Title: **Developing Automated Intelligence Collection Plans from Probabilistic Behavior Estimates**

Topic 6: **Modeling and Simulation**

Authors:

Georgiy M. Levchuk
Aptima Inc.,
12 Gill Street, Suite 1400
Woburn, MA 01801
Phone: 781-496-2467
Fax: 781-935-4385
e-mail: georgiy@aptima.com

Scott Galster
Aptima Inc.,
3100 Presidential Drive, Suite 220
Fairborn, OH 45324
Phone: 937-306-7773
Fax: 781-935-4385
e-mail: sgalster@aptima.com

Krishna R. Pattipati
Professor, ECE Dept., UCONN
Storrs, CT
Phone: 860-486-2890
Fax: 860-486-5585
e-mail: krishna@engr.uconn.edu

Correspondence:

Georgiy M. Levchuk
Aptima Inc.,
12 Gill Street, Suite 1400
Woburn, MA 01801
Phone: 781-496-2467
Fax: 781-935-4385
e-mail: georgiy@aptima.com

This paper was cleared by 88 ABW/PA on 25-MAR-09 as Document Number 88 ABW-09-1181.

Developing Automated Intelligence Collection Plans from Probabilistic Behavior Estimates

Georgiy Levchuk[1a], Scott Galster[a], and Krishna R. Pattipati[b]
[a]Aptima Inc.
[b]University of Connecticut

Abstract

Modern warfare is intensely information centric with vast amounts of data transmitted around the battlespace. Within this environment, it is critical to provide predictive and timely intelligence for planning and execution of operations. Due to the changing nature of the asymmetric battles, the intelligence analysis and collection operations are integral part of one another. However, current technologies and tactics, techniques, and procedures (TTPs) decompose these two types of activities, resulting oftentimes in disjointed solutions. For example, information collection planning is often done without accounting for the results of the intelligence analysis, and collected information does not get immediately incorporated to update intelligence estimates. This may result in collecting information that is not critical to situation understanding and delays in changing the situation estimates.

In this paper, we describe a decision support tool for intelligence analysts and collection planners integrating automated behavior pattern identification with intelligence collection planning in a close-loop solution. This technology promises to provide warfighters with a comprehensive, accurate, and cost-effective solution for intelligence gathering, analysis, and operations planning.

Motivation: Integrating Disruption and Collection Plans with Intelligence Estimates

Due to the changing nature of the asymmetric battles, the intelligence analysis and collection are integral part of one another. However, current technologies and tactics, techniques, and procedures (TTPs) decompose these two types of activities, resulting oftentimes in disjointed solutions. For example, information collection planning is often done without accounting for the results of the intelligence analysis, and collected information does not get immediately incorporated t update intelligence estimates. This may result in collecting information that is not critical to situation understanding and delays in changing the situation estimates.

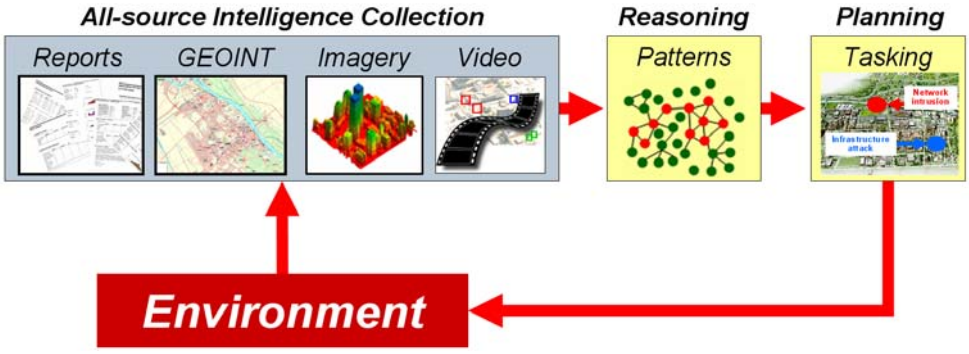


Figure 1: Collection, reasoning, and planning in a close-loop workflow of operational and strategic decision making

The *collection*, *reasoning* and *planning* are three elements in the close-loop workflow of operational and strategic decision making (Figure 1). First, the intelligence collections can be conducted with varied sensors

[1] georgiy@aptima.com; phone 781-935-3966x267; fax 781-935-4385; www.aptima.com

and produce data in different formats. This data will have large information gaps: missing events, errors in classifying what events and actors are, ambiguous information, and irrelevant data. Second, the intelligence analysis processes observed data through reasoning to identify actors and behavior patterns. Finally, current estimates of possible patterns can be used to identify information elements that are most critical to current situation understanding. This criticality means that collection of such data may improve the current understanding of the situation by disambiguating between multiple hypotheses that may currently seem equally likely. To develop efficient automated decision support systems that can reason about environment and design collection actions, we need to establish tight dependencies between various decision phases. This can be done by establishing direct input-output data flow and feedback between decision phases.

Method: Integrating Collection Planning with Behavior Pattern Recognition

Under a project called Contextualized Pattern Recognition (CoPR) sponsored by the Air Force Research Lab (AFRL), Aptima Inc. is currently developing a decision support system (Figure 2) for intelligence analysts and collection planners to *find structures and patterns* in all-source multi-scale data and *identify critical information collection requirements*. This technology probabilistically maps hypothesized adversarial behavior signatures against observed actor activity and interaction networks. The mappings are used to identify information for intelligence collection that disambiguates current predictions and improves situation understanding. We envision the CoPR system as a multi-user collaborative decision support tool, which must enable the following three types of uses (Figure 2):

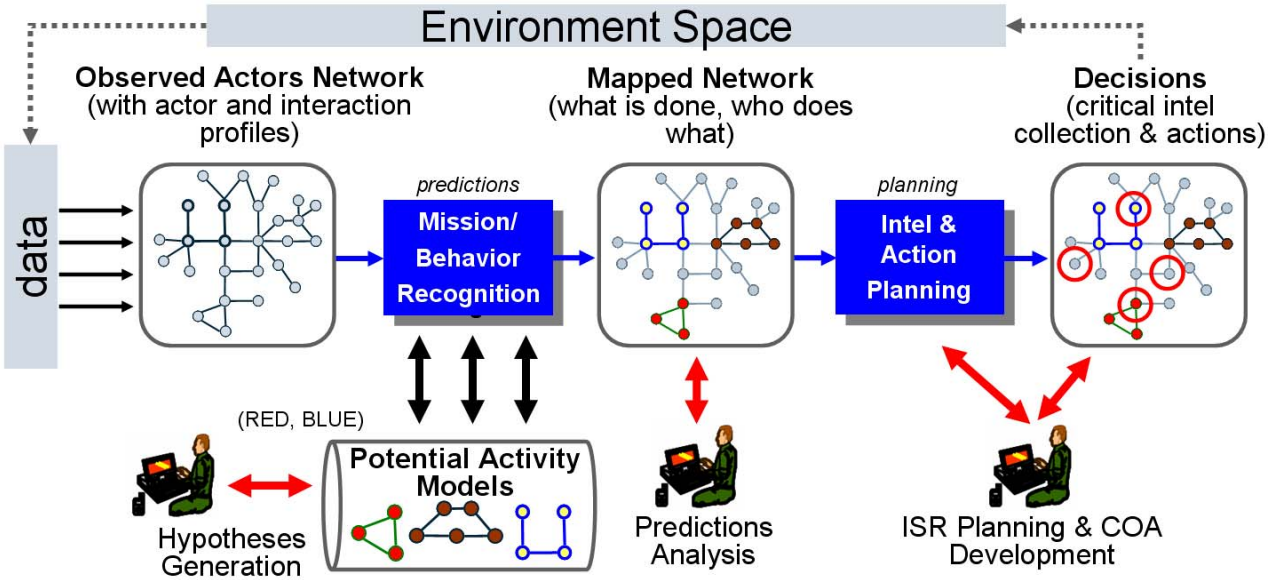


Figure 2: CoPR system workflow

Use 1: Hypotheses Generation. The CoPR system will allow analysts and commanders to define the hypotheses in the form of potential adversarial activity patterns (which we call behavior signatures) that may take place and/or are of interest to the users. This functionality allows different users to define their own distinct hypotheses, share them with other users, and refine them and their estimates over time as more intelligence becomes available that either confirms or contradicts the hypotheses. The activity patterns are represented in the form of the networks consisting of possible hostile operations, their profiles, and temporal and relational dependencies among these operations. Thus, the structure of the activity patterns is natural for modeling decision and action steps defining how the adversaries might achieve their objectives.

Use 2: Adversarial Analyses. As hypothesized hostile behavior signatures are defined, CoPR will analyze available intelligence to estimate which of these hypotheses are likely to be present. That is, CoPR will automatically identify, given observed events, which adversarial operations are taking place, where they

may occur, what is their status, who is involved in those operations, and forecast which operations will happen next.

Use 3: Intelligence, Surveillance, and Reconnaissance (ISR) Planning and Course of Action (COA) Development. As behavior signature recognition algorithms make estimates of potential hostile activities, many alternative predictions may seem equally likely due to information gaps in collected intelligence. The CoPR system will facilitate the design of the intelligence collection plans to conduct investigative actions to obtain new information that can disambiguate between different predictions. The functionality is supported by automated feature extraction and criticality assessment algorithm, which designs ISR collection plan and orders tasking to maximize information gain from collected intelligence. The users can also use disruption planning algorithms to determine which actors, operations, and resources of the adversaries to disrupt achieving highest impact on the adversaries.

Accordingly, the CoPR system will help the users generate the following products:

Product 1: Identify adversarial behavior signatures. CoPR behavior recognition algorithms will identify which of the hypothesized patterns of hostile activities are taking place.

Product 2: Identify the roles of places and actors. CoPR will produce the mapping of observed actors and places to the hostile activities, thus specifying *who did/does what* in the enemy organization, and *where operations have been conducted*.

Product 3: Forecast future operations. CoPR will estimate the state of adversarial missions, identifying which operations have completed and forecasting which of the operations will be conducted next. In addition, CoPR will map these operations to actors and places, specifying *who* will conduct operations and *where* they could occur.

Product 4: Develop ISR pans. CoPR intelligence collection planning algorithms will define the tasking and structure of the ISR collection, specifying *what needs to be collected*, *where*, and *what impact* the collected intelligence may have on the situation understanding.

The CoPR model is based on the following three elements:

(1) Representation of the adversarial activities and state of the environment as *networks*;

(2) *Probabilistic network pattern matching* to identify the adversarial activities and the actors involved in the activities; and

(3) *Information gain maximization* algorithm to design ISR collection activities to improve situation understanding.

In the following, we describe these elements in more detail.

Environment representation

In representing the environment and behaviors at different levels of granularity, we define the following three concepts: (i) adversarial behavior signatures (to which we often refer as adversarial missions or plans); (ii) observed networks; and (iii) environment state.

Adversarial behavior signature, also termed adversarial *mission*, is a collection of *tasks* (individual element operations) that adversaries plan to perform to achieve desired objectives. A *task* is an activity that entails the use of relevant resources (provided by the adversaries), and is conducted by individual actors or at specific locations. Tasks usually are defined by the set of resource requirements that must be satisfied to complete them (e.g., see task requirement modeling in (Levchuk et al., 2002)). Additional information can be specified, including task duration, location, deadlines, etc. Mission structure incorporates the temporal planning process and dependencies of tasks on each other (e.g., information/material flows between tasks, precedence constraints, etc.). The simplest mission model representation is a task precedence graph; a more complex representation that can be tied to objectives, events, conditional contingencies, and represented at different granularity levels.

