Journal of System Simulation

Volume 32 | Issue 3 Article 12

3-25-2020

Parallel Computing of Multi-Agent System in Spatially-Explicit Modeling and Simulation

Hongchun Qu
1. College of Automation, Chongqing University of Posts and Telecommunications, Chongqing 400065, China;;

Xianhui Yao
2. College of Computer Science and Technology, Chongqing University of Posts and Telecommunications, Chongqing 400065, China;

Yin Li
2. College of Computer Science and Technology, Chongqing University of Posts and Telecommunications, Chongqing 400065, China;

Parallel Computing of Multi-Agent System in Spatially-Explicit Modeling and Simulation

Abstract

Abstract: Aiming at the synchronization in the spatially-explicit parallel simulations, a limited optimistic synchronization algorithm based on Agent sensing region is proposed. The algorithm inherits the advantages of the optimistic synchronization algorithm and the security-guaranteed conservative synchronization algorithm. The algorithm categories all simulated agents into the security agent region and non-security agent region. The simulation is performed through the collaboration between the dispatch server and the execution servers where simulation tasks are scheduled and executed. The dispatch server determines whether an agent is in a security zone according to the security radius. The execution servers are responsible for performing the behavior of the agents. The algorithm achieves the better performance claimed in the limited optimistic synchronization, reduces the network traffic in the parallel simulation, ensures the causality constraint relationship in the process of simulations and decreases the simulation time cost. Experiments in GAMA platform demonstrate the effectiveness of the algorithm.

Keywords

parallel simulation, spatially-explicit model, large scale, limited optimism, perceive radius

Recommended Citation

Qu Hongchun, Yao Xianhui, Yin Li. Parallel Computing of Multi-Agent System in Spatially-Explicit Modeling and Simulation[J]. Journal of System Simulation, 2020, 32(3): 446-454.

第 32 卷第 3 期 系统仿真学报© Vol. 32 No. 3
2020 年 3 月 Journal of System Simulation Mar., 2020

多 Agent 系统在空间直观仿真建模中的并行化

屈洪春[1]，姚献慧[2]，尹力[2]
（1. 重庆邮电大学自动化学院，重庆 400065，2. 重庆邮电大学计算机科学与技术学院，重庆 400065）

摘要：针对空间并行仿真技术中的同步问题，设计基于 Agent 感知区域的有限乐观同步算法。该算法继承乐观同步算法的积极性、保守同步算法的安全性，并将仿真中的 Agent 划分为安全 Agent 区域和非安全 Agent 区域，同时利用调度服务器和执行服务器分工协作的方式完成仿真计算。*调度服务器根据不同 Agent 的安全半径筛选确定其是否属于安全区域，执行服务器负责执行 Agent 的行为。该算法实现仿真技术中积极且有限的乐观同步，降低并行仿真中网络的通信量，确保仿真程序执行过程中的因果约束关系，有效地缩短了仿真时间，*在 GAMA 仿真平台实现并验证方法的有效性。

关键词：并行仿真；空间直观模型；大规模；有限乐观；感知半径
中图分类号：TP391.9 文献标识码：A 文章编号：1004-731X (2020) 03-0446-09
DOI: 10.16182/j.issn1004731x.joss.18-0231

Parallel Computing of Multi-Agent System in Spatially-Explicit Modeling and Simulation

Qu Hongchun[1], *Yao Xianhui*[2], *Yin Li*[2]
(1. College of Automation, Chongqing University of Posts and Telecommunications, Chongqing 400065, China;
2. College of Computer Science and Technology, Chongqing University of Posts and Telecommunications, Chongqing 400065, China)

Abstract: Aiming at the synchronization in the spatially-explicit parallel simulations, a limited optimistic synchronization algorithm based on Agent sensing region is proposed. The algorithm inherits the advantages of the optimistic synchronization algorithm and the security-guaranteed conservative synchronization algorithm. The algorithm categories all simulated agents into the security agent region and non-security agent region. The simulation is performed through the collaboration between the dispatch server and the execution servers where simulation tasks are scheduled and executed. *The dispatch server determines whether an agent is in a security zone according to the security radius. The execution servers are responsible for performing the behavior of the agents. The algorithm achieves the better performance claimed in the limited optimistic synchronization, reduces the network traffic in the parallel simulation, ensures the causality constraint relationship in the process of simulations and decreases the simulation time cost.* Experiments in GAMA platform demonstrate the effectiveness of the algorithm.

