An ACP-Based Approach to Intelligence and Security Informatics

Article · September 2015

DOI: 10.1007/978-3-319-08624-8_3

3 authors, including:

Fei-Yue Wang
Qingdao academy of intelligent indistries
989 PUBLICATIONS **10,496** CITATIONS
SEE PROFILE

Wenji Mao
Chinese Academy of Sciences
121 PUBLICATIONS **1,079** CITATIONS
SEE PROFILE

An ACP-Based Approach to Intelligence and Security Informatics

Fei-Yue Wang, Xiaochen Li and Wenji Mao

1 Introduction

The field of Intelligence and security informatics (ISI) is resulted from the integration and development of advanced information technologies, systems, algorithms, and databases for international, national, and homeland security-related applications, through an integrated technological, organizational, and policy-based approach [2]. Traditionally, ISI research and applications have focused on information sharing and data mining, social network analysis, infrastructure protection, and emergency responses for security informatics. Recent years, with the continuous advance of related technologies and the increasing sophistication of national and international security, new directions in ISI research and applications have emerged that address the research challenges with advanced technologies, especially the advancements in social computing. This is the focus of discussion in the current chapter.

As a new paradigm of computing and technology development, social computing can help us understand and analyze individual and organizational behavior and facilitate ISI research and applications in many aspects. To meet the challenges and achieve a methodology shift in ISI research and applications, in this chapter, we shall propose a social computing-based research paradigm consisting of a three-stage modeling, analysis, and control approach that researchers have used successfully to solve many natural and engineering science problems, namely the ACP (Artificial societies, Computational experiments and Parallel execution) approach [10–14].

F.-Y. Wang (✉) · X. Li · W. Mao
The State Key Laboratory of Management and Control for Complex Systems,
Institute of Automation, Chinese Academy of Sciences, Beijing, China
e-mail: feiyue.wang@ia.ac.cn; feiyue@gmail.com

X. Li
e-mail: xiaochen.li@ia.ac.cn

W. Mao
e-mail: wenji.mao@ia.ac.cn

F.-Y. Wang
The Research Center for Computational Experiments and Parallel Systems Technology,
The National University of Defense Technology, Changsha, Hunan, China

R.R. Yager et al. (eds.), *Intelligent Methods for Cyber Warfare*,
Studies in Computational Intelligence 563, DOI 10.1007/978-3-319-08624-8_3

Based on the ACP approach, in this chapter, we shall focus on behavioral modeling, analysis and prediction in security informatics. We shall first present a knowledge extraction approach to acquire behavioral knowledge from open-source intelligence and facilitate behavioral modeling. On the basis of behavioral modeling, we shall then present two approaches to group behavior prediction. The first approach employs plan-based inference and explicitly takes the observed agents preferences into consideration. The second approach employs graph theory and incorporates a graph search algorithm to forecast complex group behavior. We finally provide the results of experimental studies to demonstrate the effectiveness of our proposed methods.

2 The ACP Approach

The ACP approach [10–14] is composed of three interconnected parts: artificial societies for modeling, computational experiments for analysis and parallel execution for control. We shall discuss them in detail below.

2.1 Modeling with Artificial Societies

In the literature, there are no effective formal methods to model complex social-techno systems, especially those heavily involving human and social behavior. The ACP framework posits that agent-based artificial societies are the most suitable modeling approach to social modeling and social computing. An artificial society-based approach has three main components: agents, environments, and rules for interactions. In this modeling approach, how accurately the actual system can be approximated is no longer the only objective of modeling, as it is the case in traditional computer simulations. Instead, the artificial society developed is considered as an actual systemłan alternative possible realization of the target society. Along this line of thinking, the actual society is also considered as one possible realization. As such, the behaviors of the two societies, the actual and the artificial, are different but fit different evaluation and analysis purposes. Note that approximation with high fidelity is still the desired goal for many applications when it is achievable but can be relaxed otherwise, representing a necessary compromise that recognizes intrinsic limits and constraints of dealing with complex social-techno-behavioral systems.

2.2 Analysis with Computational Experiments

Traditional social studies primarily rely on passive observations, small-scale human subject experimental studies, and more recently computer simulations. Repeatable experiments are very difficult to conduct, due to a number of reasons including but not limited to research ethics, resource constraints, uncontrollable conditions, and unobservable factors. Artificial societies can help alleviate some of these prob-

lems. Using artificial societies as social laboratories, we can design and conduct controllable experiments that are easy to manipulate and repeat. Through agent and environmental setups and interaction rule designs, one can evaluate and quantitatively analyze various factors and what-if scenarios in social-computing problems. These artificial society-based computational experiments are a natural extension to traditional computer simulation. Basic experimental design issues related to model calibration, analysis, and verification need to be addressed. Furthermore, design principles such as replication, randomization, and blocking, guide these computational experiments just as they would guide experiments in the physical world.

