Plan Detection under Partially Observable and Cluttered Conditions

George M. Mathews, David Nicholson, Andrew McCabe, Mark Williams
BAE Systems, Advanced Technology Centre, Filton, U.K.
george.mathews@baesystems.com

Abstract – *This paper examines the problem of detecting the execution of plans performed in partially observable and cluttered environments. In particular, a plan is defined as a series of tasks that must be executed according to a known precedence relation and build toward some final objective. The goal of a plan detection algorithm is to detect the execution of the plan from the available ambiguous and incomplete data before it reaches the terminal event. This paper presents a Monte Carlo inference algorithm capable of estimating the belief that the plan is currently being executed and how much progress has been made. The performance characteristics of the algorithm are tested for a variety of simulated data sets containing different signal to noise ratios.*

Keywords: Plan Recognition, Plan Detection, Monte Carlo Filtering. Non-Linear Estimation.

1 Introduction

The ability to recognise the execution of a complex plan from incomplete and cluttered observations is important for many applications. These range from providing assistive robots the ability to understand the behaviour of their owner; to organised crime detection and prevention [1]; network and computer intrusion detection systems [2], [3]; uncovering terrorist activities [4], [5], [6]; and inferring enemy objectives in military scenarios [7], [8], [9].

Of interest in this paper are plans that build toward some ultimate event, for instance a terrorist bombing, hijacking, or insurgent ambush. These plans require several tasks to be performed before the final event can be carried out. However these tasks are rarely directly observable and the available information is usually ambiguous and incomplete.

This paper examines the inference or fusion problem that involves piecing together the observed data which, considered on its own, could have come from benign and normal activity, but taken together within the context of the plan may represent activity with a potentially dire outcome.

This work builds on that of Godfrey *et al.* [4], [5] and defines a plan as a series of tasks, each with a defined but unknown duration. In addition, the tasks have a series of known precedence relationships between them (e.g. components must be obtained before they can be assembled) such that the overall plan can be represented as a Gantt chart. Intelligence gathering and investigative activities provides partial information about the execution of the individual tasks. However this information is ambiguous as it is possible for it to have been generated from benign background activity that is unrelated to the execution of the plan.

The developed inference algorithm extends the Monte Carlo filtering approach of [4] by explicitly modelling how the observations are affected by background clutter. This enables the system to not only estimate the progress of the plan but also estimate the probability that the plan is actually not underway, implying that the collected data is obtained from benign background activity.

The Monte Carlo filtering approach used in this work allows greater complexity in how plans can be modelled than comparable techniques based on hidden Markov models (e.g. [6]). For instance concurrent plan activities and non-Markovian transition probabilities can easily be incorporated.

The paper is organised as follows: Section 2 further defines the problem and the models of the plan and observables. Section 3 provides details of the inference algorithm. Details of a test scenario are given in Section 4 with the results and performance characteristics of the approach given in Section 5. Conclusions and future work are discussed in Section 6.

2 Problem Definition

To define the inference problem, three aspects must be considered: (i) the structure of the plan and the dependencies between the sub-tasks; (ii) the form of the gathered observables and their dependency on the plan sub-tasks; and (iii) the dependency of observables on the background or noise process. The overall model of the inference problem is depicted in Fig 1.

2.1 Plan

The structure of the plan is modelled as a Gantt chart or project plan. This assumes a task can only start once all its dependent tasks are completed. Once started, each task

requires a given execution time before it is complete. Thus, the evolution of the plan can be completely specified by knowing the start time of the plan (i.e. the start time of the tasks with no dependencies), the dependency structure of the plan and the duration of each activity.

Hence, to estimate the progress of a hidden plan of a given structure, all that is required is to estimate the plan start time and duration of each task. Thus, the variable of interest for the inference problem is defined by the vector:

$$\left[t_{start}, d_1, d_2, \ldots\right]^{\mathrm{T}}. \qquad (1)$$

Here t_{start} defines the start time of the plan and d_i defines the duration of task i. This vector defines the schedule of the plan, determining its complete time evolution. This is similar to [4]. However, in addition to this, non-plan instances will be explicitly modelled by defining the state as the empty set $\varnothing$. This addition allows scenarios that do not have any plan activity to be explicitly modelled and will allow an estimate to be generated over whether the plan is actually underway. Thus, the state space over which inference must be performed consists of the empty set and set of all plan schedules, that is over all $x \in \mathbb{X}$, where

$$\mathbb{X} \triangleq \varnothing \cup \mathbb{R}^{N_{tasks}+1}$$

and N_{tasks} defines the number of tasks in the plan.

