Real-time Decision Making For Shipboard Damage Control

Vadim Bulitko
Department of Computing Science
University of Alberta
Edmonton, Alberta T6G 2H1, CANADA
bulitko@ualberta.ca

David Wilkins
Beckman Institute
University of Illinois at Urbana-Champaign
Urbana, IL 61801, USA
dcw@uiuc.edu

Abstract
This paper presents a formalism called Time Interval Petri Nets (TIPNs) and shows their efficacy for model-based envisionment of real-time concurrent interacting processes with temporal constraints. The TIPN formalism is developed by extending Petri Nets to Time Interval Petri Nets. The process of TIPN construction from a state-process domain model is described. Experimental results in the domain of ship crisis decision-making show five orders of magnitude speed up over a first-principals numerical physical system simulator in a complex domain. An advantage of TIPNs is a graphical network representation that facilitates model construction, refinement, and comprehension.

Keywords: Limited Rationality, Real-time Decision Making, Petri Nets, Intelligent Systems, Knowledge Acquisition, Temporal Reasoning, Envisionment-based Control Policies.

1 Introduction
Decision-making systems in artificial intelligence can often make better decisions when it is possible to envision the consequences of alternative actions prior to making a decision (Russell & Wefald, 1991b,a; Korf, 1990). Physical system and behavioral system simulators are one approach to the achievement of envisionment (Shou et al., 2001; Heath, 1996; NIST, 2002). A limitation of this approach in complex domains arises in real-time decision making, when the time available for simulation is very limited and there is a high-cost in not making a quick decision (Bulitko & Wilkins, 1999). This paper shows that Petri Nets (Peterson, 1981; Murata, 1989; Donatelli & Kleijn, 1999) are suitable for envisionment for decision-making domains where the modelling of real-time concurrent processes plays a central role in the reasoning.

This paper is organized as follows. In section 1, the envisionment task is motivated using the domain of ship damage control, a classic example of a domain where there is a need for real-time decision making involving concurrent processes. In section 3, the knowledge representation and inference mechanism of TIPNs is presented within the context of a fire spread example that extends the formalism of Place-Transition Petri Nets (Peterson, 1981) to Time Interval Petri Nets. Section 4 describes the empirical evaluation, and covers the degree of speedup, scalability, and accuracy. Section 5 highlights methods of TIPN model construction. Section 6 covers related research while section 7 gives a summary and directions for future research.

2 Real-Time Decision Making and Ship Damage Control
The proof-of-concept domain used in this study is ship damage control (Wilkins et al., 2001). The decision-making goal is to manage crises to a ship that involve fire, smoke, flooding, pipe ruptures, hull ruptures, stability, and electrical and mechanical system deactivation. Toward this end, the main decision-maker, called a damage control assistant, must determine the nature and extent of the crisis from ship sensors and human investigators, and must then address the crisis using a limited number of automated systems and human repair parties.

The decision-making task is very time critical because the more time is spent, for example, in determining the exact location and size of a fire, the larger the fire will usually become. The decision-making involves reasoning under uncertainty, since current sensors and human investigator reports are known to be often unreliable during a major crisis.

The decision-making involves dealing with resource tradeoffs, since the resources such as the number of investigator and repair teams, and suppression methods such as water hoses are limited. Finally, there are interactions between events, such as determining whether to use scarce electricity to operate fire pumps needed to produce water pressure for fires, or for air conditioners that cool sensitive equipment.

An envisionment system can provide answers to these key questions. In the subsequent sections we will present an envisionment approach allowing for speed ups of five orders of magnitude over the existing numerical physical simulators and behavioral “intelligent agent” symbolic simulators for this domain.

How much speedup is desired? In the domain of ship damage control, at a given point in time there are usually about 50 alternative decisions for which it would be helpful to envision 20 minutes into the future. If the goal is to do the total envisionment in 5 seconds, this would require an envisionment system that can operate 50 × 20 × 60/5 = 12,000 times faster than real-time. In the experimental evaluation section, it will be seen that the implemented TIPN system is able to achieve this speedup even for complex crises. In other domains it is easy to imagine it being advantageous...
3 Extending Petri Nets to Time Interval Petri Nets

The knowledge representation and inference methods of Time Interval Petri Nets (TIPNs) will be introduced within the context of a simple fire spread example. The process will be modelled using the knowledge representation and inference of traditional Petri Nets and then this will be extended so as to allow handling the complexities of the fire spread example.

