for planning the next increment of search.

In subsequent sections, we describe SIM and the Planner. The EDS and GUI are not discussed in detail in this paper.

2 SIM

The U.S. Coast Guard planning process calls for information about the location and movement of the search object to be collected into scenarios. The information is often inconsistent but tends to form itself into sets of consistent information that we call scenarios. Each scenario tells a "story" about what may have happened to the object.

The simplest scenario is defined by a last known position (LKP) and a time which specify where and when the distress incident is believed to have occurred. It is useful to consider this example for the discussion on scenarios in the rest of this section. The next section discusses this and other scenarios in more detail.

Uncertainties in the scenario information are quantified by probability distributions. On the basis of these uncertainties and probabilistic models of drift, SIM produces a Monte Carlo set of paths for each scenario. Each path has a weight or probability assigned to it. These paths comprise a discrete sample path approximation to the stochastic process that represents the object's position and motion over time. Note the sample paths or particles move through time and space continuously. It is only the space of sample paths that is discrete.

Each scenario is given a weight by the user and the weights add to 1.0. The user can choose to have 2,500, 5,000, or 10,000 particles per scenario. The collection of weighted scenarios forms the prior distribution for object location and type. Type is discussed later. Information from unsuccessful searches is incorporated into the weights on the particles in a Bayesian manner as discussed below. The updated weights or probabilities on the particles form the posterior distribution that is used by the Planner.

Probability maps are displayed in grid cells in the SAROPS' GUI, but the particles and their weights form the actual distributions, and these are used by the Planner.

There are many parts to SIM, but we will concentrate on the notions of scenario, hazard, previous searches, and the effects of the currents and winds on the distribution of the particles.

2.1 Scenario and Pre-Distress Motion

SIM considers two types of motion, pre-distress and drift. The former models what the object did while it was in control and navigating, and the latter models the motion of the object when it has no power and simply moves according to the currents and winds. SIM models both types of motion, as well as the transition from pre-distress to drift.

The pre-distress motion is modelled by scenarios. SAROPS has several templates, called scenario-types, which the user can choose for defining a scenario. Each scenario-type has a set of parameters whose values must be specified.

In an LKP scenario, there is no pre-distress motion. The parameters specify the starting position and time of the drift. Hence, the only parameters for an LKP scenario are the position, position uncertainty, time of distress, and the time uncertainty. SIM uses the position and position uncertainty to create a bivariate normal distribution and random draws are made from this distribution for the starting position of the drift. In addition a normal distribution for distress incident time is created from the time uncertainty parameters. For each position draw, a draw is made from the time distribution. These draws determine when and where a particle starts its drift.

A second scenario-type is the "Area Scenario," or simply "Area." The difference between this scenario-type and the LKP, is that the initial position of the drift is drawn from a region bounded by a polygon and the distribution of the starting points is uniform over this region. Hence, the parameters for an Area Scenario are identical to those of an LKP scenario, except that the initial position is determined by the corners of a polygon. Again, there is no pre-distress motion.

In SIM, position draws from a uniform 2-dimensional polygon are made in the following way. Let us designate the two axes as *x* and *y*. For a polygon P, SIM computes the cumulative marginal distribution function along the *x*-axis and the inverse of this function. These functions are denoted cdf_x and cdf_x^{-1} respectively. For each x, SIM computes the cumulative distribution function along the *y* axis given *x* and its inverse; these are denoted $cdf_{y|x}$ and $cdf_{y|x}^{-1}$ respectively.

In this paper, all 1-dimensional uniform random variables are uniform over the interval $[0,1]$. Given the functions of the previous paragraph, and two independent uniform random numbers u_1 and u_2, SIM obtains the point (x_1, y_1) inside P by letting $x_1 = cdf_x^{-1}(u_1)$ and $y_1 = cdf_{y|x_1}^{-1}(u_2)$.

