一种基于并行存储的多分辨率建模方法

赵小松 李国辉
(华中科技大学计算机学院 武汉 430074)

摘 要 多粒度仿真是当今仿真研究的重点和难点,其在仿真模拟领域的重要性逐渐显现出来。在该领域已提出了很多方法,其中聚合解聚法因简单易行且通信开销小而引起大家的注意,但该方法导致的一致性问题很难解决。针对聚合解聚法和多分辨率实体法提出了基于并行存储的多分辨率建模方法,此方法在一定程度上解决了聚合解聚法引起的一致性问题,同时不像多分辨率实体法那样引起很大的资源及通信开销。

关键词 多分辨率建模,多粒度仿真,并行存储,聚合解聚
中图法分类号 TP391.9 **文献标识码** A

Multi-resolution Modeling Based on Parallel Storage

ZHAO Xiao-song LI Guo-hui
(School of Computer Science & Technology, Huazhong University of Science & Technology, Wuhan 430074, China)

Abstract Multi-resolution simulation is one of the emphases and difficulties in simulation research and its importance gradually appears. Among the existing means in the field, because the aggregation-disaggregation is simple and practicable, it rivets people's attention, but the consistency problem is serious. Based on the analysis of the problems in current methods, a new concept named parallel storage was proposed. This method solves the consistency problem in some way while costing less resource.

Keywords Multi-resolution modeling, Multi-resolution simulation, Parallel storage, Aggregate and disaggregate

1 引言

随着当今模拟仿真系统规模的扩大和对仿真度要求的提高,传统的固定分辨率建模不能很好地解决系统复杂度与资源有限性的矛盾问题日益严重。一个复杂而开放的大型系统会涉及到诸多层次。不同的层次上,该系统体现出不同的细节;在不同的时刻,我们关注该系统的层次也不同,这就需要在模拟仿真中实现各仿真层次之间的转换。在大型的仿真系统中,不同的层次上需要展现不同的属性及特征。而满足这一过程的多粒度仿真中,应用最普遍的便是聚合解聚法。同时,多分辨率实体法虽然实施起来难度大,但它也以一种独特的视角引起人们的注意。本文在国内外已有研究成果的基础上,提出一种新的仿真策略,以更好地解决一致性问题和通信开销问题。

2 研究现状及相关工作

2.1 研究现状

多分辨率仿真是指从多个抽象层次上进行仿真,分辨率的大小表示对事物抽象程度的高低。在国防领域,由于作战仿真系统的复杂性,战场情况也是复杂多变,可能某时刻我们需要知道整个连的作战能力和防御能力,但下个时刻我们更关注该连中某炮兵排的具体情况。在这个转变过程中,我们感兴趣的对象发生了变化,这就需要我们对对象的抽象程度加以变化。用多分辨率的概念建立模型可针对同一系统在不同的层次上建立不同分辨率的模型。不同的分辨率关注的细节不同,低分辨率抽象程度高,更关注整体的情况。低分辨率模型的运行需要较少的资源和开销,而高分辨率模型抽象程度低,能让我们更多地了解局部细节。由于高分辨率模型包含了更多的信息,因此它的运行需要更多的资源和开销。

针对作战系统的多分辨率仿真的研究起源于20世纪80年代,当时的代表性成果是美国兰德公司的兰德战略评估系统(RSAS)。90年代初期到中期这段时间,对于多分辨率仿真的研究进入一个停滞阶段。中后期随着大型分布式交互仿真的蓬勃发展,对该领域的研究再度繁荣。目前国外典型应用有美国的军用建模框架MMF、美国空军实验室和MITRE公司实现的战术级仿真和战役级仿真系统的互联。国内对该领域的研究始于20世纪90年代中期军事科学院的张最良和马欣明,其目前也取得一些成果,如"长城一号"、"长虹一号"等应用系统已经成功构建。国内对该领域的研究起步较晚,尚处于起步阶段,还有许多问题需要进一步研究。其中的聚合解聚问题是该领域的关键,涉及到"聚合和解聚模型、聚合什么、解聚什么、什么时候聚合、什么时候解聚、怎么聚合、怎么解聚"等。