In order to model the process of fire spread with traditional modelling tools one needs to set up a numerical simulator to compute combustible material distribution, gas zones, plumes, heat transfer, wall temperatures, ignition properties, fire suppressant effects, and so forth (Shou et al., 2001). Since fire fighting personnel are involved in the process, another behavioral simulator is needed to model personnel travel time, human fatigue, communication mistakes, injuries, oxygen-breathing apparatus operation, and many other non-trivial aspects. The inherent complexity of these types of simulations prevent multiple alternatives extending, say 20 minutes into the future, from being simulated in a matter of seconds, hence the motivation for the TIPN Petri Net approach. As with all abstractions of first-principles models, the TIPN approach is not as accurate, but it provides sufficient accuracy in a limited amount of time to support the decision-making.

The process of constructing a TIPN begins with a model of the qualitative-type patterns augmented with temporal information that underlie the phenomena in question. Using this approach, the complex processes outlined in the previous paragraph are abstracted into the following simple domain dependency:

"for any compartment X if X is hot and the fire boundaries around X are not set then the fire will spread to a neighbor compartment Y in about 3 to 4 minutes."

In the first order predicate logic (FOPC) it can be expressed as follows (here and below we use the Prolog notation):

\[
\text{ignited}(Y,T_{\text{new}}) :- \\
\text{hot}(X,T_{\text{old}}), \\
\text{not fire_boundaries}(X,T_{\text{old}}), \\
\text{neighbor}(X,Y), \\
\text{delay}(T_{\text{new}},T_{\text{old}},3,4).
\]

We will introduce TIPNs by starting out with a simplified dependency expressible in propositional logic and moving to FOPC. This gradual upgrade will be paralleled by the transition from classical Petri Nets, also known as place-transition or P/T nets (Peterson, 1981), to Time Interval Petri Nets.

3.1 Place Transition (P/T) Petri Nets

Figure 1 presents the first step in this process. We start out with no variables and time delays. At this stage we are expressing merely that "if it is hot and no fire boundaries are set then it becomes ignited".

The propositional logic sentence \(\text{ignited} :- \text{hot}, \text{no_fbs}\). relates to the Petri Net as follows. Each large circle (called “place”) corresponds to an atomic sentence. If a small black dot (called “token”) is present inside the circle then the corresponding atomic sentence is considered to hold. The horizontal bar (called a “transition”) represents the implication and the arrows (called “arcs”) represent the preconditions and post effects. Once all preconditions are met (i.e., there are tokens in all relevant places) the transition “fires”. The process of firing retracts tokens from all enabling places (i.e., preconditions) and deposits tokens to the output place (i.e., the effect place). Thus, the tokens can be said to flow from the enabling places to the output place.

It is worth noting that unlike in Bayesian Nets, where arcs represent conditional dependencies, and in Artificial Neural Nets, where arcs indicate signal transmission, arcs in Petri Nets indicate event preconditions and don’t carry numeric values. However, in all of these formalisms current state of the modelled system is represented by the current state of the nodes.

3.2 Colored Petri Nets

There are several aspects of the fire spread example that the representation described in the previous section cannot model. One such aspect is an inability to model explicit context. This section shows how this limitation is overcome by allowing the introduction of compartment variables and time stamps. The addition of these variables to our propositional statement makes it into the first-order predicate calculus. In our evolving example, the variable \(T\) will express time while the variable \(X\) will take its values over the set of compartments. Our sentence becomes \(\text{ignited}(X,T) :- \text{hot}(X,T), \text{no_fbs}(X,T)\).

Correspondingly, the Petri Net model is upgraded with so-called “colors” (Jensen, 1997). Colors are arbitrary

*Setting fire boundaries on a compartment involves cooling down the walls of the compartment to prevent the fire spread.
data tags attached to the tokens. The data includes compartment data (e.g., 3–370–0–E) and time stamps (e.g., [2:32, 3:14]). Time stamps are represented with intervals accounting for the temporal uncertainty of the predicate’s becoming true. The extended representation is illustrated in Figure 2. By extending the Petri Net in such a way, we gain the ability to use a single network for modelling many different compartments, similar to using a single algorithm to compute square root of many different numbers.