There are two reasons for using this method, rather than simply drawing a pair of uniform random variables from the smallest enclosing rectangle and discarding the resulting point it if it is not within P. The first reason is that P might be a tiny part of the enclosing rectangle. This would happen if P were used to model a small strip of water surrounding an island. In this case, a great many draws would be necessary to create a single draw within the polygon, and performance would suffer.

The second reason involves the scenario-type voyage, which is the next scenario-type discussed. That scenario-type requires some "correlation" between two position draws, and the mechanism set up above allows us to incorporate this correlation.

Note that the use of inverse cumulative distribution functions, as outlined above, will always produce correctly distributed draws for positions, no matter what the underlying distribution is. Here, it was used for a uniform polygon, but it will work for any distribution for which cdf_x^{-1} are $cdf_{y|x}^{-1}$ are available. In fact, SIM uses this technique to make bivariate normal draws.

The voyage scenario-type is the first one for which there is pre-distress motion. In the manner discussed in section 2.2 below, a random time is drawn for each particle to determine when pre-distress motion ends and drift begins. Drift is modelled as with the previous two scenarios.

The pre-distress motion of a particle in a voyage scenario is determined as follows. A voyage is defined by a set of regions determined by polygons or circles as shown in Figure 2.1. To draw a single path, SIM randomly selects points from these regions and connects them.

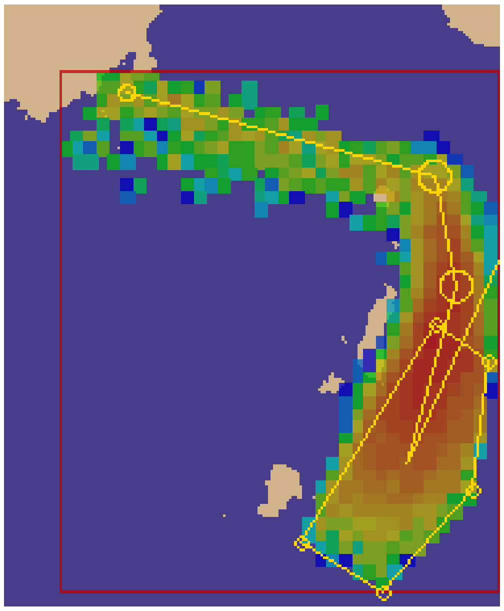
Figure 2.1. Voyage Scenario

It is important that the draws from each of these regions correctly adhere to their respective distributions. However, these draws cannot be independent of each other. If they were, there would be an unfortunate and unintended effect called "hour-glassing." To illustrate, consider Figure 2.2.

In this figure, random points are independently drawn from the two regions and connected. Since roughly half the time the draws will cross over the middle, this produces an inordinately high density of "traffic" near the center of the path connecting the two regions. To avoid this, SIM uses *cdf*s, as discussed above. The x and y in that discussion are perpendicular and parallel to the line connecting the centers of mass of the two distributions. This allows SIM to follow the principle "if a draw starts to the right, it generally stays to the right" and thus avoid hour-glassing. Still, the marginal distribution of the draws within any particular region follows its prescribed distribution.

The critical correlation here is in the 1-dimensional component that is perpendicular to the line connecting the centers of mass of the two areas. SIM changes coordinates and deals with the random variables expressed in those coordinates. The correlated draws in these coordinates are implemented by using two uniform draws and the *cdf*s discussed above.

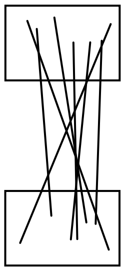
Figure 2.2: Hour-glassing

A parameter within the code determines how strictly SIM will adhere to the "starts-to-the-right-stays-to-the-right" concept. This parameter can be set anywhere from -1 to 1. If it is set to -1, a particle that starts to the right goes to the left; set to 1, it stays exactly "as far to the right" as it started. The parameter is currently set to 0.7.

A distribution resulting from a voyage scenario is shown by the color coded cells in Figure 2.1 with red cells indicating high probability and shading down to blue cells indicating low probability. Note the effect of drift on particles that transition to drift motion early in the scenario.