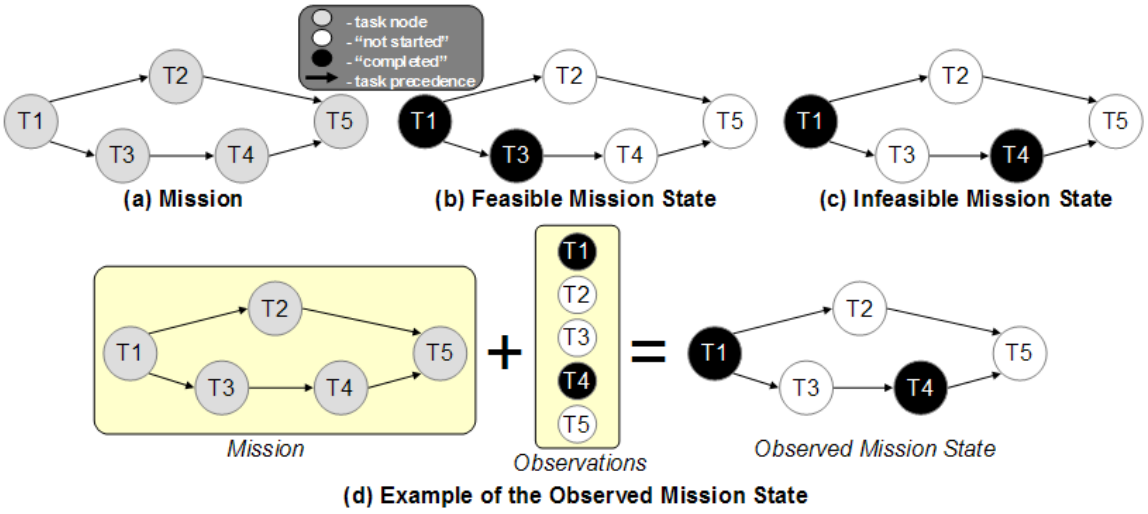


Figure 3: Example of mission and mission state

For precedence graph-based modeling (Levchuk et al., 2008), we represent a mission as a directed acyclic graph $G_M = (V_M, E_M, A_M)$, where the set of graph nodes $V_M = \{T_1, T_2, ..., T_N\}$ represents the tasks of the plan, a set of directed edges $E_M = \{e_{ij}^M = < T_i, T_j >\}$ represents the temporal relationships among tasks, and a set of attributes $A_M = \| a_{ij}^M \|$ represents task requirements and relationships signatures (a_{ii}^M is a vector of attributes for task T_i, and a_{ij}^M is a vector of attributes of dependency between T_i and T_j). The *state of mission* G is defined as a **network with node states**, represented by the tuple (V, E, A, Ω), where $\Omega = \{\omega_i^T \mid i = 1, ..., N\}$ is a set of task states. The state ω_i^T of task T_i is an indicator of the task's progress towards completion, and can incorporate success/failure information. To simplify our notation, we describe our approach based on a 2-state task representation (i.e., $\omega_i^T = 1$ if task T_i has been completed successfully and $\omega_i^T = 0$ otherwise). Figure 3 shows an example of the mission and its state, which can be feasible or infeasible. In a feasible mission state (Figure 3(b)), task execution satisfies precedence constraints (that is, if $\omega_j^T = 1$ then $\omega_i^T = 1, \forall i : e_{ij}^M \in E$ - i.e., all parents of the completed task must themselves be completed). Otherwise, the mission state is infeasible. For example, in Figure 3(c), task T4 is marked as "completed", and its predecessor T3 is not.

The mission signatures are sometimes referred to as *models networks* – that is, the known hypotheses against which the observed data must be tested.

Observed networks: We aggregate the environment observations into networks with attributes on nodes and links (Figure 4). The *nodes* in the network can represent groups, organizations, government, states, individuals, facilities, equipment, geo-political entities, etc. Node attributes can define size/membership of the groups, beliefs of people, economic power of organizations, social identities of individuals and groups, knowledge and materials, observed actions of the actor(s), capabilities of individuals/resources/areas, and other economic and social features. The links in the network represent who does what to whom in the area of interest – social interactions and influences, economic transactions, behavioral interactions, political events and activities, traffic, material exchange, etc., with link attributes defining the frequency and types of interactions among the nodes.

The set of observed events is obtained by sensors, which may include human collection teams, tactical units, unmanned aerial vehicles, radars, cyber nodes, etc. These events contain three information entries: (i) *geo-spatial information* – indicating the location of activities; (ii) *temporal information* – indicating the time of activities; and (iii) *feature information* – indicating the type of actions, participating units, resources

used, etc. Then, the network data is constructed from observed events, of which we highlight three main event classes:

- Capabilities events, which identify "*who can do what*" in the environment; for example, this data can include "city U has continued religious violence", "individual X is a truck driver", "building A has wide entrance and can be used as a storage facility", etc.
- Interaction events, which define "*who is connected to/interacts with whom*", where connections can be of several classes, e.g., socio-economic influence, information flow, materials exchange, command and synchronization of activities, etc.; for example, interactions can include communication transactions, such as "country X has been engaged in military action with country Z", "members of a militant wing engaged in a meeting with weapons suppliers at 11:35 am for 35 min to procure explosives"; financial transactions, such as "a report of a money transfer from accounts of political support groups to an organization of interest"; or geo-spatial link, such as "a member of potential terrorist cell has been seen at the same time in a village where IED attacks occurred"
- Action events, which specify "*who did/does what*"; for example, action events can include intel about individual and joint operations of adversaries, such as "organization X has staged a political protest that turned violent", "BLUE team discovered a safe house and apprehended RED operatives attempting to manufacture weapons", "trucks from company Z were used for transporting refugees", etc.

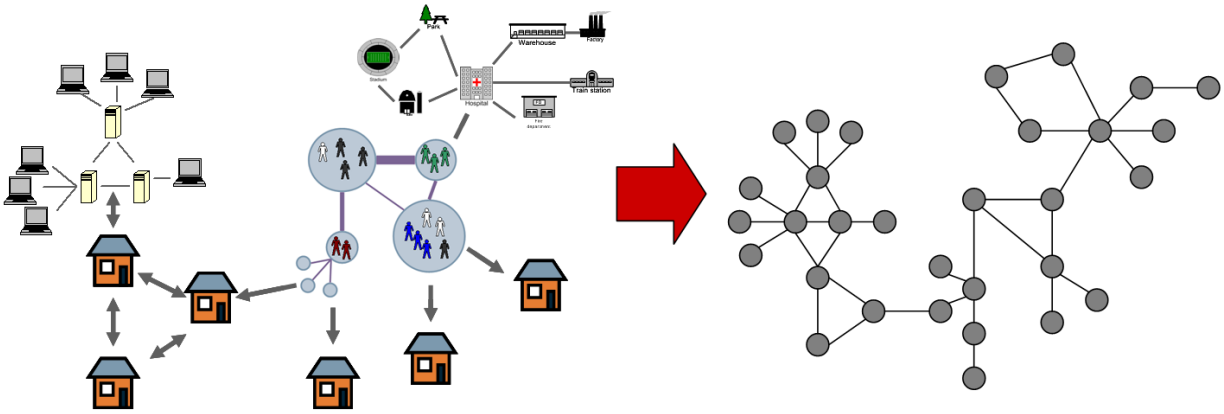


Figure 4: Example of observed network

Formally, we represent the observations network as a graph $G_D = (V_D, E_D, A_D)$, – a *data network* where V_D is the set of nodes, E_D is the set of links among them, and $A_D = \| a_{ij}^D \|$ is a matrix of observed attributes on nodes and links (a_{ii}^D is attributes vector for node i and a_{ij}^D is attribute vector for link between nodes i and j).

Environment state: To achieve desired objectives, we need to quantitatively represent the state of the environment. There can be many different state representations. For example, the state of the enemy's plan can be of interest, or the state of the BLUE's plan may need to be monitored and controlled. For CoPR, we define the state of environment as a true state of actors and their interactions. This state can be obtained by removing the noisy events from the observations and filling in information gaps. This can be achieved by identifying the pattern and state of adversarial behavior and mapping it against observed state of the environment. This would allow assessing the direct outcome of the enemy's behavior on the environment. The state of the environment can be desired or undesired, and we will define the reward (or penalty) of achieving each state. Without loss of generality, we define a set of all feasible states the environment can take $\{x_i, i = 1,..., N\}$.

Identifying adversarial actors and activities

The knowledge of the adversarial mission and its state is needed to understand the progress of adversaries towards their objectives, predict future operations, and identify resources and actors that will be used to perform the corresponding actions. The problem is that the observed data is very noisy: there are many irrelevant “normal” events, much information is not collected due to limited information resources, and the enemy is concealing their operations. As the result, the available data cannot be straightforwardly used to predict adversarial actions.

When the intelligence analysts using the CoPR system define hypotheses about adversarial missions, the CoPR system will rank-order these hypotheses, report the mostly likely current state of each mission, and identify the mapping of the tasks to actors and areas in the environment. The rank-ordering of the missions is based on their likelihood scores – probability that the mission has generated the observed data about the actors and their interactions.

The true state of the adversarial activities can only be found if observed data is aligned with model behavior signature data. To do this, we need to map behavior activities to actors and places of the environment. That is, we need to discover the *mapping* (Figure 5) of the potential adversarial tasks (nodes of model network) to the observed actors (nodes of data network). This is accomplished by finding an assignment matrix $S = \|s_{ki}\|_{k \in V_M, i \in V_D}$, where $s_{ki} = 1$ if model node k is mapped to data node i i.e., the actor i is identified with activity k. We also can restrict a task to be performed by (mapped to) a single node, in which case $\sum_i s_{ki} = 1$.

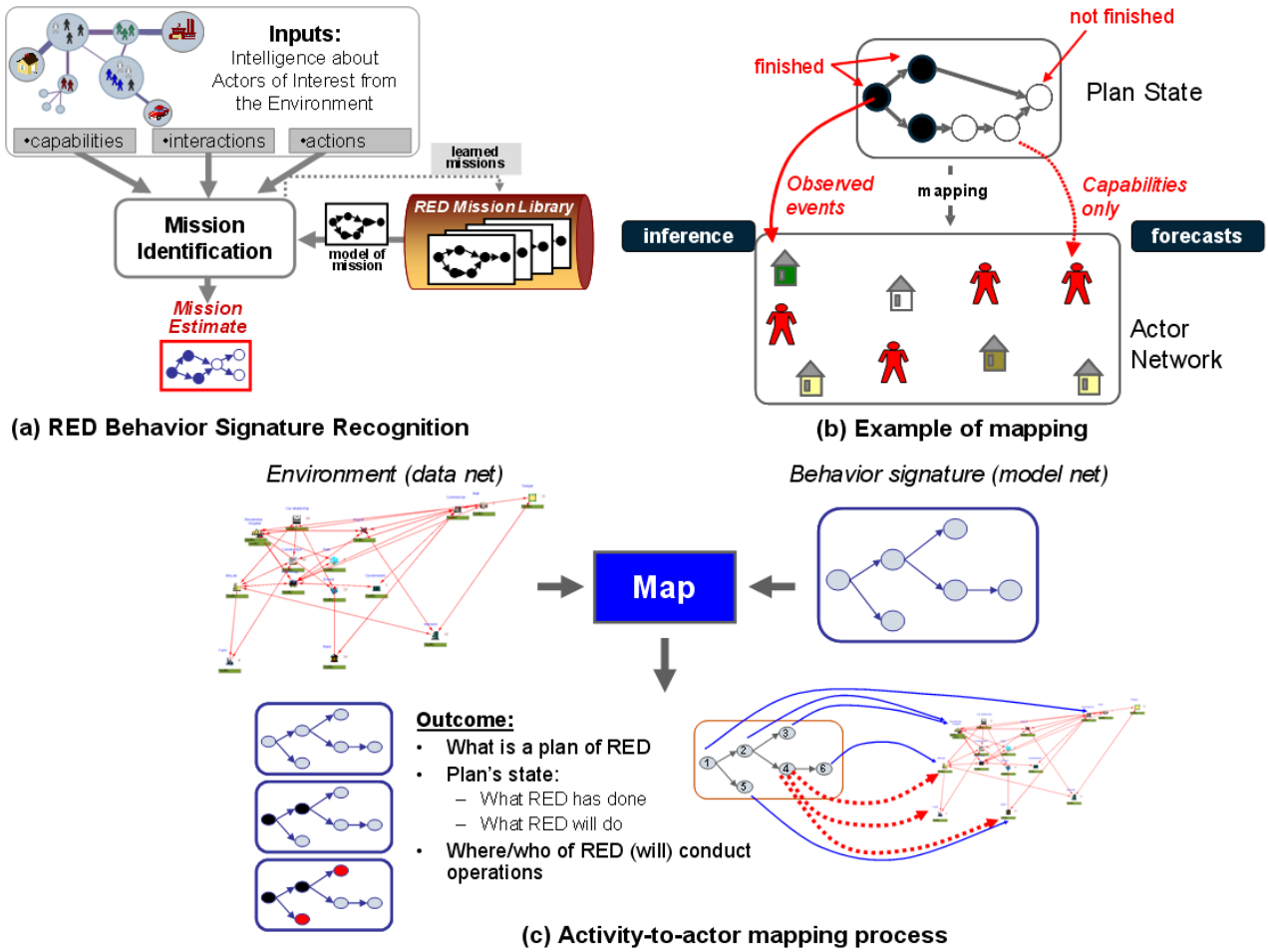


Figure 5: Activity-to-actor mapping

As the result, we can find the most likely mission, its mapping, and its state in the following two steps:

Step 1: Find mission state and activity-actor mapping for each mission
$(\hat{\Omega}_m, \hat{S}_m) = \arg\max_{\Omega,S} \mathrm{P}(\Omega, S \mid G_D, G_m)$

Step 2: Find most likely mission (behavior pattern) $\hat{m} = \arg\max_{m} \mathrm{P}(\hat{\Omega}_m, \hat{S}_m \mid G_D, G_m)$

Mission estimate indicates the hostile behavior pattern, the mission state estimate indicates "*what is/will be done*", and task-to-actor mapping indicates "*who does/will do what*". Quantitatively, solutions to above steps follow from the fact that we can rewrite probability function

$$\mathrm{P}(\Omega, S \mid G_D, G_m) = \frac{\mathrm{P}(G_D \mid G_m, \Omega, S)\mathrm{P}(\Omega \mid G_m, S)\mathrm{P}(G_m, S)}{\mathrm{P}(G_D, G_m)} = const \cdot \mathrm{P}(G_D \mid G_m, \Omega, S)\mathrm{P}(\Omega \mid G_m, S)$$ (which is true assuming that all mappings are equally likely).