3.3 Time Interval Petri Nets

The model described in the previous section is able to handle explicit context but cannot model temporal delays, which is vital in domains of time-critical decision making. In FOPC this can be addressed by adding an extra predicate delay(TimeOld, TimeNew, MinimumDelay, MaximumDelay). Within the TIPN formalism this is handled by the addition of a delay interval to the transition as shown in Figure 3. As the tokens move through the transition their time stamps get appropriately adjusted. Now our TIPN model has the ability to adjust time stamps of the tokens passing through the transitions. So in the example above, we can easily model that fire spread takes between 3 and 4 minutes.

In the evolving Petri net representation, it is still not easily possible to specify the spatial aspect of fire spread. In other words, we are in the need to adjust not only the time stamps but also the colors of the tokens passing through the transition. This can be achieved by attaching an “output operator” to the transition’s arcs leading to its output places. In Figure 4 we will simply attach a box labelled “Neighbors” to the arc leading to place “Ignited”. The operator will spread the fire from any compartment X to its neighbor Y. In FOPC the parallel effect is achieved through introducing an extra predicate neighbor/2:

\[
\text{neighbor}(X, Y) :\text{hot}(X, T), \text{not fbs}(X, R, T), \text{delay}(T, T', 3, 4), \text{neighbor}(X, Y).
\]

3.4 TIPNs with Inhibitory Arcs

At this point both our models (TIPN and FOPC) are very close to what we originally desired. Currently, however, tokens always get retracted as the transition fires. This does not correspond to the reality as the fact that fire spreads from compartment X to compartment Y does not render compartment X cold (i.e., not hot). This drawback can be addressed by introducing so-called double-ended arcs (Figure 5). They work just like regular arcs except the corresponding enabling tokens get to remain in their places. The reader might have also noticed that we currently have no way to express negation but through introducing negated concepts (atoms) such as “No Fire Boundaries Are Set”. It would be more convenient to represent negation explicitly with a different type of arc. We will call these “inhibitory arcs” and mark them with a tilde in our graphic diagrams. In the FOPC we will simply add Boolean negation:

\[
\text{ ignored}(Y, T') :\text{hot}(X, T), \text{not fbs}(X, R, T), \text{delay}(T, T', 3, 4), \text{neighbor}(X, Y).
\]

The evolving example is now complete. This concludes the introduction of Time Interval Petri Nets by use of a detailed example.

4 TIPN Experimentation

The issues of practical applicability of TIPNs as an envisionment module include: (a) the degree of speed-up over traditional numerical simulators, (b) TIPN scalability, and (c) the
Hot
Fire
Boundaries
Ignited
Spread

Figure 4: Step 4: just like delay interval [3,4] increases passing tokens' time stamps, arc operator “Neighbor” affects the color of tokens. It will spread fire from one compartment to another. In the FOPC the parallel effect is achieved through introducing an extra predicate neighbor/2:

\[
\text{ignited}(Y,T') \leftarrow \text{hot}(X,T), \neg \text{fbs}(X,R,T), \\
\text{delay}(T,T',3,4), \text{neighbor}(X,Y).
\]

Figure 5: Step 5: We finish the extension process by adding two new arc types: inhibitory arcs marked with tilde and double-ended arcs. The corresponding FOPC clause is this:

\[
\text{ignited}(Y,T') \leftarrow \text{hot}(X,T), \neg \text{fbs}(X,R,T), \\
\text{delay}(T,T',3,4), \text{neighbor}(X,Y).
\]
alleviated by 16 runs with the TIPN envisioner.

**Degree of Speed-Up and Scalability.** The numerical simulator was able to keep up its real-time operation even when simulating 20 simultaneous primary damage events that would in turn result in many more uncontrolled fire spreads. Table 1 shows the timings for the experiments and the degree of speed up. As mentioned above, primary fires were introduced at the very beginning of each scenario and were let burn uncontrollably. The envisionment/simulation was terminated at the end of the envisionment interval. For the purpose of this paper the numbers for the numerical simulator computation time are approximate using the simulation time. In practice, the actual computation time was within 10% of the estimate.