To perform Bayesian inference over this space will require a prior probability density function $p(x)$ that captures any specific domain knowledge that is not captured in the structure of the plan or observables. This must include when a plan could be expected to start (if at all) and how long may each task take. This prior density should come from domain experts or historical data.

2.2 Observables

Observables primarily come from the execution of the plan tasks. However, false observations are also generated from unrelated background activity. Although these are unrelated to the plan, these observations will have an interpretation linked to possible plan activity. To formally define the model of the observation process, firstly let z_{t_k} represent an observation at discrete time t_k. The set of possible observables is assumed discrete and modelled by $\mathbb{O} = \{o_A, o_B, o_C, \ldots\}$ (continuous extensions are also possible however not considered here). To model times periods where no evidence is uncovered, the observation variable z_t will be set to the empty set $\varnothing$. This enables the system to explicitly reason over missing information. Thus, the set of possible values z_{t_k} can take is given by

$$\mathbb{Z} \triangleq \varnothing \cup \mathbb{O} = \{\varnothing, o_A, o_B, o_C, \ldots\}.$$

Now, the detective work required to uncover the observables is modelled by the conditional probability

$$p(z_t \mid x).$$

This describes the probability of discovering the observable z_t when the true underlying plan being executed is described by x, where x may be the null plan or a particular plan schedule defined by the vector in (1). It needs to be pointed out that this model assumes the multiple observations taken during the whole investigative process are conditionally independent of each other once the state of the plan x is known. In practice this is unlikely to be true as additional factors that are not

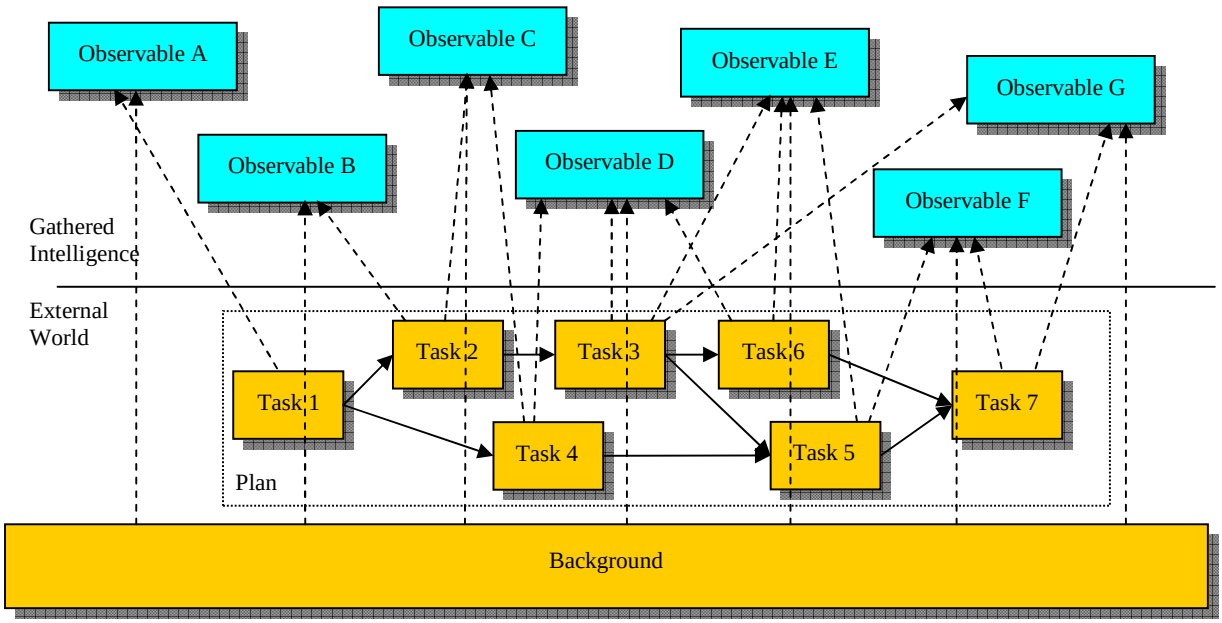

Figure 1. Aspects of the inference problem. The plan comprises a set of individual activities or tasks that must be carried out according to a set of dependencies. The observables are dependent on whether a given activity is currently underway, but may also be generated by a background noise process.

currently considered in the state x may link the way observables are uncovered. However, in this work this assumption will be used.

