基于 OODA 环的分布式作战仿真时间管理算法*

马 也[1,2],常天庆[1],范文慧[2],李 婕[3],孔德鹏[4]
(1.陆军装甲兵学院 兵器与控制系,北京 100072;2.清华大学 自动化系,北京 100084;
3.解放军总医院京南医疗区 杜家坎门诊部,北京 100072;4.92942 部队,北京 100161)

摘 要:在信息化战争时代,OODA 指挥控制环在军事仿真中的应用日益广泛,分布式仿真的灵活性及高效的并行性对作战仿真的发展提供了便利。面对作战仿真的需求,对分布式仿真系统的核心技术——时间管理算法进行研究。针对 OODA 指挥控制环的特点,提出了一种自适应移动时间窗算法。算法根据模型自适应调整时间窗大小,可避免频繁同步并对内存进行精准清除。为进一步优化时间管理算法,提出了一种改进全局虚拟时间算法,对全局虚拟时间同步计算中的阻塞过程进行改善。详细分析该算法的步骤及原理,并通过实例进行验证。结果表明,该算法可有效保证分布式仿真的运行,运行速度优于其他传统时间管理算法。

关键词:OODA 环;时间管理;移动时间窗;全局虚拟时间
中图分类号:TP391 **文献标识码**:A DOI:10.16157/j.issn.0258-7998.211344

中文引用格式:马也,常天庆,范文慧,等. 基于 OODA 环的分布式作战仿真时间管理算法[J].电子技术应用,2021,47(5):86-91.
英文引用格式:Ma Ye,Chang Tianqing,Fan Wenhui,et al. Time management algorithm of distributed combat simulation based on OODA loop[J]. Application of Electronic Technique,2021,47(5):86-91.

Time management algorithm of distributed combat simulation based on OODA loop

Ma Ye[1,2],Chang Tianqing[1],Fan Wenhui[2],Li Jie[3],Kong Depeng[4]
(1.Department of Weapons and Control, Academy of Army Armored Force,Beijing 100072,China;
2.Department of Automation,Tsinghua University,Beijing 100084,China;
3.Jingnan Medical District Dujiakan Clinic,PLA General Hospital,Beijing 100072,China;
4.Unit of 92942,Beijing 100161,China)

Abstract: In the information-based war era, OODA command and control loop is widely used in military simulation. The flexibility and efficient parallelism of distributed simulation promote the development of combat simulation. According to the requirement of combat simulation, this paper studies the core technology of distributed simulation system, which is time management algorithm. Based on the characteristics of OODA command and control loop, this paper proposes an adaptive moving time window algorithm. The algorithm adaptively adjusts the size of time window according to the model, which can avoid frequent synchronization and clear the memory accurately. In order to further optimize the time management algorithm, the author also proposes an improved global virtual time algorithm to improve the blocking process in global virtual time synchronization calculation. Through detailed analysis of the steps and principles of the algorithm, and verification by an example, the results show that the algorithm can effectively guarantee the running of distributed simulation, and the running speed is better than other traditional time management algorithms.

Key words: OODA loop;time management;moving time window;global virtual time

0 引言

OODA 环理论是由 BOYD J 提出的军事作战中的战略决策理论[1],成为对作战中出现的冲突问题进行描述的一种科学方法。该理论一经提出,便立即被整个西方世界的军事界所认可,至今仍被广泛应用于军事作战仿真问题的研究[2]。众多学者对其进行研究并提出模块化 OODA 环、智能 OODA 环及认知 OODA 环等衍生理论[3-5]。在信息化战争时代,对多国联合军事演习、协同军事打击以及与其他军兵种部队的共同作战需求的增加,使得普通仿真难以应对[6-7]。分布式仿真因其更高的性能及灵活性越来越多地被军事作战仿真所采用。分布式仿真的技术难点是如何确保系统的因果逻辑关系,正确及可重复地在并行系统上高效合理地处理事件并推进仿真时间。时间管理技术是分布式仿真的一项重点技术,用

*基金项目:国家重点研发计划(2017YFB1400105)