As a summary, speed-ups of three to five orders of magnitude were feasible with the TIPN envisionment module. It should be also noticed that increasing number of primary fires did slow down the TIPN module though it was still above the desired speedup of 12,000x described in Section 2 of this paper.

**Accuracy of Envisionment.** The accuracy of prediction results are summarized in Table 2.

As the table illustrates, the agreement between numeri-
As the next step, we tested the TIPN envisionment module in more realistic and practically useful settings of the ship damage control domain. A much greater set of physical and personnel activities phenomena were handled (Bulitko, 1998). The results presented below suggest that the TIPNs can be a useful envisionment tool in a challenging real-world domain.

TIPNs were used as an envisioner within a rule-based real-time decision-making system Minerva 5.2 (Bulitko, 1998, 2000). The system’s tasks were (1) to maintain situation awareness by taking in damage reports and sensor readings aboard a naval vessel as well as actively verify crisis reports by sending investigator teams and (2) to provide a casualty response by dispatching damage control teams and activating damage control devices (Wilkins et al., 2001). On every decision-making cycle the deliberation module of the system would suggest a set of feasible actions to take. The TIPN envisionment module was used to predict the effects of the actions and select the most promising one.

To evaluate the system’s performance, the input was 160 scenarios that had been run within the DC-Train ship damage control simulator at the Navy’s Surface Warfare Officers School (SWOS) in Newport, Rhode Island. These scenarios were judged by experts to be realistic and approximately 60 different Navy officers solved them. We compared the performance of Minerva-DCA (with the TIPN scheduler) to that of Navy officers at SWOS on these scenarios. The results are shown in Figure 7. Any scenario had one of the three outcomes: “ship lost”, meaning that a major disaster such as a missile compartment explosion was reached; “ship possibly saved”, meaning that at 25 minutes scenario time the ship was still alive yet there were active crises; and “ship saved”, meaning that there were no active crises at the 25-minute mark.

The use of Minerva with the TIPN envisionment module resulted in 117 out of 160 ships being saved. This is a 318% improvement over human DCAs wherein 28 ships were saved. Likewise, TIPN-equipped Minerva lost 21 ships, or 46% fewer than the Navy officers did.

### 5 Knowledge Acquisition for TIPNs

The previous sections have introduced the TIPN formalism and illustrated how a TIPN can be used to simulate fire spread on a ship, which can be used for example, to envision or predict the future course of the fire spread. In this section we highlight the guidelines for manual TIPN design. Automated methods including machine learning for TIPNs are presented in (Bulitko, 2000; Bulitko & Wilkins).

TIPN-based envisionment models can be handcrafted in a number of ways. In this section we will formulate some intuition and guidelines for doing so based on practical experience.

In the approach one typically begins by identifying domain states (described with a set of attributes), state changes, causal dependencies, and temporal information. As an example, we will use the fire spread TIPN presented in Figure 5. For the example, one would first identify state attributes of interest: (1) compartment temperature (hot/not hot); (2) compartment fire boundaries (set/not set); (3) compartment ignition (ignited/not ignited).

Then one identifies relevant state changes: fire spread (a compartment becomes ignited).

A domain expert would then inform the designer of the following causal dependency: fire spreads (i.e., a compart-

---

<table>
<thead>
<tr>
<th>Number of primary fires</th>
<th>Envisionment interval (minutes)</th>
<th>False positives</th>
<th>False negatives</th>
<th>TOTAL discrepancy</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>10</td>
<td>11</td>
<td>0</td>
<td>11</td>
</tr>
<tr>
<td>5</td>
<td>20</td>
<td>19</td>
<td>4</td>
<td>23</td>
</tr>
<tr>
<td>10</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>10</td>
<td>5</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>10</td>
<td>10</td>
<td>13</td>
<td>6</td>
<td>19</td>
</tr>
<tr>
<td>10</td>
<td>20</td>
<td>20</td>
<td>13</td>
<td>33</td>
</tr>
<tr>
<td>20</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>20</td>
<td>5</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>20</td>
<td>10</td>
<td>5</td>
<td>11</td>
<td>16</td>
</tr>
<tr>
<td>20</td>
<td>20</td>
<td>7</td>
<td>23</td>
<td>30</td>
</tr>
</tbody>
</table>

Table 2: Envisionment accuracy of the TIPN module versus the numerical simulator measured in the number of fires. The total discrepancy is the sum of false positives (i.e., the fires envisioned by TIPN but not the simulator) and false negatives (i.e., the fires envisioned by the simulator but not TIPN).
ment becomes ignited) if its neighboring compartment is hot and no fire boundaries are set around it and the associated temporal information: the process of fire spread takes 3 to 4 minutes.