An additional problem with formulating the model of the observation process as the distribution $p(z_t \mid x)$, is that it is a very complicated function defined as a mapping from the product set $\mathbb{X}\times\mathbb{Z}$ to the real line. This can be simplified by following [4] and assuming the probability of discovering a given z_t is only dependent on whether a given task is either not-started, underway, or complete. Here however, we make the further assumption that it is only dependent on whether a given task is underway. Thus, the observation model can be written as

$$p(z_t \mid x) = p(z_t \mid T).$$

Where $T \in \{\varnothing, 1, 2, \ldots, N_{tasks}\}$ denotes whether a given task is underway and is determined by the state x and the observation time t. The distribution $p(z_t \mid T)$ can now be defined as a simple conditional probability table.

It is noted that the special case of $T = \varnothing$ is explicitly considered. This extension will allow the system to reason over false observations, that is observables that have been uncovered and thought to describe behaviour consistent with the execution of the plan but have actually been generated by normal background activity that is unrelated to the plan. In this paper the combination of these observables are modelled as uncorrelated random events; however more complicated models, that explicitly model individual behaviours, are also possible but will require a more powerful model than the table $p(z_t \mid T)$.

3 Inference Algorithm

The inference problem that is of interest in this paper is that of determining the posterior probability over the plan state x given the uncovered observables.

$$p(x \mid z_{t_1}, z_{t_2}, \ldots)$$

This will allow answers to questions like "What is the probability that the plan is currently being executed?" and "What tasks are likely to be currently underway?" to be calculated.

In what follows it will be assumed that there is available a prior density $p(x)$ and a model $p(z_t \mid T)$ that accurately describes the observation process. In the examples considered in this paper, these models coincide exactly with the models used to produce the data. Determining the robustness of the algorithm to modelling inaccuracies is left for future work.

The approach taken to perform inference over the hidden plan state x builds on the Monte Carlo estimation or particle filtering framework of [4]. This is extended to incorporate the more general models described in Section 2, in particular the null plan $x = \varnothing$ and the null observation $z = \varnothing$. The method is a version of the sampling-importance-resampling (or bootstrap) algorithm [10]; however as the entire plan state is estimated there is no need for a dynamics model and the associated sampling step[1]. The algorithm is recursive and starts with a set of samples drawn randomly from the prior distribution $p(x)$. Observations are fused into the sample population by repeatedly updating the weights followed by a possible resampling operation. To describe the process in more detail, first assume a set of weighted particles exists representing the prior

$$p(x) \approx \sum_{s=1}^{N_s} w_s^0 \delta(x - x_s^0).$$

One method of obtaining the sample set $\{x_s^0, w_s^0\}_{s=1}^{N_s}$, is to first draw from the distribution over whether there is a plan or not (i.e. a binary variable). If the plan state is sampled, a further start time and set of task durations are sampled to completely populate the sample state x_s.

As observables $z_{t_1}, z_{t_2}, \ldots, z_{t_k}, \ldots$ are collected, the samples and weights are updated in a sequential fashion such that they approximate the posterior probability distribution

$$p(x \mid z_{t_1}, z_{t_2}, \ldots, z_{t_k}).$$

This is achieved using Bayes rule

$$\begin{aligned} p(x \mid z_{t_1}, z_{t_2}, \ldots, z_{t_k}, z_{t_{k+1}}) = \\ \lambda\, p(z_{t_{k+1}} \mid x)\, p(x \mid z_{t_1}, z_{t_2}, \ldots, z_{t_k}). \end{aligned} \quad (2)$$

Where λ is a normalisation constant. It is noted that it is this step that requires the conditional independency between the observations given the plan. If this does not hold this step must be modified appropriately.