对于多分辨率仿真建模的理论研究目前有视点选择法、

到稿日期:2011-06-02 返修日期:2011-08-04
赵小松(1977—),男,博士生,讲师,主要研究方向为系统仿真,E-mail:hustkent@sina.com;李国辉(1973—),男,博士,教授,博士生导师,主要研究方向为数据库理论、软件工程、仿真等。

聚合解聚法、多分辨率实体法等经典方法，但每种方法都存在若干问题。视点选择法使模型一直运行在高分辨率状态，过于单一；聚合解聚法方法简单，但频繁的聚合解聚造成的一致性问题难以解决；多分辨率实体法耗费太多的资源，可行性不高。而基于并行存储的多分辨率仿真通过将属性分配到不同分辨率层次模型存储从而避免了频繁的加聚解聚开销，在整体运行性能及资源和时间开销上都有所改观。下面分别介绍这几种方法。

2.2 经典的多分辨率建模方法

模型和仿真是密切相关但又不同的概念，在模型中是把一个系统作为研究对象，对该系统的某些特性进行抽象从而形成一个模型，而仿真便是通过对该模型的研究来达到研究系统的目的。对多分辨率仿真的研究主要就是对多分辨率建模方法的研究。多分辨率建模是多分辨率仿真研究中的重点内容之一，现在主要的方法有视点选择法、聚合解聚法、多分辨率实体法等。

2.2.1 视点选择法

视点选择法是多分辨率建模的传统方法。该方法更加关注事物的细节，从始至终一直运行在高分辨率模型中。当需要与低分辨率模型进行交互时，将高分辨率模型信息进行更高一级的抽象，生成相应的低分辨率模型信息，来和低分辨率模型交互。由于时刻运行于高分辨率，故该方法的信息准确度高，对真实系统的仿真程度高，思路也比较简单，在运行过程中模型不需要频繁变化，一致性也容易维护，但耗费资源和开销较大，缺乏灵活性，可重用性差。在作战这种信息资源异常珍贵的情况下使模型时刻运行于高分辨率也是没有必要的。

2.2.2 聚合解聚法

聚合是将高分辨率模型合并成低分辨率模型的过程，是对所需信息进行二次抽象的过程。解聚是聚合的逆过程，将低分辨率模型分解成高分辨率模型。因为对信息的抽象程度不同，通常不允许高分辨率模型直接与低分辨率模型进行交互，所以要使实体在同一层次上交互则需要模型的聚合和解聚过程。聚合解聚法[9]主要包括完全聚合解聚、部分聚合解聚、区域聚合解聚等。聚合解聚法表现直观，易于理解，也是多分辨率仿真中常用的方法，但是这种方法也存在许多问题，比如数据不一致性、链式解聚、交互冲突等。

2.2.3 多分辨率实体法

多分辨率实体法[10]的基本思想是建立一个能使高分辨率模型和低分辨率模型同时运行的实体。在该实体内，两种不同层次的模型并行运行，并保持数据一致性。由于两种不同层次的模型同时运行需涉及较多的属性，因此引入属性核的概念。属性核是一个属性集合，通过这个属性集合能即时生成所有分辨率下的所有属性，而且任何层次间的交互结果必须反映到模型的其他层次上。多分辨率实体法能较好地保持数据的一致性，但它需要极大的资源及通信开销，而且对于属性核的实现难度很大。

3 基于并行存储的多分辨率建模

以往对于多分辨率仿真的研究有了以上多种经典的方法，但还是有许多方面需要进一步研究。本文针对以上几种经典的建模方法在某些方面进行了改进，进而提出了一种新的方法，使实体可以在多种分辨率模型中转换而不需要频繁地聚合解聚，降低了加聚解聚开销，而且可以实现跨分辨率交互，还能够解决交互冲突问题。本文提出了如下并行存储的概念。

