Dynamic-Stochastic Influence Diagrams: Integrating Time-Slices IDs and Discrete Event Systems Modeling

Xin Zhao, Yin-fan Zhu, Wei-ping Wang, and Qun Li

Abstract—The Influence Diagrams (IDs) is a kind of Probabilistic Belief Networks for graphic modeling. The usage of IDs can improve the communication among field experts, modelers, and decision makers, by showing the issue frame discussed from a high-level point of view. This paper enhances the Time-Sliced Influence Diagrams (TSIDs, or called Dynamic IDs) based formalism from a Discrete Event Systems Modeling and Simulation (DES M&S) perspective, for Exploring Analysis (EA) modeling. The enhancements enable a modeler to specify times occurred of endogenous events dynamically with stochastic sampling as model running and to describe the inter-influences among them with variable nodes in a dynamic situation that the existing TSIDs fails to capture. The new class of model is named Dynamic-Stochastic Influence Diagrams (DSIDs). The paper includes a description of the modeling formalism and the hiberarchy simulators implementing its simulation algorithm, and shows a case study to illustrate its enhancements.

Keywords—time-sliced influence diagrams; discrete event systems; dynamic-stochastic influence diagrams; modeling formalism; simulation algorithm

I. INTRODUCTION

THE weapon and equipment system of systems (SoS) demonstration is a high level complex systems analysis issue and involves vast uncertain factors. In the modern complex world, decision makers are faced with an ever increasing number of situations that require a all-around response, ranging from terrorism to local battles and perhaps campaign between multi-national organizations for the employment of military forces to achieve or maintain demonstration objectives. To handle them effectively, Rand has gave a Exploring Analysis (EA) methodology for Capabilities-Based Planning (CBP)[2][3]. As a model-based method, EA needs suitable model form for analysis supporting. The model’s characters include: lower resolution; reflecting cause and effect relations among entities but also complex relationships, it’s not easy to reflect its ‘time’ by a time-slice concept, it still can’t be fit for all kinds of processes description. This is its inherent deficiency for complex cooperated processes description.

In other words, although TSIDs connects ‘uncertainty’ with ‘time’ by a time-slice concept, it still can’t be fit for all kinds of high level modeling. If the ‘system’ not only has complex entities but also complex relationships, it’s not easy to reflect its process clear. Take military campaign for example, if it contains interactional crossed loops that caused by different weapons and troops, it’s hard to abstract them.

The extended Time-Sliced Influence Diagrams (TSIDs) [5][6] allows the existence of dynamic feedback loop circle to classic IDs[7] for graphic modeling. It’s a special case of Dynamic Bayesian Networks (DBNs) [8][9], that increases the decision nodes and the objective nodes for optimization of decision-making, not just probabilistic evaluation only. The TSIDs allows a system analyst to observe how uncertainties of variables influence model’s behavior with time changing.

TSIDs had been experimentally used in the area of EA for describing important factors of campaign and their relations dynamically [10][11]. The TSIDs-based formalism is appropriate to answer influences queries and to perform what-if analyses.

Despite its ability to model complex situations in a compact and easy to read manner, TSIDs fails to capture variance of time sequences among events of dynamic situations. Furthermore, from another view, the object that TSIDs could describe is usual an independent process or subpart of the high-level process. This is its inherent deficiency for complex cooperated processes description.

Moreover, the relationship description between times or influence factors are settled in TSIDs. It’s not enough to describe the dynamic and stochastic cooperated operations. As Fig.1, TSIDs just runs short of variational temporal sequences among different time-slices.

This shortcoming of TSIDs may turn out to be unrealistic in...
many real world situations. The paper proposes syntax and semantic enhancement to TSIDs to overcome the limitation. The enhancement would enable a system analyst to model the impacts of different actions on the desired effect in a dynamic uncertain situation.

The rest of the paper is organized as follows. Section II discusses the technical background—IDs, TSIDs, and the combination for TSIDs and DES. Sections III and IV describe the proposed modeling formalism and its simulation algorithm, respectively. Section V illustrates the proposed enhancements by a case study that realized in EASim—an IDs modeling tool. Finally, Section VI concludes the paper and points towards the future research direction.