To apply this step to the sample set $\{x_s^k, w_s^k\}_{s=1}^{N_s}$, which is distributed according to the current prior $p(x \mid z_{t_1}, z_{t_2}, \ldots, z_{t_k})$, and obtain a set of weighted samples distributed according to the posterior defined in (2) requires reweighting the samples according to the observation likelihood

$$w_s^k \propto w_s^{k-1}\, p(z_{t_k} \mid T(x_s^{k-1}, t_k)).$$

Where $T(x_s^{k-1}, t_k)$ determines the task that is being executed at time t_k if the true plan state is described by x_s^{k-1}. After the weighting is applied the weights are renormalised such that $\sum_{s=1}^{N_s} w_s^k = 1$.

To avoid degenerating to a single sample with weight equal to one, the sample points are resampled according to the new weights. However, as any resampling process can potentially remove useful samples, it is only carried out when a significant number of the samples have negligible weight. This is determined by calculating the effective number of samples and comparing to a threshold fraction of the actual number of particles (e.g. one third)

[1] In essence, the inference method used is comparable to performing conventional target tracking by performing estimation directly on the trajectory of the target and thus no prediction step is required.

$$N_{\text{eff}}^{k} = \frac{1}{\sum_{s=1}^{N_s} w_s^k}$$

The resampling process employed is based on [4] and modifies each selected sample by randomly generating a new set of durations for all the future tasks that have not yet been observed. This ensures the sample set has adequate variety.

This process of reweighting the samples, followed by the resampling, defines a recursive process that can be used to fuse any number of observables into the estimate. In addition the process can easily handle out of sequence and delayed data as there is no prediction step.

The set of sample points and their associated weightings completely define the estimated progress of the hidden plan. However, to answer questions like: "What is the probability that the plan is being executed?" or "What is the probability that task 1 has finished?" requires further processing. This processing is effectively a marginalisation operation of the joint distribution represented by the weighted samples. For instance for a given time t, to determine the probability that the plan has not started, simply requires summing all the weights of the samples that represent no plans ($\varnothing$) and plans that will start in the future ($t < t_{start}$). In a similar manner the probability that a given task has not yet started, is underway, or has finished can also be calculated (see [4] for additional details).

It is noted that to perform this inference process, the full observation model $p(z_t \mid T)$ is not explicitly required as each operation only needs the un-normalised likelihood

$$l^{z_t}(T) \propto p(z_t \mid T)$$

This means that the full table $p(z_t \mid T)$, which may be very large as all possible observations must be enumerated, never needs to be explicitly constructed. All that is required is the ability to construct the un-normalised likelihood $l^{z_t}(T)$ over $T \in \{\varnothing, 1, 2, \ldots, N_{tasks}\}$ for all the observables z_t that are actually obtained. Thus, whatever data is represented by z_t, whether it be from ELINT, HUMINT, text, video, etc, needs to be translated into a probabilistic likelihood over what tasks (and the background) could have generated it. This could be performed by a human analyst on an individual basis, or if operating within a larger data processing system be generated from some predefined template models.

4 Example Scenario

A scenario of interest examined in this work to test and evaluate the approach involved piecing together intelligence reports to uncover enemy activity preparing an ambush in an asymmetric urban conflict. Such preparations can be broken down into several distinct tasks ranging from the initial planning and financing, to recruiting, bomb manufacturing, reconnaissance, and finally deployment of gunman to the attack area. Each of these tasks, when performed, may give rise to observables that can be detected e.g. by informants, passing patrols, intercepted communications, etc. The set of plan tasks and the observables that they may give rise to are depicted in Fig. 2.

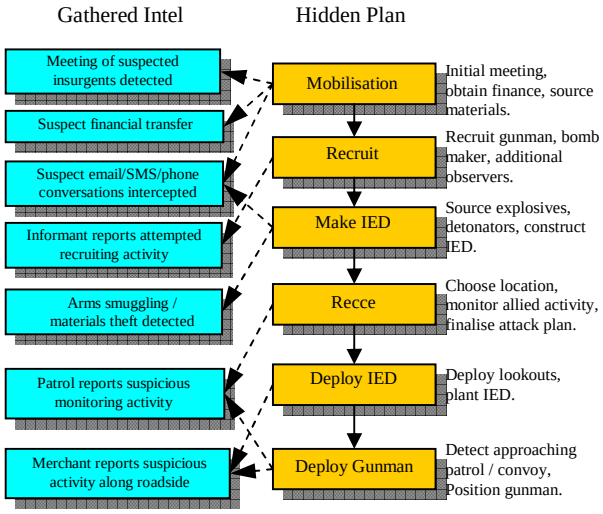

Figure 2. Example intelligence reports and their dependency on the hidden plan activities.