定义 1(并行存储) 根据属性的特性进行归类，以确定其存储模型的分辨率层次，使得实体的属性分别分布于高、低分辨率模型的存储策略。

并行存储就是根据使用情况将实体属性分别存储在高分辨率模型或者低分辨率模型中。高分辨率模型和低分辨率模型都不存储实体的所有属性。将在高分辨率模型中经常使用的属性存储在高分辨率模型，在低分辨率模型中经常使用的属性存储在低分辨率模型，两种层次的模型中的属性集构成互补的关系。比如一个飞机连队由 10 架飞机组成，每架飞机的攻击力不同，假如根据任务需要更关注每架飞机具体的攻击力，则将 10 架飞机的位置分别保存在 10 个高分辨率模型中，而低分辨率模型不保存攻击力信息，当需要低分辨率攻击力信息时，由高分辨率信息进行聚合。一个后勤保障组由控制物资承载能力、补给供给能力、维修保障能力的 3 个实体组成，而一般情况下我们更关注整个组的后勤保障能力，对具体的物资承载、补给供给、维修保障不感兴趣。这时将后勤保障能力属性存储在低分辨率模型中，在高分辨率模型中不存储具体的后勤信息，当需要高分辨率模型相关信息时，由低分辨率信息根据环境和参数进行解聚。

3.1 数据的并行存储策略

实体的属性分开存储在两个层次的模型中，为了便于交互，每个实体模型中留出部分存储单元，用于存储临时数据。临时数据是在需要跨分辨率交互时用来保存同一实体不同分辨率模型的相关信息的。

为了确定某属性应该存储的分辨率层次，需要构造一个函数，将属性的特征作为函数的参数，根据函数的输出来确定其存储的分辨率层次。假设构造的判断函数为 IsHResolution()，则对于属性集 R 中的任何一个属性 r_i，如果 IsHResolution(r_i)$\geqslant$1，则将该属性存储在高分辨率实体中，否则将其存储在低分辨率实体。

在高级体系结构(HLA，High Level Archicture)中，可将低分辨率对应的全部高分辨率实体模型作为一个联邦成员，低分辨率实体模型作为一个联邦成员，两个联邦成员通过联邦运行支撑环境(RTI，Run-time Infrastructure)进行交互。互联接口采用高分辨率模型接口方案，这样在 RTI 中传输的是低分辨率信息，可以有效减少 RTI 负载。设该实体有 3 个属性 r_1，r_2，r_3，其中 IsHResolution(r_1)$>$1，IsHResolution(r_2)$<$1，IsHResolution(r_3)$<$1。系统整体框架如图 1 所示。

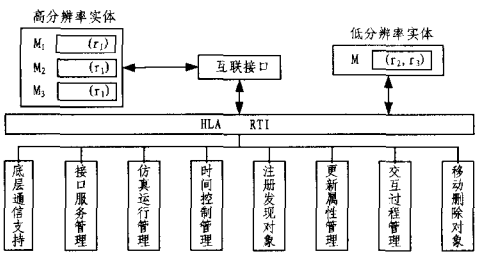

图 1 多分辨率系统架构图

3.2 交互方式

当需要交互的两个实体处于同一层次时，分辨率可以直

接交互，这时与加聚解聚法中的交互没有区别。当两个实体不在同一层次上时，临时存储单元便发挥了作用。

跨分辨率交互如图 2 所示。当低分辨率模型 Q 需要和高分辨率模型 A 进行交互来获取 A 中 a 属性时，首先和低分辨率模型 P 进行交互，告知 P 它需要 A 的 a 属性信息。P 模型获取到该请求信息后，向 A 发送请求 a 属性的信息。A 接受该信息并将 a 属性反馈给 P，P 将属性保存在空白存储单元 f 中。然后 P 将 f 单元中的信息与 Q 进行交互，Q 便得到了 A 中的属性信息。同理，当高分辨率模型 M 与低分辨率模型 P 进行交互时，先将该请求发送给 Q，Q 和 P 交互后，Q 再和 M 进行交互，M 便得到 P 的属性信息。

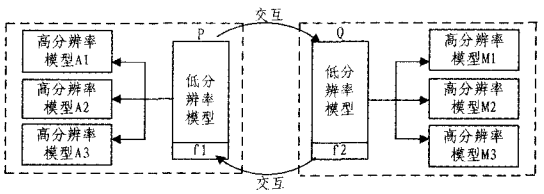

图 2　跨分辨率交互图

3.3　一致性问题

一致性问题是多分辨率仿真中最重要的问题之一，也是最难解决的问题之一。多分辨率实体法能较好地解决一致性问题，但它的开销注定了目前无法真正应用到实际的仿真中。广义的一致性包括模型和真实世界间的一致性、仿真模型和数学模型间的一致性、相同实体在不同抽象层次上即不同分辨率下的一致性。本文所指的一致性主要是指同一实体在不同分辨率下的一致性。