2.3 Control and Management Through Parallel Execution

Parallel execution refers to the fact that long-lived artificial systems can run in parallel and co-evolve with the actual systems they model. This is a generalization of controllers as used in classical automation sciences, which use analytical models to drive targeted physical processes to desired states. This parallel execution idea provides a powerful mechanism for the control and management of complex social systems through co-evolution of actual and artificial systems. The entire system of systems can have three major modes of operations. In the learning and training mode, the actual and artificial systems are disconnected. The artificial systems can be used to train personnel. In the experimenting and evaluating mode, connections or syncing between the actual and artificial systems take place in discrete times. Computational experiments can be conducted between these syncs, evaluating various policies. In the controlling and managing mode, the artificial systems are used as the generalized controllers of the actual systems with two systems constantly connected. Social computing applications, especially those involving security, control and management of social activities, can benefit directly from parallel execution.

3 Modeling Organizational Behavior

Action knowledge has been widely used in modeling and reasoning about agent's behavior. Action knowledge is typically represented using plan representation, which includes domain actions and the states causally associated with the actions (i.e., action preconditions and effects) [3]. Action precondition is the condition that must be made true before action execution. Action effect is the state achieved after action execution. Since action knowledge is the prerequisite of various security-related applications in behavior modeling, explanation, recognition and prediction, in this section, we present a knowledge extraction approach to acquire action knowledge, making use of the massive online data sources. The action extraction procedure includes action data collection, raw action extraction and action refinement [7]. Below we introduce the automatic extraction of action preconditions and effects from online data.

3.1 Extracting Action Knowledge from the Web

Extracting causal relations has been studied in previous related research (e.g., [4, 6, 8]). The focus of our work is different from those of previous research in two aspects. First, instead of finding general causal relations between two clauses or noun phrases, our focus is to find the causal relations between actions and states C action knowledge for behavioral modeling. Second, we need to acquire richer knowledge types C not only causal relations, but also goals, reasons and conditions associated with the actions. Sil et al. [9] propose a SVM-based approach to build classifiers for identifying action preconditions and effects. As their work only tests a small number of actions all selected from one frame in FrameNet, and all the actions are treated as single verbs, the performance of their approach in complex and open domain is unclear.

In extracting action preconditions, we differentiate several types of precondition: necessity/need, permission/possibility and means/tools. We classify the patterns into four categories based on their types and polarities. Tables 1 and 2 shows the linguistic patterns we design for extracting action preconditions and effects. To ensure the quality of the extracted causal knowledge, we prefer rule-based approach which can achieve relatively high precision. On the other hand, as our work is based on the open source data, recall rate could be compensated by the huge volume of online resources.

3.2 Computational Experiment on Terrorist Organization

As a great amount of reports about this group and its historical events are available online, we employ computational methods to automatically extract group actions and causal knowledge from relevant open source textual data. The textual data we use are the news about Al-Qaeda reported in *The Times*, *BBC*, *USA TODAY*, *The New York Times* and *The Guardian*, with totally 953,663 sentences.

Among the official investigation reports, 13 real attacks perpetrated by $Al-Qaeda$ have relatively complete descriptions. Intelligence analyst helped us manually compose the action knowledge of each attack based on these descriptions, and these form the basis of our experiment. We evaluate the performance of our method by checking how many actions and states specified in each attack example are already covered by the domain actions and causal knowledge we extract. Table 3 shows the results of the experimental study. The average coverage rates of the actions, preconditions, effects and states (preconditions plus effects) are 85.8, 74.1, 78.7 and 75.6 %, respectively. In general, the experimental results verify the effectiveness of our approach.

After action knowledge acquisition, we collect organizational behavior knowledge with quality. Based on the action knowledge we collect, we further employ planning algorithm to generate attack plans about this group and construct the plan library