When considered separately each observable will not provide enough information to infer that there is any enemy activity. To be able to successfully identify that there is enemy activity underway, all the data must be considered within the context of the plan.

4.1 Plan Model

Within this scenario, the prior distribution $p(x)$ defines when it is believed that a plan may start and how long each of the tasks may take. Here, it is assumed that it has been identified from historical data and expert predictions that there is a 50% belief that there will be no enemy activity within the time period of interest (e.g. the next two months). For the other 50% it is assumed that a plan will start according to a uniform distribution over that same time period.

To model the internal task durations, a separate prior distribution over each task must be specified [4]. For instance, a gamma distribution could be used to model the task durations from an estimate of the mean and variance. In this work, it is assumed that the prior distribution for each task duration is the same and has a mean length of 3 days with a variance of 1 day. Thus, the prior can be specified as

$$p(x) = \begin{cases} 0.5 & \text{if } x = \varnothing \\ 0.5\, p(t_{start})\, p(d_1)\, p(d_2) \cdots & \text{otherwise} \end{cases}$$

Where $p(t_{start})$ is a uniform distribution over the two month period and $p(d_i)$ is a gamma distribution with

mean 3 days and variance 1 day for all tasks i. It is noted that with this prior a plan is, on average, 18 days long.

4.2 Observation Model

The observation model specifies the probability that an observable is generated either by the background or an active task within each hourly time step. This is specified by defining two separate probabilities, one that relates the probability that the active tasks generate an observable, which is assumed to be the same for all the tasks and will be referred to as the signal strength, and another which defines the probability of the background generating an observable, referred to here as the noise strength. These are combined to produce the overall conditional probability table $p(z_t \mid T)$.

4.3 Simulated Data

Using the plan model and observation models described above random datasets can be generated. Such datasets are shown in Fig. 3 and consist of two months worth of simulated observables. These 4 example data sets have been generated according to a single pair of signal and noise strengths. The effects caused by different signal and clutter strengths are explored in Section 5.2.

5 Results

The output of a typical run of the algorithm is given in Fig. 4. This figure shows the time series of input data (corresponding to the top dataset in Fig. 3), and overlaid on it is the evolution of the belief that a plan is currently on going (at the time shown, this is approximately 55%). Also displayed is the decomposition of this belief into the estimates of the progress made on the individual tasks and the actual ground truth schedule of the hidden plan. These results indicate that if conditioned on the plan being underway (which at this time is believed to be true with a probability of 55%), the algorithm has estimated that the tasks 'Mobilisation' and 'Recruit' are complete (with almost 100% conditional belief), the 'Make IED' is most likely complete with a conditional probability of approximately 80%. There is more uncertainty in whether the remaining tasks of 'Recce', 'Make IED' and 'Deploy Gunman' are either not started, underway or complete. This is understandable due to the uncertainty in the input data, and is consistent with the current ground truth task 'Recce'.

5.1 Plan Detection Performance

The results discussed above and shown in Fig. 4 demonstrate the representative performance of the inference algorithm for the majority of data sets like the ones displayed in Fig. 3. For these types of problems there is just sufficient information contained in the observables to allow useful conclusions to be drawn, e.g. the estimated belief that a plan is going on is higher when there actually