II. TECHNICAL BACKGROUND

A. Classic IDs

Classic Influence Diagrams (IDs) [7] is a directional acyclic graph for decision-making issue modeling. It’s also a probability net being used for solving uncertainty and reflecting qualitative & quantitative knowledge synthetically and an effective tool for information communication. Decision-making analysis based on IDs is usual at strategic level. The classic IDs is formed by directional arcs and nodes with data structures, as Fig.2.

![Fig. 2 Graphical elements of classic IDs](image)

The developing process of IDs is field and application oriented basically. Their evolvement separately or simultaneously occurred at relation, function and value level, including[12][13][14][15][16][17][18]: Probability IDs, Fuzzy IDs, Rough IDs, Potential IDs, Monotonic IDs and Ignorant IDs etc. No matter what requirements of application they fulfilled or what levels they occurred at, the essential purpose is to utilize the IDs’ powerful abilities to graphically capture causal and influence relationships for analytical modeling that is a kind of methodology for analyzing complex strategic decision-making issue. Among them, Time-Sliced IDs (TSIDs) is especially worthy to get attention for model-based high level issue decision-making analysis.

B. Time-Sliced IDs

By introducing time concept and dynamic character, TSIDs extended classic IDs and could support differential equation and continue system modeling for analysis. This also made it more fit for abstracting averaged process, and improved cause and effect influence net’s description comparing with static analytic IDs, as Fig.3.

![Fig. 3 Classic IDs to TSIDs](image)

An TSIDs is a four-tuple $\text{TSIDs} = \langle D, M, A, h_N \rangle$ where
- $D$ is the index set of nodes;
- $M = \{M_d \mid d \in D\}$ is the set of variable nodes;
- $A$ is the set of influence arcs;
- $h_N$ is the iterative step.

C. Combination of TSIDs and DES M&S

The capability to enhance that TSIDs needs is just what the Discrete Event Systems (DES) M&S owned. DES [19][20][21] can express the input and intrinsic random, time delay, and queue. And so it can describe simultaneous, cooperating and multilevel complex processes. It’s a good reference for TSIDs improvement. The combined modeling of TSIDs and DES can form a new formalism to enhance the top-down complex system modeling. So, when you do a high level complex issue modeling, the TSIDs part can be used to model the issue’s macrostructure parts well, and the DES part can be used to model other detailed parts.

1) Relationship of ‘time’ conception: The two parts (TSIDs and DES) should have accordant ‘time’ conception. So ‘Etime’ is introduced to represent event time comparing with ‘Ftime’ that representing feedback time, as Fig.4.

![Fig. 4 Event time and feedback time](image)

2) Combination mode: From the graphic syntax and semantic point of view, the nodes in DES M&S must have bidirectional relationship with the variables of TSIDs, whether events, states, activities, or processes they figured. Considering discrepancies of two parts, their elements may have two kinds of combination modes: one is node-combined, what need holistic simulated evaluation, and is a multi-granularity description to the same level and towards behavior (event) principally; the other is module-combined, what need separated simulated evaluation, and is a multi-resolution description to different levels and towards entity principally. The former mode is flexible and suit for complex system description especially to our situation.

3) Principles: As amelioration to existing method, it just should
- Be graphic, and needs less extension to TSIDs;
- Be intuitionist;
- Be prone to understand, and be convenient for model structure redress and relationship explanation;
- Unitive rule, and be convenient for software design and implementation.
III. PROPOSED MODELING FORMALISM

In this paper, the improved method is called as ‘Dynamic-Stochastic Influence Diagrams (DSIDs)’, for its ability for dynamic arranging of stochastic events, processes and influ-
ences, and supporting their random treating. To describe the characters of DES, DSIDs add a ‘Time node’ at TSIDs foundation, expressing stochastic events by time list directly, as Fig.5. This is propitious to assert with nodes of TSIDs, and to transfer evaluating engine.

![Fig. 5 Primary variable nodes of DSIDs](image)

The DSIDs describes pivotal elements of issue by these graphic nodes, and the connection of them form a DSIDs net model. In addition, as a graphic modeling method, it should be considered that doing some structural processing for input and output interfaces during the description of DSIDs formalism. A standard DSIDs model specification is a structure:

\[
\text{DSIDs}_i = \langle D, M, A, I, Z, h_N, t, t_a \rangle
\]

Where

- \( D \) is the index set of nodes;
- \( M = \{ M_d \mid d \in D \} \) is the set of variable nodes;
- \( A \) is the set of influence arcs;
- \( I = \{ I_d \mid d \in D \} \) is the set of nodes that influence node \( M_d \);
- \( Z = \{ Z_d \mid d \in D \} \) is the interface mapping set of node \( M_d \);
- \( h_N \) is the iterative step;
- \( t \) is the current time;
- \( t_a \) is the time advance function.