Table 1 Patterns for extracting action preconditions

Precondition	Necessity/need	<action (verb-ing+object/verb-ing) set> *require* | *demand* | *need* <precondition set> <node-name> *need* <precondition set> *to* <action (verb+object/verb) set>
	Permission/possibility	<precondition set> *allow* <node-name> *to* <action (verb+object/verb) set> <precondition set> *enable* | *create the possibility for* <nodename> *to* <action (verb+object/verb) set>
	Means/tools	<node-name> *use* <precondition set> *to* <action (verb+object/verb) set> *provide* | *supply* | *offer* <precondition set> *for* <node-name> *to* <action (verb+object/verb) set> *provide* | *supply* | *offer* <node-name> *with* <precondition> *to* <action (verb+object/verb) set>
	Negative patterns	<¬precondition set> *prevent* | *stop* <node-name> *from* <action (verb-ing+object/verb-ing) set> <¬precondition set> *disable* | *undermine* <node-name> *to* <action (verb+object/verb) set> *lack of* <precondition set> *prevent* | *stop* <node-name> *from* <action (verb-ing+object/verb-ing) set> *the shortage of* <precondition set> *disable* | *undermine* <action (verb-ing+object/verb-ing) set> *cannot* <action (verb+object/verb) set> *without* | *unless* <precondition set>

Table 2 Patterns for extracting action effects

Effect	Causation	<action (verb-ing+object/verb-ing) set> *bring about* | *lead to* | *result in* | *trigger* | *cause* | *produce* | *give rise to* <effect set> <effect set> *be caused* | *produced* | *triggered* | *brought about by* <action (verb-ing+object/verb-ing) set> <node-name> <action (verb+object/verb) set> *to cause* | *bring about* | *produce* | *trigger* <effect set> <node-name> <action (verb+object/verb) set>, *causing* | *producing* | *triggering* | *resulting in* | *leading to* <effect set> <node-name> <action (verb+object/verb) set>, *which* | *that bring about* | *lead to* | *result in* | *trigger* | *cause* | *produce* <effect set> <effect set> caused | produced by <action (verb-ing+object/verb-ing) set> *What bring about* | *lead to* | *result in* | *trigger* | *cause* | *produce* | *give rise to* <effect set> be <action (verb-ing+object/verb-ing) set>
	Reason/goal	*the reason of* | *reason for* | *cause of* <action (verb-ing+object/verbing) set > *be* <effect set> <node-name> <action (verb+object/verb) set> *because of* | *on account of* | *due to* <effect set> <action (verb-ing+object/verb-ing) set> *be due to* <effect set> <node-name> <action (verb+object/verb) set> *for the purpose of* | *in an attempt to* | *in an effort to* | *in order to* | *so as to* | *in the cause of* <effect set>

Table 3 Experimental results on causal knowledge and action extraction

Attack example	Action converge	Precondition converge	Effect converge	State converge
1	0.833	0.818	0.833	0.824
2	0.800	0.778	0.800	0.786
3	0.900	0.727	0.900	0.781
4	0.889	0.737	0.778	0.750
5	0.857	0.733	0.714	0.727
6	0.833	0.727	0.833	0.765
7	0.875	0.765	0.750	0.760
8	0.833	0.667	0.833	0.722
9	0.889	0.684	0.889	0.750
10	0.900	0.800	0.800	0.800
11	0.857	0.750	0.714	0.739
12	0.857	0.692	0.714	0.700
13	0.833	0.750	0.667	0.722
Average	0.858	0.741	0.787	0.756

[7]. Plan library represents the groups strategic plans and behavioral patterns, which are the key of organizational behavior modeling. Below is an example plan in this groups plan library (The rectangles denote actions and the rounded rectangles denote preconditions and effects).

4 Forecasting Group Behavior via Plan Inference

Group behavior prediction is an emergent research and application field in intelligence and security informatics, which studies computational methods for the automated prediction of what a group might do. As many security-related applications could benefit from forecasting an entitys behavior for decision making, assessment and training, it is gaining increasing attention in recent years. Recent progress has made it possible to automatically extract plan knowledge (i.e., actions, their preconditions and effects) from online raw textual data and construct group plans by means of planning algorithm, albeit in the restrictive security informatics domain (see Sect. 3). On the basis of this, we present two plan-based approaches to group behavior forecasting in this section. The first approach is based on probabilistic plan inference, and the second approach is aimed at forecasting complex group behavior via multiple plan recognition (Fig. 1).