Keywords: parallel simulation；spatially-explicit model；large scale；limited optimism；perceive radius

引言

随着空间直观仿真在各行业的大规模应用，仿真系统规模越来越庞大，导致系统中 Agent 数量巨增。传统仿真方法一般采用串行仿真技术，一个时间段内只能处理有限的 Agent，而单个 Agent 本身具有复杂的规则集，因此多 Agent 系统常常超过单个处理器的运算能力，导致仿真运算时间呈指数增加。因此，多 Agent 系统的并行化仿真逐渐成为了大规模空间直观仿真的主流方向。

收稿日期：2018-04-23 修回日期：2018-11-09；
基金项目：国家自然科学基金(61871061)；
作者简介：屈洪春(1979-)，男，四川南充，博士，教授，博导，研究方向为面向大数据的仿真与预测模型；姚献慧(1991-)，女，山东临沂，硕士生，研究方向为并行仿真模型。

实现多 Agent 系统并行化仿真不仅需要考虑到仿真性能需求、实际应用环境以及时间和空间上的复杂度，还需要解析多 Agent 系统中的结构层次和它们之间潜在的冲突和协商路径等。近年来多 Agent 仿真系统的研究已经取得了长足进展。Haddadi 等对通信交互进行了严格定义，提出了合作与交互理论，设计了一种比较适合复杂系统的 Agent 机制，但对通信合作过程没有给出与环境相关的具体规则[1]。Huhns 从分布计算的角度分析了服务器 Agent 与客户端 Agent 之间的合作方式[2]。韩玉平在云环境下提出了基于 JADE 的多 Agent 分布式仿真系统，通过 FIPA 标准确保 Agent 之间的交互[3]。刘家学提出了一种电子仪表系统的建模方案，利用混合型 Agent 设计了系统结构[4]。但上述多 Agent 系统仿真方法都是基于串行模式，难以克服仿真效率低、仿真周期长的缺陷。

多 Agent 系统的并行分布仿真技术是目前国内外仿真领域中研究的热点之一。并行分布仿真技术把一个完整的、复杂的仿真系统划分成不同的逻辑进程(Logical process，LP)。LP 根据一定的分配策略分别在不同的处理机上运行，各个处理机或 LP 之间通过带有时戳的信息来进行相互通信[5]。由于处理机的性能存在差异，各个 LP 上仿真程序运行的速度也不可避免存在差异，这会导致 LP 间的消息传递表现出延时性、随机性、多发性等特点，因此不同 LP 之间如何进行消息和时间同步成为仿真领域研究亟待解决的关键问题。并行分布仿真时间同步算法大致可以分为保守同步算法、乐观同步算法和混合同步算法。

由 Chandy，Misra 以及 Bryant 提出的保守同步算法，其逻辑进程之间通过带有时戳的消息来提供事件之间的因果关系，只有当逻辑进程保证不会收到时戳小于本地局部仿真时间 LVT 的消息时，才允许执行当前事件[6]。由于保守算法严格遵守本地因果约束条件，因此需要不断地判断是否可以安全地处理事件，即确保可能影响该事件发生的所有事件都已经完成了，因此非常容易造成死锁。

乐观同步算法中并不能严格的遵守因果关系约束的条件。不同的 LP 在运行时可能会收到违反因果关系的消息，此时就需要使用回退机制，通过回退机制将系统恢复到出现错误时的时间点，并按照重新规划的路径推进仿真的运行。由 Jefferson 提出的 Time Warp[7]算法是非常著名的乐观算法，当出现了违反因果关系的消息时，算法就会执行核心机制“超前-回退”。该算法的局限性在于一方面需要大量的内存空间来存储需要回退的数据和状态信息，另一方面难以满足在空间直观仿真中要遵循的仿真实时性要求，显然这种回退并不是一个最佳的选择。

混合同步算法是将乐观同步算法和保守同步互相融合，取长补短，借鉴了乐观同步算法的超前性，同时也借鉴了保守同步算法的约束性，典型的如周期时间桶算法(BTB)[8]。周期时间桶算法在乐观协议的基础上提出了改进方案，在没有消息反馈的基础上就能完成系统事件的回退。这很好地解决了发送风险消息和乐观处理事件的问题。但是周期时间桶算法中事件限只能在周期开始后才能确定下来，由于引入了过多的周期，所以周期时间桶算法的同步开销的占比也相对过大。

为提升保守同步算法效率，同时减小乐观同步算法在放松约束上的回退风险，以及克服周期桶算法同步开销过大的缺陷，提出了基于 Agent 感知区域空间的有限乐观同步算法。该算法保证了仿真程序运行中的因果关系约束，对一些实时交互，不适合回退、大规模的直观仿真模型适用性更强。

1 基于 Agent 的空间直观仿真建模

基于 Agent 的空间直观仿真模型适合处理复杂分布问题，其建模思想是将复杂系统从上至下分解为多个 Agent，通过微观 Agent 的行为和交互进行建模，最终涌现出系统的宏观行为[9]。Agent 具有行为独立性，能够感知环境，并能对环境变化及时做出反应，并通过学习相关知识适应新的环境。同时 Agent 可以具有目标导向性，自主根据决策执

DOI: 10.16182/j.issn1004731x.joss.18-0231

第 32 卷第 3 期 系统仿真学报 Vol. 32 No. 3
2020 年 3 月 Journal of System Simulation Mar., 2020

行相应的动作。仿真 Agent 所具有的动作应与实际系统中实体动作相一致，Agent 动作的执行结果会反馈到仿真系统中，为关联 Agent 的信息更新提供直接依据。