于解决分布式仿真的时间推进问题，它决定了分布式仿真系统的整体性能，被该领域的研究者广泛关注[8-9]。

典型的时间管理算法有保守算法及乐观算法两种。保守算法虽可以严格保证系统因果约束，正确地进行时间推进。但其依赖前瞻量和空消息机制，给系统带来无法避免的死锁问题，严重影响系统性能。乐观算法允许违反系统因果约束，通过回退机制来解决因果关系错误。虽能够积极地处理事件并避免保守算法带来的死锁问题，但回退机制会占用系统内存，过度的乐观又会造成多级回退现象，从而不利于系统的并行性并使性能大幅下降。后续的研究者针对以上问题提出了一些解决方法，大多存在系统实现复杂、在牺牲系统性能的前提上保证因果约束以及需要改变模型来获取计算性能等缺点。

本文综合考虑保守及乐观算法的优缺点，针对 OODA 环作战仿真的特点，提出一种基于改进 GVT 的自适应时间窗混合推进时间管理算法并对全局虚拟时间算法的计算进行改进。算法能够改善传统时间窗算法频繁同步问题，同时避免出现传统时间管理算法的死锁及多级回退等缺点。

1 OODA 环作战模型的构建

BOYD J 提出的 OODA 环理论指出，军事行动基本上是由一系列决策过程或循环组成的。基础 OODA 环主要包括侦察(Observation)、判断(Orientation)、决策(Decision)和行动(Action)四部分[10-11]。具有指挥控制特点的军事作战 OODA 环按照上述四部分进行循环的战斗，在一个循环结束后便开始新的循环，如此往复。循环周期的时间长短与作战单位的兵力规模、作战环境的范围以及作战的方式相关。

1.1 OODA 环作战因果关系

在作战过程中，己方和敌方将持续进行战场观察获取作战信息，对信息进行判断和分析后进行决策并采取相应的行动。其对抗冲突主要体现在 OODA 的周期长短上，优先采取行动的一方将获取主动权和优势，双方都将尽可能地缩短自己的循环周期。OODA 环作战双方交战因果关系如图 1 所示。

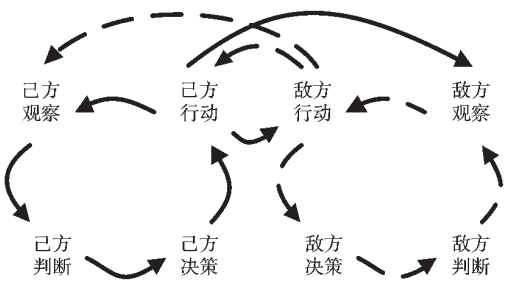

图 1 OODA 环作战双方交战因果关系

1.2 OODA 环作战模型

模型采用美国海军分析中心(Center for Naval Analyses，CAN)提出的经典 Agent 作战模型 EINSTein 的框架[12]，在该框架的基础上构建一个基于多 Agent 的作战仿真模型，该模型用于测试分布式仿真时间管理算法的有效性。模型中每个 Agent 代表一个作战兵力，它有 3 种状态，分别为存活、受伤和死亡。每个 Agent 赋予一个特征权值向量：

$$\vec{\omega}=(\omega_{\mathrm{AF}},\omega_{\mathrm{AE}},\omega_{\mathrm{IF}},\omega_{\mathrm{IE}}) \tag{1}$$

式中，$-1\leqslant\omega_x\leqslant 1$，$\sum|\omega_x|=1$。$x$ 的取值为{x|AF，AE，IF，IE}。每个权值向量由 4 个分量组成，分别为己方存活权值分量(ω_{AF})、敌方存活权值分量(ω_{AE})、己方受伤权值分量(ω_{IF})、和敌方受伤权值分量(ω_{IE})。Agent 将根据各自特征权值向量的值决定自己在所处战场环境中的移动策略。当分量值为正时，代表该 Agent 有靠近分量给定特征的 Agent 的意愿；当分量值为负时，则为撤退意愿。Agent 所有可能进行的移动策略将由惩罚函数 z 来排序，通过计算惩罚函数 z 的最小值确定 Agent 具体的移动方向。惩罚函数 z 定义为：