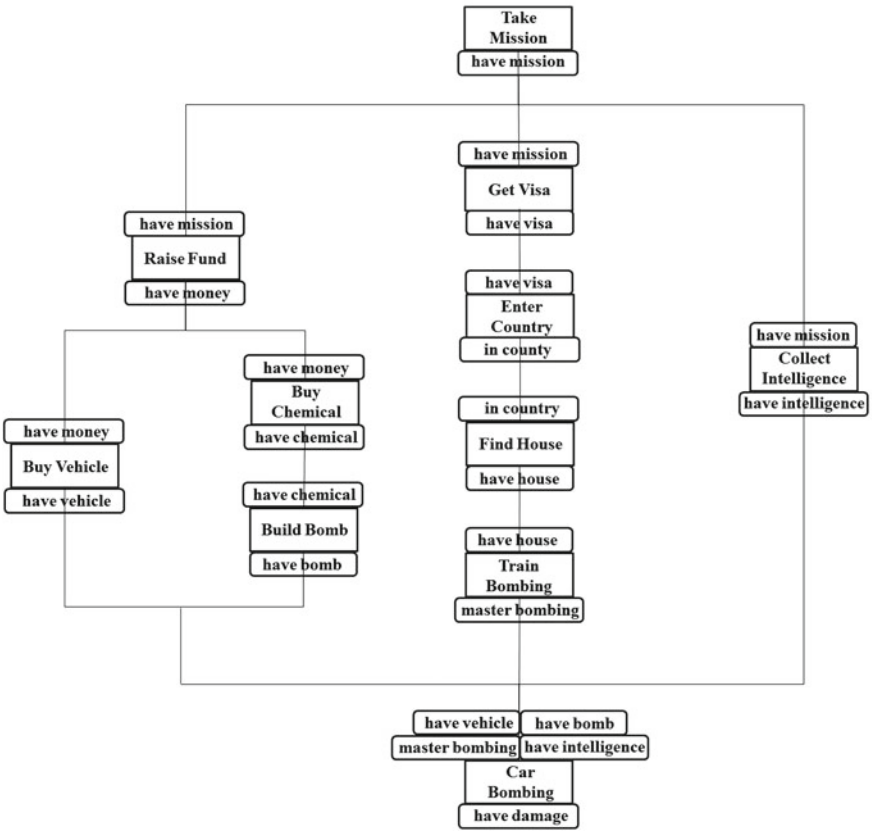

Fig. 1 An example strategic plan of the group

4.1 The Probabilistic Plan Inference Approach

Plan representations are used by many intelligent systems. In a probabilistic plan representation, the likelihood of states is represented by probability values. To represent the success and failure of action execution, we use execution probability $P_{execution}$ to represent the likelihood of successful action execution given action preconditions are true. An action effect can be nondeterministic and/or conditional nondeterministic. We use effect probability P_{effect} to represent the likelihood of the occurrence of an action effect given the corresponding action is successfully executed, and conditional probability $P_{conditional}$ to represent the likelihood of the occurrence of its consequent given a conditional effect and its antecedents are true. The desirability of action effects (i.e., their positive/negative significance to an agent) is represented by utility values. Outcomes are those action effects with non-zero utility values. We use expected utility (EU) to represent the overall benefit or disadvantage of a plan.

Our approach is based on the fundamental MEU (maximum expected utility) principle underlying decision theory, which assumes that a rational agent will adopt a plan maximizing the expected utility. The computation of expected plan utility captures two important factors. One is the desirability of plan outcomes. The other is the likelihood of outcome occurrence, represented as outcome probability. We use the observed evidence to incrementally update state probabilities and the probabilities of action execution. The computation process is realized through recursively using plan knowledge represented in plans.

4.1.1 Probability of States

Let E be the evidence. If state x is observed, the probability of x given E is 1.0. Observations of actions change the probabilities of states. If action A is observed executing, the probability of each precondition of A should be 1.0, and the probability of each effect of A is the multiplication of its execution probability and effect probability. If A has conditional effects, the probability of a consequent of a conditional effect of A is the product of its execution probability, conditional probability and the probabilities of each antecedent of the conditional effect.

- IF $x \in precondition(A)$, $P(x|E) = 1.0$
- IF $x \in effect(A)$, $P(x|E) = P_{execution}(A|precondition(A)) \times P_{effect}(x|A)$
- IF $x \in consequent(e) \wedge e \in conditional - effect(A)$, $P(x|E) = P_{execution}(A|precondition(A)) \times P_{conditional}(x|antecedent(e), e) \times \prod_{e' \in antecedent(e)} P(e'|E)$

Otherwise, the probability of x given E is equal to the prior probability of x.

4.1.2 Probability of Action Execution

If an action A is observed executed, the probability of successful execution of A given E is 1.0, that is, $P(A|E) = 1.0$. If A is observed executing, $P(A|E)$ equals to its execution probability. Otherwise, the probability of successful execution of A given E is computed by multiplying the execution probability of A and the probabilities of each action precondition.

$$P(A|E) = P_{execution}(A|precondition(A)) \times \prod_{e \in precondition(A)} P(e|E)$$

4.1.3 Outcome Probability and Expected Utility of Actions

The probability changes of action execution impact the calculation of outcome probabilities and expected utilities of actions. Let O_A be the outcome set of action A,

and outcome $o_i \in O_A$. The probability of o_i given E is computed by multiplying the probability of executing A and the effect probability of o_i.

