On Temporal Analysis of Timed Influence Nets Using Point Graphs

Sajjad Haider **Abbas K. Zaidi** **Alexander H. Levis**

System Architectures Lab
George Mason University, Fairfax, VA
{shaider1, szaidi2, alevis}@gmu.edu

Abstract

This paper demonstrates the use of Point Graphs (PG) and temporal logic for analyzing courses of action (COA) in a Timed Influence Net (TIN) that models a dynamic uncertain situation. The current practice in courses of action analysis looks at the impacts of actions on the likelihood of the desired effects over a period of time. The impact of time, however, is not studied explicitly. This paper presents an algorithm that generates a corresponding Point Graph for a Timed Influence Net. This graph-based knowledge representation and reasoning formalism is shown to help reveal temporal behavior of the modeled system. A temporal language is also shown to interact with the graphical representation. An analysis on the graph addresses user-defined 'what-if' scenarios for a better understanding of the temporal relationships between certain actions that may result in a desired effect at a particular time instant.

1. Introduction

Timed Influence Nets (TINs) have been used experimentally in the area of Effects-Based Operations (EBO) (Wagenhals and Levis 2002, Wagenhals et al. 2003). They are used as a decision aid for modeling and analyzing uncertainties involved in a complex dynamic situation. Furthermore, once a Timed Influence Net (TIN) is used to model a situation, it allows a system modeler to evaluate the performance of different courses of action in terms of their impacts on the likelihood of achieving some desired effect(s). The TIN formalism originated from a general class of probabilistic reasoning framework, known as Bayesian Networks (Pearl 1987, Jensen 2001, Neapolitan 2003). Bayesian Networks (BNs) were originally designed to capture uncertainty in static (time-independent) situations. During the last decade, much effort has been focused on integrating the notions of time and uncertainty into a single analytical framework. Most of this work can be classified into one of the following two categories: (i) adding constructs to the static BN formalism to capture temporal dependencies (Hanks et al. 1995, Santos and Young 1999) and (ii) developing algorithms that can compute the likelihood of variables of interest in a reasonable amount of time (Kjaerulff 1992, Boyen and Koller 1998). BNs with the additional temporal constructs are typically referred to as Time Sliced Bayesian Networks (TSBNs). Recently, it is shown (Haider and Zaidi 2004) that a transformation exists from a TIN to an equivalent TSBN. It should be noted that inference in a general class of BNs, even for static situations, is NP-Complete. A lot of reported work is, therefore, focused on developing efficient techniques for approximate inference that can be helpful for modeling real time situations.

The term temporal logic has been broadly used to cover all approaches for the representation of temporal information within a logical framework. A temporal logic can be defined as a language for encoding temporal knowledge about an application system and as a tool for reasoning about temporal relations among the system entities. (Galton 1999) Many schemes have been suggested to represent time in the AI literature for both a qualitative and a quantitative treatment of time. Zaidi (Zaidi 1999) presented a Point Interval Temporal Logic (PITL) based on Allen's ontology of time (Allen 1983). PITL incorporates both qualitative and quantitative temporal aspects associated with points and intervals in a system specification. The tool, TEMPER (Zaidi and Levis 2001), automates the inference mechanism of PITL. It takes input in PITL language, interprets it, and transforms the temporal statements into an equivalent graphical structure. The graph, called Point Graph (PG), not only implements the axiomatic system of PITL, but also helps verify system integrity before inference making. A temporal inference engine answers user-defined queries by exploring structural properties of the graph.

Few efforts have been made for integrating temporal logic with the Bayesian approaches or vice versa. The work of Santos and Young (1999) focuses on using Allen's interval logic for knowledge elicitation, while Burns and Morrison (2003) have proposed a template, based on Allen's interval logic, for structured temporal reasoning.

This paper explores the use of PITL for TIN models in analyzing the temporal impact of a certain course of actions on variables of interest. The inference mechanism of PITL is used to find, at a particular time instant, the source of a change in the likelihood of a variable of interest. The PITL inference engine achieves this task by analyzing the relationships that exist between actionable events and the variable of interest. The paper also demonstrates the use of PITL for performing what-if analysis on user-defined input/output scenarios. The analysis helps a system modeler in developing a better understating of the temporal relationships that must exist, at a particular time instant, between certain actions required to achieve a desired effect.