现在通常不能同时运行高分辨率模型和低分辨率模型，这样就会存在信息的聚合或者解聚。高分辨率模型中包含了更多的细节信息，从高分辨率模型到低分辨率模型的转换就必然存在信息的丢失。而低分辨率模型到高分辨率模型的解聚过程是根据概括性信息生成细节信息的过程，需要附加额外的数据、规则或条件。同时，不同分辨率的模型关于对象的侧重点并不相同，建模机理也可能不同。例如编队级作战模型大多采用兰彻斯特方程，而平台级作战模型大多采用概率论方法。不同分辨率层次的模型之间的完全一致性是很难保证的，我们要做的便是将偏差控制在条件允许的范围内。

一个由 3 个排实体 M_1，M_2，M_3 组成的机步连实体 M 与另一实体 N 作战，机步连实体的攻击力属性保存在高分辨率模型中。M 先解聚为 3 个排实体 M_1，M_2，M_3，经过交战，M_1 攻击力损伤 30%，其他两个实体完好。然后 3 个实体重新聚合为 M，则 M 攻击力损伤 10%。接着 M 与另一实体 T 作战，损伤 20%，交战结束。此时解聚只需将第二次损伤的 20%解聚到 3 个排实体中，因为第一次的损伤数据是记录在 3 个排实体中，聚合后排实体中的数据并没有删除。由于攻击力属性保存在高分辨率模型中，因此当高分辨率模型聚合后，低分辨率模型的攻击力属性值由高分辨率攻击力属性聚合而成，是一个虚拟值，并没有真正保存在模型中，仅仅当其发生变化时才将变化量反馈到高分辨率模型中。相比聚合解聚法，上述方法由于记录了第一次交战的信息，因此能更好地满足模型与现实世界之间的一致性。

传统的加聚解聚法和并行存储法的算法描述如下。

```
//as1:M1 开始时受到的损伤
//as2:M2 开始时受到的损伤
//as3:M3 开始时受到的损伤
//a:低分辨率实体受到的损伤
//a1:M 刚聚合后低分辨率实体受到的损伤
//a2:M 聚合为低分辨率实体后再次受到的损伤
//ae1:M1 所有交互结束时受到的损伤
//ae2:M2 所有交互结束时受到的损伤
//ae3:M3 所有交互结束时受到的损伤
// STATECHANGE():模型运行层次切换函数
```

具体算法描述如下：

```
聚合解聚方法中加聚算法(struct * p1,struct * p2,struct * p3)
{
  STATECHANGE(s,x,t);
  struct * p;
  p=(struct * )malloc(Number);
  if(! p)
    return false;
    p->a1= p1->as1+p2->as2+p3->as3;
  free(p1);
  free(p2);
  free(p3);
  while(p->a2! =NULL)
  p->a= p->a+p->a2;
  return p->a;
}
并行存储方法加聚算法(struct * p,struct * p1,struct * p2,struct *
p3)
{
  STATECHANGE(s,x,t);
  p->a1= p1->as1+p2->as2+p3->as3;
  while(p->a2! =NULL)
    p->a=p->a+p->a2;
  return p->a;
}
加聚解聚法解聚算法(struct * p)
{
  STATECHANGE(s,x,t);
    struct * p1;
    struct * p2;
    struct * p3;
    p1=(struct * )malloc(Number);
    p2=(struct * )malloc(Number);
    p3=(struct * )malloc(Number);
    while(a2! =NULL)
    p->a=p->a+p->a2;
    p1->as1=a/3;
    p2->as2=a/3;
    p3->as3=a/3;
    free(p);
    return (p1->as1,p2->as2,p3->as3);
}
并行存储法解聚算法(struct * p,struct * p1,struct * p2,struct * p3)
{
  STATECHANGE(s,x,t);
  double float m1=0;
  while(a2! =NULL)
    m1+=p->a2;
  p1->as1+=m1/3;
  p2->as2+=m1/3;
  p3->as3+=m1/3;
```

```
  return (p1->as1,p2->as2,p3->as3);
}
```