SIM has one more scenario-type, the LKP plus dead-reckoning scenario-type which will not be discussed here. Other scenario-types are being considered and developed. In the most recent version of SIM, 1.3.0.0, there is an LOB (or line-of-bearing) scenario-type, which will be used for flare sightings as well as radio transmission receptions. Models for "swimmer" and "sailboat" are being considered for inclusion in future versions of SIM.

2.1.1 Scenario Weights

Scenarios can be weighted and combined in SIM. Scenarios with heavier weights will contribute more to the final distribution; the particles within these scenarios, although the same in number, carry greater weight and cause the probability distributions to emphasize their positions more than those from scenarios having lighter weights.

2.2 Hazards and Time of Distress

In scenarios where there is pre-distress motion, SIM must determine the time at the distress occurs and the drift begins. After creating a path for the pre-distress motion as described above, SIM computes how much time it would take to complete the path. In the absence of hazards, a uniform draw over that time is made to place the time of distress. The position along the particle's path corresponding to that time is computed to obtain the distress incident position.

Another part of SIM's model is the notion of a hazard. This expresses the idea that there are times and places where distress is more likely to occur. A hazard is defined by a region, an effective time interval, and an intensity. The hazard is in effect in the region only during the effective time interval.

The intensity determines how likely it is that a particle going through the hazard during the effective time interval will go into distress, Currently, there are four intensities available in SAROPS; 1, 3, 5, and 10. The greater the intensity, the greater the chance that the particle will go into distress while in the hazard.

SIM determines when a particle goes into distress as follows. Consider a hazard defined by a region H with hazard intensity κ. Assume that a path T starts at $t_0 = 0$, and that t_1 is the time that the path enters H, t_2 is the time that the path leaves H, t_3 is the time of the end of the path, and that $[t_1, t_2]$ is contained in the effective time interval for the hazard. The time of the distress is obtained as a draw from the distribution with the density function

$$f(t) = \begin{cases} a & \text{for } t \in [0, t_1] \cup [t_2, t_3] \\ \kappa a & \text{for } t \in (t_1, t_2) \\ 0 & \text{otherwise} \end{cases}$$

where a is chosen so that f integrates to 1.

Note that $\kappa = 1$ is equivalent to no hazard. Furthermore, intensities are multiplicative. If H_1 is a hazard with intensity κ_1 and H_2 is a hazard with intensity κ_2, and $H = H_1 \cap H_2$, then H acts as a hazard with intensity $\kappa_1 \kappa_2$.

2.3 Post-Distress Object-Type

When a particle goes into distress, it will have an object-type associated with it. This will determine both the effect that the winds and currents have on it, and the impact that previous searches have on the probability distribution for object location.

As an example, suppose a person is involved in a distress incident but it is not known whether the person is floating in the water or was lucky enough to get into a life raft. In the former case, the currents will have the most influence on the position of the object at a later time and the winds will have relatively little influence. But if the person is in a raft, and there are strong winds, the winds will have a substantial effect on the person's position.

Similarly, a search, even under excellent conditions, can easily overlook a person floating in the water. However, a search over an area with good visibility that fails to detect a raft substantially reduces the chances that there is a raft in that area. Hence an unsuccessful search will affect the probability distribution far more in the case of a raft than it will in the case of a person in the water because rafts are much easier to detect than persons in the water.

SIM allows the user to specify a probability distribution on the object type that results from a distress incident. These probabilities are scenario-dependent. SIM treats the random variables that determine the time of distress and the object type that results from the distress, as independent. For example, the probability of a particle transitioning to a raft type is the same whether the distress occurs early or late, in a hazard or out of a hazard.

2.4 Effects of Unsuccessful Search

As mentioned in the discussion on Scenarios, SIM assigns probabilities to the particles; particles that have high probability influence the probability distribution more than those with low probabilities. After an unsuccessful search, the posterior particle probabilities are computed using Bayes' rule. Particles that lie in regions that have been well searched, will tend to have lower posterior probabilities than ones that do not.