The rest of the paper is organized as follows: Sections 2 and 3 provide an introduction to TIN and PG,

respectively. Section 4 presents the algorithm that generates a Point Graph (PG) from a TIN. The resultant PG is shown to answer temporal queries and to perform what-if analyses. Finally, Section 5 presents the conclusion and proposes some future research directions.

2. Timed Influence Nets

The modeling of the causal relationships in TINs is accomplished by creating a series of cause and effect relationships between some desired effects and the set of actions that might impact their occurrence in the form of an acyclic graph. The actionable events in a TIN are drawn as root nodes (nodes without incoming edges). A desired effect, or an objective in which a decision maker is interested, is modeled as a leaf node (node without outgoing edges). Typically, the root nodes are drawn as rectangles while the non-root nodes are drawn as rounded rectangles. Figure 1 shows a partially specified TIN. Nodes B and E represent the actionable events (root nodes) while node C represents the objective node (leaf node). The directed edge with an arrowhead between two nodes shows the parent node promoting the chances of a child node being true, while the roundhead edge shows the parent node inhibiting the chances of a child node being true. The inscription associated with each arc shows the corresponding time delay it takes for a parent node to influence a child node. For instance, event B, in Figure 1, influences the occurrence of event A after 5 time units.

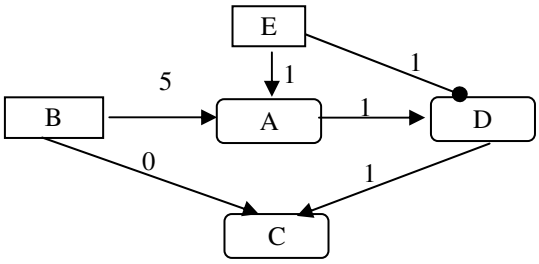

Figure 1: An Example Timed Influence Net (TIN)

The purpose of building a TIN is to evaluate and compare the performance of alternative courses of actions. The impact of a selected course of action on the desired effect is analyzed with the help of a *probability profile*. Consider the TIN shown in Figure 1. Suppose the following *input scenario* is decided: actions B and E are taken at times 1 and 7, respectively. Because of the propagation delay associated with each arc, the influences of these actions impact event C over a period of time. As a result, the probability of C changes at different time instants. A probability profile draws these probabilities against the corresponding time line. The probability profile of event C is shown in Figure 2.

The following items characterize a TIN:

1. A set of random variables that makes up the nodes of a TIN. All the variables in the TIN have binary states.
2. A set of directed links that connect pairs of nodes.
3. Each link has associated with it a pair of parameters that shows the causal strength of the link (usually denoted as g and h values).
4. Each non-root node has an associated baseline probability, while a prior probability is associated with each root node.
5. Each link has a corresponding delay d (where $d \geq 0$) that represents the communication delay.
6. Each node has a corresponding delay e (where $e \geq 0$) that represents the information processing delay.
7. A pair (p, t) for each root node, where p is a list of real numbers representing probability values. For each probability value, a corresponding time interval is defined in t. In general, (p, t) is defined as
   $([p_1, p_2, \ldots, p_n], [[t_{11}, t_{12}], [t_{21}, t_{22}], \ldots, [t_{n1}, t_{n2}]])$,
   where $t_{i1} < t_{i2}$ and $t_{ij} > 0 \; \forall$ i = 1, 2, …., n and j = 1, 2

The last item in the above list is referred to as input scenario, or sometimes (informally) as course of action. Formally, a TIN is described by the following definition.