A. Nodes Set

There are five kinds of nodes in DSIDs, \( M = N_C \cup N_Y \cup N_P \cup N_D \cup N_T \), namely chance node, variable node, decision node, objective node, and time node. Hereinto, the variable form of time node is chain list for value storage representing the time that event will take place.

To \( \forall d \in D, M_d = \langle n, X_d, Y_d, V_{\text{init}}, V_d, \Delta_d, \lambda_d, t, t_a \rangle \), in which

- \( n \in NP \) is the name of node, and \( NP \) is the name space;
- \( t \) is the current time;
- \( h \) is the current actual step;
- \( V_{\text{init}} \) is the initialized value, maybe a number or a depending function to other nodes;
- \( V_d \) is the value of node to each time;
- \( X_d = \{ (p, v) \mid p \in IPorts_d, v \in X_{p,i} \} \) is the multiva-

variable set of inputs of the node with \( IPorts_d \);
- \( Y_d = \{ (p, v) \mid p \in OPorts_d, v \in Y_{p,i} \} \) is the multiva-

variable set of outputs of the node with \( OPorts_d \);
- \( \Delta_d \) is the internal logic function;
- \( \lambda_d \) is the output mapping function.

Of course, to different nodes, the detailed definitions are different that be omitted for conciseness here.

The semantic executing of node can described as follow: After computed the time advanced by node net (DSIDs), nodes receive the value of \( t \) and \( h \); when received the value \( X_{(i)} \) from all input ports, based on themselves’ forepassed value, them transact internal logic by \( \lambda_d \), calculate output values by \( \Delta_d \) and map them to output ports.

B. Arcs Set

The influence relationships are figured by arcs in DSIDs and different for the sort of nodes they connected, to form the net structure together with all nodes. In DSIDs, \( A = A_f \cup A_c \cup A_d \cup A_c \cup A_e \cup A_f \), as Tab.I.

The former four are the same as those of TSIDs, and the later

<table>
<thead>
<tr>
<th>Arc</th>
<th>Definition</th>
<th>Signification</th>
</tr>
</thead>
<tbody>
<tr>
<td>( A_f )</td>
<td>( {(x, y), x \in N_C \cup N_Y, y \in N_T } )</td>
<td>relevancy</td>
</tr>
<tr>
<td>( A_c )</td>
<td>( {(x, y), x \in N_C \cup N_Y, y \in N_T } )</td>
<td>information</td>
</tr>
<tr>
<td>( A_d )</td>
<td>( {(x, y), x \in N_D, y \in N_T } )</td>
<td>influence</td>
</tr>
<tr>
<td>( A_e )</td>
<td>( {(x, y), x \in N_P, y \in N_T } )</td>
<td>order</td>
</tr>
<tr>
<td>( A_c )</td>
<td>( {(x, y), x \in N_C, y \in N_T } )</td>
<td>condition</td>
</tr>
<tr>
<td>( A_f )</td>
<td>( {(x, y), x \in N_T } )</td>
<td>action</td>
</tr>
</tbody>
</table>

two are what DSIDs has peculiarity. Respectively, \( A_f \) indicates condition that its source node provides to target node, and \( A_f \) indicates action that its source node affects to target node when the event happened.

In addition, there is also a loop restriction in DSIDs as follow: In the net connected by \( A_f \), \( A_c \), \( A_d \), \( A_c \), and correlative nodes, there couldn’t exist circle depending that without time delay. Namely, to \( \forall i, j \in D \), if the calculation of \( V_{ij} \) indirectly depends on \( V_{ji} \), then the calculation of \( V_{ij} \) couldn’t depends on \( V_{ji} \) synchronously, no matter directly or indirectly.

C. Other Constituents

1) The set of nodes that influence \( M_d \) : To \( \forall d \in D : I_d = \{ i \mid ((i, \text{fromport}), (d, \text{toport})) \in I \} \), where \( I_C \subseteq \{ ((a, o_p), (b, i_p)) \mid a, b, d, o_p, i_p \in OPorts_a, i_p \in IPorts_b \} \) that expressing linked relationships.