To perform the Bayesian update for unsuccessful search, SIM needs to know the path of the Search and Rescue Unit (SRU) performing the search, and the probability that the SRU can detect the object. The latter is determined from a Lateral Range Curve (LRC). Each SRU has an LRC for each object type. The LRC also depends on environmental conditions such as sea state.

The LRC is the function λ that gives the probability of detection as a function of the closest point of approach of the SRU to the search object on a "long" straight search leg. A discussion of the form of the lateral range curve in use by SAROPS can be found in [3]. The SRU's route is divided into legs, where each leg is the part of the route between consecutive waypoints. The closest point of approach (or CPA) of a particle to the leg of an SRU's is the smallest distance that the particle and the SRU are apart from each other during that leg. Computing this distance is a fairly simple geometry problem.

Consider a particle p and a search by an SRU consisting of K straight line legs. Let d_k be the distance at the closet point of approach of the SRU to p on leg k. Then $1 - \lambda(d_k)$ is the probability the SRU fails to detect the particle on the k^{th} leg. Different legs are considered to have independent detection opportunities so the probability of the SRU not detecting p is

$$pfail(p, sru) = \prod_{k=1}^{K} \left(1 - \lambda(d_k)\right). \qquad (1)$$

If there are multiple SRUs, then the *pfail* of the particle is computed by multiplying the failure probabilities for all SRUs to obtain

$$pfail(p) = \prod_{sru} pfail(p, sru). \qquad (2)$$

The prior probability of the particle p is multiplied by $pfail(p)$ and normalized in the usual Bayesian fashion to produce the posterior probability for p.

2.5 Environmental Effects on Drift

Once a particle enters a state of distress and has an object type associated with it, it moves as the currents and winds move it. At this point, all particles move the same way, regardless of their scenario-type. In this section, we describe how SIM simulates that movement.

There are two forces acting on a drifting particle; currents and winds. The effect of currents is more straightforward. Once a value for the velocity of the current is obtained, the particle's velocity due to the current is the equal the velocity of the current.

The data is received in the form of a grid of values in space and time. Each grid point has values for speed towards the east (called the *u*-speed) and the speed towards the north (called *v*-speed). These values are given for a set of times which are the same for all grid points.

If SIM needs to know the *u*-speed and the *v*-speed at one of these times, SIM takes the three closest grid points and uses a weighted average of the three values, where the weights are the inverses of the distances to the chosen grid points.

Usually SIM will need the *u*-speed and *v*-speed for times different than those given in the EDS data. To get the speeds in this case, SIM obtains the speeds for two closest times from the EDS and interpolates between these values.

SIM inserts a random effect on the resulting *u*-speed and *v*-speed. For every time step in the simulation, and every particle, SIM computes the *u*-speed and the *v*-speed, and perturbs them by using a random draw from a normal distribution. However, within a particle, these draws are not independent. In other words, the random draws for the *u*-speed of a particular particle p at one time-step, and the random draw for the *u*-speed for p at the next time step, are correlated. Specifically, if Δt is the difference in time, measured in minutes, between the two time steps, then the correlation, $\rho(\Delta t)$, is given by

$$\rho(\Delta\tau) = e^{-\alpha\Delta\tau}$$

where α is chosen so that $e^{-\alpha 60} = 1/2$.

The second force acting on a particle is the wind, and this effect is more complicated. The same statements about interpolation and correlation hold, but the effect of the *u*-speed and *v*-speed on the particle are more complicated.

The effect that the wind has on an object is called the leeway. While it is reasonable to expect that a current of 3 knots will push an object at a speed of 3 knots, the same is not true for wind due to the balance of forces between those of the wind acting on the exposed surfaces of the object and the drag of the water acting on the submerged surfaces of the object.