$$P_{action}(o_i|E) = P(A|E) \times P_{effect}(o_i|A)$$

4.1.4 Outcome Probability of Plans and Expected Plan Utility

Let O_P be the outcome set of plan P, and outcome $o_j \in O_P$. Let $\{A_1, ..., A_k\}$ be the partially ordered action set in P leading to o_j, where o_j is an action effect of A_k. The probability of o_j given E is computed by multiplying the probabilities of executing each action leading to o_j and the effect probability of o_j (Note that $P(A_i|E)$ is computed according to the partial order of A_i in P).

$$P_{plan}(o_j|E) = (\prod_{i=1,...,k} P(A_i|E)) \times P_{effect}(o_j|A_k)$$

The expected utility of P given E is computed using the utilities of each plan outcome in P and the probabilities with which each outcome occurs.

$$EU(P|E) = \sum_{o_j \in O_P} (P_{plan}(o_j|E) \times Utility(o_j))$$

4.2 The Multiple Plan Recognition Approach

In real-world situations, a group often engages in complex behavior and may pursue multiple plans/goals simultaneously. These complex group behaviors can hardly be captured by conventional plan inference approaches as they often assume that an agent only commits to one plan at a time. To achieve complex group behavior forecasting, we propose a novel multiple plan recognition approach in this section.

From a computational perspective, multiple plan recognition poses great challenge. For observed group actions, the hypothesis space of multiple plan recognition turns out to be rather huge and the computational complexity is extremely high. To address the challenge, our approach consider using searching techniques to efficiently find the best explanation. Intuitively, if we view the actions of plans as vertexes and links between actions as edges, we can convert plans into a graph. We intend to map multiple plan recognition into a graph theory problem and adopt graph search techniques to find a near best explanation.

Below we first give the problem definition and represent the hypothesis space of input observations as a directed graph (i.e. explanation graph). We then describe how to compute the probability of an explanation. We finally present an algorithm for finding the best explanation.

4.2.1 Problem Definition

Our approach adopts the hierarchical plan representation. A hierarchical plan library, PL, is a set of hierarchical partial plans. Each partial plan is composed of abstract and/or primitive actions. The actions in the partial plan form a tree-like structure, where an abstract action corresponds to an AND node (i.e., there exists only one way of decomposition) or an OR node (i.e., multiple ways of decomposition exist) in the plan structure. At an AND node, each child is decomposed from its parent with decomposition probability 1. At an OR node, each child is a specialization of its parent. The sum of specialization probabilities of each child is 1.

For each observation, it can be either a primitive action or a state. Given a plan library, an explanation, SE_i, for a single observation, O_i, is an action sequence starting from a top-level goal, G_0, to O_i: $SE_i = \{G_0, SG_1, SG_2, ..., SG_m, O_i\}$, where SG_1, SG_2,..., and SG_m are a set of abstract actions. There can be multiple explanations for a single observation. An explanation, E_j, for an observation set $O = \{O_1, O_2, ..., O_n\}$ is defined as $E_j = SE_1 \cup SE_2 \cup ... \cup SE_n$, where SE_i is an explanation for the single observation O_i. If $SE_1 = SE_2 = ... = SE_n$, the explanation E_j corresponds to a single plan. Otherwise it corresponds to multiple plans.

We define the multiple plan recognition problem as follows. Given a hierarchical plan library PL and an observation set O, the task of multiple plan recognition is to find the most likely explanation (best explanation), E_{max}, from the explanation set, E, for O

$$E_{max} = \underset{E_i \in E}{argmax}\ P(E_i|O)$$

4.2.2 Constructing Explanation Graph

Given a set of observed actions and a plan library, the procedure of constructing explanation graph is as follows:

Step 1. Construct an explanation graph EG which is an empty graph and add all observations to the bottom level of EG.

Step 2. Expand the parents of each observation following a breadth-first strategy and add these parents to EG. Decomposition/specialization links between actions are treated as directed edges and are also added to EG. The direction of the edges denotes decomposition or specialization relation. A decomposition/specialization probability is attached to each edge. Duplicate actions and edges are combined during expansion.

Step 3. Apply this breadth-first expansion strategy on EG until all the actions in EG are expanded.

Step 4. Then add a dummy node on the top of the graph and connect the dummy node to all the top-level goals. The edges from the dummy node to top-level goals are associated with the prior probabilities of each top-level goal.

Now our approach constructs an explanation graph which contains all the possible explanations for the given observed actions.