Hence, we obtain

$$(\hat{\Omega}_m, \hat{S}_m) = \arg\max_{\Omega,S} \mathrm{P}(\Omega, S \mid G_D, G_m) = \arg\max_{\Omega,S} \mathrm{P}(G_D \mid G_m, \Omega, S)\mathrm{P}(\Omega \mid G_m)$$

$$\cong \arg\max_{\Omega,S} \prod_{ijkm} \left[P(a_{ij}^D \mid a_{km}^M, \omega_k^T, \omega_m^T)\right]^{s_{ki}s_{mj}} \prod_i P(\omega_i^T \mid \omega_{N\{i\}})$$

$$= \arg\min_{\Omega,S} \sum_{ik} s_{ki} \log P(a_{ii}^D \mid a_{kk}^M, \omega_k^T) + \sum_{ijkm} s_{ki}s_{mj} \log P(a_{ij}^D \mid a_{km}^M, \omega_k^T, \omega_m^T) + \sum_i \log P(\omega_i^T \mid \omega_{N\{i\}})$$

$$\approx \arg\min_{\Omega,S} \sum_{ik} s_{ki}\left(\omega_k^T \parallel a_{ii}^D - a_{kk}^M \parallel + (1-\omega_k^T) \parallel a_{ii}^D \parallel\right) + \sum_{ijkm} s_{ki}s_{mj}\left(\omega_k^T \parallel a_{ij}^D - a_{km}^M \parallel + (1-\omega_k^T) \parallel a_{ij}^D \parallel\right)$$

$$+ const \cdot \sum_i \log P(\omega_i^T \mid \omega_{N\{i\}})$$

One approach to solve this problem is to iteratively update the mission state variables $\{\omega_i^T\}$ and mapping variables $\{s_{ki}\}$. Then, mission state variables can then be found using belief propagation algorithm, and the mapping solution found for the quadratic assignment problem (Grande et al., 2008; Levchuk et al., 2008).

Planning intelligence collection actions

In the previous section, we described how mission, state, and mapping estimates can be derived. In the case the data has large noise and/or missing information, many behavior patterns, states, and mapping would seem equally likely. When this occurs, the model has a low confidence and large ambiguity, and additional information is needed to improve predictions.

To discover the adversarial activity pattern, mission state, task-to-actor mapping, and even true state of an environment (e.g., actors, their relationships, their views, intent, and power balance in the society), we can apply actions that are analogous to injecting test signals into physical systems. Such *probing signals* force the system to reveal itself. Mathematically, the main objective of probing actions is to **maximize the information gain** for current estimate of the environment's state. This can be achieved by finding those probing actions that reduce the entropy, a measure of uncertainty, of the current estimates the most. An example of a probing action could be providing information that coerces adversaries to communicate more and hence to reveal their relationships, commit actions that could be observed, or even change some of their plans. The outcome of probing is an improved understanding of the environment and, consequently, a better ability to prevent instability and/or control crises.

Let's denote the probability of being in state $i \in \mathrm{N}$ at the current time as $p_t(i)$ (here, the word *state* is used loosely, and can be interpreted as "mission pattern", "mission state", "mission mapping", etc.). To

distinguish the states and determine the information collection and probing actions, we define the vector of features $\varsigma^i = \{\varsigma_1^i, ..., \varsigma_N^i\}$ for each state i. For simplicity of notation, we can assume that each feature ς_f^i can be collected using some action (e.g., probing), and that the observation ξ_f will be obtained with the probability $p_f(\xi \mid \varsigma) = \mathrm{P}(\xi_f = \xi \mid \varsigma_f^i = \varsigma)$. The actions to collect the feature information can involve both types of investigative actions described above. For the purposes of the mathematical formulation, we ignore the differences between passive information collection and probing, but we note that the main differences in real-world settings would involve the probability of collecting correct intelligence, cost of the actions, and the action specifics.

Without loss of generality, we assume that all environment states are among the current potentials (alternatively, we could have limited the analysis to a subset of sates that are currently very likely – such as with likelihood above certain threshold). We use the entropy as a score of ambiguity of current predictions, as it characterizes "*how much uncertainty is there in predicting the true state given the data already collected*?". When the entropy is high (i.e., there is a high uncertainty that the current predictions can achieve a correct result, so $H_t = -\sum_i p_t(i) \log p_t(i)$ is close to $\log |\mathrm{N}|$), the likelihood estimates of the environment state predictions are similar. If these states carry significantly different projections in terms of the threat and consequent preventive actions, i.e. the best preventive policies for each state are significantly different and none is satisfactory for all states, we cannot rely with confidence on the preventive policy derived from the current state prediction. Since significant uncertainty in predictions is often due to missing data, we can instead attempt to identify the features that are critical to prediction, so that the collection of these features would achieve the largest reduction in the prediction's ambiguity.

First, we perform feature extraction by selecting the subset of all features $F \subset \{1, ..., N\}$, where for each selected feature $f \in F$ there exist at least two states i, j such that $\varsigma_f^i \neq \varsigma_f^j$. Second, we order these features to achieve the most distinguishability among state predictions while satisfying resource constraints. The measure of the benefit for conducting the investigative action for feature $f \in F$ is then the information gain:

$$gain(f) = H(\mathrm{N} \mid O) - H(\mathrm{N} \mid O, f) = -\sum_i p_t(i) \log p_t(i) + \sum_{x,y} \frac{\left|k \in S : \{\varsigma_f^k = \varsigma\}\right|}{|\mathrm{N}|} p_f(\xi \mid \varsigma) \sum_{i \in \mathrm{N}: \{x_f^i = x\}} p_t(i \mid \xi_f = \xi) \log p_t(i \mid \xi_f = \xi)$$

In the above,

$$p_t(i \mid \xi_f = \xi) = \frac{p_f\left(\xi \mid \varsigma^i\right) p_t(i)}{\sum_{j \in S} p_f\left(\xi \mid \varsigma^j\right) p_t(j)}$$

Finally, we can construct the plan as a conditional sequence of collection actions by defining a decision tree (Figure 6), where each internal node corresponds to the collection action (probe), and the links out of the nodes correspond to the action outcomes (collected information). The leaf nodes of the collection tree correspond to the environment states or groups of states. In the latter case, the leaf node is a belief about being in each of the states of the set, and we can compute the probability of each of those states.

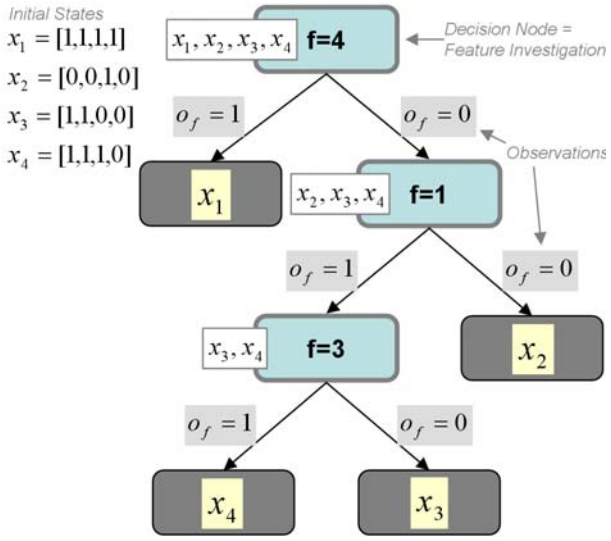


Figure 6: Example of an intelligence collection plan structure

The probing plan tree can be constructed using several approaches. One of the computationally simplest algorithms selects the probes greedily based on the maximum reduction in the entropy. Another approach can consider the outcomes of the probes in the future, so that the policy that maximizes the long-term reduction in the entropy is used. We can use a combination of the roll-out and greedy methods to design such a probing plan.

Execution of the probing plan is accomplished by following the decision tree and the outcomes of the investigative actions. Each decision to collect the data splits the set of current hypotheses (environment states) into several subsets (two subsets if we have binary outcomes of the collection), and results in a reduction of the entropy (or increased information gain).

In CoPR, we define the states as *probabilistic behavior signature estimates* (the hypothesized mission pattern, corresponding mapping, and concomitant probability score obtained during pattern matching), or in other words a pair of mission and its mapping (M, S). We define state features using the mapping of mission tasks to actors $\| s_{ki} \|$ and corresponding attributes of the mission tasks a_{mk}^{M}. For each probabilistic behavior signature estimate, we define the actor and relationship **behavior signature profiles** $\rho_{ij}^{s} = \sum_{nm} a_{nm}^{M} s_{ni} s_{mj}$. This creates a matrix of features that defines each state (behavior estimate). In this case, the feature types are actors and their relationships against which intelligence collection can be conducted, and values ρ_{ij}^{s} are feature values for a state in the prediction. Example of the resulting feature extraction in the case of defining task and actor attributes corresponding to the classes of activities is shown in Figure 7. The *behavior signature profiles* are then analyzed to identify the variables (actors, relationships, and their attributes) that have not been collected and that distinguish the behavior signatures with maximum total information gain.

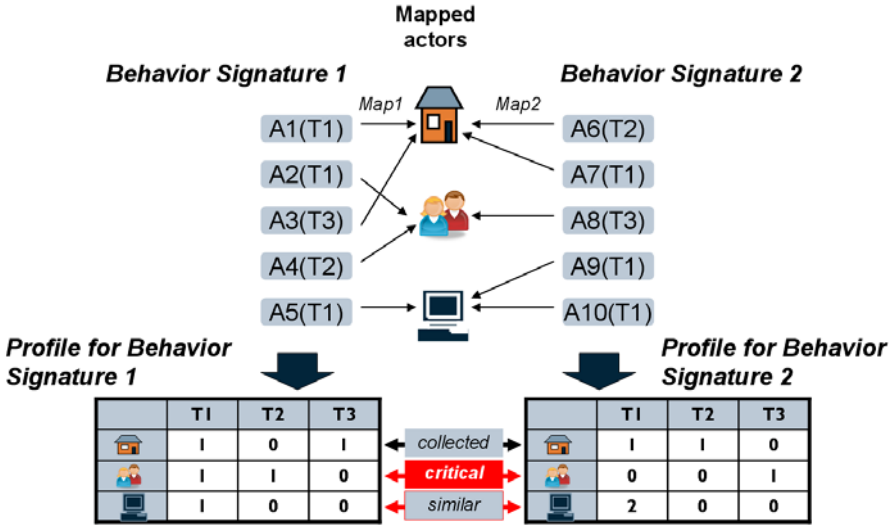


Figure 7: Extracting features and mission-based behavior signature profiles

To find what intelligence collection actions need to be conducted, the algorithm first filters out the actors and relationships for which data has been collected, and second filters our the actors and relationships for which the profile vectors ρ_{ij}^{s} are the same. Then, we find a set Ω of distinct values of vectors ρ_{ij}^{s} for all behavior signatures.