The wind doesn't push an object at the wind's speed and it often doesn't push an object exactly in the downwind direction. SAROPS' model for the effect of wind is based on Appendix H of [4]. It involves the two component forces, downwind and crosswind. The downwind component is, of course, in the direction toward which the wind is blowing. The crosswind component is perpendicular to the downwind component, but that statement leaves two possible directions. SIM switches between these two directions, and the time between the switches is exponentially distributed. It remains to state how the magnitudes of the downwind and crosswind components are computed.

Let ψ be the wind speed obtained from the EDS. The downwind slope ν is a scale factor that, when multiplied by ψ, gives the speed of the downwind component: downwind speed$= \nu\psi$. The value of ν is different for each particle and is determined by parameters specific to the type of object and a Gaussian draw. The parameters for the slope calculation are the nominal speed q, the slope $\bar{m}$ at speed q, and the standard deviation σ. There are two ways of computing the downwind slope for an individual particle.

The first method is the standard method. Let $z \sim N(0,1)$ be a random draw for this particle (this is a constant and is set once for each particle, not once for each time step). The effective downwind slope for this particle is given by $\nu = \bar{m} - z\sigma / q$.

The second method is the Rayleigh method. In the Rayleigh method, the downwind slope is given as follows. Let R be a draw from a Rayleigh distribution with density function

$$f(x) = x\exp\left(-\tfrac{1}{2}x^2\right) \text{ for } x \geq 0.$$

Set the downwind slope to be $r = \bar{m}\sqrt{2/\pi}R$. Note that $E(r) = \bar{m}$, and that $r \geq 0$.

SIM uses the Rayleigh method for object types that have a very low values of $\bar{m}$. For such object types, the standard method can lead to downwind slopes that are negative, and using the Rayleigh method avoids this.

The crosswind slope is computed using the standard method with different values for standard deviation and average slope. The nominal speed is the same as for the downwind slope.

3 Planner

The Planner is responsible for providing suggestions for where search effort should be expended. Information about the availability and capability of the searching units or SRUs is combined with the particle distribution to produce a suggested assignment of the SRUs to rectangles. The availability of the SRUs determines which units are to be used and when they can arrive on scene for the search. The capabilities of the units determine the speed and endurance (time on station) of each SRU. Using detection models that depend on the search platform, sensor, object type, and environmental conditions, SAROPS determines the detection capability of each SRU in terms of lateral range functions. From this information, it recommends a search plan for each SRU and provides an estimate of success (detection of the search object) for the overall search effort. It attempts to do this in a manner that maximizes the increase in success probability for this increment of search

There are two aspects to our discussion about Planner: the description of the problem that planner addresses, and the heuristic algorithms that Planner uses.

Planner uses the particles along with their probabilities as produced by SIM as an input. In addition, planner is given a collection of SRUs. Typically, these are helicopters or fixed wing aircraft or vessels that can search for the missing object.

The Probability of Success (or POS) for a search is computed as follows. Let $w(p)$ be the probability of particle p. Suppose we have specified the search paths for the SRUs. Then we calculate the probability $POS(p)$ of at least one of the SRUs detecting the particle p using the method in section 2.4. In particular using (1) and (2) we compute

$$POS(p) = 1 - \prod_{sru} pfail\left(p, sru\right). \qquad (3)$$

To obtain POS we compute

$$POS = \sum_{p} w(p) POS(p).$$

Planner attempts to maximize POS by placing the SRUs in rectangles and following the constraints given in Table 3.1 below. The following sections describe how Planner obtains its solution which is a local maximum.

3.1 Solution Approach by Planner

The term "configuration" for an SRU means the rectangle and its induced path. The induced path has parallel segments called *search legs,* connected by segments that are at right angles to the search legs. These connecting segments are all the same length and are called *cross legs*. The length of the search legs is called the *search leg length* and the length of the cross legs is called the *track-spacing*.

1. Inside its rectangle an SRU must search in a pattern of equally spaced parallel paths.
2. The total length of the pattern must equal the available path length of the SRU, minus one track-spacing.
3. Rectangles that correspond to different aircraft must not overlap. Similarly, rectangles that correspond to different surface ships must not overlap.
4. Each SRU has a minimum track-spacing and the track-spacing must be at most the search leg length.