GAMA 平台是基于 Agent 建模语言的仿真建模开发平台，特别适合 Agent 行为和地理空间信息的建模[10]。在 GAMA 平台中用户不仅能自由地使用复杂的 GIS 数据作为 Agent 仿真环境，还支持运行仿真系统中高达百万级别的 Agent 群体。GAMA 平台具有友好的仿真界面、模型浏览窗口以及多视窗同步仿真。在仿真运行过程中，用户可以适时地与 Agent 进行互动，同时 GAMA 平台对地理矢量数据的操作和处理能力非常强大，而且提供了 Agent 架构和空间分析函数扩展库等。GAMA 已被用于各种大型仿真应用中，同时也可以作为各行业决策工具的开发平台，如 MAELLA 项目。GMAL 是 GAMA 平台的建模语言，GAMA 简单易用，较少的代码就能实现复杂的功能。本文以 GAMA 平台为依托，用 GMAL 语言对多 Agent 系统进行建模，并以野生蓝莓模拟仿真项目验证和测试了算法的并行性能。

2 基于感知区域的有限乐观同步算法

基于 Agent 的功能特性与离散时间仿真系统乐观同步算法[11]，本文通过引入感知区域的建模限制，提出一种新的基于感知区域的并行化空间直观仿真同步算法。基于感知区域的 Agent 模型，将并行仿真系统中以 LP 为基本时间处理单元改为以单个 Agent 为基本时间处理单元，采用了不同于传统并行仿真系统的仿真时间计算模型，也就避免了以 LP 为基本时间处理单元所引起的回退问题和 LP 之间的消息传递瓶颈问题。

2.1 算法角色

本法中有 2 种角色：一是主服务器；二是执行服务器。主服务器负责筛选安全 Agent，并且调度逻辑进程(LP)执行计算任务，执行服务器负责本 LP 事务的执行。主服务器与执行服务器之间通过 Agent 的调度、注册、管理等活动进行直接通信，而执行服务器之间只能通过主服务器进行间接的通信，通过主服务器的管理调度使不同执行服务器之间能够进行 Agent 转移和通信。图 1 展示了仿真系统中主服务器(Main container)与执行服务器(LP container)之间互相通信的过程。

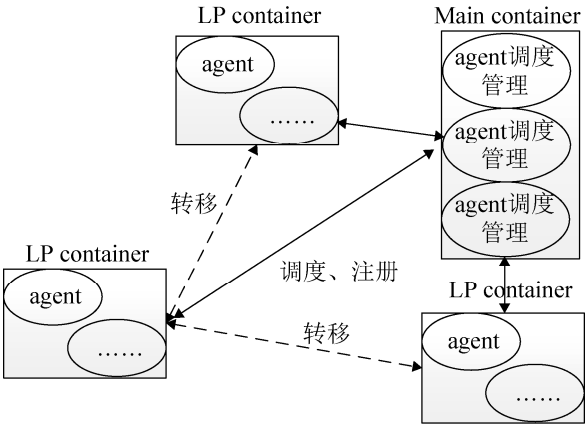


图 1 服务器间通信示意图
Fig. 1 Inter-server communication diagram

主服务器提供了 3 种管理服务：Agent 调度管理、全局坐标管理和消息传送系统。Agent 调度管理服务负责调度不同 LP 中的 Agent；全局坐标管理系统可以对 Agent 的坐标位置进行计算；并行空间仿真系统中，每个 LP 在全局坐标系中是固定的，Agent 的坐标是基于 LP 的局部坐标系位置确定的。当 Agent 转移到所在 LP 的空间区域范围之外时，需要主服务器根据 Agent 的全局系统坐标确定转移的目标节点地址，转移过程中用到主服务器的消息传送系统。

在仿真系统中，Agent 可以处于以下 6 种状态中的任意一种：

创建：创建 Agent 对象，并在 Agent 管理系统上注册，对没有注册的 Agent 系统不允许与其他 Agent 进行通信。

活动：Agent 对象在 Agent 管理系统上注册，在管理系统中，注册过的 Agent 可以与其他 Agent 进行通信，并拥有所有系统特征。

挂起：Agent 对象当前被停止，其内部线程在系统中被暂停，Agent 行为无法被执行。

等待：Agent 对象被阻塞，它的内部线程处于睡眠状态，当条件满足时唤醒 Agent。

删除：Agent 对象死亡后，内部线程终止执行，Agent 无法再注册在 Agent 管理系统上。

转移：当移动的 Agent 向其他 LP 进行转移时，系统会缓存其信息，并将其发往不同的 LP 地址上。

2.2 感知区域

在改进的 Agent 并行仿真模型中，建模时需定义感知区域属性。常规 Agent 触发事件的范围半径定义为：

AOI：Area Of Interest 表示一个 Agent 在某个执行周期中的触发事件所影响范围的半径。

AOI_i^j：表示一个 Agent 在仿真周期 i 到仿真周期 j 中可能触发事件的范围半径，AOI_i^j 计算公式如下：

$$AOI_i^j = speed * duration * (j - i) + AOI$$

式中：*speed* 为 Agent 的线速度；*duration* 为 Agent 仿真的周期时间，感知区域定义了 Agent 行为影响的空间范围大小，根据感知区域属性，安全 Agent 的定义为：