For each actor and/or relationship (i, j) (for actor i we have index (i, i)), we define subsets $R_1^{(i,j)}, ..., R_{|\Omega|}^{(i,j)}, R_{\varnothing}^{(i,j)}$, where $R_r^{(i,j)}$ as the set of behavior signatures for which $\rho_{ij}^{s} = r \in \Omega$ and $R_{\varnothing}^{(i,j)}$ is the set of behavior signatures (pattern-mapping pairs) that are not mapped to actor/relation (i, j). The

algorithm then finds the actor/relationship (i^*, j^*) which maximizes the information gain. Approximately, we can compute information gain as $gain(i,j) = -\sum_s p_s \log p_s + \sum_{r\in\Omega}\left(\frac{1}{\sum_{s\in R_r^{(i,j)}} p_s}\sum_s p_s \log p_s - \log \sum_{s\in R_r^{(i,j)}} p_s\right)$.

The algorithm then terminates if set $R_\varnothing^i$ is empty. If it is not, then the algorithm takes that subset and proceeds in the same manner to find next data collection target (actor or relationship).

Results: Assessing the Effectiveness and Sensitivity of Automated Intelligence Collection Planning

We have conducted three experiments incorporating several different types of hostile behaviors. In the first and second experiments, we generated random RED mission plans and conducted sensitivity analyses with different levels of information noise. In the first experiment, we tested the ability of CoPR to correctly map the operations to the actors. In the second experiment, we assessed ability of CoPR to identify correct RED mission states. In the third experiment, we have used an example dataset (see Appendix for more details) to illustrate the workflow of the system, process of generating ISR plans, and the assessment of ISR planning effectiveness. The results of these experiments appear below.

Experiment 1: Analysis of task-to-actor mapping accuracy with random RED mission networks

We conducted the first experiment to identify the mapping of tasks in randomly generated missions (model networks) to actors in the environment (data networks). We assessed the task-actor mapping accuracy against several uncertainty levels (**Error! Reference source not found.**). The signal-to-noise (SNR) ratio was a good measure of the quality of the data; however, our results showed that the effect of noise on the pattern in the data has a higher impact on the accuracy than the amount of noise alone.

Table 1: Sample/comparison points for random missions in experiment 1 (SNR shown in dB)

Data Spec	No Noise	rndNodeAttrErr = 3.0; rndLinkAttrErr = 3.0; probMiss = 0; probFalse = 0;	rndNodeAttrErr = 3.0; rndLinkAttrErr = 3.0; probMiss = 10; probFalse = 10;	rndNodeAttrErr = 3.0; rndLinkAttrErr = 3.0; probMiss = 15; probFalse = 15;	rndNodeAttrErr = 5.0; rndLinkAttrErr = 5.0; probMiss = 15; probFalse = 15;	rndNodeAttrErr = 5.0; rndLinkAttrErr = 5.0; probMiss = 20; probFalse = 20;	rndNodeAttrErr = 8.0; rndLinkAttrErr = 8.0; probMiss = 20; probFalse = 20;
SNR	-	31.17834879	8.725442499	3.936605233	3.199784794	0.966333683	0.084709562

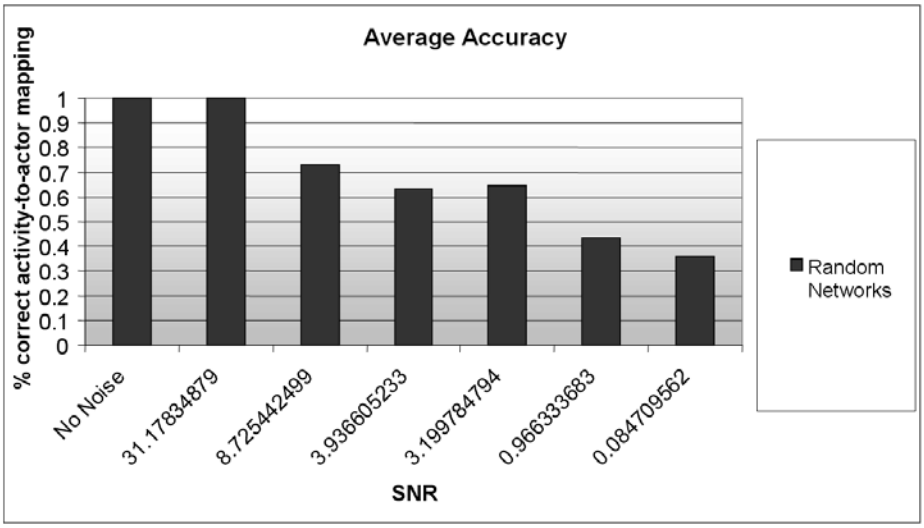


Figure 8: Sensitivity of mapping accuracy to data noise

The tests on random networks (Figure 8) was conducted by randomly generating the model networks of 20 nodes with network link density equal to ten percent (10%) (on average two links per node). The true mapping was then randomly generated, and a random noise added to obtain observed data network. The outcome from CoPR's network mapping was then compared to the ground truth. The results of 30 runs for each data sample in **Error! Reference source not found.** are shown in Figure 8. We can see that CoPR solution is perfect in finding permutations (case of no noise) and at low noise (31 dB SNR). CoPR's network mapping algorithm accuracy is good at medium noise (>73% accuracy in 9 dB SNR) and at significant noise (>64% accuracy in 3 dB SNR, which had 15% miss, 15% deceptions, 25% errors). However, CoPR needs improvements in convergence for larger noise levels to achieve higher prediction accuracy.

Experiment 2: Analysis of mission state identification accuracy with random RED mission networks

We conducted a second experiment to predict the states of randomly generated missions. We assessed the mission state predictive accuracy against several uncertainty levels (**Error! Reference source not found.**). Three measures of the accuracy were used in this experiment: *mission ID accuracy* was equal to the ratio of correctly identified RED mission patterns to the total number of test points; *state ID accuracy* was equal to the % of task states identified correctly (= 0 if the wrong mission pattern was selected), and *state ID accuracy for correct missions* was calculated as average of correct task state identifications over only correctly identified mission patterns. The SNR was used to measure quality of the data.

First, we randomly generated 30 distinct feasible tasks. Next, we randomly generated seven mission models as hypotheses, selecting one as a true mission plan. Each mission was comprised of 15 tasks randomly selected from original 30-task set and had 20% precedence constraint density. The prior probability of task state = 1, given that all predecessor tasks have state = 1, was 80%. Using this parameter, we generated the state of the selected true mission, and finally generated a mission state observation. As shown in **Error! Reference source not found.**, we varied the probability of missing the task execution and probability of falsely identifying task state as = 1 to generate observed mission state with different noise levels. The number of randomly generated tasks and number of tasks selected for mission models were chosen to produce a significant similarity between the true behavior and alternative hypotheses, so that on average every hypothesized mission had half of the task overlapping with the true hostile mission. This by intention introduced the challenge for the predictive algorithm, because the models it had to select from to match the observed data were similar to one another.

Table 2: Sample/comparison points for random missions in experiment 2 (SNR shown in dB)

Data Spec	probMiss = 0.1 probFalse = 0.1	probMiss = 0.2 probFalse = 0.2	probMiss = 0.3 probFalse = 0.3	probMiss = 0.4 probFalse = 0.4	probMiss = 0.5 probFalse = 0.5
SNR	4.988658376	2.384504666	0.479444814	-1.248397521	-2.225833734

Figure 9 shows the average accuracy (over 100 Monte-Carlo simulations) of identifying the mission and its tasks' states (= 1 or 0) for the observations with noise levels of **Error! Reference source not found.**. In our tests, the accuracy of mission identification and mission state identification algorithms was high (over 80% and 75% respectively) at significant noise level (2.38 dB, see Figure 9), but started to decrease sharply afterwards. However, for the situations in which the mission was identified correctly, the accuracy of recognizing its state (the state of all the tasks in the mission) was relatively high (over 65%) for a very high noise level (-1.25 dB SNR); the average accuracy for all cases has dropped at that point to just below 45%. The measure of accuracy is equal to the percentage of the tasks with correctly identified state (= 1 or 0) among all tasks that were identified, which represents the true measure of correctly predicting the hostile operations that have occurred and .the ones that are planned but have not occurred yet.

The measures of precision (% of correctly predicted tasks with state = 1 among all tasks predicted to have state = 1) and recall (% of tasks correctly predicted tasks with state = 1 among all tasks with true state = 1) have followed the same pattern as the accuracy metric. Precision measure in our context assessed the percentage of correct predictions of hostile operations among all alarms about adversarial behavior, while the recall measure assessed the percentage of correctly identified hostile operations among all such operations that occurred. Both precision and recall measures dropped abruptly to 0 at the last test point (SNR=-2.23 dB). This indicates that almost no correct detection about hostile operations could be done with such noise in the data.

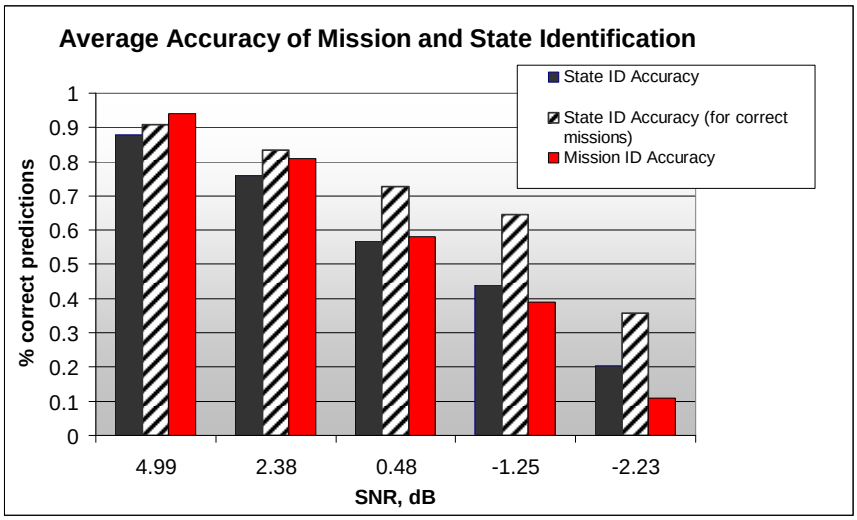


Figure 9: Predicting random adversarial missions --- sensitivity of mission identification algorithm's accuracy to data quality

Experiment 3: Analyses with CoPR dataset

In this section, we describe the example of analyses and supported decision making by CoPR prototype using an example dataset (see Appendix for details). In the following, the analysis was done to find the mappings of activities to geo-spatial areas/actors (analyzing *where activities took place* rather than *who did them*). Mapping activities to RED team members in addition to the physical areas will be our consideration for follow-on work.

We start by showing an example of how a simulation component of the prototype generated RED actions.

Example of the simulation process and output: The RED behaviors in the CoPR prototype were generated using C2 simulator that selects tasks from the RED mission to execute, finds what areas will be used for these tasks, identifies RED resources and actors, creates corresponding commands, and simulates how the task execution will happen geographically and over time. Figure 10 illustrates an example of the simulated mission execution in the CoPR prototype, where we show the target engagements and geo-spatial movements of RED team members. Figure 11 shows how the activities are done over time, in what areas and by what actors.

Example of behavior predictions: activity pattern and action-area mapping: To illustrate how the behavior pattern analysis is done, we first explain how the actor/area profiles are defined. The following analysis is for RED mission "*Cyber Attack on Critical Financial Infrastructure*" (see Appendix, Figure 16). In Table 3, we show the example of the actor profiles for only attributes relevant to analyzed mission (behavior pattern). The *true* profile vectors are shown in Table 3 broken into subvectors specifying counts of actor capabilities and events. Yellow color highlights the areas that are not involved in the hostile activities for analyzed example of behavior instance. The *observed* actor/area profile vectors will be of the

same form, but the values of attributes would be based on whether corresponding features and events were observable and on the errors in observations.