2) The interface mapping set of \( M_d \) : To \( \forall d \in D : Z_d = \{ x_{i_d} \} Y_{i_d} = X_d \), \( Z_d (\ldots, Y_{(i, \text{fromport})}, \ldots) = x_{d(\text{toport})} \), and

\[
x_{d(\text{toport})} = \bigoplus_{i \in I_d} Y_{i_d(\text{fromport})} = (x_{\text{fromport}})_{(i, \text{fromport})}^{(d, \text{toport})} \cdot y_{(\text{fromport})}
\]

Where \( \bigoplus x = \text{true} \); if \( x \neq \text{false} \) it \( i \Rightarrow x_{i_d} = \text{true} \), then \( \bigoplus x, x_2, \ldots, x_n = x_1 \), else \( \bigoplus x, x_2, \ldots, x_n = \text{false} \). ‘ \( \oplus \) ’ allow
several output ports linked to one input port. It’s helpful to DES expression [21].

3) andNh: t is the current time, Nh corresponds to the ‘Time’ conception that mentioned before, they are calculated by function ‘ta’.

4) Time advance function ‘ta’: The time-advance of DSIDs needs to consider time values updating and events arising. This situation is similar to formalism M&S of discrete time and discrete event systems. There is a similar concept ‘state event’ too. The implementation of function ta followed the ‘bisection method’ arithmetic that the article [21] mentioned. It’s the function of t, Nh and Vd of all nodes, and so

\[ ta: \prod_{d=1}^{n} V_{d(i)} \times t \times h_{N} \rightarrow R_{h_{N}}. \]

IV. SIMULATION ALGORITHM FOR DSIDs EVALUATION

The algorithm of DSIDs simulated evaluation could be described as Fig.6. To see the detail of TSIDs evaluation algorithm implementation, you may take [5][6][9][22] for references. The algorithm and the ‘forward calculate’ will be materialized in the description of simulators next.

![Fig. 6 The logistic process of DSIDs evaluation algorithm](image)

The ‘LastTime’ of algorithm expresses the latest calculating time that between ‘Time’ and ‘Time-Step’ (Step=1 here), as Fig.7.

![Fig. 7 The ‘time’ in evaluation processing](image)

The algorithm is compatible for TSIDs, and so they (TSIDs & DSIDs) are equivalent without ‘time node’.

A. Hiberarchy

The simulators of DSIDs may have clear hiberarchy as Fig.8. The DSIDs simulator structure is compartmentalized into ‘node-simulator’ for ‘node’ and ‘net-coordinator’ for ‘net’ that formed by nodes. The former control the operation of nodes, and the later define the connection of them. Moreover, ‘net-coordinator’ also takes charge for time advancing, state events scheduling, and messages transferring etc. In addition, a ‘root-simulator’ is introduced to control simulation repetition which will not be discussed detailed in this paper.

![Fig. 8 Hiberarchy of DSIDs simulators](image)

The simulation algorithm of DSIDs is implemented based on the following seven kinds of messages. DSIDs scheduling process is realized by messages communication.

<table>
<thead>
<tr>
<th>Name</th>
<th>Message</th>
<th>Send and Receive</th>
</tr>
</thead>
<tbody>
<tr>
<td>i</td>
<td>Initialization</td>
<td>upper to lower</td>
</tr>
<tr>
<td>w</td>
<td>Calculation enabled</td>
<td>net-coordinator to node-simulator</td>
</tr>
<tr>
<td>e</td>
<td>Event schedule</td>
<td>net-coordinator to node-simulator</td>
</tr>
<tr>
<td>r</td>
<td>Earliest time</td>
<td>net-coordinator to node-simulator</td>
</tr>
<tr>
<td>se</td>
<td>Failure of state event</td>
<td>node-simulator of Net to node-coordinator</td>
</tr>
<tr>
<td>x</td>
<td>Value input</td>
<td>net-coordinator to node-simulator</td>
</tr>
<tr>
<td>y</td>
<td>Value output</td>
<td>node-simulator to net-coordinator</td>
</tr>
</tbody>
</table>

Tab.II shows symbols of seven messages on which the DSIDs evaluation is based, while their sending and receiving mode are described in Tab.III.