在仿真程序的执行过程中，假设任意两个 Agent 为 A、B，如果 A、B 是同步的，在仿真周期为 i 时调度 A 和 B 执行仿真，如果 B 先于 A 进入仿真周期 i+1，且有：

$$AOI_i^{i+1}(A) + AOI_i^{i+1}(B) < AB$$

式中：AB 为 A 到 B 的直线距离，则 A 与 B 在第 i+1 个执行周期中互为安全 Agent。在执行 $B(A)$的第 i+1 个执行周期时，不会与 $A(B)$发生交互，不需要等待 A 程序就可以安全的调度 B 执行周期 i+1，在 A 到达执行周期 i+1 时相对于 B 依然可以安全地执行仿真而无需回退。

2.3 算法描述

调度服务器计算安全 Agent，并且调度 LP 执行，执行服务器负责系统中不同仿真区域的执行。调度服务器，执行服务器的运行流程图如图 2 所示。

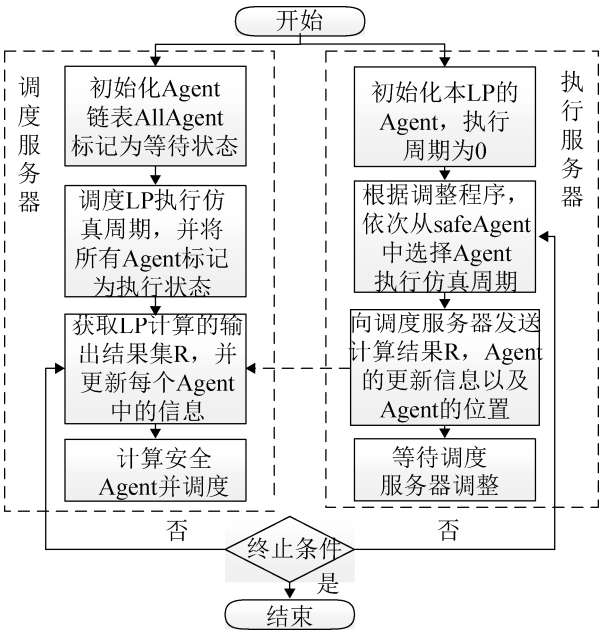


图 2 算法流程图

Fig. 2 Algorithm flow chart

在第 0 个执行周期，所有 Agent 均被标记为安全 Agent，调度服务器调度所有 Agent 在不同的 LP 上执行仿真周期 1。如果某个 LP 先执行完毕，找到这个 LP 中的安全 Agent(记为 S_1)，通过调度这些 Agent 执行第二个执行周期。根据定义，当其他 LP 中的 Agent 完成执行周期 1 之后，这些 LP 的 Agent 相对于 S_1 是安全的，因此可以调度这些 Agent 执行下一个周期 2。以此类推，调度过程总是以安全 Agent 的计算为基础，所以安全 Agent 的计算与筛选是算法的关键。对于有大量 Agent 的空间直观仿真系统，不需要每次都重新计算所需要调度的 Agent，系统可以通过利用之前的计算信息，不断地筛选安全 Agent 进行调度。因为系统在调度之前就确保了被调度的 Agent 的安全性，所以这些 Agent 以基于安全判断被执行若干次调度之后，执行周期为 i 的 Agent 相对于执行周期为 $j(j>i)$ 的 Agent 都是安全的。

具体地，Agent 被第一次调度时，被调度的 Agent 相对于其他所有执行的 Agent 都是安全的，可以直接进行调度而无需计算；其次，以后的调度

DOI: 10.16182/j.issn1004731x.joss.18-0231

第 32 卷第 3 期 系统仿真学报 Vol. 32 No. 3
2020 年 3 月 Journal of System Simulation Mar., 2020

Agent 会两两计算与之相关的 Agent 是否为安全，或通过已知安全 Agent 信息选择所需要执行的 Agent 执行下一个周期；最后，如果某个 LP 先执行完毕，在计算安全 Agent 时只需要计算当前仿真时间小于本 LP 上 Agent 仿真时间的 Agent，而不需要每次都重新计算所需要调度的 Agent，可以通过利用之前的计算信息，来判断 Agent 是否安全。因此，通过对安全 Agent 计算方法进行改进，本算法只计算不同的元素对一次仅且一次，算法的时间复杂度为 $o(1)+o(2)+\cdots+\cdots+o(n)=o(n^2)$。