$$z(B_{xy})=\frac{1}{\sqrt{2r_{\mathrm{F,S}}}}\left[\frac{\omega_{\mathrm{AF}}}{N_{\mathrm{AF}}}\sum_{i\in\mathrm{AF}}^{N_{\mathrm{AF}}}D_{i,B_{xy}}+\frac{\omega_{\mathrm{AE}}}{N_{\mathrm{AE}}}\sum_{j\in\mathrm{AE}}^{N_{\mathrm{AE}}}D_{j,B_{xy}}\right]+\frac{1}{\sqrt{2r_{\mathrm{E,S}}}}\left[\frac{\omega_{\mathrm{IF}}}{N_{\mathrm{IF}}}\sum_{k\in\mathrm{IF}}^{N_{\mathrm{IF}}}D_{k,B_{xy}}+\frac{\omega_{\mathrm{AE}}}{N_{\mathrm{AE}}}\sum_{l\in\mathrm{IE}}^{N_{\mathrm{IE}}}D_{l,B_{xy}}\right] \tag{2}$$

其中，B_{xy} 是 Agent 所处环境的坐标位置；AF、IF、AE、IE、Nx 分别定义为当前 Agent 感知范围内的己方存活的 Agent 集合、己方受伤的 Agent 集合、敌方存活的 Agent 集合、敌方受伤的 Agent 集合以及 x 类型 Agent 的总数；$1/\sqrt{2r_{\mathrm{F,S}}}$ 为己方比例因子；$1/\sqrt{2r_{\mathrm{E,S}}}$ 为敌方比例因子；$D_{X,Y}$ 为 X 到 Y 的距离；惩罚函数 z 可以有效权衡 Agent 与其他敌我双方在内的 Agent 的距离，其下标与 Agent 特征权值向量下标相同。通过惩罚函数，Agent 将移动到惩罚最低的方向，该方向能最大程度地满足其特征权值向量的大小所期望的方向。

惩罚函数同时也受 μ 规则的限制作出部分改变，Agent 将根据环境的动态变化改变个性权值。μ 规则如下式所示：

$$\begin{cases}N_{\mathrm{F}}(r_{\mathrm{F}})-N_{\mathrm{E}}(r_{\mathrm{E}})\geqslant\Delta\rightarrow w_{\mathrm{AE(default)}},w_{\mathrm{IE(default)}}\\N_{\mathrm{F}}(r_{\mathrm{F}})-N_{\mathrm{E}}(r_{\mathrm{E}})<\Delta\rightarrow-|w_{\mathrm{AE}}|,-|w_{\mathrm{IE}}|\end{cases} \tag{3}$$

其中，$N_{\mathrm{F}}(r_{\mathrm{F}})$为在己方观察范围内己方的 Agent 数量；$N_{\mathrm{E}}(r_{\mathrm{E}})$为敌方观察范围内敌方的 Agent 数量；$\Delta$ 为战斗阈值数量。μ 规则可描述为，当 Agent 的观察范围内存在的己方数量比敌方数量具有不到战斗阈值的优势，那么该 Agent 将远离敌方 Agent。

模型中的战场地形为离散的二维网格，初始状态下，敌方和我方以中轴线为轴，随机分布在其所属的地形中。Agent 初始默认状态为存活状态，当其被攻击命中后，Agent 的状态将降级为受伤状态，若再次受到攻击，将降级为死亡状态，死亡状态的 Agent 在仿真结束前不会再出现。处于受伤状态的 Agent 的作战命中率为存活状态的命中率的一半。每个 Agent 又包括观察范围、移动范围