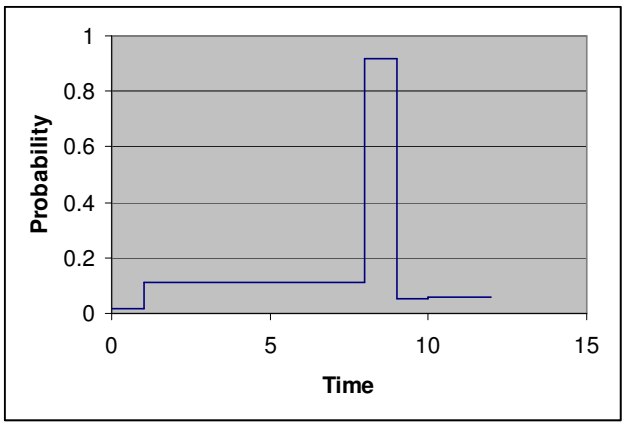

Figure 2: Probability Profile for Node C

Definition 1 Timed Influence Net (TIN)

A TIN is a tuple $(V, E, C, B, D_E, D_V, A)$ where

V: set of Nodes,

E: set of Edges,

C represents causal strengths:

$E \rightarrow \{ (h, g)$ such that $-1 < h, g < 1 \}$,

B represents Baseline / Prior probability: $V \rightarrow [0,1]$,

D_E represents Delays on Edges: $E \rightarrow Z^+$

(where Z^+ represent the set of positive integers),

D_V represents Delays on Nodes: $V \rightarrow Z^+$, and

A (input scenario) represents the probabilities associated with the state of actions and the time associated with them.

$A: R \rightarrow \{([p_1, p_2, \ldots, p_n],[[t_{11},t_{12}], [t_{21},t_{22}], \ldots,[t_{n1},t_{n2}]])$ such that $p_i = [0, 1]$, $t_{ij} \rightarrow Z^*$ and $t_{i1} \leq t_{i2}$, $\forall$ i = 1, 2, …., n and j = 1, 2 where $R \subset V \}$

(where Z^* represent the set of nonzero positive integers)

3. Point Graphs and Temporal Language

This section presents a graph formalism called Point Graph (PG) that is used to represent, both qualitatively and quantitatively, temporal information between points and intervals in a system. The intervals may represent activities and points instantaneous events. Formally, a Point Graph is defined as follows.

Definition 2 Point Graph (PG)

A Point Graph, PG (V, E_A, D, T) is a directed graph with:

V: Set of vertices with each node or vertex v ∈ V representing a point on the real number line. Two points pX and pY are represented as a composite point [pX;pY] if both are mapped to a single point on the line.

E_A: Union of two sets of edges: $E_A = E \cup E_{\leq}$, where

E (LT edges): Set of edges with each edge $e_{12} \in E$, between two vertices v1 and v2, also denoted as (v1, v2), representing a relation '<' between the two vertices—(v1 < v2);

$E_{\leq}$ (LE edges): Set of edges with each edge $e_{12} \in E_{\leq}$, between two vertices v1 and v2, also denoted as (v1, v2), representing a relation '≤' between the two vertices—(v1 ≤ v2).

D (Length): Edge-length function (possibly partial): $E \rightarrow \Re^+$

T (Stamp): Vertex-stamp function (possibly partial): $V \rightarrow \Re$

The following temporal language (Definition 3) can be used to describe temporal aspects/requirements of a system either already represented as a PG, or to be input to the PG representation.

Definition 3 Temporal Language

The lexicon consists of the following primitive symbols:

Points (Event): A point X is represented as [pX, pX] or simply [pX]. Several labels p1, p2, …, pn, representing a single point are represented as a composite point [p1;p2;…;pn].

Intervals: An interval X is represented as [sX, eX], where 'sX' and 'eX' are the two end points of the interval, denoting the 'start' and 'end' of the interval, s.t. sX < eX.

Point Relations: These are the relations that can exist between two points. The set of relations R_P is given as:

R_P = {Before, Equals, Procedes}

Functions: Interval length function that assigns a non-zero positive integer to a system interval, e.g.,

Length X = d, where X = [sX, eX], d ∈ $\Re^+$

The stamp function assigns an integer number to a system point, e.g., Stamp p1 = t, t ∈ $\Re$

A temporal statement in this language either takes the form of a function statement, or 'X Ri Y' where X and Y are points and Ri ∈ R_P.

The temporal relation 'Before' corresponds to the '<' edge in the PG definition. Similarly, the relation 'Precedes' corresponds to a '≤' edge, and the temporal relation 'Equals' results in a composite point (vertex) in the PG representation. The two functions for the quantitative information directly map to the identically named functions in the PG definition. Figure 3 shows the correspondence.