Table 3.1. Statement of Planner Problem

Planner creates a solution by trying many combinations of rectangles for the SRUs. Planner continues to try combinations until a pre-determined period of time has elapsed or the user has terminated the optimization. When planner quits, it computes statistics (including POS) about its best solution, and reports the solution and these statistics.

Planner does not randomly choose combinations of configurations as potential solutions. Rather, it has a heuristic for placing the SRUs one at a time, and then an algorithm that refines these configurations until it stops improving the solution. It then makes a "big jump" of one of the SRUs, and refines that solution.

In Table 3.2, we list the steps in Planner's algorithm. Violations of the constraint in item 4 of Table 3.1 contribute to a measure called the Track Spacing Violation (TSV).

1. Get an initial solution.
2. Refine this solution to eliminate overlap and TSV.
3. Improve POS, introducing as little overlap as possible while not introducing any TSV.
4. Go to Step 2 if it is reasonable to do so.
5. Otherwise, create another initial solution, and then go to Step 2.

Table 3.2. Steps in Planner Algorithm

The following sections describe the mathematical framework, how planner gets its initial configurations (Step 1), the "refinement" processes of Steps 2 and 3, how planner decides whether or not it is reasonable to go back to Step 2 (in Step 4), and how it creates another initial solution (Step 5).

3.2 Mathematical Framework

Note, rectangles are used only for determining the three items listed in Table 3.3.

1. The SRU path is derived from the rectangle.
   a. This path determines POS for each particle, and hence POS.
2. From the induced track-spacing and search leg length, the TSV is computed.
3. Overlap is computed from the rectangles.

Table 3.3: Use of Rectangles

Put another way, the constraints are in terms of the rectangles; the rectangles induce the paths; and these paths are used to compute POS. Still, the optimization problem is most easily stated by considering the variables that specify the rectangles.

It takes five variables to specify a single rectangle; its center (two variables), the length and width of the rectangle (two more variables, called *ell* and *w*) and the orientation (θ). But these five variables are not quite enough to specify the path, since they do not specify which corner to start the path from, and whether the first turn is left or right. Both of these considerations can be "encoded" by adopting some conventions and allowing the width of the rectangle to be negative; this requires some explanation.

Planner uses the convention that the first leg runs in the direction of θ, and the rectangle is oriented so that the side of length *ell* is parallel to the first leg. This leaves two possibilities for a path; first turn is left or first turn is right. Another planner convention is that a positive value for w indicates that the first turn is right, and a negative value indicates that it is left (see Figures 3.1 and 3.2). Hence, the path (and rectangle) are specified by the five variables, where w is allowed to be negative. By contrast, *ell* must always be positive.

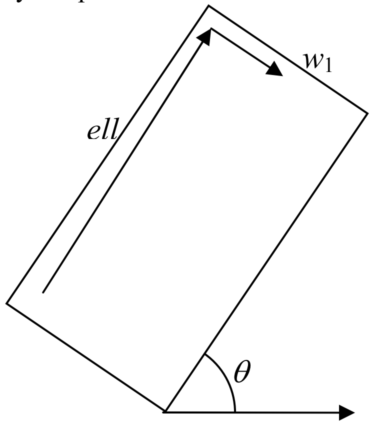

Figure 3.1: First-turn Right

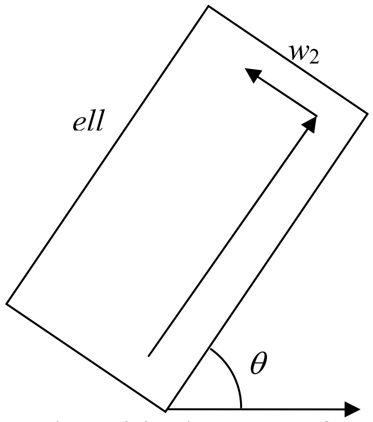

Figure 3.2: First-turn Left

There are actually very few (*ell*,*w*) combinations that are admissible. To state what they are, some notation is necessary. We denote the track-spacing by s, and the search leg length by t. We take the effective path length (L) of an SRU to be 85% of its speed times time-on-scene. 85% is a Coast Guard factor to allow time for investigating sightings and general maneuvering. Finally, we are given a constant γ that is the minimum allowable value for track spacing. We have the following constraints on s and t:

$\gamma \leq s \leq t$

$L/(t+s)$ is an integer

Table 3.4: Constraints on s and t.