和火力范围的3种距离范围，用于进行作战OODA环的执行。Agent的攻击目标为其火力范围内所有的敌方Agent，每次随机从中挑选一个进行攻击。

OODA环作战模型作战流程如图2所示。

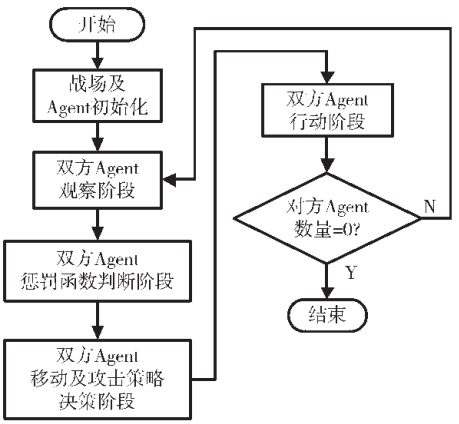

图2 OODA环作战模型流程图

首先对战场及作战兵力进行初始化，随后进入OODA环循环过程。每个Agent根据观察范围观察战场情况；随后通过惩罚函数z对接收到的战场情况进行判断，计算移动范围内所有的移动策略并进行排序；当判断阶段结束，进入决策阶段。根据惩罚函数z的最小值确定Agent移动策略并选择火力范围内的攻击目标。若无攻击目标，则不进行选择；最后，根据决策内容进行相应的移动和攻击行动；行动阶段结束后对敌方Agent数量进行统计，若还存在敌方Agent，则进入下一轮OODA循环，否则结束作战。

2 基于改进GVT的自适应移动时间窗算法

2.1 自适应移动时间窗算法

因分布式仿真系统的特点，需要使用时间同步技术(即时间管理算法)对系统时间进行同步，保证仿真的时间推进过程正确高效[13]。保守算法虽能够严格确保时间推进正确，但其效率较低，不能适应分布式作战仿真的需求，因此仿真将采用乐观算法。

移动时间窗算法(Moving Time Window，MTW)算法是对乐观算法的乐观推进程度进行控制的先进乐观时间管理算法[14-15]。它能够控制乐观执行程度的算法，设置可以超过其他逻辑进程仿真时间的上限，通过上限对时间推进程度进行控制，该算法需要计算全局虚拟时间(Global Virtual time，GVT)。对于乐观算法，当出现迟到消息时，会进行回退操作保证消息处理按照时间戳有序进行。GVT计算就是得到目前仿真过程中未来可能出现回退操作的时间戳下限。

移动时间窗算法的思想主要是对某个仿真进程的仿真时间能够超过其他仿真进程的仿真时间的上限进行设置，在尚未达到上限时，仿真可乐观地推进，一旦有进程达到上限便强制阻塞，直到其他所有落后的进程都推进到上限。普通移动时间窗的下限设置为GVT，上限设置为GVT+W，W为自行设定的时间窗值。所有进程不得超过上限，随着GVT的推进而移动推进。它操作简单，可有效解决个别进程推进过快而导致的回退现象，但存在时间窗在程序运行过程中无法改变、频繁进行GVT同步计算的问题。

本文提出的自适应移动时间窗算法不以仿真时间来定义时间窗，而采用事件。由于在仿真执行的过程中，通常无法准确地推算处理每个事件的时间，结合基于OODA环的作战仿真的特点，它具有明确的作战周期及事件，因此，算法将结合OODA环的周期采用周期循环方式进行仿真推进，自行调整时间窗大小。仿真模型中每个Agent都有一个单独的进程，每个进程的推进周期可以表述为：

(1)将一个推进周期定义为OODA环中四个部分中的一个部分，每个部分有一个待处理事件队列。每个推进周期产生的所有未处理的事件和其时间戳存放在所述部分的事件队列中。同时，事件队列中还包含在每个部分仿真开始和结束阶段发送的当前部分开始和结束的标志事件及相应的时间戳。

(2)当收到OODA环下一部分开始标志事件时，停止当前进程的推进。

(3)将下一部分开始标志事件和大于下一部分开始标志事件时间戳的事件从当前待处理事件队列移至下一部分待处理事件队列中。按照时间戳对当前部分的待处理事件队列中的事件进行排序。

(4)开始当前逻辑进程OODA环的下一轮推进周期。

对推进周期进行上述设定，可保证各个推进周期之间不会产生延迟消息造成作战仿真因果关系错误从而引发进程回退。自适应移动时间窗算法对事件处理进行了规定，所有进程推进的虚拟时间小于GVT的待处理事件队列中的事件可以移动至处理事件队列中进行乐观的处理，其他待处理队列中的事件不允许进行处理；所有事件及事件处理产生的过程结果不进行删除，保证回退操作能够顺利进行；当某一进程收到OODA环一下部分开始标志事件时，将对其他进程发送消息；若其他所有进程都已收到，将启动GVT计算，对系统进行时间同步。