改进后的模型仿真算法由 Algorithm1 描述：

输入：Agent 属性 *speed*，*AOI*，*duration*，*cycle* 的初始值；

输出：安全 Agent 信息；

1) 根据初始值初始化 Agent 链表，将 Agent 放置于二维链表 all-Agent 上；

2) 在执行服务器上执行本机二维链表 all-Agent 上所列出的 Agent；

3) 调度服务器判断事件触发范围，为执行服务器筛选安全 Agent，并将其放入二维链表 safe-Agent；

4) 执行服务器从 safe-Agent 中取出安全区域内的 Agent 执行；

5) 若有 Agent 跨 LP 转移，则将 Agent 从原 LP 的 all-Agent 上销毁，并在移入的 LP 中创建 Agent；

6) 执行服务器执行完一个 *cycle*，将 *cycle*，duration 进行累加；

7) 判断仿真是否结束，若否，则转步骤 3。

2.4 算法测试和比较

测试模型在单机运行和多机运行条件下执行，实验的软硬件环境为 Inter Pentium CPU G645 (主频为 2.90 Hz)、内存 2.0 GB、Windows XP 操作系统，软件平台为 GAMA 仿真平台。

本节通过仿真实验来比较 Agent 感知区域的有限乐观算法与基本乐观算法 Time Warp[7,12]、保守同步算法以及 BTB 算法在空间直观并行仿真中的性能。通过对时间的推进速度(TAS)、丢包影响率的比较来评价仿真算法性能，其中时间推进速度是比较时间同步算法的常用性能指标。

时间推进速度定义为单位时间内仿真推进的步数，它测量了仿真节点的同步效率，时间推进速度可以描述为：

$$Tv = N / (t_2 - t_1)$$

式中：N 为仿真推进的步数；t_1 到 t_2 为选取的仿真时间段，利用 GAMA 平台运行仿真案例测试具有 2 个节点的仿真系统。实验列举了具有 4，10，16，25 或 40 个 Agent 等不同的仿真系统，得到如图 3 所示的时间推进速度测试结果。从图 3 可看出 Agent 有限乐观的仿真同步推进速度大大高于保守同步算法与 BTB 算法，总体上也明显高于基本乐观算法 Time Warp。而且随着仿真成员 Agent 数量的增加，保守同步算法执行速度 Agent 很快下降，BTB 算法的推进速度也随仿真成员数量的增加而急剧降低，而基于 Agent 感知区域的有限乐观算法的推进速度与基本乐观算法 Time Warp 却不因仿真成员数量的增加而降低，其性能明细优于保守算法与 BTB 算法。

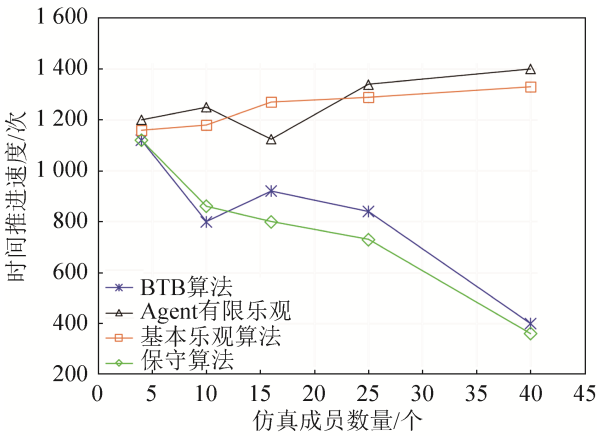


图 3 不同算法的仿真推进速度比较

Fig. 3 Simulation speed comparison of different algorithms.

丢包影响率是仿真程序运行中节点的丢包推进速度与无丢包时推进速度的结果比值，通过其的百分比来衡量丢包率对时间管理的影响:

$$Lost = Ylost / Nlost$$

Lost 为丢包率，单位为百分比，*Ylost* 为仿真产生丢包时的速度，*Nlost* 为仿真无丢包时的推进速度，图 4 展示了采用 800 个 Agent 模拟 10 种丢包状况下的仿真丢包影响率，从图 4 可知，丢包率在 0%~20%之间，4 种算法的丢包影响率基本保持一致，当丢包率大于 20%时，Agent 有限乐观算法的丢包影响率明显下降(40%)，基本乐观算法 Time Warp 也有所下降，但下降幅度较小(35%)，BTB 算法和保守算法的丢包影响率下降不是很明显(分别为 16%和 24%)。基于以上分析，可知提出的 Agent 有限乐观算法对丢包率是不敏感的。

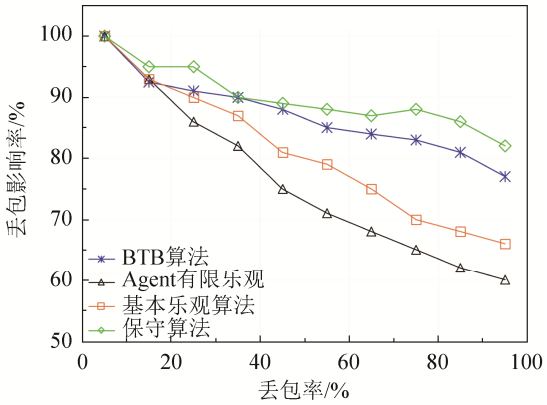


图 4 丢包率影响测试结果
Fig. 4 Packet loss rate impact test results

3 仿真应用

3.1 野生蓝莓传粉系统仿真模型

野生蓝莓管理与种植系统是由野生蓝莓与各种传粉昆虫组成的、具有极高价值的复杂农业生态系统，其每年的收益超过 200 亿美元。由于野生蓝莓种植系统存在基因异质性、复杂的空间分布模式、多种传粉昆虫共生的特点，极易由传粉昆虫引发疾病传播，且受非常容易受气候变化的影响，因此其管理与种植是一项非常复杂的系统工程。传统的生态学和农业观测实验难以获取全面的数据，也不能有效地检测出影响种植收益和成本之间的隐含因素及其关系，因此采用仿真模型来探索系统并预测便成了一种高效的手段。