These are constraints on s and t, but they induce constraints on *ell* and *w* since s and t are derived from *ell* and *w*. To describe this derivation, assume that $\theta = 0$, w is positive, and the lower-left hand corner of the rectangle is at the origin. The general case is obtained by a simple change of coordinates. The Coast Guard convention for putting a path inside of a rectangle assumes the following:

1. The path starts at $(s/2, s/2)$
2. Set *ell* = $t+s$
3. Use all but s of the effective path length L.

Table 3.5: Construction of the Induced Path

Given these requirements, it is not hard to show that $w = Ls/(t+s)$.

There is one more constraint in the optimization problem that Planner solves; the overlap constraint. This constraint requires that the SRUs corresponding to aircraft must not have overlapping rectangles, and the SRUs that correspond to surface ships must not have overlapping rectangles. The mathematical statement of Planner's problem is to maximize POS subject to the constraints listed in Table 3.4, and these "overlap constraints."

3.3 Steps 1 – 5 of the Algorithm

This section describes how steps 1 – 5 of table 3.2 are performed.

Step 1. For each SRU, a sweep width is calculated from the lateral range function. This is multiplied by the search speed and the effective search time on station of the SRU to obtain swept area. Each time the planner considers a rectangle for placement of the SRU, it computes the ratio of swept area to the area of the rectangle and enters this ratio into the exponential detection function [1] to compute probability of detection (POD) given the target is in the rectangle. POD is multiplied by the probability of containment in the rectangle (POC) to obtain POS. Starting with a one cell rectangle located at the highest probability cell, planner performs an accordion search to determine the best size and location of the rectangle.

An accordion search proceeds by adding or subtracting one row or column of cells from the rectangle and re-computing the POS. If an improvement is made this becomes the new rectangle. When no further improvement in POS is obtained, the search stops. If there is more than one SRU, the next rectangle is started in the cell with the highest probability given failure of the search by the previously located rectangles. Again Planner proceeds to determine this rectangle by an accordion search. This is continued until initial rectangles are determined for all SRUs. During this process checks are made to see that the rectangles chosen do not overlap "too much" with previous rectangles.

Steps 2 - 4. The "refinement" steps 2 and 3 of the algorithm in Table 3.2 are the core of the optimization algorithm. In these steps, SAROPS perturbs each box individually and checks whether an improvement is made. There are 12 types of perturbations called *moves*.

If SAROPS finds a move that improves both POS and overlap, SAROPS immediately makes that move. (Note, for steps 2 and 3 POS is computed using the method described below as apposed to the simple method for initial placement described above.) Otherwise, if it is eliminating overlap (Step 2), SAROPS will use a move that reduces overlap while minimizing the decrease in POS (over all the moves that reduce overlap). To eliminate overlap, SAROPS first computes the area that is contained in the intersection of pairs of rectangles. It then finds a move that decreases that area but restricts itself to moves that do not change the shape or size of a rectangle.

If SAROPS is improving POS (Step 3), it will use the move that improves POS but increases overlap as little as possible. For both Steps 2 and 3, the refinement quits when there is no move that improves the current objective. If SAROPS quits while improving overlap, and it hasn't eliminated the overlap, it jumps to Step 5. If SAROPS eliminates overlap (Step 2) and has not improved POS over the last time that it completed Step 2, then it jumps to Step 5.