算法的计算步骤如下：

(1)仿真中的所有进程按照乐观的方式同步推进，更新各自的推进周期，若有延迟消息，则进行回退；

(2)当某一进程收到下一推进周期开始标志事件时，对所有进程发送消息并回到步骤(1)，若所有进程都收到下一推进周期开始标志事件，进行GVT计算，同步系统时间；

(3)对上一推进周期的事件队列及过程结果进行删除，清理内存；

(4)所有仿真进程回到步骤(1)，开始下一轮仿真。

算法伪代码如下：

```
while (1){
  if one of the LP receive the start flag event about next advancing cycle then
    if all the LP receive the start flag event about next advancing cycle then
      calculate GVT;
      delete the event queue and process result of the last advancing cycle;
  break;
    else
      break;
else
    break;
}
```

2.2 改进 GVT 算法

常规 GVT 计算分为同步法和异步法，同步法在发送开始 GVT 计算的消息后，会等待所有处理器返回接收消息才会开始计算，同时暂停所有处理器正在进行的事件处理，会造成严重的阻塞。异步法不存在阻塞问题，允许在计算 GVT 的同时继续处理事件。两种算法都容易产生暂态消息问题。

为了避免在接到 GVT 计算消息后还存在已经发送但尚未接收的暂态消息未被统计而导致的 GVT 计算错误问题。较为经典的异步 GVT 算法为 Mattern GVT 算法，该算法通过对分布式仿真的每个处理器设置“切分点(Cut Point)”来计算 GVT。如图 3 所示，CP1 和 CP2 为两个切分点。在 CP1 和 CP2 之间为灰色区域，由白色区域发送的消息为白色消息，灰色区域发送的消息为灰色区域。算法将系统中所有的处理器按照逻辑排列成环结构，由系统中的控制器沿环上的顺序发送 GVT 计算消息得到各自处理器，处理器接收该消息的时刻设置为 CP1。CP2 的设置是为了将所有暂态消息都包括在 GVT 计算中。当所有处理的 CP1 确认后，系统控制器将再次沿着环在各个处理器上确定 CP2，当发送给当前处理器的白色消息和当前处理接收的白色消息的数值相等时，设置当前时刻为 CP2。在 CP1、CP2 确认完毕后计算所有灰色消息及 CP2 之后未处理消息的最小时间戳值，该值即为 GVT 值。

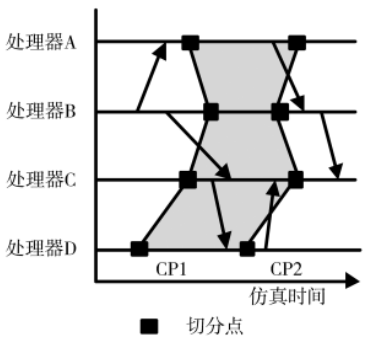

图 3 Mattern GVT 算法切点示意图

该算法存在需要控制器对处理器构建所有处理器的环结构，并逐一确定 CP1、CP2 两个切点的值。同时，Mattern GVT 算法还需要区分不同消息的颜色，并统计 CP2 之后所有未处理消息的时间戳值，计算量较大。根据上述两个缺点，对 Mattern GVT 算法进行了两点改进。

(1)本文提出的自适应移动时间窗算法中，CP1 的确定由算法的步骤(2)自动确定，并不需要控制器沿环逐一计算。

(2)对 GVT 的计算方法进行改进，不进行消息颜色判断及 CP2 之后所有未处理消息的最小时间戳统计。当所有处理器的两个切点确定后，若下一个事件的时间戳值大于 CP1 时，则 GVT 设置为所有进程中最小的本地虚拟时间。

通过对 Mattern GVT 算法的改进，可避免控制器对处理器构造环结构，能够自发地进行切点确认及 GVT 计算。同时，无需进行消息颜色判断及统计所有未处理事件的时间戳造成的计算冗余，防止计算阻塞，有效优化 GVT 计算的复杂度。算法伪代码如下：

```
while(1){
  if receive calculation GVT message then
  CP1←localt;
While ( the number of white message sent = the number of white message received) {
      CP2←localt ;
}
Loop : if the timestamp of the next event is later than CP1 then
          GVT←min{locate};
      else
          goto Loop;
}
```