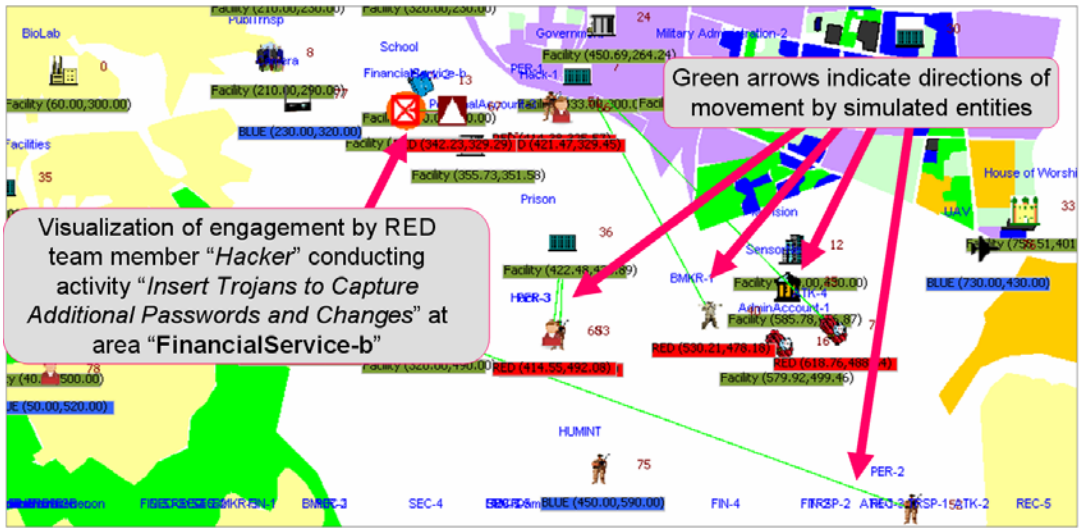


Figure 10: Example of the simulated dynamics

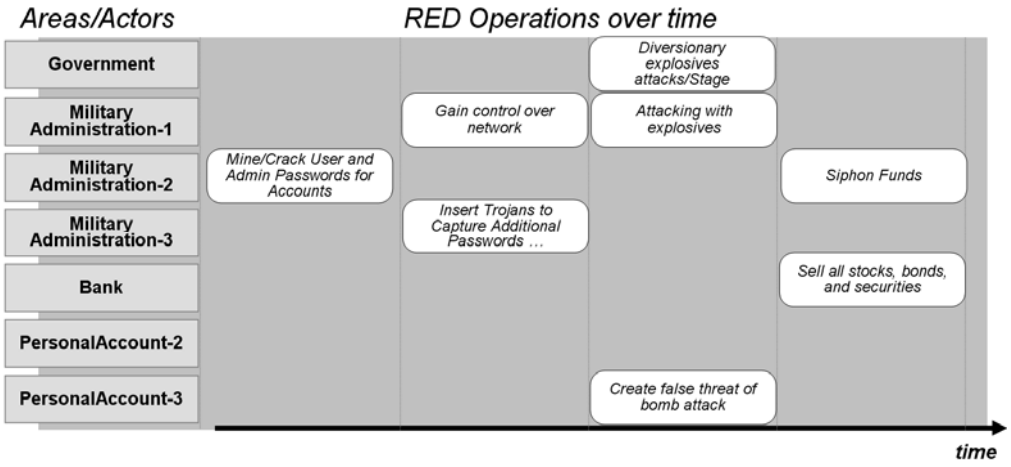


Figure 11: Example of the simulated action-actor/area assignment and schedule

Table 3: Example of true actor/area attributes

Accumulative Behavior Attributes	*4*	*2*	*2*	*3*	*1*	*1*	*1*	*1*	*3*	*2*	*1*	*1*	*3*	*5*	*2*
	Capability of Area/Actor						**Current events of Area/Actor**								
Actors/Areas	VAL	GOV	AINF	PINF	CSVC	CINFR	KNW	ATK	AINF	PINF	BACT	CSVC	CINFR	HACK	PER
Airport	2	0	0	0	0	1	0	0	0	0	0	0	0	0	0
Park	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Residential	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Commercial	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Government	3	1	1	1	1	1	1	0	0	0	0	0	0	0	1
Mansion	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
School	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0
Bank	1	0	1	1	1	0	0	0	0	1	0	1	1	1	0
AdminAccount-1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
PersonalAccount-1	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0
PersonalAccount-2	0	1	1	0	0	0	0	0	0	0	0	0	0	1	0
PersonalAccount-3	0	1	1	0	0	0	0	0	1	0	0	0	1	0	0
PersonalAccount-4	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0
FinancialService-a	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0
SensorNet-A	0	1	1	0	0	1	0	0	0	0	0	0	0	0	0
Military Administration-1	5	0	0	1	0	1	0	1	1	0	0	0	0	1	1
Military Administration-2	5	0	0	1	0	1	0	0	0	1	1	0	1	1	0
Military Administration-3	5	0	0	1	0	1	0	0	1	0	0	0	0	1	0
Telecommunications	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0
Processing/Treatment	3	0	1	0	0	0	0	0	0	0	0	0	0	0	0

Next, we illustrate how the parameters for pattern matching algorithm can be computed. Let's assume that 100% accurate data is available (that is, actor/area profile observations are equal exactly to the values in Table 3). We now calculate the mismatch between the profiles of actors and behavior tasks/activities using observed actor profiles and known model task profiles (task profiles for RED mission "*Cyber Attack on Critical Financial Infrastructure*" are shown in Table 4; here, *requirements for target* in task profile are equivalent to actor/area capabilities profile, and *requirements for resources* are equivalent to actor/area event profile).

Table 4: Example of the activities/task profiles

	Requirements for Target						**Requirements for Resources**								
Tasks/Activities	VAL	GOV	AINF	PINF	CSVC	CINFR	KNW	ATK	AINF	PINF	BACT	CSVC	CINFR	HACK	PER
Attacking with explosives	3	0	0	0	0	0	0	1	0	0	0	0	0	0	0
Diversionary explosives attacks/Stage	1	1	0	0	0	0	1	0	0	0	0	0	0	0	1
Mine/Crack User and Admin Passwords for Accounts	0	0	1	0	0	0	0	0	0	0	0	0	0	1	0
Insert Trojans to Capture Additional Passwords and Changes	0	0	0	1	0	0	0	0	1	0	0	0	0	1	0
Create false threat of bomb attack	0	1	0	0	0	0	0	0	1	0	0	0	1	0	0
Sell all stocks, bonds, and securities	0	0	1	1	1	0	0	0	0	1	0	1	1	1	0
Siphon Funds	0	0	0	1	0	0	0	0	0	1	1	0	1	1	0
Gain control over network	0	0	0	0	0	1	0	0	1	0	0	0	0	1	1
Accumulative Behavior Attributes	***4***	***2***	***2***	***3***	***1***	***1***	***1***	***1***	***3***	***2***	***1***	***1***	***3***	***5***	***2***

Error! Reference source not found. shows the example of the mismatch function coefficients c_{ki}. We have used the weighted sigmoid function for these computations. Similarly, parameters $c_{km;ij}$ can be computed based on profiles of task relationships and actor/area links. We do not present them in this document due to large size of corresponding matrices.

Table 5: Example of the mismatch parameters used by predictive algorithm

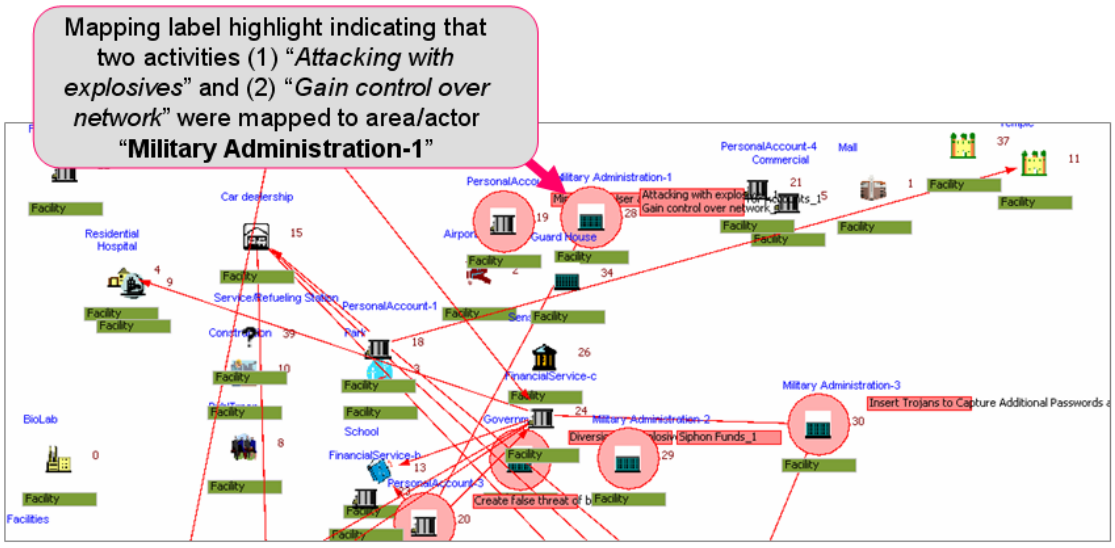
Mismatch Function Parameters Cki	Actors/Areas																			
Tasks/Activities	Airport	Park	Residential	Commercial	Government	Mansion	School	Bank	AdminAccount-1	PersonalAccount-1	PersonalAccount-2	PersonalAccount-3	PersonalAccount-4	FinancialService-a	SensorNet-A	Military Administration-1	Military Administration-2	Military Administration-3	Telecommunications	Processing/Treatment
Attacking with explosives	27.31	70	70	70	20	70	28.81	28.81	70	70	70	70	70	70	70	0	20	20	28.81	20
Diversionary explosives attacks/Stage	90	140	140	140	0	140	90	90	140	90	90	90	90	140	90	70	90	90	90	90
Mine/Crack User and Admin Passwords for Accounts	70	70	70	70	20	70	70	0	70	20	0	20	20	20	20	50	50	50	70	20
Insert Trojans to Capture Additional Passwords and Changes	90	90	90	90	40	40	40	20	90	90	70	70	90	90	90	0	20	0	40	90
Create false threat of bomb attack	90	90	90	90	40	90	90	70	90	40	40	0	40	90	40	70	70	70	90	90
Sell all stocks, bonds, and securities	230	230	230	230	80	180	180	0	230	180	160	160	180	130	180	160	120	160	180	180
Siphon Funds	130	130	130	130	80	80	80	20	130	130	110	110	130	130	130	60	0	60	80	130
Gain control over network	60	110	110	110	40	110	110	90	110	110	90	90	110	110	60	0	40	20	110	110

In **Error! Reference source not found.**, we highlighted in yellow the actors/areas not used for hostile actions. We can clearly see that the mismatch function parameters for these actors are large. Then, we indicate in RED the mismatch values = 0. If the links between tasks and areas were not accounted for, there will be ambiguity in mapping the activities/tasks to actors/areas. For example, activity “*Mine/Crack User and Admin Passwords for Accounts*” could be mapped similarly to actors “Bank” or “PersonalAccount-2” because the mismatch parameters in both cases are = 0. This ambiguity is due to the fact that we have ambiguous (although correct!) observations about the types of events taking place at these areas. The true mapping is to associate activity “*Mine/Crack User and Admin Passwords for Accounts*” with “PersonalAccount-2”, as indicated in **Error! Reference source not found.** with white cell. The algorithm is able to make mapping correction because it also considers the link mismatches between tasks and areas.

When the observed event and concomitant actor/area profile information is noisy, a single mapping of tasks to actors may not be efficient. Therefore, we use the algorithm to generate multiple mappings, and rank-order them with the value of corresponding mismatch function (equivalent to log-likelihood). An example of three alternative mappings generated by the algorithm is shown in 12(a). We can see that the first mapping had 100% accuracy of associating adversarial activities with the actors/areas, while other two mappings had 75% accuracy (yellow marking indicates incorrect mappings). 12(b) shows how the mapping is presented in CoPR prototype to the user in the form of the overlay labels on the geo-spatial terrain. Generating multiple mappings can reduce the probability of missing the activity, although increasing potential for false alarms. The false alarms are later removed by conducting additional ISR collection.