Having this information at hand, one can start building a TIPN. The following conversion guidelines apply: (1) state attributes become TIPN places (e.g., places “Hot”, “Fire Boundaries”, “Ignited”); (2) state attribute types become TIPN token colors (e.g., tokens in place “Hot” would be colored with compartment tags); (3) state changes become transitions (e.g., transition “Spread”); (4) temporal information becomes transitions’ delay intervals (e.g., [3,4]); (5) causal information determines the TIPN arcs (e.g., enabling arc “Hot” to “Spread” and disabling arc “Fire Boundaries” to “Spread”). It also determines the arc operators (e.g., operator “Compartment” spreads the fire to a neighboring compartment).

Figure 8 represents this process graphically. The intuition is as follows: the marking (i.e., the set of tokens) of a TIPN represents the current domain state. If a token is present in a place then the relevant state attribute is considered to hold. The color of the token specifies the context. Example: a token tagged with 3-370-0-E, [5:00, 6:15] in the place “Ignited” represents the fact that compartment 3-370-0-E is known to become ignited between scenario times 5:00 and 6:15. Domain state changes correspond to marking changes. A TIPN marking changes when a transition fires and tokens flow through the transition. Thus, domain state changes correspond to TIPN transition firing. Each domain state change is causally dependant on some preconditions. In the TIPN world such dependency is expressed by linking enabling and disabling places to a TIPN transition. Thus the fact that fire spreads if a compartment is hot and no fire boundaries are set will lead to having one enabling input arc and one disabling input arc linked to the transition “Spread”.

6 Related Research

There exists extensive research in domain modelling for prediction-based decision-making. Often more accurate and detailed first-principle simulators (NIST, 2002; Shou et al., 2001; Heath, 1996) are compiled into less accurate and computationally effective models. The resulting trade-offs are an interesting research topic on its own (Russell & Wefald, 1991b,a). A detailed treatment of the major approaches is beyond the scope of the paper and can be found in (Bulitko, 2000).

Previous work that develops Petri Nets extensions for AI intelligent reasoning includes (Sil, 1995) for probabilistic reasoning, (Zhang & Murata, 1996) for rule-based logic systems, (Medeiros, Xexe, & de Souza, 1999) for fuzzy workflow management, and (Costa Miranda, 1999) for multi-agent systems. In the remainder of this section we will survey efforts on extending Petri Nets as a domain modelling formalism.

6.1 Timed Colored Petri Nets (TCP-nets) By Jensen

In (Jensen, 1997) Jensen suggests a set of extensions to his formalism of Colored Petri Nets (CP-nets) dealing with time. The new model is termed Timed Colored Petri Nets (TCP-nets) and contains the following main elements:

1. tokens with point-value timestamps indicating when the tokens are ready to bind;
2. a CP-net state characterized with not only the marking but also the central model time (a point value);
3. input arc expressions that evaluate to timed multi-sets. In other words, the expressions not only specify the tokens to be presented at the input places but also their timestamps relative to the model time;
4. output arc expressions that also evaluates to a timed multi-set. In this case the time values of the multi-set are treated...
A domain expert would then inform the designer of the following causal dependency:

- fire spreads (i.e., a compartment becomes ignited)
- compartment ignition (ignited/not ignited)
- compartment fire boundaries (set/not set)
- compartment temperature (hot/not hot)
- fire spreads (i.e., a compartment becomes ignited) if its neighboring compartment is hot and no fire boundaries are set around it

Temporal Information:
- the process of fire spread takes 3 to 4 minutes.

### Domain State Attributes:
- compartment fire boundaries (set/not set)
- compartment temperature (hot/not hot)
- compartment ignition (ignited/not ignited)

### Domain State Changes:
- fire spread (a compartment becomes ignited)

### Domain Causal Dependencies:
- fire spreads (i.e., a compartment becomes ignited) if its neighboring compartment is hot and no fire boundaries are set around it

### Figure 8: Creating a TIPN network from a domain model

![TIPN Network Diagram](image)

as the time delay values;

5. transitions eager to fire, meaning that the time enabled step with the earliest enabling time will occur (fire) first (if it does).