<table>
<thead>
<tr>
<th>Message</th>
<th>node-simulator</th>
<th>net-coordinator</th>
<th>root-simulator</th>
</tr>
</thead>
<tbody>
<tr>
<td>(i, t)</td>
<td>↓</td>
<td>↓</td>
<td>→</td>
</tr>
<tr>
<td>(*, t)</td>
<td>↓</td>
<td>↓</td>
<td>→</td>
</tr>
<tr>
<td>(c, t)</td>
<td>--</td>
<td>--</td>
<td>--</td>
</tr>
<tr>
<td>(r, t)</td>
<td>--</td>
<td>--</td>
<td>→</td>
</tr>
<tr>
<td>(se, t)</td>
<td>--</td>
<td>--</td>
<td>←</td>
</tr>
<tr>
<td>(x, t)</td>
<td>--</td>
<td>--</td>
<td>↓</td>
</tr>
<tr>
<td>(y, t)</td>
<td>--</td>
<td>--</td>
<td>↓</td>
</tr>
</tbody>
</table>

Legends: ↓: receive; ←: send; --: void.

B. Description of Simulators

The following sections will show detail of algorithm realization for each simulator.

1) Node-Simulator: There are four node-simulators for different kinds of nodes. They all base on the basic
node-simulator and have each peculiarity. Fig.9 to Fig.14 shows
details.

Fig. 9 Modality of basic node-simulator

node - simulator for \( n \in N_c \)
when receive \( \ast \)-message \((\ast,t)\)
if \( Y_d = \emptyset \)
error : bad synchronize action
\( Y_d = V_{d0} = A_d(t,h,V_d) \)
send \( y \)-message \((Y_d,t)\) to parent
\( tl = tn \)
\( tn = t \)

Fig. 10 Node-simulator for chance node

node - simulator for \( n \in N_p \)
when receive \( \ast \)-message \((\ast,t)\)
if \( Y_d = \emptyset \)
error : bad synchronize action
if \( X_d = \emptyset \)
\( Y_d = V_{d0} = A_d(t,h,V_d) \)
send \( y \)-message \((Y_d,t)\) to parent
\( tl = tn \)
\( tn = t \)

Fig. 11 Node-simulator for decision node

node - simulator for \( n \in N_v \)
when receive \( x \)-message \((x,t)\) at time \( t \)
if \( Y_d = \emptyset \)
error : bad synchronize action
if \( X_d = \emptyset \)
\( X_{d\{x\}\{p\}} = x_{d\{x\}\{p\}} \)
\( \exists x_{d\{x\}\{p\}} = \emptyset \) then
\( Y_d = V_{d0} = A_d(t,h,V_d,X_{d0}) \)
send \( y \)-message \((Y_d,t)\) to parent
\( tl = tn \)
\( tn = t \)

Fig. 12 Node-simulator for variable node

node - simulator for \( n \in N_o \)
when receive \( x \)-message \((x,t)\) at time \( t \)
if \( Y_d = \emptyset \)
error : bad synchronize action
if \( X_d = \emptyset \)
\( X_{d\{x\}\{p\}} = \emptyset \) then
\( Y_d = V_{d0} = A_d(t,h,V_d,X_{d0}) \)
send \( y \)-message \((Y_d,t)\) to parent
\( tl = tn \)
\( tn = t \)

Fig. 13 Node-simulator for objective node

node - simulator for \( n \in N_t \)
variable es :
\( tle \) // t ime of last event
\( ar_{us} \) // accuracy reached judgement function
\( P \) // priority

cancel \( ml \) // cancel event
when receive \( e \)-message \((e,t)\) at time \( t \)
if \( Y_d = \emptyset \) or \( X_d = \emptyset \)
error : bad synchronize action
\( X_{d\{e\}\{p\}} = \emptyset \) then
\( Y_d = V_{d0} = A_d(t,h,V_d) \)
send \( y \)-message \((Y_d,t)\) to parent
\( tl = tn \)
\( tn = t \)

Fig. 14 Node-simulator for time node
What needs to be specially explained is that the simulator for time node: differentiate input values that from Nf and non-Nf; includes two special functions, arr(t, t) for state event accuracy judgment and cancel(t, V, t, h, X, X0) for event canceling; and owns several output ports to drive other time nodes or influence linked variable nodes.