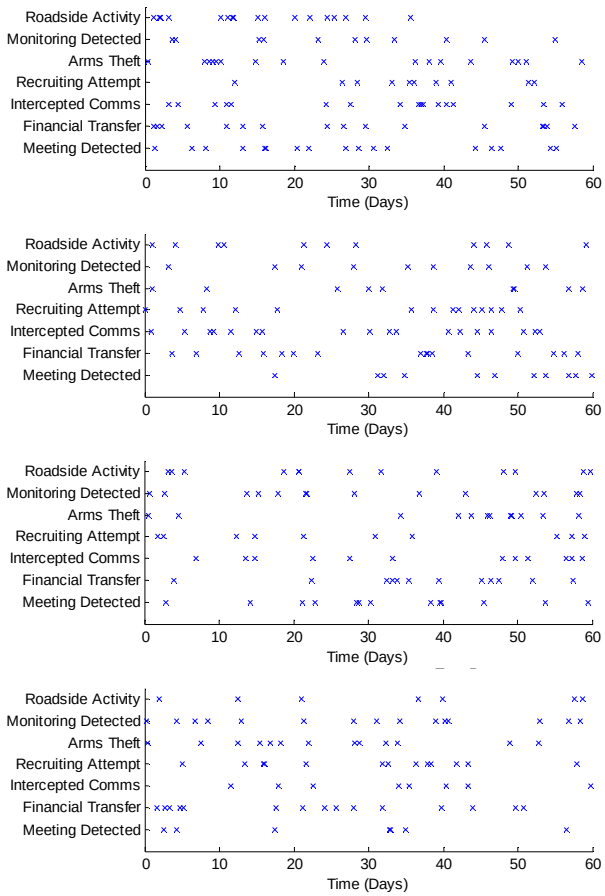

Figure 3. Example test data. Plans exist in the top two data sets, while the bottom two contain only false observables from the background.

is a plan being carried out, than during periods when there was no plan activity. However to understand how often the algorithm misses plans or falsely identifies a plan that is not actually occurring the overall performance of the algorithm needs to be determined.

Thus, to characterise the power of the algorithm to be able to differentiate between plan activity and no plan activity, the inference algorithm will be considered a binary classifier that determines whether the plan is currently being executed or not. Although this ignores the extra information regarding the belief about which tasks are being executed, it allows a simple and well known methods to be used, in particular a receiver operator characteristic (ROC) curve. A ROC curve describes the performance of a binary classifier by plotting the rate of false positives vs. true positives. The curve is generated by varying the decision threshold that converts the degree of belief into the binary decision of 'There is something going on' or 'Nothing is going on'. An ROC curve is displayed in Fig. 5 and is calculated from 50 random datasets with similar characteristics to those in Fig. 3 (i.e. plan structure and signal/noise strengths). The ROC curve

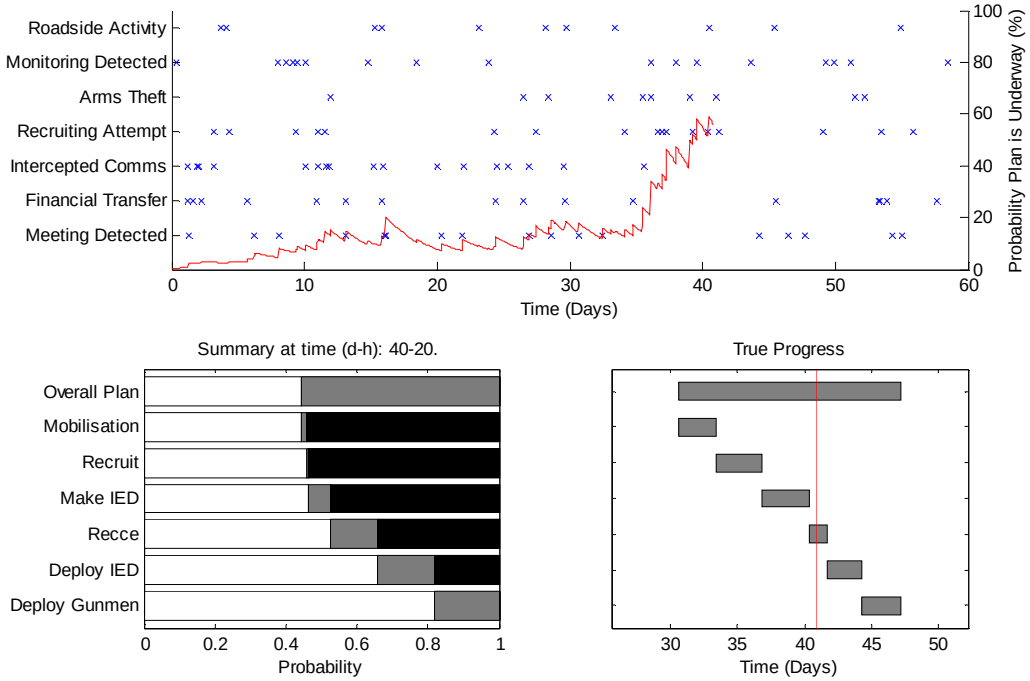

Figure 4. Typical run of the algorithm on the first data set from Fig. 3. Top: Time series of input data (left axis), overlaid with the time evolution of the predicted belief that a plan is currently on going (right axis). The snapshot was taken at day 40, when the belief is approximately 55%. Bottom Left: Belief that the plan and each of the individual tasks are 'not underway', 'underway' or 'completed'. Bottom Right: True progress of the hidden plan, the vertical line indicates the current time. The actual task being executed is 'Recce' and is consistent with the estimate.

graphically displays the trade-off between wanting high true positive rate and low false positive rate.