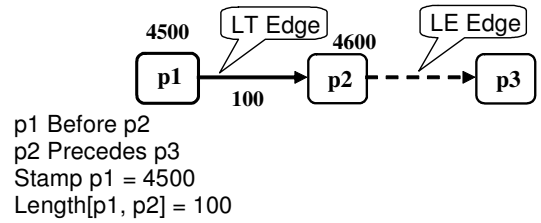

Figure 3: Point Graph and Corresponding Temporal Statements

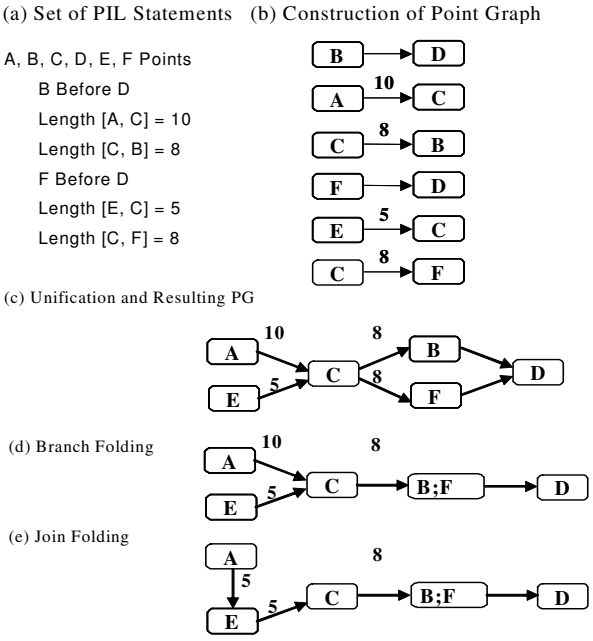

Figure 4: Steps in PG Construction

A set of PIL statements can now be represented as a set of PGs where each PG corresponds to a single statement in the temporal system. A consolidated PG for the entire temporal system can be constructed by *unifying* and *folding* the individual PGs (Zaidi and Wagenhals 2004). The unification looks at the nodes of a set of PGs and merges the nodes with identical node labels or the ones with equality relation between them. The folding process, on the other hand, looks at the quantitative information on nodes, and edges, of a PG and folds the edges based on the available information. The process establishes new relations among system points, inferred through the quantitative analysis of the known relations specified by interval lengths and stamps. Figure 4 illustrates the process of constructing a PG for a set of PIL statements with the help of an example.

The presence of inconsistent information in a temporal system results in an *erroneous* PG, which may result in erroneous inferences and/or analyses preformed on the PG. It is, therefore, imperative to identify and correct the inconsistent cases prior to any analysis. Theorem 1 characterizes the temporal inconsistencies in a Point Graph.

Theorem 1 Temporal Inconsistency (Zaidi and Wagenhals 2005)

A Point Graph contains inconsistent information iff

(a) for a point (vertex) p1, the system calculates two different stamps; *or*
(b) for some points pi and pj, 'p1 < p2', the system can determine two different lengths for the interval [pi, pj].

A polynomial-time path-consistency algorithm is presented in Zaidi and Wagenhals (2004) for identifying the erroneous cases in the PG representation.

4. Temporal Analyses of Timed Influence Nets

This section explains how the integration of TIN and PG formalisms adds new suit of techniques for analyzing complex uncertain situations. The proposed techniques aid a system modeler in gaining a better insight of the impact of a selected course of action on desired effect(s). The PG representation of a corresponding TIN answers queries regarding certain temporal characteristics of an effect's probability profile. The methodology is explained in Section 4.2. Furthermore, the PG aids a system modeler by explaining what needs to be done for achieving a certain effect at a specific time instant. If the requirements for achieving effects at certain time instants are not temporally consistent, then the PG helps in understanding the reasons for inconsistencies. This approach is presented in Section 4.3. Both types of temporal analyses (Sections 4.2 and 4.3) assume that a corresponding PG has been constructed from a TIN. The construction of a PG is the subject of the following sub-section.