There are several important observations to make about TCPNs:

1. All the timestamps and delays are point-value with no intervals or probabilistic distributions and thus don’t easily allow for uncertainty to be taken into account.
2. The model operates with a single global clock meaning that the occurrence (firing) rules are non-local and the entire operation is ordered along the time dimension. In other words, a TCPN cannot simultaneously model two completely independent processes happening, for instance, 10 days apart.
3. Each TCPN has a corresponding CPN readily derivable by dropping all the timing information. An occurrence graph of a TCPN (without the timing information included in the nodes) is a subset of the occurrence graph of the corresponding CPN since TCPN firing rules are stricter than their CPN counterparts.
4. For a cyclic TCPN the occurrence graph is likely to become infinite since the timestamps are likely to continue to increase. In such a case either an OE-graph should be devised or the scenario timeline should be artificially limited to make the graph suitable to analysis.

#### 6.2 Timed Colored Petri Nets (ITCP-nets) by van der Aalst

One of the important drawbacks of representing time stamps and time delays as point values is that in reality, the durations of subprocesses typically vary. This fact might have an adverse effect on accuracy and convenience of modelling with TCPNs. Several extensions have been proposed including probabilistic time delay distributions and interval calculus. There are several strong assumptions one has to make in order to use probabilistic distribution methods for transition time delays and still the analysis methods might not be always adequate (van der Aalst, 1993). In this section we will consider an alternative formalism called Interval Time Colored Petri Nets (ITCPN) suggested by van der Aalst in (van der Aalst, 1993) which is to represent transition time-delays as intervals. Below we will present the highlights of the framework illustrated with a simple example (Figure 9):

1. Like with the TCP-net formalism presented above, the tokens are point time stamped and colored. Arc expressions are implemented through the transition function which translates a multi-set of tokens to be consumed into a multi-set of tokens to be produced. In our example the transition function is not given explicitly which means, by default, that it takes a single token from each input places (i.e., \( p_1 \)) and outputs a single token into each of the output places (i.e., \( p_2 \) and \( p_3 \)).
2. One of major differences between TCPNs and ITCPNs lies with the transition delays. In ITCPNs the transition delays are represented as closed intervals. Each output arc is assigned a time delay interval. In our example arc \( (t, p_2) \) has the delay interval of \([1, 2]\) and arc \( (t, p_3) \) has the delay interval of \([3, 4]\).
3. The enabling time of an event (or step in TCP-net terminology) is the maximum of all timestamps of the tokens to be consumed. van der Aalst also defines the model time as the minimum of all enabling times. Thus, the
An example of Interval Timed Colored Petri Net (ITCPN).

The model time advances only when an event happens. In our example there are two events (steps): one is with token \( (a, 5) \) as the enabling token and the other has token \( (b, 10) \) as the enabling token. In the first case the enabling time is \( \max\{5\} = 5 \) and in the second case the enabling time is \( \max\{10\} = 10 \). The model time will be equal to \( \min\{5, 10\} = 5 \) and thus the first event is bound to occur before the second one.

4. Once an event happens (or step occurs) the new tokens are put into the output places. Each of them will have a new timestamp calculated as follows: \( t_{\text{new}} = t_{\text{firing}} + t_d \) where \( t_{\text{new}} \) is the new point-valued timestamp, \( t_{\text{firing}} \) is the firing time, and \( t_d \) is the actual delay value which is required to fall into the delay interval \( [\text{delay}_{\text{min}}, \text{delay}_{\text{max}}] \). In our example if the first event (with \( (a, 5) \) as the enabling token) happens then the resulting marking will look as shown in Figure 10 if the actual delays happen to be \( 1.79 \in [1, 2] \) for the first output arc and \( 3.2 \in [3, 4] \) for the second output arc. It is important to realize that there are non-countably many markings resulting from the event as the delays can take on any real numbers in ranges \([1, 2]\) and \([3, 4]\) correspondingly. So in reality we get the following set of markings:

\[
\{ p_1 : (b, 10) , p_2 : (a, 5 + d_1) , p_3 : (a, 5 + d_2) \mid d_1 \in [1, 2], d_2 \in [3, 4] \}
\]

The ITCPN extensions are undoubtedly a significant advance over the PT nets. The time stamping, coloring, and interval calculus provided for successful applications of the formalism (van der Aalst, 1993). It should be also noted that the ITCPN and TIPN formalisms while developed independently, share a number of common attributes and strengths. In the following we will point out some differences among TIPNs, ITCPNs, and TCPNs.