5.2 Overall Performance Characterisation

The above analysis provides an insight into the performance of the algorithm for a given scenario with fixed plan and observation models. However, when attempting to determine what investigative process to take, which in turn determines what observables could be collected and their signal and clutter characteristics, it is important to known how the observation model affects the ability of the algorithm to draw useful conclusions.

To accomplish this, the performance of the algorithm will be summarised further by integrating the ROC curve. This produces a single number representing the performance of the algorithm for a fixed set of models and varies from 0.5 for a purely random classifier to 1 for a perfect classifier. It can also be interpreted as the probability that the belief score assigned to a positive instance (at a time when the plan is underway) is higher than that assigned to a negative instance (when the plan is not underway). This statistic measures how accurately the two classes can be resolved from the data. For the curve displayed in Fig. 5 the area is 0.83. This statistic is plotted in Fig. 6 for difference types of data with different observation models (and hence different signal and noise characteristics). The star (☆) represents the characteristics of the data and results depicted in Fig. 3 and 4.

As expected, for high signal strength (i.e. intelligence can be easily gathered regarding which tasks are being executed) the algorithm achieves almost perfect performance. However, as the signal strength is reduced, and less information is available to the inference algorithm, the performance degrades. Also as expected, the rate of degradation is faster when the noise is higher.

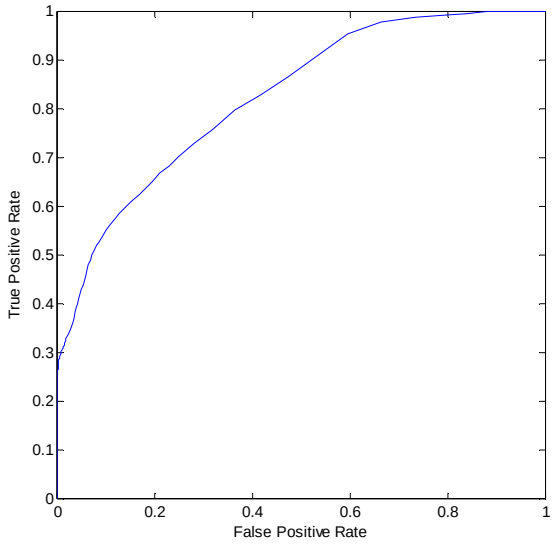

Figure 5. ROC-Curve. Displays the trade-offs between true positive and false positive rates that different decision thresholds will produce.

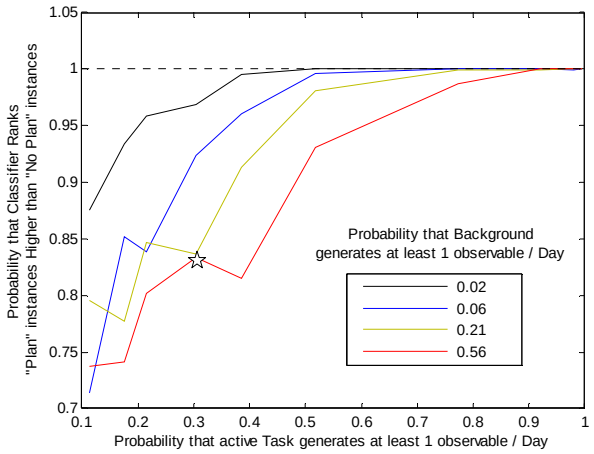

Figure 6. Performance characteristics of the inference algorithm for different signal and noise strengths. The signal is related to the strength of the observables on the plan tasks and is plotted along the x-axis. The noise is related to the strength of the observables to the background activity and varies according to the colour of the line.