对于一个简化的 DEVS 模型 GM 描 G 述如下：

$M=\langle X,Y,R,s,S,\delta,T\rangle$

其中：

X：输入集合

Y：输出集合

R：属性集合

s：当前运行状态，$x\in X$

S：运行状态集合，$S=\{s1,s2\}$表示高分辨率，低分辨率两个层次

δ：状态转移函数，$\delta:S\times X\times T\rightarrow S$

T：时间函数

```
STATECHANGE(s,x,t)
{
  if(x∈X &&δ(s,x,t)符合状态转换条件)
    if s=s1
      then active(s2)
        s←δ(s1,x,t)
      R={r2,r3}
  else if s=s2
      then active(s1)
        s←δ(s2,x,t)
      R={r1}
    else return false
  else continue
}
```

3.4 并发冲突问题

并发交互是指同一实体的不同分辨率模型同时向其他实体发送交互信息。当该次交互会引起不同分辨率模型中同态属性变化时，就造成并发冲突。因为同态属性是不同分辨率下同一实体的同一内在属性的表示，而同一属性应该在同一时间范围内只由一个模型来影响，反馈到另外一个模型。

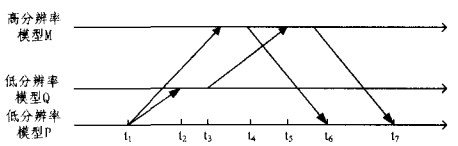

图 3　交互冲突示意图

在图 2 中，如果低分辨率模型 P 同时与 Q 和 M 交互，而且交互过程中会引起某属性的变化，如图 3 所示，t_1 时刻实体 P 需要 30 枚炮弹，P 与 Q 发生交互，P 同时与 M 发生交互；t_2 时刻 Q 接到 P 的请求，将 P 的请求解聚，分别让 M_1，M_2，M_3 各供给 10 枚；t_3 时刻 M 接到 P 的交互；t_4 时刻 M 供给 P30 枚炮弹；但 t_5 时刻 M 又收到 Q 解聚后的交互。而这并非真实情况，应该避免。

由于在并行存储多粒度仿真中同一个属性只存储于一个分辨率层次的模型中，因此不会造成同一实体不同分辨率层次中存储的同态属性的同时更新。但对于在高分辨率层次中存储的属性信息，仍然可能加聚后在低分辨率层次中得到更新，然后将更新反馈到高分辨率层次中去，此时需要加入验证机制。当反馈到高分辨率层次中时，验证高分辨率模型是否已经针对同性质的、同实体的交互发生了更新。如果发生了，就不再更新，否则便更新。

4　实例性能分析

下面以 3.3 节中的实例来比较并行存储的多分辨率仿真和最常用的加聚解聚法。

加聚解聚法中，属性的保存是与使用频率无关的。当模型运行于低分辨率时，属性就保存在低分辨率模型中。当模型运行于高分辨率时，属性就保存在高分辨率模型中。而并行存储方法是根据属性的使用特性将其分别保存在高或低分辨率模型中。当模型运行在高分辨率却需要低分辨率模型中保存的属性时，将该属性解聚反馈给高分辨率模型。当模型运行在低分辨率时，情况类似。

4.1 在一致性上进行比较

假设 t 时刻 M 和 N 交互。该过程中 M_1 受到 30% 的损伤，交互完毕后，M 和 N 进行聚合。此刻另一实体 H 需要 M_1 的攻击力信息，此时需要将 M 模型解聚，来得到高分辨率模型，然后将 M_1 的信息与 H 交互。然而 M 聚合时，M_1 受到 30% 的损伤加聚为 M 受到 10% 的损伤，M_1 受到损伤的信息已经丢失。在解聚时，M 受到 10% 的损伤将被理解为 3 个实体各自受到 10% 的损伤，H 得到的信息 M_1 受到 10% 的损伤，结果的误差很大。而且整个 M 模型的解聚也会造成交互延迟。并行存储方法中，攻击力信息保存在高分辨率实体中，t 时刻 M_1 受到的 30% 的损伤没有因为模型的聚合而丢失，即使当时 H 运行于低分辨率模型，H 向 M 请求 M_1 的信息，然后 M 将该请求传达给 M_1，M_1 将该信息反馈到 M 中的 f 里，M 将该信息交互给 H，并没有将整个 M 解聚，大大节省了时间。其过程如图 4 所示。