但是，实现野生蓝莓传粉系统的有效仿真是非常困难的任务，原因在于系统中有多种生物存在，且具有不同的行为，考虑到昆虫的飞行距离，必须要足够大的区块才能够正确地仿真系统传粉动力学行为。而大的区块，如 1 公顷，野生蓝莓的花朵数量将超过 2 亿朵，而这 2 亿朵花之间因传粉导致的基因流向不是对称的，这显然需要非常大规模的并行仿真才能实现。本文将使用该仿真系统来验证算法的有效性。

在野生蓝莓仿真系统中，Agent 是仿真架构的基本组件，该模型由 2 种类型的 Agent 组成：

(1) 常规 Agent，例如：蓝莓田、蓝莓植物(簇/Clone)、茎、花以及蜜蜂传粉者，这些都是现实中可见的实体；

(2) 虚拟 Agent，例如：环境、系统调度、天气以及物候，这些提供协调常规 Agent 之间交互的空间和时间参考。

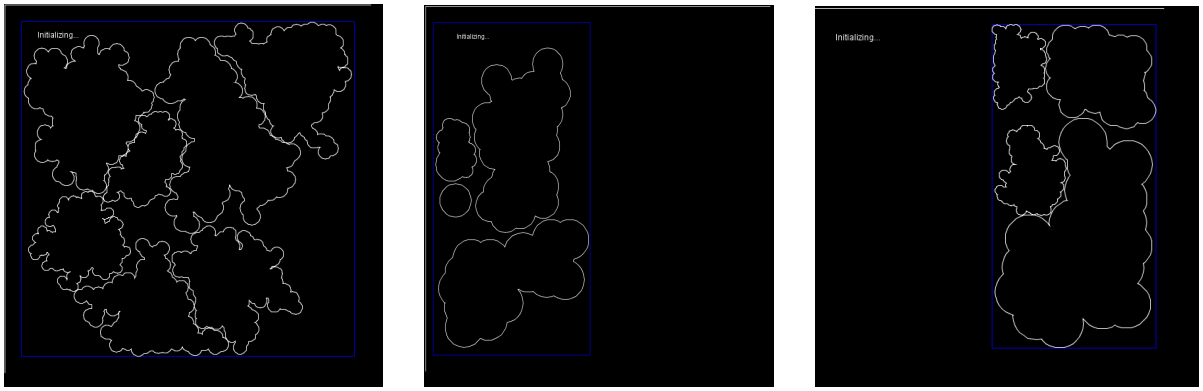
本项研究对大规模蓝莓传粉系统进行仿真，将面积为 50 m×50 m=2 500 m^2 的蓝莓田拆分成面积为 50 m×25 m，50 m×25 m 两块面积相同的蓝莓田，并行的 2 台主机之间采用基于 Agent 感知区域的有限乐观同步算法进行通信，并在 GAMA 平台上实现了基于感知区域有限乐观同步的并行仿真。

图 5(a)展示了单机运行的蓝莓植被初始状态的仿真效果图，图 5(b)展示了多机并行时蓝莓克隆扩散生长过程的仿真效果图。蓝莓簇在扩散过程中会在 2 个 LP 图 5(b,c)之间进行初始的信息通信，不断交互彼此的密度，判断蓝莓是否达到发芽的时间，调度服务器进行信息处理，若 2 个 LP 都达到标准，则进入蓝莓的下一个周期。

图 6(a)展示了单机运行的蓝莓繁殖生长阶段状态的仿真过程，图 6(b,c)展示了多机并行的蓝莓繁殖生长仿真过程运行状态，此时蜜蜂进入授粉状态，蓝莓进入发芽、花期，并陆续进入结实状态。绿点是正在发芽阶段(开花前)的样方，黄点是茎正在开花的样方，红点是茎上的花已经成为果实(开花后)的样方。混合的黄色(花)和绿色(芽)显示在 Clone 内正处于在开花阶段。

https://dc-china-simulation.researchcommons.org/journal/vol32/iss3/12
DOI: 10.16182/j.issn1004731x.joss.18-0231