4.1 Creating a Point Graph from a Timed Influence Net

The steps involved in generating a PG from a corresponding TIN are presented in Table 1. The following is the illustration of the conversion approach with the help of the sample TIN of Figure 1. For the example case: R = {B, E} and F = {C}.

Step 1: In this step, all the paths from the root nodes to the leaf nodes are determined. For instance, there are four distinct paths in the TIN of Figure 1:

(i) B – A – D – C (ii) B – C
(iii) E – A – D – C (iv) E – D – C

Step 2: This step transforms each path into a corresponding PG. A unique subscript is added to all the nodes in the path during this transformation. Thus, path B-A-D becomes B1-A1-D1; path B becomes A2-D2-C2; and so on. The delays attached with each arc in the TIN are transformed to length expressions in PG. The corresponding PIL statements are given below:

Length[B1,A1] = 5 Length[A1,D1] = 1
Length[D1,C1] = 1 Length[E3,A3] = 1
Length[A3,D3] = 1 Length[D3,C3] = 1
Length[E4,D4] = 1 Length[D4,C4] = 1
B2 Equals C2

It should be noted that the time delay between B and C is 0 time unit. Thus, both events represent the same temporal point. The PGs obtained as a result of Step 2 are shown in Figure 5.

Table 1: A PG Construction from a TIN

Given a TIN
R: Set of Root Nodes (Actionable Events)
F: Set of Leaf Nodes (Desired Effects)

1. For each r ∈ R find all the paths leading to a f ∈ F. Apply this step for all f ∈ F.
2. Add a unique subscript to each node in an individual path obtained in Step 1.
3. Represent each path as a PG where a node in the path becomes a vertex and a delay d (d >0) on an arc between two vertices v1, v2 becomes Length(v1,v2)=d in the PG.
4. For each set of vertices in PG that represent a root node in TIN, add a temporal equality constraint 'Equal' among its elements.

The following step is executed once an input scenario is provided.

5. Based on the input scenario, assign time stamps to vertices representing root nodes .
6. Construct an aggregate PG using temporal statements provided in Steps 3-5 after applying the unification and folding operations.

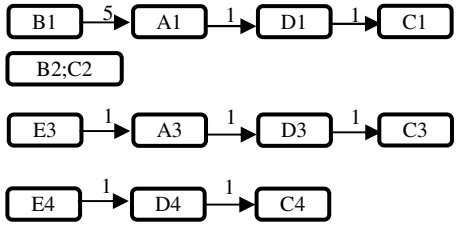

Figure 5: PGs Corresponding to Paths in the TIN

Step 3: This step adds temporal relation 'Equals' between points that represent a particular root node in the corresponding TIN. Since R = {B, E}, the following information is provided to the PIL engine:

B1 Equals B2
E3 Equals E4

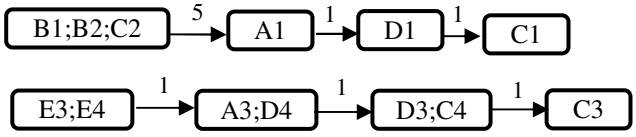

Figure 6: Branch Folded PGs

Based on the given information, unification and folding operations are applied on the PGs of Figure 5. The resulting PGs are shown in Figure 6.

Step 4: Let an input scenario be given: suppose B occurs at time 1, while E occurs at 7. This information is added to the set of temporal statements described in the earlier steps. Thus, the following statements are added:
Stamp[B1] = 1
Stamp[E3] = 7
This last set of information results in further unification and folding of the PGs of Figure 6. The final consolidated PG is shown in Figure 7.

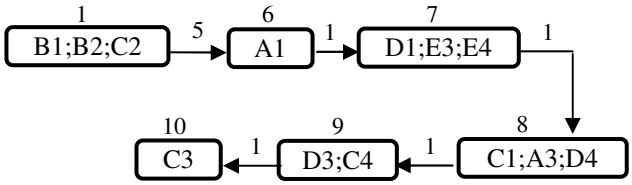

Figure 7: PG with Input Scenario

4.2 Temporal Queries

Once a PG is obtained from a TIN, it can then be used to explain certain temporal characteristics of a probability profile. Consider the profile shown in Figure 2. Suppose a system modeler is interested in knowing what causes a change in the probability of event C at time 8. The algorithm that answer this and similar queries is presented in Table 2 and is explained below, with the help of an example.