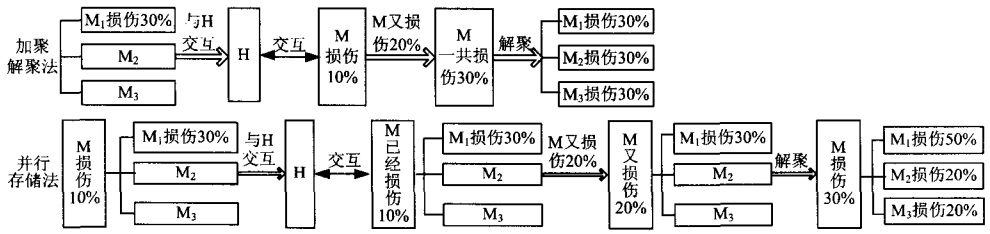

图 4　加聚解聚法与并行存储法结果对比

为了比较模型在一致性上的表现，通常采用一致度的概念进行量化分析。本次分析采用 Hamming　I/O 一致度方法[1]。

设 M，M' 是两个不同分辨率层次的模型，其中：

$M=\langle X,Y,\Omega,\lambda,T\rangle$

$M'=\langle X',Y',\Omega',\lambda',T'\rangle$

X 为模型的输入，Y 为模型的输出，Ω 为该分辨率层次下的输入集合，λ 为输入到输出的映射，T 为时间的范围。设

$Y=(y_1,y_2,y_3,\cdots,y_n)$

$Y'=(y_1',y_2',y_3',\cdots,y_n')$

不同分辨率模型 M 与 M' 的 I/O 一致度定义如下：

$$C(M,M')=1-\frac{2}{n}\sum_{i=1}^{i=n}\left|\frac{y_i-y_i'}{y_i+y_i'}\right|$$

本次分析比较加聚解聚法和现实数据的一致度与并行存

储法和现实数据的一致度，即将真实世界作为模型 M'，计算 M 与 M' 之间的一致度。下面分别采用多组数据并比较一致度的结果，如表 1 所列。

表 1　加聚解聚法与并行存储法输出结果对比表

序号	高分辨率实体	运行于高分辨率受到损伤	运行于低分辨率受到损伤	真实损伤	加聚解聚法输出结果	并行存储法输出结果	加聚解聚法一致度	并行存储法一致度
	M_1	10%	15%	25%	25%	25%		
1	M_2	10%	15%	25%	25%	25%	100%	100%
	M_3	10%	15%	25%	25%	25%		
	M_1	20%	30%	50%	30%	40%		
2	M_2	5%	20%	25%	30%	25%	55.3%	75.9%
	M_3	5%	10%	15%	30%	25%		
	M_1	50%	15%	55%	30%	60%		
3	M_2	5%	10%	15%	30%	15%	24.9%	83.8%
	M_3	5%	5%	10%	30%	15%		

并行存储法记录了高分辨率交互信息，因此能表现出与现实世界之间更高的一致性。当运行高分辨率层次时，各实体受到的损伤区别很大；运行于低分辨率层次时，受到的损伤区别较小，并行存储法的性能表现最优。图 5 表示了经历过加聚解聚过程的模型的一致度对比，紫线为并行存储方法，黄线为加聚解聚方法。

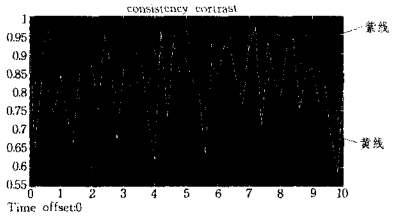

图 5　一致度分析结果对比图

在本次试验中，随机产生一个 3×3 矩阵，矩阵第一列为模型运行于高分辨率层次时高分辨率的 3 个实体分别受到的损伤，第二列为模型运行于低分辨率层次时 3 个实体分别受到的损伤，第三列为模型运行于高分辨率层次时 3 个实体分别受到的损伤。经过一次聚合与解聚，来对两个建模方法的仿真度进行比较，对比结果如表 1 所列。