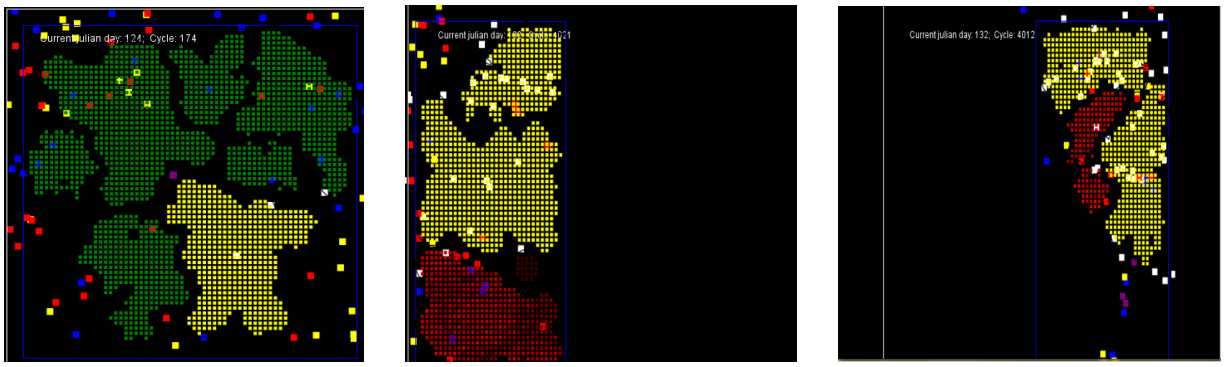
第 32 卷第 3 期 系统仿真学报 Vol. 32 No. 3
2020 年 3 月 Journal of System Simulation Mar., 2020



(a) 全局仿真区域 (b) 局部仿真区域(左) (c) 局部仿真区域(右)

图 5 蓝莓簇扩散生长过程仿真

Fig. 5 Simulation of diffusion growth process of blueberry clone



(a) 蓝莓全局仿真实例 (b) 蓝莓局部仿真实例(左) (c) 蓝莓局部仿真实例(右)

图 6 蓝莓繁殖生长阶段仿真

Fig. 6 Propagation and growth stages of blueberry clone

图 7 展示了并行运行的 2 个 LP 之间进行互相通信时信息交互内容，图 7 显示 2 台主机在第 621，622，623 仿真步数交互时发送的信息，信息中对 2 个 LP 的蓝莓是否达到花期进行了交互，通过彼此交互的信息，调度服务器可以计算花的开放密度。当 2 个 LP 上蓝莓花的开放达到一定百分比后，接下来就会触发蜜蜂授粉事件。对跨 LP 移动的蜜蜂进行了位置的交互，每个 LP 上都设有一个位置临界值，当蜜蜂跨过这个临界值后，则同时需要对蜜蜂进行创建与销毁，创建蜜蜂时根据蜜蜂的种类以及相应的位置，在所移动到的 LP 中进行创建，同时需要保存蜜蜂原有的基本属性，比如：蜜蜂的离巢距离，蜂巢的位置以及蜜蜂花篮中采蜜的数量等。这些基本属性都需要在原 LP 中发送信息时一同发送给新的 LP。同时原 LP 中要销毁原蜜蜂信息。

3.2 仿真应用的性能分析

本文通过将仿真程序部署在不同数量的逻辑处理器上并观测其执行时间来测试并行算法的效能。实验采用的 Agent 数量为 1000，按照 LP 的数量均匀分配 Agent，误差为 1%以内。执行时间为 80 000 个虚拟时间所对应的物理时间，单位为 s。图 8 给出了逻辑处理器数量与执行时间的关系。当逻辑处理器 LP 数量较小时，因为管理和协调负载相对于并行执行所带来的效率提升并没有明显增加，所以总的仿真时间呈下降趋势，但整体性能并不是线性增长的。当逻辑处理器 LP 的数量增大时，管理和协调负载成本逐渐超过并行执行效率提升的优势，因此总的仿真时间逐渐增加。在本实验中，LP 数量为 5 是一个可以明确观测到的临界值，可以认为是管理和协调负载与并行效能的平衡点。


```
send621
621·flower_in_bloom:true·percent_flower_open:0.10609480812641084·Honeybee:location=
true
0.10609480812641084
createBee:0
Honeybee609{5.0,5.0,0.0}:{1.0671672039662408,2.848287457837822,0.0}:[{1.06716720396
createBee:0
Honeybee610{5.0,5.0,0.0}:{1.0671672039662408,2.848287457837823,0.0}:[{1.06716720396
true
send622
622·flower_in_bloom:true·percent_flower_open:0.10609480812641084·Bumblebee:location
true
0.10609480812641084
createBee:0
Bumblebee445{1.6745947549527962,1.082526943832199,0.0}:{1.6745947549527962,1.082526
true
send623
623·flower_in_bloom:true·percent_flower_open:0.10609480812641084·Honeybee:location=
true
0.10609480812641084
createBee:0
Honeybee611{5.0,5.0,0.0}:{2.93643000702709,8.63977826024133,0.0}:[{1.06716720396624
createBee:0
Osmia221{-0.3138699059036693,8.21045824963505,0.0}:{-0.3138699059036689,8.210458249
true
```

图 7　并行仿真过程中属于不同 LP 的蓝莓对象之间的交互信息

Fig. 7　Information exchange between blueberries simulated in different logical process (LP)