Table 2: Answering Temporal Queries using a PG

Given a PG, a TIN, v: variable of interest,
t: time of interest, C: list of Causes
1. Initialize C to null.
2. Determine the subscripts of v at time t. Let S = [s1, s2, …, sn] be the list of subscripts.
3. For each element s in S:
   (i) Starting from the root of the PG, search the PG until the first variable with the subscript s is identified. Let x be such a variable.
   (ii) Let m be the time stamp associated with x.
   (ii) Add (x, m) to C.
4. Report the list C.

The algorithm first identifies the subscript(s) of the variable of interest for a given time stamp. For instance, the subscript of C at time 8 is '1'. Starting from the root of the PG, the algorithm, in the next step, searches the graph for the subscript until it finds the first variable (or set of variables if they share the subscript) that matches the subscript. For instance, in the PG of Figure 6, the first element having subscript '1' is B. The time stamp associated with B1 is 1. Thus, a change in the profile of C at time 8 is because of action B occurring at time 1. If more than one action cause a change in the probability of C at time 8, then all of them are reported along with the time of their occurrences. Furthermore, if multiple paths exist between an action node and a desired effect, in a TIN, the algorithm of Table 2 can be used to identify the path through which the action has impacted the effect node. For the example under consideration, the path through which B impacted C at time 8 is: B – A – D – C.

4.3 What-If Analysis

The PG obtained in Section 4.1 can also aid in performing what-if analyses. Suppose after observing the probability profile of Figure 2, the system modeler is interested in knowing what needs to be done, in order to combine the impacts that reach C at times 8 and 9. The algorithm that accomplishes this task is presented in Table 3.

Table 3: What-If Analysis Using a PG

Given a TIN, a PG G1
R: Set of Root Nodes (Actionable Events)
F: Set of Leaf Nodes (Desired Effects)
V: List of variables of interest
T: List of times of interest
S: List of variables with equal time stamp
1. For each $r \in R$ find all the paths leading to a $f \in F$. Apply this step for all $f \in F$.
2. Add a unique subscript to each node in an individual path obtained in Step 1.
3. Represent each path as a PG where a node in the path becomes a vertex and a delay d (d >0) on an arc between two vertices v1, v2 becomes Length(v1,v2)=d in the PG.
4. For each set of vertices in PG that represent a root node in TIN, add a temporal equality constraint 'Equal' among its elements.
5. For each element v in V,
   (i) Find its subscript at the corresponding time t ($t \in T$) in G1. Let s be the subscript.
   (ii) Add variable v with subscript s to S.
6. Add temporal equality constraint 'Equals' among the elements of list S.
7. Construct an aggregate PG G2 using temporal statements provided in Steps 3-6.

The algorithm assumes that a PG based on an input scenario has already been constructed (Figure 7 in the current context). As stated above, the modeler is interested in combining the impacts that reach node C at time 8 and 9; thus, list V has elements [C, C], while list T has elements [8, 9]. The first four steps of the algorithm are the same as the algorithm given in Table 1, and therefore, are

not explained below. In Step 5, the subscripts of elements in V at corresponding times, described in T, are determined. Thus, C at time 8 has subscript '1', while C at time 9 has subscript '4'. As a result of this step, list S consists of [C1, C4]. Step 6 adds the following statement: C1 Equals C4

The PG resulted from the temporal statements provided in Steps 2-6 is shown in Figure 8. The PG can now be used to answer the query the system modeler is interested in. For instance, the length between points representing events B and E is 5 time units. Therefore, to combine the impacts that affect node C at time 8 and 9, B must be executed 5 time units before E.

The what-if analysis not only identifies the temporal relationships that should exist between two actionable events for achieving a desired impact at a certain time instant, but it also tells a system modeler if the given set of requirements are temporally inconsistent.