4.2　在交互冲突上进行比较

加聚解聚法常用的解决交互冲突的方法为加锁机制。当 N 与 M 进行交互时，先将 M_1，M_2，M_3 中攻击能力的同态属性进行加锁，交互完成后才能解锁。此时首先需要将 M 解聚，然后计算 M_1，M_2，M_3 中哪些属性是攻击能力的同态属性，进而加锁，过程很清晰。解聚后还要重新回到低分辨率模型，一次加聚解聚和对同态属性的验证需要较长时间。并行存储方法采用验证机制，避免了重复更新。当 N 与 M_1 中的攻击能力交互时，更改 M_1 中的攻击能力，同时更改 M 中对 M_1，M_2，M_3 的攻击能力聚合后的信息。由于对 M 中该信息更改只是更改的虚拟值，因此当将该虚拟值的变化反馈到高分辨率层次中时才进行验证。如果需要更新便更新，否则弃掉此次变化并通知 N，这在很大程度上节省了解决冲突的时间。

结束语　在多分辨率仿真领域，数据的一致性问题一直是难以解决的问题，而如何能更高效地解决并发冲突问题，也作为一个课题被提出来。本文在已有成果的基础上提出了基于并行存储的多分辨率仿真模型，并提出了新的解决思路。当前，基于并行存储的多分辨率仿真也面临一些挑战。如何根据属性的特性来判断是否存储在高分辨率仿真中，这个标准不易实施。为了将高分辨率模型和低分辨率模型中的数据独立地存储，每个模型需要两个独立的存储器，如何处理两个存储器之间的联系，也有待进一步探索。

参 考 文 献

[1] 刘宝宏，黄柯棣. 多分辨率模型系中的一致性问题研究[J]. 系统仿真学报，2005，17(9)：2057-2059

[2] 尹华飞，吕品，等. 多分辨率仿真模型互联运行问题研究[J]. 中国体视学与图像分析，2010，15(3)：269-273

[3] 周华任，马亚平，等. 战争模拟多分辨率建模研究[J]. 系统仿真学报，2009，21(21)：6833-6836

[4] 朱松岩，江敬灼，等. 聚合解聚及其一致性问题研究[J]. 军事运筹与系统工程，2008，22(3)：33-38

[5] 刘宝宏. 多分辨率建模的理论与关键技术研究[D]. 长沙：国防科学技术大学，2003

[6] 周彦，戴剑伟. HLA 仿真程序设计[M]. 北京：电子工业出版社，2002，6

[7] Drewry D T, Reynolds P F, Emanuel W R. Optimization as a Tool for Consistency Maintenance in Multi-resolution Simulation[R]. University of Virginia, 2002

[8] Franceschini R W. Computational for Disaggregation[C]//Paper of the 9th CGF &BR Conference. 2000

[9] Natrajan A. Consistency Maintenance in Concurrent Representation [D]. USA: School of Engineering and Applied Science, The Univ. of Virginia, 2000

(上接第 127 页)

[2] 汪浩，谢军凯，高仲仪. 遗传算法及其在软件测试数据生成中的应用研究[J]. 计算机工程与应用，2001，15(12)：64-68

[3] 董敏，毕盛，齐德昱. 基于正则表达式的测试数据自动生成技术[J]. 计算机工程，2009，35(16)：29-31

[4] 王捷民，等. 基于改进的自适应遗传算法 HCGA 的测试数据自动生成[J]. 北京理工大学学报，2007，27(10)：883-885

[5] Mansour N, Salame-Jaffa M. Data Generation for Path Testing [J]. Software Quality Journal, 2004, 12: 121-136

[6] Miller J, Reformat M, Zhang H. Automatic test data generation using genetic algorithm and program dependence graphs [J]. Information and Software Technology, 2006, 48: 586-605

[7] Korel B. Automated software test data generation[J]. IEEE Trans on Software Engineering, 1990, 16(8): 870-879

[8] Wegener J, Baresel A, Sthamer H. Evolutionary test environment for automatic structural testing[J]. Information and Software Technology, 2001, 43(14): 841-854

[9] 陈继锋，朱利，沈钧毅，等. 一种基于分支覆盖的测试数据自动生成算法[J]. 计算机科学，2006，33(12)：261-264

[10] 金虎，李志蜀，张磊，等. 基于面向路径的遗传算法的测试用例自动生成[J]. 计算机工程，2007，33(3)：21-23

[11] 巩敦卫，张岩. 一种新的多路进覆盖测试数据进化生成方法[J]. 电子学报，2010，38(6)：15-16

[12] 白凯，崔冬华. 基于遗传算法的测试数据自动生成[J]. 计算机与数字工程，2010，38(2)：52-54