2.3 分布式作战仿真运行机制

仿真将采用基于改进 GVT 的自适应移动时间窗算法进行推进，其运行机制如图 4 所示。

仿真中由时间管理控制 Agent 对时间推进线程的后

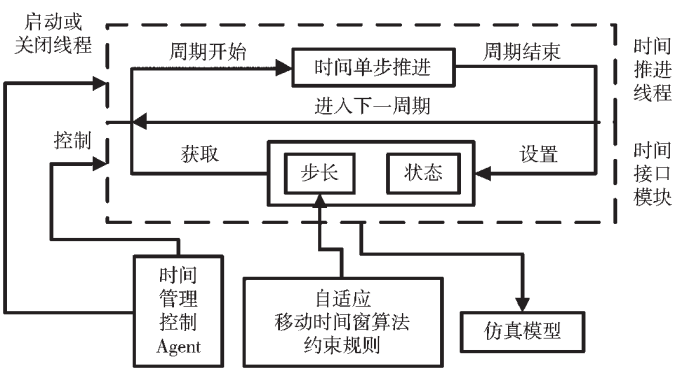

图 4 分布式作战仿真的运行机制

动和关闭进行管理和控制;时间管理算法对仿真的步长进行约束;仿真的状态由不同的线程进行初始化和设置;最终时间管理服务将作用在仿真模型中,保证分布式仿真在运行的过程中,事件因果逻辑及消息收发正确且高效。

3 算法分析

3.1 时间异常问题

对于分布式作战仿真,尤其重视仿真可能出现的时间异常问题。主要的时间异常有内部延时异常和信息延迟异常两种。

两种时间异常都会带来严重的因果关系错误问题,本文提出的算法可有效防止该问题的出现,如图 5 所示。算法将所有事件存放在 OODA 四部分的事件队列中,在上一部分处理完毕后,会触发 GVT 计算,同时对待处理事件队列进行排序,避免出现时间异常及回退。

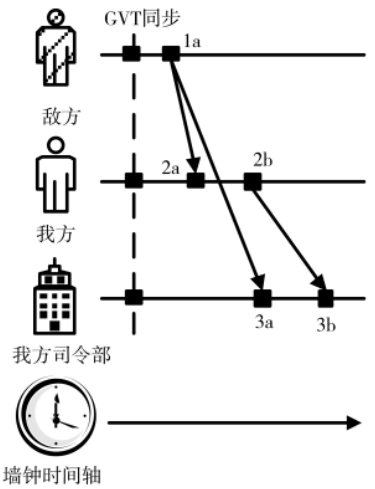

图 5 本文算法对时间异常的修复示意图

3.2 内存过载问题

乐观算法使用时间弯曲逻辑进程(TWLP)并构成时间弯曲系统,TWLP 与串行仿真的区别在于,当前逻辑进程(LP)的事件可以是其他 LP 调度产生的,同时,处理完毕的事件不进行立即删除操作,会存在暂时队列中。经典乐观算法能够对保守算法的效率进行改善,但会随着仿真的推进带来内存占用过量及多级回退问题。

本文提出的算法只需要保存上一部分的事件处理数据,在 GVT 计算过后,将对之前所有处理事件产生的数据进行清除,释放内存,开始新一轮推进周期,有效解决乐观算法带来的内存过载问题。

4 算法分析

对作战仿真模型的参数进行设置用于测试算法性能。模型的战场由二维离散点构成,战场大小为 100×100。战场中分布两种作战 Agent,分为红军和蓝军,每个 Agent 代表一个兵力,数量各 100 个。当 Agent 死亡,在本轮仿真中将永久退出战场。双方观察范围均为 15,运动范围为 1,火力范围为 5,战斗阈值为 1,特征权值为$\vec{w}$=(0.25,0.25,0.25,0.25)。仿真运行 500 步并重复实验 100 次。仿真基于分布式 Agent 开发技术框架 JADE 平台,用 Java 语言编写。每个Agent 为一个单独进程,并被划分在 3 个 JADE 容器中,模拟 3 台计算机进行分布式仿真。对 100 次仿真结果进行平均,得到的存活 Agent 数量趋势如图 6 所示。