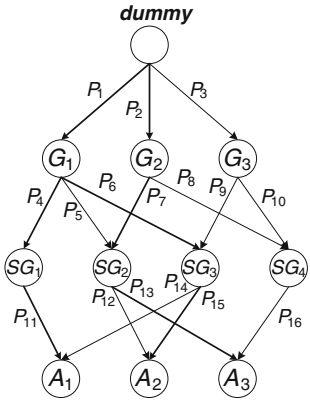

Fig. 2 Illustration of an explanation graph

Figure 2 is an explanation graph for the observed actions A_1, A_2 and A_3. It is a directed graph with bold lines denoting an explanation. The symbols P_1, P_2,..., P_{16} denote the decomposition/specialization probabilities associated with edges. An explanation corresponds to a connected sub-graph in the explanation graph containing the dummy node, top-level goals, sub-goals, and all the observations. In the explanation, the nodes with input degree 1 correspond to observations.

Here we define an explanation for an observation set as a tree in an explanation graph, in which the root is a dummy node and the leaves are all the observations. The tree exactly specifies an explanation for each observation.

4.2.3 Computing the Probability of an Explanation

Let $O_{1:i} = \{O_1, O_2, ..., O_i\}$ be observed actions, the probability of an explanation E_j is computed as

$$P(E_j|O_{1:i}) = P(E_j, O_{1:i})|P(O_{1:i}) = P(O_{1:i}|E_j)P(E_j)|P(O_{1:i})$$

As $1/P(O_{1:i})$ is a constant for each explanation, we denote it as K. $P(O_{1:i}|E_j)$ is the probability that $O_{1:i}$ occurs given the explanation E_j and is 1 for all hypotheses. $P(E_j)$ is the prior probability of TH explanation, i.e., the probability of entire tree of the explanation graph. For explanation E_j, let $G_{1:m} = \{G_1, ..., G_m\}$ be top-level goals and $SG_{1:n} = \{SG_1, ..., SG_n\}$ be sub-goals. We denote the vertex set of the tree E_j as $V = dummy \cup G_{1:m} \cup SG_{1:n} \cup O_{1:i}$. Let $E = \{e_1 = dummy \rightarrow G_1, ..., e_s = SG_x \rightarrow SG_y, ..., e_t = SG_z \rightarrow O_i\}$ be the set of edges of E_j, where $1 \leq x, y, z \leq n$ and $1 \leq s \leq t$. Here we assume the decomposition of each action is directly influenced by its parent node. The prior probability of the explanation E_j is

$$\begin{aligned} P(E_j) &= P(V|E) \\ &= P(O_i, e_t|V/O_i, E/e_t) * P(V/O_i, E/e_t) \\ &= P(e_t) * P(V/O_i, E/e_t) \\ &= ... = P(dummy) * \prod_{edge \in E} P(edge) \end{aligned}$$

$P(O_i, e_t|V/O_i, E/e_t)$ is the conditional probability that the decomposition rule, e_t, activates and O_i is decomposed given the tree $(V/O_i, E/e_t)$. This is equal to $P(e_t = SG_z \rightarrow O_i)$ according to our decomposition assumption. In addition, $P(edge)$ is the probability of the edge in the explanation and $P(dummy)$ is the prior probability that an observed agent will pursue goals. This is constant for each explanation.

4.2.4 Finding the Best Explanation

Now the problem of finding the best explanation can be formulated as

$$\begin{aligned} E_{max} &= argmax\, P(O_{1:i}|E_j)P(E_j)|P(O_{1:i}) \\ &= \underset{E_j \in E}{argmax} \prod_{edge_i \in E_j} P(edge_i) \\ &= \underset{E_j \in E}{argmax} \sum_{edge_i \in E_j} ln(P(edge_i)) \\ &= \underset{E_j \in E}{argmax} \sum_{edge_i \in E_j} ln(P(\frac{1}{edge_i})) \end{aligned}$$

where $P(edge_i)$ is the decomposition probability associated with $edge_i$. We denote $ln(1/P(edge_i))$ as the weight of $edge_i$. As $0 < P(edgei) < 1$, we get $ln(1/P(edge_i)) > 0$. For explanation graph EG, we attach the weight $ln(1/P(e))$ to each edge $e \in EG$ (where $P(e)$ is the probability on edge e) and then we can convert an explanation graph to a directed weighted graph. Now the problem of finding the most likely explanation is reformulated as finding a minimum weighted tree in the explanation graph with the dummy node as the root and observations as leaf nodes.

Finding a minimum weight tree in a directed graph is known as the directed Steiner tree problem in graph theory [1, 15]. It is defined as follows: given a directed graph, $G = (V, E)$, with weights, $w(w0)$, on the edges, a set of terminals, $S \subseteq V$, and a root vertex, r, find a minimum weight tree, T, rooted at r, such that all the vertices in S are included in T. A number of algorithms have been developed to solve this problem. In our approach, we employ an approximation algorithm proposed by Charikar et al. [1].