Task Name	**Mapped Area/Actor (1)**	**Mapped Area/Actor (2)**	**Mapped Area/Actor (3)**
Attacking with explosives	**Military Administration-1**	**Military Administration-1**	**Military Administration-1**
Diversionary explosives attacks/Stage	**Government**	**Government**	**Government**
Mine/Crack User and Admin Passwords for Accounts	**PersonalAccount-2**	**PersonalAccount-3**	**Bank**
Insert Trojans to Capture Additional Passwords and Changes	**Military Administration-3**	**Military Administration-1**	**Bank**
Create false threat of bomb attack	**PersonalAccount-3**	**PersonalAccount-3**	**PersonalAccount-3**
Sell all stocks, bonds, and securities	**Bank**	**Bank**	**Bank**
Siphon Funds	**Military Administration-2**	**Military Administration-2**	**Military Administration-2**
Gain control over network	**Military Administration-1**	**Military Administration-1**	**Military Administration-1**
% correct	100%	75%	75%

(a) Mapping of Actions to Actors (yellow cells indicate incorrect predictions)



(b) Mapping Visualization in CoPR Prototype

Figure 12: Example of the mapping obtained by the algorithm

Example of the analysis for different levels of data uncertainty: In the previous subsection, we showed an example of analysis when all data has been observed accurately although there was an ambiguity of the correct association between RED mission tasks and actors/areas. The algorithm was able to find the mapping with 100% accuracy for that situation. In this section, we show how the % of missing data affects the analysis and how the behavior identification can proceed over time.

In Figure 13, we show an example of how the accuracy of activity mapping changes with time of RED mission progress. The test points indicating changes in the accuracy corresponded to the times when new observations of activities were received. This happened as the RED executed next tasks in its mission. Thus, the X-axis can also be considered as the % of data available for analysis (this can also be considered as data completeness). The Y-axis plots the % of correctly mapped activities. In this diagram, there are three functions: blue line plots the accuracy when a single “best” mapping was taken as a solution; red line plots the accuracy when 3 best mappings were taken and then the mapping of activity was considered accurate if it was present in any of the three mappings; and green line plots the completeness of data available for analysis.

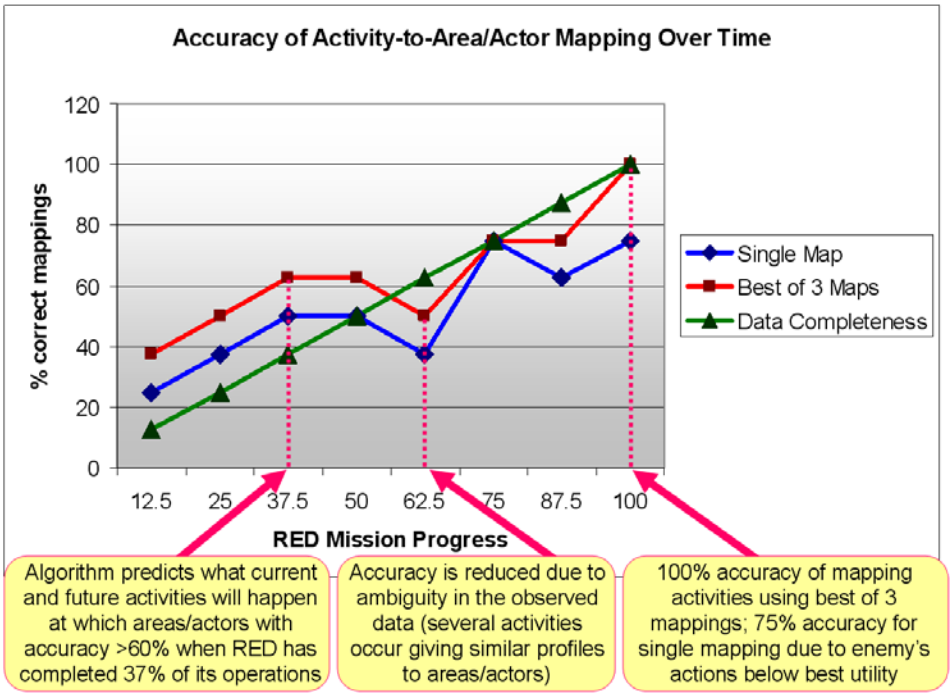


Figure 13: Example of changes in prediction over time as adversaries progress with their operations

From Figure 13, we notice the following three important observations:

(a) Algorithm is able to correctly map >62% of activities (current and future) when only 37% of RED operations have been completed. This indicates the ability of the algorithm to reason well with very scarce data;

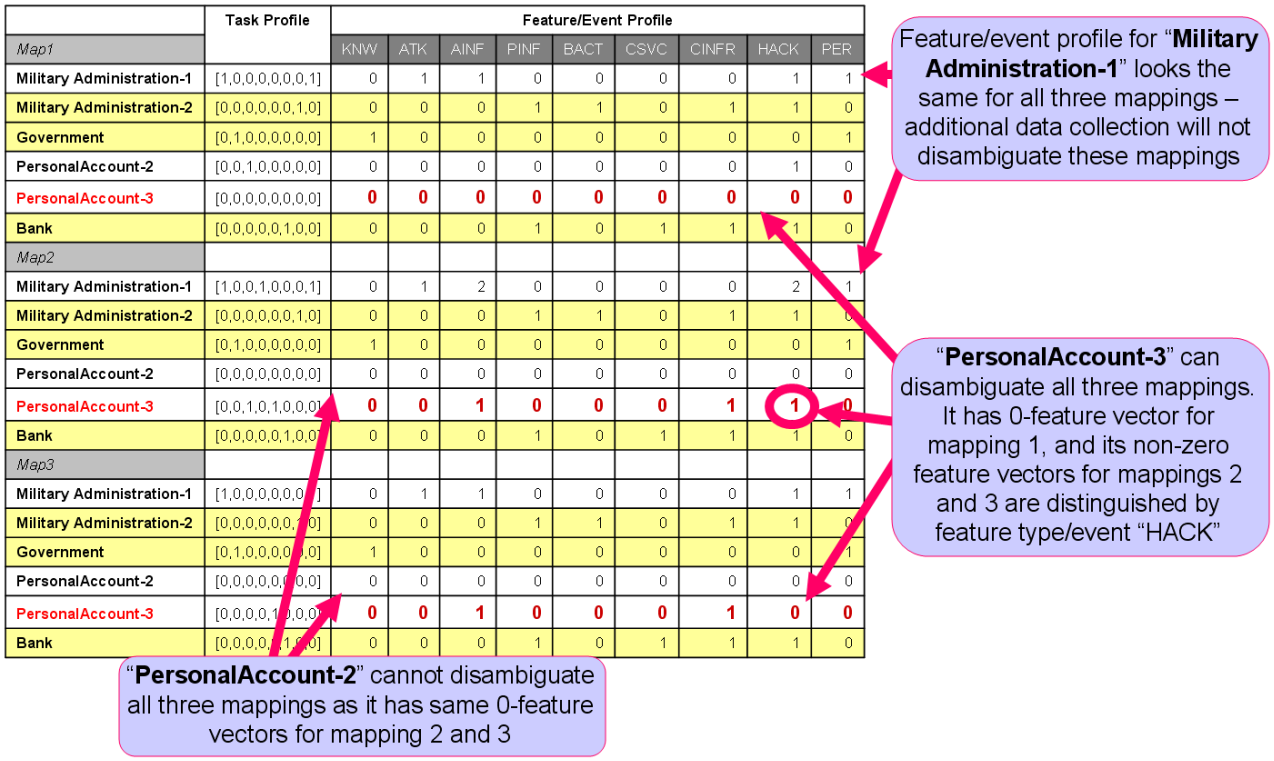
(b) Algorithm accuracy reduced when more data was presented. We analyzed this behavior and found that this was due to the ambiguity that new events introduced into the data. This ambiguity was caused by similarity of actor/area profiles given the set of events. In this situation, additional data collection is needed to disambiguate among many possibilities;

(c) While the 3-mapping option gives 100% accuracy of activity mapping when all data is available, a “single-best” mapping achieves only 75% accuracy. This was due to the fact that RED team did not act rationally in the simulation, and selected several intermediate options for its activities that were not geo-spatially efficient to the total RED mission execution. In this case, the “score of mismatch” computed by the algorithm, which accounts for the observation signal as well as “utility” (goodness of fit between the area/action and a task and links between tasks and actors), would score the incorrect mapping higher than the true one as performed by RED in the simulation.

Example of intelligence collection planning process: The intelligence collection planning algorithm identifies most critical information that can achieve highest impact on disambiguating current predictions. We use the information gain metric to score different ISR options. The process starts with the set of current predictions, from which we derive the feature profiles of actors/areas. Figure 14 shows an example of the area profiles using three mappings from Figure 12(a). The vectors for “task profile” are 0-1’s corresponding to the tasks that the area/actor is mapped to. We immediately see that actors “Bank”, “Government”, and “Military Administration-2” have exactly the same task profiles and therefore are the same for all mappings. Collecting data for these actors will not change the relative scores among these three mappings and therefore cannot disambiguate them. Hence, three potential areas of interest remain: “Military Administration-1”, “Personal Account-2”, and “Personal Account-3”.

We now need to perform a more detailed analysis than task profiles would allow. For this purpose, we use the feature/event profile which we calculate as the *sum of all resource requirements of tasks mapped to this area/actor*. While task profiles for “Military Administration-1” are different for all three mappings, the

feature/event profiles are similar for all three mappings. Hence, collecting intelligence at this actor/area will not change the mismatch functions and therefore will not result in distinguishing any of the mappings. Next, we look at event/feature profile for “Personal Account-2” and notice that while it can disambiguate mapping 1 from the other two, it has the same (zero) feature profile vector for mappings 2 and 3, and therefore this actor/area cannot differentiate all three mappings at the same time. Collecting intelligence at this actor/area will result in some reduction of the entropy, but may not produce the maximum information gain possible. Finally, we look at event/feature profile for “Personal Account-3” and see that it is clearly different for all three mappings: profile for mapping 1 has all 0’s, and profiles for mappings 2 and 3 are distinct by the “HACK” feature. Therefore, collecting data at “Personal Account-3” will completely disambiguate all current mapping predictions, and therefore we designate this actor/area for the ISR collection action. Specifically, the collection will include instruction to collect values (events) for either “AINF” or “CINFR” feature as well as “HACK” feature; no other features are necessary, and accordingly the actions and ISR asset allocations can be specified. As the result, the ISR collection planning process will define *where* the collection must be conducted, *what* must be collected, and *what will happen* if sought information is obtained.



	Task Profile	Feature/Event Profile								
Map1		KNW	ATK	AINF	PINF	BACT	CSVC	CINFR	HACK	PER
Military Administration-1	[1,0,0,0,0,0,0,1]	0	1	1	0	0	0	0	1	1
Military Administration-2	[0,0,0,0,0,0,1,0]	0	0	0	1	1	0	1	1	0
Government	[0,1,0,0,0,0,0,0]	1	0	0	0	0	0	0	0	1
PersonalAccount-2	[0,0,1,0,0,0,0,0]	0	0	0	0	0	0	0	1	0
PersonalAccount-3	[0,0,0,0,0,0,0,0]	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
Bank	[0,0,0,0,0,1,0,0]	0	0	0	1	0	1	1	1	0
Map2										
Military Administration-1	[1,0,0,1,0,0,0,1]	0	1	2	0	0	0	0	2	1
Military Administration-2	[0,0,0,0,0,0,1,0]	0	0	0	1	1	0	1	1	0
Government	[0,1,0,0,0,0,0,0]	1	0	0	0	0	0	0	0	1
PersonalAccount-2	[0,0,0,0,0,0,0,0]	0	0	0	0	0	0	0	0	0
PersonalAccount-3	[0,0,1,0,1,0,0,0]	**0**	**0**	**1**	**0**	**0**	**0**	**1**	**1**	**0**
Bank	[0,0,0,0,0,1,0,0]	0	0	0	1	0	1	1	1	0
Map3										
Military Administration-1	[1,0,0,0,0,0,0,1]	0	1	1	0	0	0	0	1	1
Military Administration-2	[0,0,0,0,0,0,1,0]	0	0	0	1	1	0	1	1	0
Government	[0,1,0,0,0,0,0,0]	1	0	0	0	0	0	0	0	1
PersonalAccount-2	[0,0,0,0,0,0,0,0]	0	0	0	0	0	0	0	0	0
PersonalAccount-3	[0,0,0,0,1,0,0,0]	**0**	**0**	**1**	**0**	**0**	**0**	**1**	**0**	**0**
Bank	[0,0,0,0,1,0,0,0]	0	0	0	1	0	1	1	1	0

Figure 14: Example of the feature profile derived from predictions

Conclusions

In this paper, we describe process, algorithms, and example analyses for recognizing adversarial behavior signatures and devising guided intelligence collection plans. We have tested behavior prediction algorithms with random data sets and have shown high accuracy of behavior/mission pattern recognition and activity mapping for large levels of data uncertainty. The ISR collection planning can improve the robustness of the behavior analyses products further by targeting the information collection most critical to current predictions. We have illustrated the process of behavior analysis and ISR planning on the example dataset developed in CoPR project. During our research, we identified several modeling challenges and additional functionality that must be implemented to improve the accuracy and reliability of the CoPR technology.