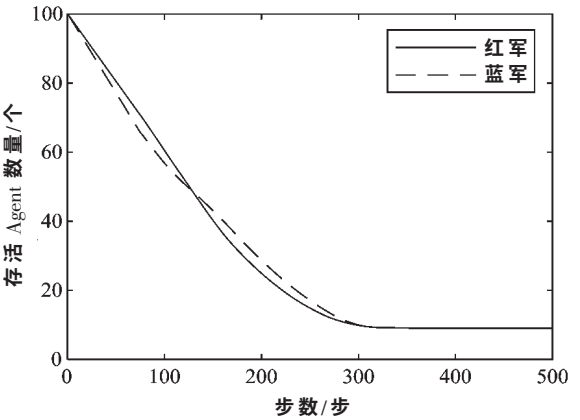

图 6 仿真过程中存活 Agent 数量趋势图

由于红蓝军设置的参数基本相同,其存活 Agent 数量趋势大致相同,在仿真运行至 300 步时人数趋于稳定,双方不再交战。

分别使用 4 种时间管理算法对模型进行仿真测试,4 种算法分别为本文提出的算法、普通移动时间窗算法、乐观算法及保守算法,实验结果如图 7 所示。

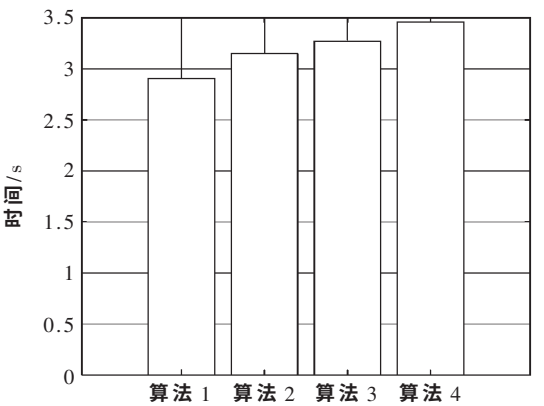

图 7 4 种算法实验结果对比图

本文提出的自适应时间窗管理算法在基于 OODA 环的仿真中运行效果最好,所需时间最短。该算法与其他 3 种算法相比运行时间分别提高了 7%、11%和 16%。

5 结论

并行分布式仿真能够提高仿真的运行效率,其重难点在于时间管理算法的性能。针对 OODA 环作战仿真的特点,本文提出一种自适应移动时间窗算法,仿真结果优于其他经典时间管理算法。算法无需修改模型内部结构,紧贴模型逻辑流程,自适应调整时间窗大小;对推进

周期进行设定，避免并行仿真的多级回退及死锁问题，同时可定期精准清除内存；改进 GVT 计算算法，无需频繁进行 GVT 同步并能自发确定切点，改善 GVT 计算复杂度。算法可为并行作战仿真的研究提供支持，对未来大规模集群仿真作战提供新的思路和参考。