4.3 Computational Experiments

4.3.1 Experimental Study 1

We still choose $Al-Qaeda$ as the representative realistic group for our study. Among the official investigation reports, 13 real attacks perpetrated by $Al-Qaeda$ have relatively complete descriptions. Based on our automatically generated plans, intelligence analyst helped choose 13 plans that match the reported real attacks. These plans form the plan library for our experimental study. We randomly generate a set of evidence using the combination of actions and initial world states in the plan library, and collect 95 lines of evidence. Each line contains either two observations (constituting 49 % of the evidence set) or three observations (constituting 51 % of the evidence set).

Four human raters experienced in security informatics participate in the experiment. According to the plan library we construct, each rater examined the evidence set line by line and predicted the most likely plans based on each line of evidence. The test set is composed of each raters predictions together with the corresponding evidence, with inter-rater agreement ($Kappa$) 0.764. The prior state probabilities, action execution probabilities and effect probabilities used by our approach (less than 100 items in total) were assigned by intelligence analyst. The intelligence analyst also assigned prior and conditional probabilities for Bayesian reasoning. Mapping plans to Bayesian networks is based on the generic method provided in [5].

Table 4 shows the experimental results using our approach and Bayesian reasoning. We measure the agreement of the probabilistic plan inference approach and each rater using the $Kappa$ statistic. The average agreements between our approach and human raters are 0.664 (for two observations) and 0.773 (for three observations), which significantly outperform the average agreements between Bayesian reasoning and the raters. As $0.6 < k < 0.8$ indicates substantial agreement, the empirical results show good consistency between the predictions generated by our approach and those of human raters.

4.3.2 Experimental Study 2

We still choose $Al-Qaeda$ as a representative group. Based on our previous work [7], we automatically extract group actions and construct group plans from relevant open source News (e.g., $Times\ Online$ and $USATODAY$). Domain experts helped connect the hierarchical partial plans in the plan library. The plan library we use for this experiment includes 10 top-level goals and 35 primitive and abstract actions (we allow primitive/abstract actions to appear in multiple plans). Although large numbers of plans are computationally feasible by our approach, we prefer a relatively small and realistic plan library so that it is tractable by human raters in the experiment. Figure 3 illustrates the plan structure for a top-level goal in the plan library.

Table 4 Kappa agreements between algorithms and human raters

Rater	Plan inference						Bayesian reasoning					
	Two observations			Three observations			Two observations			Three observations		
	$P(A)$	$P(E)$	K	$P(A)$	$P(E)$	K	$P(A)$	$P(E)$	K	$P(A)$	$P(E)$	K
1	0.826	0.108	0.805	0.878	0.106	0.864	0.436	0.078	0.388	0.469	0.106	0.406
2	0.696	0.090	0.666	0.837	0.105	0.818	0.435	0.086	0.382	0.469	0.107	0.405
3	0.609	0.096	0.567	0.776	0.097	0.752	0.435	0.098	0.374	0.490	0.102	0.432
4	0.652	0.088	0.618	0.694	0.101	0.660	0.370	0.089	0.308	0.388	0.092	0.326
Avg			0.664			0.773			0.363			0.392

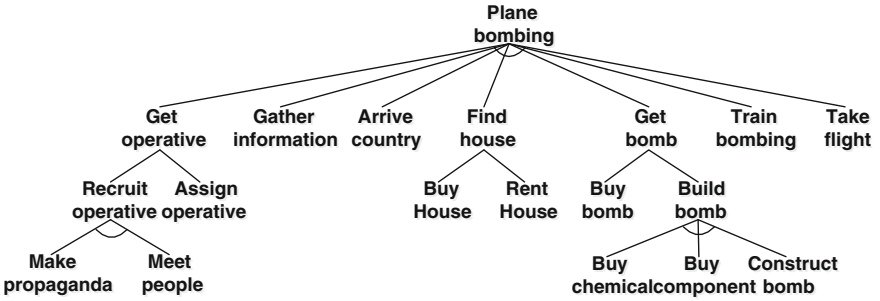

Fig. 3 Plan structure for a top-level goal in the plan library

Table 5 Agreements between MPR algorithm and human raters for various observations (obs.)