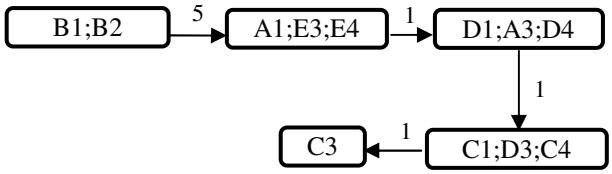

Figure 8: PG Used for What-If Analysis

5. Conclusions

This paper presents a suite of techniques that further enhances the capabilities of TIN based formalism as a modeling and analysis tool for dynamic uncertain situations. The PG based suite of techniques, when applied on a TIN, aids a system modeler in having a better understanding of the behavior of a desired effect over a period of time. The proposed technique also aids in performing 'what-if' analysis. The outcome of this analysis tells a system modeler what needs to be done for achieving a desired effect at a particular time instant, given that there are no temporal anomalies among the set of requirements provided by the system modeler. If the set of requirements are inconsistent, the PG structure would explain the reasons that resulted in inconsistencies.

Acknowledgements

This research was sponsored by the Air Force Research Laboratory, Information Directorate under Grant No. F30602-01-C-0065 and Air Force Office of Scientific Research under contract numbers F49620-01-1-0008. The authors are grateful to the anonymous reviewers for their useful suggestions. The contribution of Mr. Mashhood Ishaque of the GMU System Architectures Laboratory in the development of PIL Engine (the software implementation of PG and temporal logic) is gratefully acknowledged.

References

Allen, J. F. 1983. Maintaining knowledge about temporal intervals. *Communication ACM* 26:832–843.

Boyen, X. and Koller, D. 1998. Tractable inference for complex stochastic processes. *In Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence*.

Burns, B. and Morrison, C. T. 2003. Temporal Abstraction in Bayesian Networks. *AAAI Spring Symposium*, Palo Alto, Calif.

Galton, A. *1999. Temporal Logic: The Stanford Encyclopedia of Philosophy*. Edward N. Zalta (ed.).

Haider, S. and Zaidi, A. K. 2004. Transforming Timed Influence Nets into Time Sliced Bayesian Networks. *In Proceedings of Command and Control Research and Technology Symposium*, San Diego, Calif.

Hanks, S., Madigan, D., and Gavrin, J. 1995. Probabilistic Temporal Reasoning with Endogenous Change. *In Proceedings of the Eleventh Conference on Uncertainty in Artificial Intelligence*.

Jensen, F. V. 2001. *Bayesian Networks and Decision Graphs* Springer-Veralg.

Kjaerulff, U. 1992. A Computational Scheme for Reasoning in Dynamic Probabilistic Networks. *In Proceedings of the Eighth Conference on Uncertainty in Artificial Intelligence*.

Neapolitan, R. E. 2003. *Learning Bayesian Networks*. Prentice Hall.

Pearl, J. 1987. *Probabilistic Reasoning in Intelligent Systems: Network of Plausible Inference*. Morgan Kaufmann.

Santos Jr. E. and Young, J. D. 1999. Probabilistic Temporal Network: A Unified Framework for Reasoning with Time and Uncertainty. *International Journal of Approximate Reasoning* 2.

Wagenhals, L. W. and Levis, A. H. 2002. Modeling Support of Effect-Based Operations in War Games. *In Proceedings of the Command and Control Research and Technology Symposium*.

Wagenhals, L. W., Levis, A. H., McCrabb, M. B. 2003. Effects-Based Operations: A Historical Perspective of a Way Ahead. *In Proceedings of the Eighth International Command and Control Research and Technology Symposium*, Washington DC.

Zaidi, A. K. 1999. On Temporal Logic Programming Using Petri Nets. *IEEE Transactions on Systems, Man and Cybernetics, Part A* 29(3): 245-254.

Zaidi, A.K. and Levis, A. H. 2001. TEMPER: A Temporal Programmer for Time-sensitive Control of Discrete-event Systems. *IEEE Transaction on Systems, Man, and Cybernetics* 31(6): 485-496.

Zaidi, A. K. and Wagenhals, Lee W. 2005 (n Press). Planning Temporal Events Using Point Interval Logic. *Special Issue of Mathematical and Computer Modeling*.