2) Net-Coordinator: After the definition of node-simulators, the net-coordinator, whose task is to send and receive messages to make time advanced and nodes calculated, could be discussed. See Fig.15 for details.

```c
net - coordinator

variable es :
    parent // root - coordinator
    tl // time of last
    tn // time of now
    tf // time of future (next step time)
    fTime // FTime
DSIDs <= D,M,A,h,y,t,a,1,Z > /net
when receive i - message (i, t) at time t
for each d in D do
    send i - message (i, t) to child d
fTime me = t
    tl = t
    tn = t
    tf = 0
send * - message (* , t) to self
when receive * - message (* , t)
while t = fTime + h
do fTime = t
    tl = tn
    tn = tn
    tf = tf + h
send * - message (* , t) to self
when receive r - message (r, t) at time t
    if tn < r < tf then
        tf = r
when receive se - message (se, t) at time t
    stop the computatino n of all children
rollback all variables and events to time t = tl
    tf = fTime + h, then
send * - message(* , t) to self
when receive y - message (y, t) from d
    with output value y, y0 - to from
receivers = {t, r, toport} // root - coordinator
for each (r, toport) in receivers
    send x - messages (x, toport) to (r, toport)
end net - coordinator
```

Fig. 15 Net-Coordinator

V. A MODELING TOOL AND CASE STUDY

A. EASim - Modeling Tool for DSIDs

A software tool – EASim, was designed and implemented, as Fig.16, which provides the ability and agility to deal with complex issues of real world, by supporting DSIDs further for analytical modeling.

In EASim,

- The hiberarchy description makes it more simplified to build or manage big model and increases the ability for analytical model’s building, organizing and evaluating.
- And variables can also be vector array except scalar.
- It’s another source of flexible and abstract ability for model maker. It will redundantly make some treat-off among model particular, evaluating time, available data and dimensionality space.
- To improve representation of discrete processes, EASim not only samples at each evaluating, but also samples at every steps in each evaluating, and provides several statistical methods for analyzing.
- A script language, Python, is embedded in EASim; and extended by C++ to provide more power for analytical modeling.

In addition, the DSIDs evaluation may include vast data and complex process, especially for complex military SoS counterwork. So, EASim’s evaluating engine is not to immit all correlative data of the model into memory, but to storage model using XML format files, and then respectively evaluate these files by pretreating them to multi-batches. It not only improves running efficiency of evaluating, but also relaxes strict memory and computing power requirements to PC – analysis is often carried out in a combinatorial space of all factors’ various values and common needs to be performed just by PC.

B. DSIDs Model of Armada Area Defense Issue

An armada area defense issue is taken for DSIDs modeling example. The basic scenario is that:

- Attacking planes, from different batches, pierce through the defense lines and then fires cruise missiles to attack weapon carriers of defending side for anti-ship missile intercepting;
- Attacking side fires some ballistic missiles to attack kernel target directly at opportune time;
- Defending side intercepts the attacking planes and missiles by air-to-air battles and air defense missiles.

Fig.17 shows the macrocosm structure of the DSIDs model and the influences of factors between time nodes and other nodes, for armada area defense issue.

Fig.18 shows the gross num vs. useable num of node
‘Amount of Intercepting Weapon’ in the DSIDs model. It illustrates that the competition of war resources could come forth at any moment and this could influence the actual occurrences of events.

VI. CONCLUSION

Classic IDs is a good tool to support decision-making for its laconic graphics and effectual approximate reasoning. And the extended Time-sliced IDs has been used for strategic planning successfully. To get more creditability for special object, the weapon and equipment system of systems (SoS) demonstration, one of strategic planning issues, the shortcoming of TSIDs is discussed and has been enhanced to own more capability to describe more details of battle processes for influence factors analysis. The proposed class of model is named Dynamic-Stochastic Influence Diagrams (DSIDs). And its modeling formalism and simulation algorithm was described amply. It had been used in the martial planning example excellently. To make the new formalism could be used widely, more work have to do further. For example, as a combination of TSIDs and DES M&S, DSIDs has relationships with DBNs, DBNs, colored Petri nets, and Event Graphs etc, the conversions to these interrelated systems make it’s possible to use a variety of analysis algorithms developed for them.
ACKNOWLEDGMENT

The authors would like to gratefully acknowledge the financial support of the National Defense Pre-Research Foundation of China (Grant No. 9140C640505), the National Natural Science Foundation of China (No. 60974073 and No. 60974074).

REFERENCES