We presented the Experiment 3 above to showcase the challenges of adversarial behavior recognition. After analyzing these results, we concluded that the following improvements need to be made to the algorithms to increase the accuracy of behavior recognition and activity mapping:

(1) **Generate multiple mapping predictions for further analysis:** On one hand, having larger number of alternative mappings can reduce probability of miss; on the other hand, these mappings can be used to find information to be collected to improve the accuracy of behavior recognition;

(2) **Change which attributes are used for mismatch calculation:** When the task has not been completed, the algorithm should not expect to have observation events of corresponding activities. Therefore, event attributes should not be used for calculating the task-actor mismatch. However, when the task has completed, the preference should be given to events over standard actor/area capability profiles, as only events can indicate what actors did what activities. In this case, the capability attributes should only be used to “help” the algorithm find correct mapping when there are many missing events.

(3) **Track activity pattern over time:** The analysis and experiments presented in this paper were based on a “batch-mode algorithm” that did not consider temporal information in finding the mapping. Instead, integrating the mapping with mission state recognition would allow changing what attributes are considered during the mapping and account for sequencing of tasks at areas/actors. This will reduce the ambiguity in the data and improve the algorithm’s accuracy in computing the approximation to the likelihood function.

(4) **Use observation/event data at links between actors/areas:** In our analysis, we used event data for actors but accounted only for capability profiles for links. Including events from actor relationships will improve the algorithm accuracy.

(5) **Use different mismatch and utility functions:** We noticed the impact of different utility/mismatch functions on the accuracy of the behavior recognition and activity mapping. In our future research, we will develop alternative mismatch functions that could be tailored or trained to the domain of analysis and types of RED behaviors.

(6) **Tradeoff node and link attributes:** We noticed the impact on decision accuracy when node and link attributes are weighted differently in computation of the global pattern mismatch function. We will incorporate these weights as parameters for the behavior recognition model than can either be modified by the users or trained given training data sets of adversarial behavior.

In our current research, we are working to implement these improvements and technologies into the future CoPR system.

Acknowledgement:

This work was supported by AFRL Contract # FA8650-08-M-1334. The authors would like to thank AFRL customer Dr. Kirk Weigand, RYTC, for generating the original challenge, guiding the development of the solution, and providing us valuable feedback on the conceptual framework and results.

References:

G. Levchuk, Y. Levchuk, and K. Pattipati, “Identifying Command, Control and Communication Networks from Interactions and Activities Observations”, *Command and Control Research and Technology Symposium*, 2006, San Diego, CA.

Levchuk, G., hYu, F., Meirina, C., Singh, S., Levchuk, Y., Pattipati, K., Willett, P., & Kelton, K. (2007). Learning from the Enemy: Approaches to Identifying and Modeling the Hidden Enemy Organization. In A. Kott (Ed.). Information warfare and organizational decision-making. Norwood MA: Artech House.

Levchuk, G., D. Grande, K. Pattipati, Y. Levchuk, A. Kott (2008). Mission Plan Recognition: Developing Smart Automated Opposing Forces for Battlefield Simulations and Intelligence Analyses, to appear in Proceedings of the 13th International Command and Control Research and Technology Symposium.

Grande, D., G. Levchuk, W. Stacy, and M. Kruger, “Identification of Adversarial Activities: Profiling Latent Uses of Facilities from Structural Data and Real-time Intelligence”, Proceedings of the 13th International Command and Control Research and Technology Symposium, 2008.

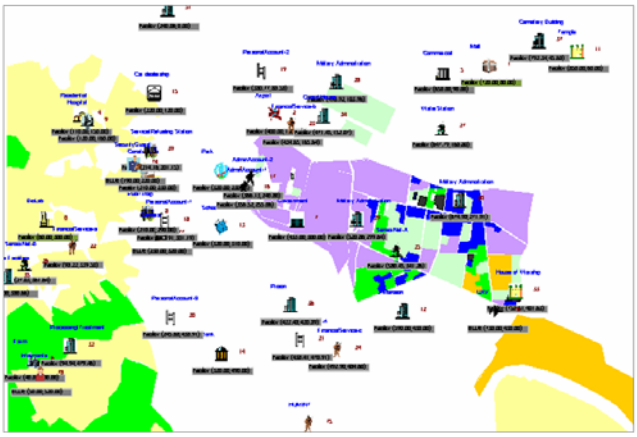
Levchuk, G., D. Lea, and K. Pattipati, “Recognition of Coordinated Adversarial Behaviors from Multi-Source Information”, Proceedings of SPIE Defense and Security Symposium, Volume 6943, Orlando, FL, 2008

Appendix: Example Dataset

In CoPR project, we developed a use-case and corresponding example dataset that utilized synthetic data intentionally designed to emulate real-world data. Synthetic data was needed to provide the ground truth about the reasons or prerequisite actions of the adversaries, which is required to measure the accuracy of predictive algorithms. In addition, we needed to create situations with diverse ground truth activities to be able to test the ability of predictive algorithms to recognize different types of hostile operations. In the following, we describe in detail the elements of the use case.

The environment: We have described the environment terrain as structural information about buildings and geo-political areas. Our data consisted of a list of buildings/areas together with their location, size, and function. We defined eight functions: plant, transportation, government, military, infrastructure, social, residential, and network. In particular, the facilities with “network” function could be used to conduct cyber attacks. Accordingly, the geo-political area of analysis contained several types of facilities and areas, including houses of worship, military administration facilities, water treatment facilities, schools, oil service facilities, shopping centers, data storage and service networks, government office buildings, police stations, parks and entertainment centers, manufacturing and laboratories, storage and warehouse facilities, automotive repair centers, financial institutions, services control stations, rental facilities, etc.

Figure 15 shows buildings/areas layout in the use case, and provides an example of their functions. In addition to size and function of buildings, we used other attributes to define the profile of buildings/areas, including data about the neighboring population, cultural trends, demographics of the region, availability of technologies at the facility, etc.



(a) Area Layout for CoPR Use-case

Area	Function
BioLab	Plant
Mall	Infrastructure
Airport	Infrastructure
Park	Social
Farm	Infrastructure
Government	Government
FinancialService	Infrastructure
Oil/Gas Facilities	Military
SensorNet	NetworkNode
Military Administration	Military
WaterStation	Infrastructure
AdminAccount	Government

(b) Example of Building List and Functions

Figure 15: Dataset area layout and functions

The actors: Buildings and areas in the use case were passive actors, to which we sometimes refer to as targets. The active actors in the dataset have been defined to represent the members of RED organizations, the elements of BLUE forces and their assets, and other actors such as non-government organizations (NGOs), normal people in the environment, etc. The adversaries were represented by a set of team-actors, including bomb makers, reconnaissance cells, support personnel, financiers, transportation, security teams, hackers, and weapons attack teams. The types of teams have been selected based on analysis of real-world terrorist networks and their operations.

The situation, RED missions, activities, and organization: In the area of interest, several adversarial attacks occurred over time. To succeed in task execution, RED team had to assign a set of actors to perform the operations (tasks of their plan) based on the resources required for execution of this operation and resources capabilities of RED actors. The application of actor's resources to the task can be viewed as an individual action by this actor. For example, task "dump contaminant" requires possession of chemical poison materials and the unloading activity; this task can then be performed with the help of a support team and any of the actors who possess "chemicals" capability (this capability must be acquired during mission with the help of a financial group). For more details on the resource-based task modeling, see Levchuk et al., 2002; 2003.

The attack events in CoPR dataset were generated using organizational performance simulation functionality of CoPR prototype, which was designed as agent-based distributed controller of simulated actors. This model implemented four main organizational and actor behaviors: (i) selecting tasks in the mission to execute, targets for corresponding operations, and assigning actors to those tasks; (ii) moving in the environment by routes and zones; (iii) engaging other simulated actors in the environment – e.g., hostile actors can attack other actors; and (iv) sensing information in the environment – e.g., actors can detect other actors, classify them, observe actions, etc. In our simulations, we have used a single mission and organization for RED at a time. The BLUE actors operated in two phases: in first phase, BLUE actors were kept stationary in the environment to provide sensed data feeds (detected entities and action events) for predicting RED's hostile behaviors; in second phase, BLUE actors would move to execute the ISR plan devised by CoPR intelligence collection algorithm.

We have developed several adversarial behavior signatures, i.e. mission patterns, and ran the simulation comparing the predictive algorithm capability against each of them. An example of five such patterns is shown in Figure 16. Mission plan specification captured the spatial information (information about task locations via specification of target requirements), temporal information (sequencing of tasks according to precedence constraints in the mission), and type information (overlap in the types of activities that need to be performed). The use case contained different modus operandi (missions) for adversaries that, while distinct in the objective for the adversaries, had overlapping operations.

RED actors can take different roles and form different adversarial organizations depending on the membership of actors in different cells and their subordination to intermediate RED commanders. For our example dataset, we defined a single RED organization and did not require identifying its structure. We will incorporate the organizational network identification in our future work.

Quantitative definition of actors, operations, and execution utility: To identify who are hostile actors and what actions they execute, we quantitatively defined models of action and actor *profiles*. The action profile definition also enabled specification of observable action *signals* – i.e., the events that can be observed about the actor executing the action. For example, if the action is to store the weapons, its might require storage facility and possession of weapon materials. Only adversarial actors possessing such materials can conduct this action, and it can only be done at a facility with existing storage capacity. The match between profile of the actor and profile of the task then defines the utility of action to the enemy. On the other hand, this action may result in the events of loading materials from trucks to the facility. Adding this information to utility match helped determine the true occurrence of the action in the area and the actors involved.

To determine how actors can be associated with actions, we defined three classes of attributes:

- *attributes describing capabilities of actors:* this data is current before the start of enemy's operations; data about facility capabilities can be collected from analyzing imagery by automated or manual means (e.g., using radar scans and intelligence data about availability of resources at facilities); data about capabilities of human actors can be collected based on intelligence reports about them, where demographic information about their areas of operation can fill the data gaps; data about groups and geo-political entities can be collected using open-source intelligence (websites, news reports, blogs, etc.).

- *attributes describing current events and actions performed by actors:* this information is dynamic and can be obtained from human collection teams, UAV data feeds, etc.
- *attributes describing previous actions of the actors:* this is historic information that could have been obtained from the past events in the area of interest.