1. Since ITCPN delay values are non-deterministically "sampled" from the non-point delay intervals the occurrence (reachability) graph will always be infinite (even non-countably infinite). That makes the reachability graph based analysis inefficient. To relax this problem van der Aalst (van der Aalst, 1993) created alternative ITCPN behavior semantics in which token timestamps are intervals and thus the entire model implements an interval calculus. A similar calculus was adapted consistently throughout TIPNs from the very beginning.

2. There is a central ITCPN model time and the transitions are eager to fire. In other words, enabled firing steps (events) will be executed in chronological order. Thus, like TCP-nets, ITCP-nets do not allow for concurrent simulations of processes which are time-separated.

3. Unlike in the case of TCPNs, ITCPNs appear to have no provision for premature token use. In other words, the input arcs carry no time expressions and thus a token can be used only on or after the model time equals its time stamp.

4. Unlike in the case of TIPNs, ITCPNs don’t provide for uncertainty with regard to token timestamps for any single marking since all the timestamps are point values and thus are assumed to be known exactly. Therefore, there is a need to represent the uncertainty with a generally non-countable set of markings.

7 Summary and Conclusions

The following main contributions were presented in the paper:

1. A Petri Nets based approach to decision-making through environment modelling is presented. Petri Nets formalism is known for its solid theoretical base, clear syntax and semantics, intuitive graphic representation, and native concurrency support (Peterson, 1981; Murata, 1989).

2. The classical Petri Net model is extended in various ways to make it suitable for reasoning within AI environments. The main extensions concern explicit temporal reasoning, context, and operator support. The new formalism is hence called Time Interval Petri Nets (TIPNs).

3. The framework has been applied in the real-time decision-making domain of ship damage control for the tasks of automated problem-solving and intelligent tutoring (advising, critiquing, and scoring). In a large exercise involving 160 simulated ship crisis scenarios, our decision-making expert system showed a 318% improvement over Navy officers by saving 89 more ships.
Subsequently, the following directions for the future research appear to be interesting:

1. A TIPN model encodes what is commonly known as the next state function \( \delta(s_t, a) = s_{t+1} \) where \( s_t \) is the world's state at time \( t \), \( a \) is the action the agent takes at time \( t \), and \( s_{t+1} \) is the state of the world at time \( t+1 \). In the scheduling framework presented in this paper we used the function to predict the effects (i.e., to calculate \( s_{t+1} \) given the current situation (\( s_t \)) and the action (\( a \)) we are considering taking. Another interesting way of using \( \delta \) is to solve the equation above for action \( a \) given the current state \( s_t \) and the desired future state \( s_{t+1} \). Using the model in such a way will allow us to produce the action to take given where we are and where we want to be. If the equation allows for multiple solutions then a separate scheduler (e.g., a more refined model) should be used to arbitrate among them.

2. A large number of formal analysis techniques have been developed for Petri nets that are relevant to modelling domains with concurrent temporal processes. Examples of formal analysis techniques include computing deadlocks, cycles, equivalence, coverability, and reachability (Peterson, 1981). A line of future research is to show how these techniques are valuable when using Petri Nets for physical and intelligent agent simulation in an intelligent system domain. Some initial efforts in this direction are reported in (Bulitko, 2000).

Acknowledgements

IRCL members Sebastian Magda, Anthony Czupryna, Jr., and David Fried have done most of the TIPN coding and have contributed numerous interesting suggestions. Anthony Czupryna’s participation deserves a special recognition. Discussions with KBS members and Drs. Valeriy K. Bulitko and Dan Roth were helpful. Detailed reviews from anonymous reviewers led the paper to its current form. The research has been supported in part by ONR Grant N00014-97-1-0749, ARL Grant DAAL01-96-2-0003, NRL Contract N00014-97-C-2061 as well as University of Alberta and NSERC grants.

References