Step 5. Planner chooses a rectangle at random to move. It initially moves this rectangle to the cell with highest posterior probability given failure to detect by *all* the rectangles. It then performs an accordion search starting with this cell using the posterior given failure by all rectangles except the one moved to determine the parameters of the moved rectangle.

Computing POS for Steps 2 and 3. The computation of POS for a given collection of SRU configurations can be very time-consuming, and Planner must compute POS for many configurations. Therefore, Planner estimates POS in the following manner and maximizes that estimate.

SAROPS estimates POS by drawing 1500 particles with replacement from the full distribution of particles and computing *POS* over this sample. SAROPS computes the variance of this estimate and increases the number of samples if the standard deviation is more than 5% of the estimated POS value. Typically, this approach reduces the number of particles used in the estimation by a factor of ten compared to using the full distribution of particles.

There is a cost to this approach however. The maximization method employed requires that the same number be obtained each time the payoff function is evaluated at the same input. Thus SAROPS keeps the same sample for all evaluations. This tends to bias the estimates on the high side because the system is optimizing on a fixed subset of the particles. In practice, this is not an issue because the final set of boxes is evaluated with respect to all the particles, and the intermediate computations of the optimization are not reported.

Figure 3.3 shows an example of the graphical output of the planner. The rectangles are ones computed by the planner. The thin paths and the dashed paths show the induced search paths produced from the rectangle as described in Table 3.5. The heavy lines show the search path followed during the interval between the two times in the upper left-hand corner of the figure.

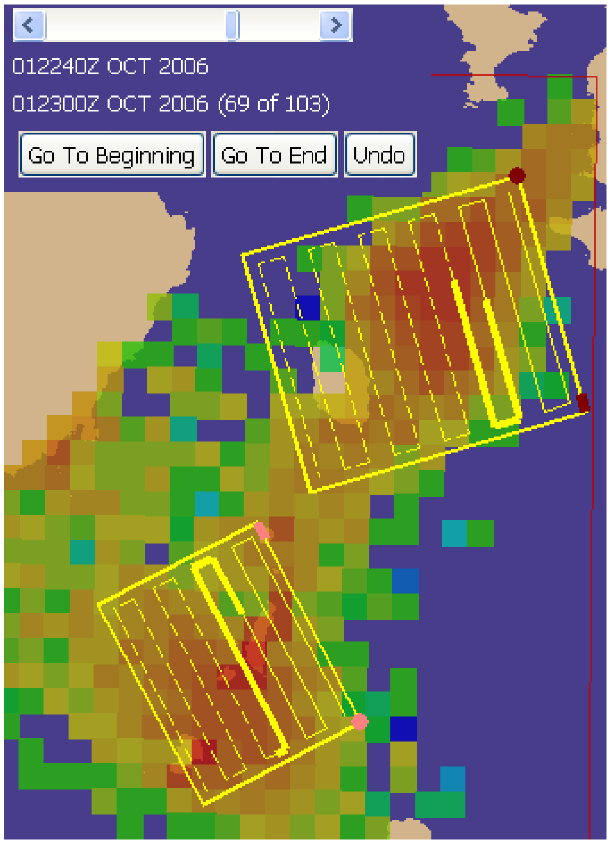

Figure 3.3 Planner Output

4 References

[1] Lawrence D. Stone, *Theory of Optimal Search*, 1975 (2nd edition, 1989).

[2] Henry R. Richardson and J. H. Discenza, "The United States Coast Guard Computer-Assisted Search Planning System (CASP)," *Naval Research Logistics Quarterly*, Vol. 27, pp.141-157, 1980.

[3] Memorandum from USCG (G-OPR-1), "SAROPS LATERAL RANGE CURVES," John R. Frost, Mar 11, 2004.

[4] U.S. Coast Guard Addendum to the United States National Search and Rescue Supplement (NSS) to the International Aeronautical and Maritime Search and Rescue Manual (IAMSAR), U.S. Department of Homeland Security, United States Coast Guard, Apr 29, 2004.