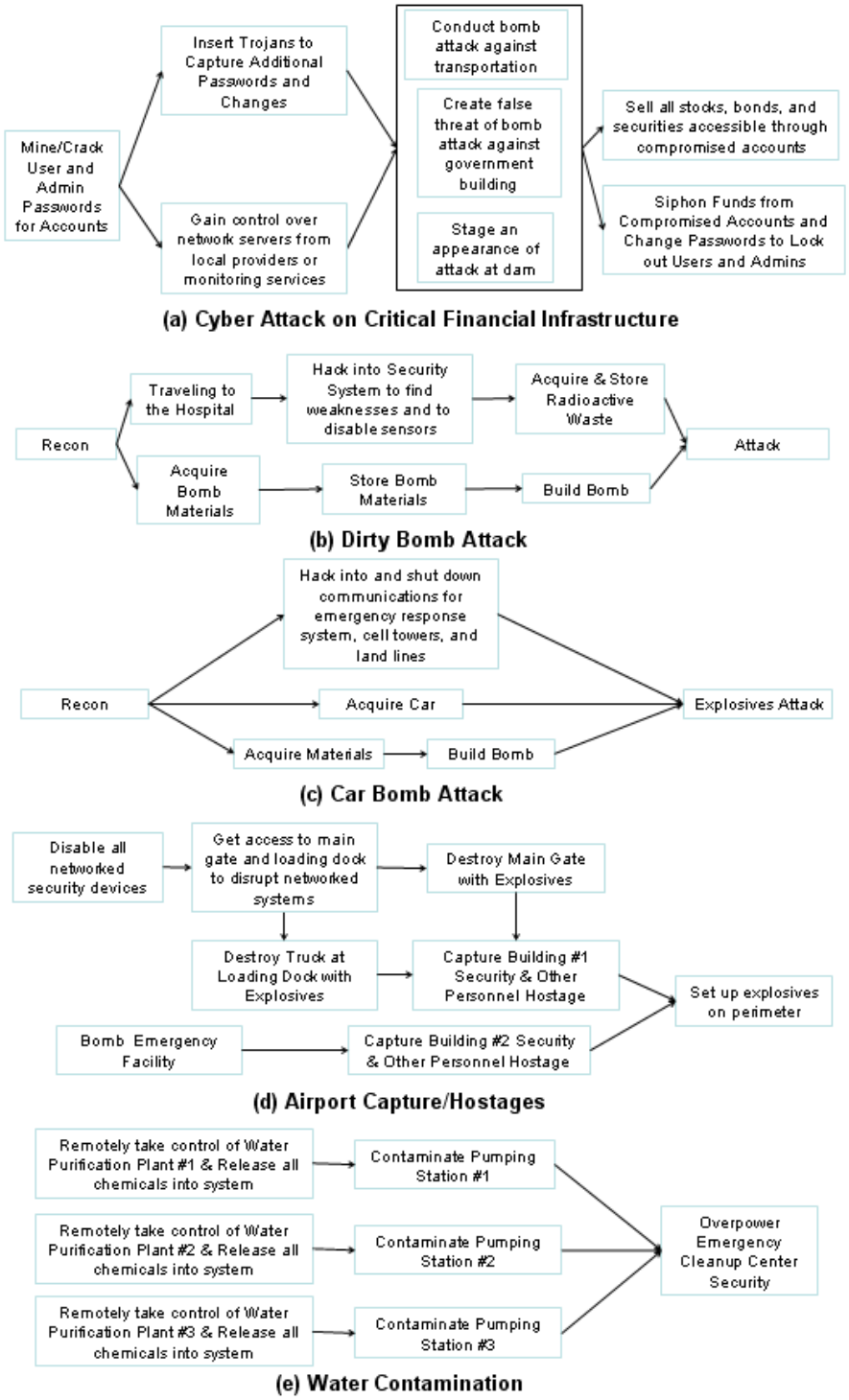


Figure 16: RED missions in CoPR example dataset

The CoPR example dataset definition started with quantitative specification of behavior signatures, including the following:

1. List of attribute types (for capabilities, current events, and historic data);

2. List of RED model tasks/activities and their attributes;
3. List of RED model organization roles and their attributes;
4. List of patterns of RED model (hypotheses) missions; and
5. List of patterns of RED model (hypotheses) organizations.

Then, we defined observed data, including the following:

1. Observed actors (including RED) and their attributes;
2. Facilities and their attributes; and
3. Events (capability, interaction, and action) and their attributes.

In our example dataset, we focused only on capabilities and current event attributes and defined the following attributes for these classes:

- *Value (VAL):* indicates the significance in attacking a target for RED
- *Transportation (TRS):* indicates the resources/availability/event of transporting the materials/bomb
- *Storage (STR):* indicated ability and conduct of storing materials for extended time periods
- *Reconnaissance (REC):* indicates a capability to conduct recon missions by RED and the needs for task definition
- *Attack (ATK):* the capability acquired when bomb is manufactured
- *Money (MON):* availability of and outcome of financial transaction
- *Security (SEC):* defined the security of conducting hostile actions for RED operatives, as well as the capability of conduct security operations
- *Materials (MAT):* defined the availability of materials that could be used to manufacture the explosives
- *Technology (TEC):* indicated availability of technology to manufacture dirty bomb or need for/availability of knowledge of how the explosives is manufactured.
- *Poison (POIS):* chemical or biological material for dirty-bomb attack or water poisoning
- *AdminInfo/Access (AINF):* indicates the ability to access network resources under administrative privileges
- *Personal & Financial Info (PINF):* indicates information about person’s profile and access to their financial and other accounts
- *Bank Accounts (BACT):* access to financial institutions and banking services
- *Control of Sensors (CSENS):* indicates ability to manipulate remote sensors
- Control of Services (CSVC): ability to control network services, and remote services, such as traffic lights, gates, bridges, water and electricity distribution, media networks, communication channels, etc.
- *Control of Infrastructure (CINFR):* ability to physically control infrastructure
- *Hacker (HACK):* expertise in password breaking and network intrusion/cyber attacks
- *Personnel (PER):* need for personnel support, e.g., for loading explosives onto the cars, etc.
- *Public Health (HELTH):* indicates the access to public safety resources, such as food, water, medication, etc.

When defining actors (humans, groups, facilities) for our dataset, we specified their profile using capabilities and current events. When defining tasks (RED operations), we split attributes notionally into two vectors:

- *resource requirement* – what actors should possess to successfully conduct the operation; this vector is matched with the actor profile; and
- *target requirements* – what facilities should possess to support the operation; this vector is matched with the facility profile.

Figure 17 shows some examples of attributes that we have defined for actors and tasks in CoPR dataset. We intentionally created tasks that had overlapping attributes, so that missing action observations could result in the confusion of associating these observations with more than one task. Based on the functions of buildings and areas, we have defined their capability vectors. The capabilities of actors have been defined based on their knowledge, skills, possessed resources, and roles in the enemy organization.

	Resource Requirements													*Target Requirements*										
Role	SZ	SEC	STR	MAT	TEC	KNW	MON	REC	POIS	AINF	PINF	BACT	CSENS	SZ	SEC	STR	MAT	TEC	KNW	MON	REC	POIS	AINF	PINF	BACT	CSENS
Acquiring poison	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
Recon	0	0	0	0	0	0	0	1	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
Storing explosive materials	0	0	0	1	0	0	0	0	0	0	0	0	0	3	0	0	0	0	0	0	0	0	0	0	0	0
Assemble bomb	0	0	0	1	0	1	0	0	0	0	0	0	0	2	0	0	0	1	0	0	0	0	0	0	0	0
Insert Trojans to Capture Additional Passwords and Changes	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0
Siphon Funds from Compromised Accounts and Change Passwords to Lock out Users and Admins	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0
Gain control over network to disable/manipulate sensors/monitoring capabilities/system	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Create false threat of bomb attack against government building	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

(a) RED Model Tasks/Activities

	Capabilities												
Facility	SZ	SEC	STR	MAT	TEC	KNW	MON	REC	POIS	AINF	PINF	BACT	CSENS
BioLab	2	0	1	0	1	0	0	0	1	0	0	0	0
Mall	4	1	3	0	0	0	2	0	0	0	0	1	0
Airport	10	0	3	3	2	0	0	0	0	0	0	0	1
Park	3	3	0	0	0	0	0	0	0	0	0	0	0
Residential	1	5	1	0	0	0	0	0	0	0	0	0	0
Commercial	3	2	3	1	2	0	1	0	0	0	0	0	0
AdminAccount	0	0	0	0	0	0	0	0	0	1	0	0	0
SensorNet	0	0	0	0	0	0	0	0	0	1	0	0	1

(b) RED Areas/Facilities

	Capabilities												
Role	SZ	SEC	STR	MAT	TEC	KNW	MON	REC	POIS	AINF	PINF	BACT	CSENS
SecurityDetail	0	1	0	0	0	0	0	0	0	0	0	0	0
Hackers	0	0	0	0	0	0	0	0	0	1	1	1	0
Attacker	0	0	0	0	0	0	0	0	1	0	0	0	0
Financier	0	0	0	0	0	0	1	0	0	0	0	0	0
Recon	0	0	0	0	0	0	0	1	0	0	0	0	0
Bombmaker	0	0	0	0	0	1	0	0	0	0	0	0	0

(c) RED Actors

Figure 17: Example of attributes for actors and tasks

Observable data was extracted from capability, action, and interaction events, which had time, location, and individuals involved in the event. To explain how we defined action events, we note that often multiple RED actors participate in the same operation due to the need to satisfy the resource requirements of tasks, while we assumed that a single facility/area is used to conduct an operation. When actors perform their portion of the operation, this is equivalent to them “applying” their capabilities to the task or target of the operation. For example, the task “*assemble bomb*” (Figure 16) requires three types of capabilities: materials, technology, and security protection. These capabilities can be brought together by explosives specialists (who possess technology capability) and support team (who possess materials and security). Thus, an observable action event is the detection of activity associated with using these capabilities. The data of action events included the following fields: i) *time* and geo-spatial *location* of event; ii) *actor* involved in the event; and iii) *capabilities* of actor *used* in the action.

Thus, for the example of “assemble bomb” operation, BLUE may detect action events describing the operatives of RED conducting security around the building, and actors who brought bomb making materials to the building, while the information about actors with technological bomb assembly knowledge might be missing. Therefore, the observed data might be incomplete, ambiguous, and noisy. Overall, CoPR can deal with four types of data collection noise:

(1) Event miss: Events about the activities are captured by sensors (SIGINT, HUMINT, IMINT, …), and not all such events might be detectable. For example: *Facility was used to hold a meeting between terrorists, but there was no UAV/patrol at the time in the area*. As an outcome, all attributes from the missed event are excluded from analysis.

(2) Attributes miss: Sensors (humans, algorithms, …) might miss an attribute present in the incoming data/event. For example: *LIDAR data was incorrectly analyzed by the image classification algorithm*. As an outcome, correct attribute was missed and excluded from the analysis.

(3) Irrelevant Attributes/events: Sensors (humans, algorithms, …) might falsely perceive that an attribute was present in the incoming data/event or might falsely add an event due to deceptive information that has never occurred or irrelevant information wrongly associated with event. For

example: *Analyst, based on studied imagery, reported a presence of hide-out at the construction site.* As an outcome, incorrect attribute is added as input and is used for analysis.

(4) Attributes errors: Sensors (humans, algorithms, …) might incorrectly assess the value of an attribute in the incoming data/event. For example: *Analyst, based on studied imagery, reported that the building had large footprint, while building had medium-to-small footprint.* As an outcome, incorrect attribute value is used as input for analysis.

The example in Figure 18 shows how the observed information about actors and facilities might get generated. We have developed the uncertainty layer component that takes the true data from the simulation and makes it noisy for the sensitivity analysis of algorithm accuracy versus different noise levels.

Hence, we extract the profiles of actors and facilities from event attribute vectors. Similar data can be collected about linkages between actors and facilities. For example, linkages between actors are related to actor interactions, and linkages between facilities are profiles from the activities on the roads between them. The profiles of actors and facilities are then organized in the form of a data network – an attributed graph where the nodes are actors/facilities and links are actor and facility interactions. The nodes and links are labeled with profiles in the form of observed attribute vectors.

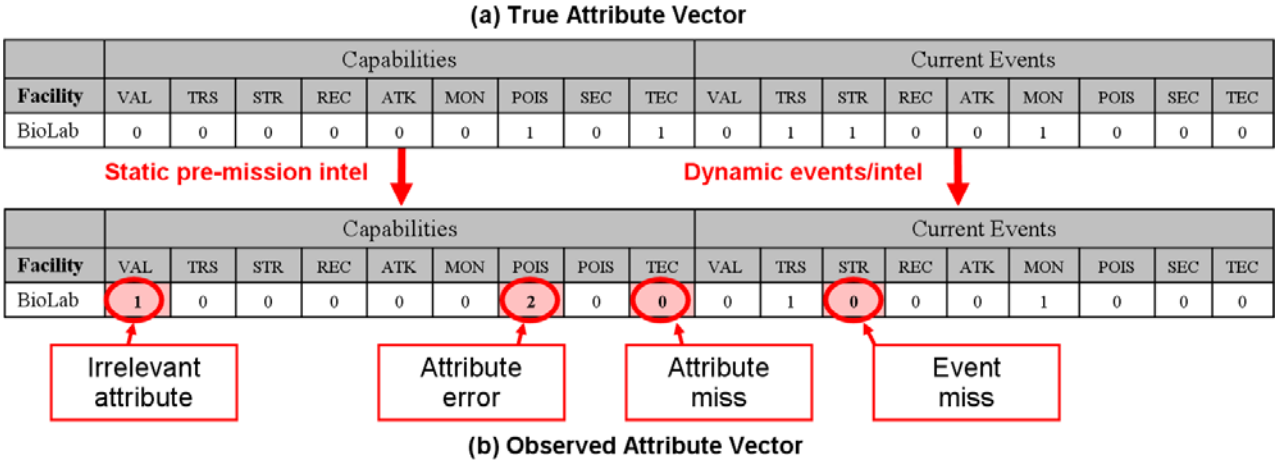


Figure 18: Example of observed data generation for task “Assemble Bomb”