This result allows a distinction to be made between scenarios and investigative processes which will produce enough information to allow useful inferences to be made vs. scenarios where the available data is simply too ambiguous and the hidden plans are effectively unobservable.

This analysis has produced a quantitative description of how the performance of the algorithm varies according to the amount of signal and noise in the system for a given scenario. However, such understanding is also required when applying this method to new domains, which may have a different plan structure and different observables with different signal and noise characteristics. To characterise the dependency of these on the performance of the algorithm a similar analysis could be performed that considers different plan structures; prior models for the plan durations; and the number of observables that could be collected and how they relate to the plan and the background. This would require a vast number of simulations to be performed and has not been done. Instead we give some general comments that relate to how these may impact the performance of the algorithm:

- The more chain like the plan is the better the algorithm will perform. This comes from the fact that chain-like plans impose greater constraints on which tasks can be executed when, than plans with greater degrees of parallelism. These extra structural constraints are naturally exploited by the algorithm resulting in improved performance.

- The greater the 'resolution' of the observables the greater the performance. Here resolution relates to the number of tasks that can generate a common observable. E.g. a scenario where every task generates a different observable will have higher resolution than a scenario where every task generates the same observable and hence better performance. This resolution of the observables also provides insight into how much detail should be used to model the plan.

6 Conclusion

This paper has examined the problem of detecting the execution of plans carried out in partially observable and cluttered environments. In particular, a plan has been represented as a series of tasks that are executed in a known order and build toward some final event. The goal of a plan detection algorithm is to detect the execution of the plan from the available ambiguous and incomplete data before it reaches the terminal event. This has been accomplished using a Monte Carlo inference algorithm. This algorithm extends the filtering approach of [4] by explicitly modelling the full observation process. This enables the system to not only estimate the progress of the plan but also estimate the probability that the plan is actually not underway.

A method based on ROC curves has been presented to examine the performance of the algorithm and how well it can classify between the two situations corresponding to whether plan activity is underway or not. A performance analysis has been carried out for a variety of simulated datasets and provides a basis to determine whether a given scenario, involving a particular plan model and investigative process, will provide sufficient information to allow useful conclusions to be drawn.

Within this work the model use to generate the data has coincided with the model used within the inference algorithm. This is unlikely to be possible in practice and future work will examine how robust the algorithm is to modelling errors.

References

[1] Cornish, D. B. & Clarke, R. V., "Analyzing Organized Crimes", *Rational choice and criminal behavior: recent research and future challenges*, Routledge, 2001, pp. 41-63.

[2] Geib, C. W. & Goldman, R. P., "Plan Recognition in Intrusion Detection Systems", in Proceedings of DARPA Information Survivability Conference and Exposition II, 2001.

[3] Geib, C. W. & Goldman, R. P., "A probabilistic plan recognition algorithm based on plan tree grammars", Artificial Intelligence, 173, pp. 1101-1132, 2009.

[4] Godfrey, G. A.; Cunningham, J. & Tran, T., "A Bayesian Nonlinear Particle Filtering Approach for Tracking the State of Terrorist Operations", in Proc. Int. Conf. on Information Fusion, 2007.

[5] Godfrey, G. A. & Mifflin, T. L., “Likelihood-based Optimization of Threat Operation Timeline Estimation”, in Proc. Int. Conf. on Information Fusion, 2009.

[6] Allanach, J.; S. Singh, H. T.; Pattipati, K. & Willett, P., “Detecting, Tracking and Counteracting Terrorist Networks via Hidden Markov Models,” IEEE Aerospace Conference, Big Sky, MT, March 2004

[7] Das, S.; Lawless, D.; Ng, B. & Pfeffer., A., “Factored particle filtering for data fusion and situation assessment in urban environments”, in Proc. Int. Conf. on Information Fusion, 2005.

[8] Suzic, R., “Representation and recognition of uncertain enemy policies using statistical models”, in Proc. of the NATO RTO Symp. on Military Data and Information Fusion, 2003.

[9] Johansson, L. R. M. & Suzic, R., “Particle filter-based information acquisition for robust plan recognition”, in Proc. Int. Conf. on Information Fusion, 2005.

[10] Ristic, B.; Arulampalam, S. & Gordon, N., Beyond the Kalman Filter: Particle Filters for Tracking Applications, Artech House, 2004.