参考文献

[1] OSINGA F.Science，strategy and war：the strategic theory of John Boyd[M].Netherlands：Eburon Academic Publishers，2005.
[2] 朱江，蔡蔚，闻传花，等.基于 OODA 指挥控制环的作战仿真实验[J].指挥控制与仿真，2015(3)：112-115.
[3] 邹立岩，张明智，柏俊汝.一种基于 OODA-L 的智能无人集群作战仿真建模框架[C].2019 中国系统仿真与虚拟现实技术高层论坛，2019：238-245.
[4] Pei Yun，Hou Tiedan，Yang Qingshan，et al.Analysis of fighter agility in beyond-visual-range air combat[J].Air & Space Defense，2019，2(1)：42-46.
[5] VETTORELLO M，EISENBART B，RASNCOMBE C.Toward better design-related decision making：a proposal of an advanced OODA loop[J].Proceedings of the Design Society International Conference on Engineering Design，2019，1(1)：2387-2396.
[6] 高昂，段莉，张国辉，等.计算机生成兵力行为建模发展现状[J].计算机工程与应用，2019，55(19)：43-51.
[7] 柴磊，王智学，何明.指挥控制协同能力需求体系形式化框架[J].计算机工程与应用，2020，56(5)：49-56.
[8] BASINGAB M，NAGADI K，RAHAL A，et al.Distributed simulation using agents for the Internet of Things and the factory of the future[J].Information(Switzerland)，2020，11(10)：458.
[9] MELNIK E V，OSTROUKHOV A Y，PUKHA I S，et al.The software structure for agent-oriented simulation with distributed dispatching[J].Journal of Physics：Conference Series，2020，1661(1)：012180.
[10] 周丰.指挥控制系统模型的分析与扩展[C].北京：第二届中国指挥控制大会论文集，2014：112-114.
[11] 张志祥，高春荣，郭福亮.人工战争：基于多 AGENT 的作战仿真[M].北京：电子工业出版社，2010.
[12] ILACHINSKI A.Artificial war：multiagent-based simulation of combat[M].World Scientific，2004.
[13] 陈西选，徐珞，曲凯，等.仿真体系结构发展现状与趋势研究[J].计算机工程与应用，2014，50(9)：32-36.
[14] TOCCI T，PELLEGRINI A，QUAGLIA F，et al.ORCHESTRA：an asynchronous wait-free distributed GVT algorithm[C].IEEE/ACM International Symposium on Distributed Simulation & Real Time Applications.ACM，2017.
[15] LI T T，HAN L，WANG J Y.Research and application of time management in parallel and distributed real-time simulations[J].Journal of System Simulation，2013，25(8)：1783-1788.

(收稿日期：2021-01-28)

作者简介：

马也(1993-)，通信作者，女，博士研究生，主要研究方向：信息感知与控制技术，E-mail：mayegf@126.com。

常天庆(1963-)，男，教授，博士生导师，主要研究方向：信息感知与控制技术。

范文慧(1968-)，男，教授，博士生导师，主要研究方向：复杂系统建模与仿真。

(上接第 85 页)

感信息，2006(5)：44-47.
[2] RICHARDSON L，RUBY S.RESTful Web Services[M].O′reilly Media Inc，2007.
[3] 李大鹏.基于 REST 风格 Web 服务的研究[J].电子商务，2010(4)：63-63.
[4] RATHOD D M，DAHIYA M S，PARIKH S M.Towards composition of RESTful web services[C].International Conference on Computing.IEEE，2016.
[5] 潘金亚.基于 Spring 的 REST 式 Web 服务研究与应用[D].西安：西安电子科技大学，2013.
[6] MAK G.Spring MVC framework[M].Spring Recipes，2008.
[7] 王丹，孙晓宇，杨路斌，等.基于 SpringBoot 的软件统计分析系统设计与实现[J].软件工程，2019，22(3)：40-42.
[8] 陈平.基于 Spring 的轻量级 Web 框架的研究与设计[D].镇江：江苏大学，2005.
[9] 肖祥红.基于 PostgreSQL 的省级像控点数据库设计与建设[J].地理空间信息，2019，17(11)：63-66，11.
[10] 刘小飞，胡珂，关昆，等.基于 Postgis 的异构地理空间数据组织与应用开发[J].测绘与空间地理信息，2014(9)：63-65.
[11] 王东兴，朱翊.GeoJSON 在异构地理信息数据集成中的应用[J].测绘与空间地理信息，2018，41(6)：138-140，146.
[12] 曾侃.基于开源数据库 PostgreSQL 的地理空间数据管理方法研究[D].杭州：浙江大学，2007.
[13] 林媛媛.基于 PostgreSQL 与 Postgis 的空间数据库设计及应用研究[D].赣州：江西理工大学，2014.
[14] 郭明强.面向高性能计算的 WebGIS 模型关键技术研究[D].武汉：中国地质大学，2013.

(收稿日期：2020-03-30)

作者简介：

王昊(1991-)，男，硕士，助理研究员，主要研究方向：Java 后端研发、遥感大数据。

特日根(1987-)，通信作者，男，博士，副高级研究员，主要研究方向：遥感大数据，E-mail：terigen@charmingglobe.com。

版权声明