Rater	$P(A)$			K		
	2 obs.	3 obs.	4 obs.	2 obs.	3 obs.	4 obs.
1	0.967	0.932	0.899	0.961	0.908	0.874
2	0.978	0.913	0.869	0.974	0.887	0.837
3	0.956	0.917	0.859	0.949	0.891	0.830
4	0.906	0.924	0.902	0.856	0.899	0.877
5	0.838	0.895	0.878	0.765	0.866	0.847

We randomly generate a number of observation sets using the combination of primitive actions in the plan library. We collect 90 lines of observation sets in total, each line corresponding to one observation set. Among them, 30 observation sets contain two observations each, 30 contain three observations each and 30 contain four observations each. Five human raters who have at least 3 years experience in the security informatics domain participated in the experiment. Based on the constructed plan library, each rater examined the observation sets one by one and predicted the most likely plans (single plan or multiple plans) based on each observation set. The test set is composed of each raters predictions together with corresponding observations (with inter-rater agreement of 0.88).

Table 5 shows the experimental results between the multiple plan recognition approach and each human rater. We measure the agreement of the results generated by our approach and those of the raters for two, three and four observations using precision, $P(A)$, and Kappa statistics, K. The agreements between our approach and human raters for two observations, three observations, and four observations are all above 0.8, thus the empirical results show good consistency between the predictions generated by our MPR approach and those of human raters.

5 Conclusion

In this chapter, we propose an ACP-based approach for behavioral modeling, analysis and prediction in security informatics. To facilitate behavioral modeling, we present a knowledge extraction approach to acquire behavioral knowledge from open-source intelligence. On the basis of behavioral modeling, we propose two plan inference approaches for group behavior forecasting. The first explicitly takes the observed agents preferences into consideration to infer the most likely plan of groups. The second employs a graph search algorithm to discover multiple intentions underlying complex group behavior. Experimental results to demonstrate the effectiveness of these computational methods we propose as well as the underlying ACP approach.

References

1. Charikar, M., Chekuri, C., yat Cheung, T., Dai, Z., Goel, A., Guha, S., Li, M.: Approximation algorithms for directed steiner problems. In: Proceedings of the Ninth Annual ACM-SIAM Symposium on Discrete Algorithms, pp. 192–200 (1998)
2. Chen, H., Wang, F.Y., Zeng, D.: Intelligence and security informatics for homeland security: information, communication, and transportation. IEEE Trans. Intell. Transp. Syst. **5**(4), 329–341 (2004)
3. Fikes, R.E., Nilsson, N.J.: Strips: a new approach to the application of theorem proving to problem solving. Artif. Intell. **2**(3), 189–208 (1971)
4. Girju, R.: Automatic detection of causal relations for question answering. In: Proceedings of the ACL 2003 Workshop on Multilingual Summarization and Question Answering, pp. 76–83 (2003)
5. Huber, M.J., Durfee, E.H., Wellman, M.P.: The automated mapping of plans for plan recognition. In: Proceedings of the Tenth International Conference on Uncertainty in Artificial Intelligence, pp. 344–351 (1994)
6. Khoo, C.S.G., Chan, S., Niu, Y.: Extracting causal knowledge from a medical database using graphical patterns. In: Proceedings of the 38th Annual Meeting on Association for Computational Linguistics, pp. 336–343 (2000)
7. Li, X., Mao, W., Zeng, D., Wang, F.Y.: Automatic construction of domain theory for attack planning. In: IEEE International Conference on Intelligence and Security Informatics, pp. 65–70 (2010)
8. Persing, I., Ng, V.: Semi-supervised cause identification from aviation safety reports. In: Proceedings of the Joint Conference of the 47th Annual Meeting on Association for Computational Linguistics, pp. 843–851 (2009)
9. Sil, A., Huang, F., Yates, A.: Extracting action and event semantics from web text. In: AAAI Fall Symposium on Common-Sense Knowledge, vol. 40 (2010)
10. Wang, F.Y.: Computational experiments for behavior analysis and decision evaluation of complex systems. Acta Simulata Systematica Sinica **5**, 008 (2004)
11. Wang, F.Y.: Social computing: concepts, contents, and methods. Int. J. Intell. Control Syst. **9**(2), 91–96 (2004)
12. Wang, F.Y.: A computational framework for decision analysis and support in isi: Artificial societies, computational experiments, and parallel systems. In: Intelligence and Security Informatics, pp. 183–184. Springer, Berlin (2006)
13. Wang, F.Y.: Parallel management systems: concepts and methods. J Complex Syst. Complex. Sci. **3**(2), 26–32 (2006)

14. Wang, F.Y.: Toward a paradigm shift in social computing: the acp approach. IEEE Intell. Syst. **22**(5), 65–67 (2007)
15. Zosin, L., Khuller, S.: On directed steiner trees. In: Proceedings of the Thirteenth Annual ACM-SIAM Symposium on Discrete Algorithms, pp. 59–63 (2002)