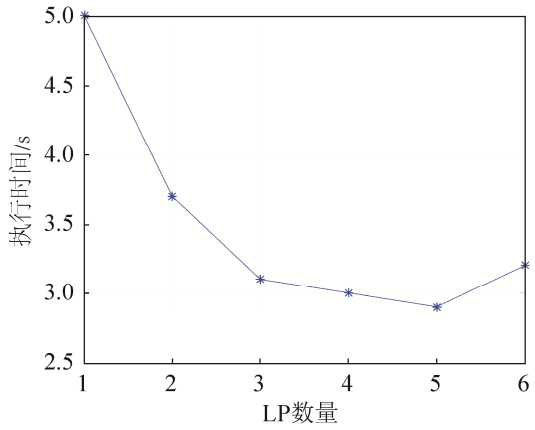


图 8　不同数量逻辑处理器(仿真节点)与执行时间的关系

Fig. 8　Execution time on different number of LPs (simulation nodes)

另一方面，加速比也是衡量算法并行效能的重要指标。加速比定义为在单处理器上执行的仿真时间 T 与在多处理器上执行的时间 T_N 之比。通过对加速比的测试可以对多处理器并行处理仿真速度的提升做比较。加速比公式为：$a=T/T_N$。

图 9 为蓝莓并行仿真在不同 LP 上执行的加速比。加速比增加表明多机仿真所用的时间相对单机仿真所用的时间更少，并行仿真优势更明显。由图中可知，随着 LP 数量的增加，加速比增加，但优势逐渐被管理和协调成本所消耗，因此加速比与 LP 数量的正相关性逐渐减弱。当 LP 数量达到当前仿真条件下的平衡点，即 LP=5 时，加速比达到顶点，然后拐头向下，不利于仿真程序的执行。因此在仿真应用中需要综合考虑 LP 数量与 Agent 数量与分布的关系，来确定合适的 LP 数量。

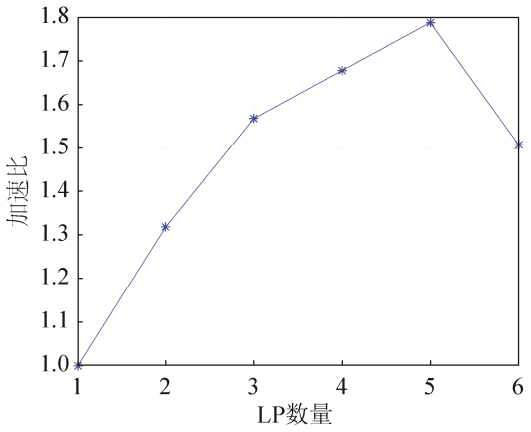


图 9　蓝莓并行仿真加速比与逻辑处理器数量的关系

Fig. 9　Acceleration ratio versus number of LPs

https://dc-china-simulation.researchcommons.org/journal/vol32/iss3/12
DOI: 10.16182/j.issn1004731x.joss.18-0231

第 32 卷第 3 期 系统仿真学报 Vol. 32 No. 3
2020 年 3 月 Journal of System Simulation Mar., 2020

4 结论

随着多学科交叉研究的进展，基于 Agent 的空间直观模型因为其机理模型的本质能够解决传统微分方程模型在空间异质性表达上的缺陷，因此得到了广泛的应用。近年来涌现出了大量的空间直观仿真模型，但是大规模空间直观仿真模型的运行需要消耗大量的计算资源，计算时间成了模型应用的主要瓶颈。为了解决这个问题，需要解决 Agent 在空间直观仿真中的并行化问题，其核心是同步问题。本文基于 Agent 感知域设计了一种有限的乐观同步算法，用以处理空间并行仿真中的同步问题，该算法继承了乐观同步算法的积极性和保守同步算法的安全性。通过将仿真代理分成安全代理和非安全代理来进行保守性。同时，与调度服务器和执行服务器一起进行仿真计算。在减少并行仿真中的网络流在执行模拟程序时，保证因果约束并有效地减少仿真时间，该算法还应用于大型蓝莓花粉化系统的模拟模型，并在 GAMA 平台上检验了该方法的执行效率。但是，由于篇幅所限，未能对 LP 数量、Agent 在 LP 上的分布等问题与管理协调成本和并行效率之间的关系做深入研究，在下一步的工作中将进行重点讨论和分析。

参考文献:

[1] Haddadi A. Communication and cooperation in Agent systems: a pragmatic theory[J]. Lecture Notes in Computer Science(S0302-9743), 1995, 1056(1): 31-4.

[2] Huhns M N. Being and acting rational [Agent design][J]. IEEE Internet Computing (S1089-7801), 2003, 7(2): 91-93.

[3] 韩玉萍. 云环境下汽车制造供应链分布式多 Agent 仿真系统开发[D]. 大连:大连理工大学, 2016.
Han Yuping. The Distributed Multi-agent Simulation System Development of Auto Manufacturing Supply Chain under Cloud Enviroment [D]. Dalian: Dalian University of Technology, 2016.

[4] 刘家学, 胡萍, 朱玉娟. 基于多 Agent 的飞机电子仪表系统仿真模型[J]. 计算机测量与控制, 2016, 24(7): 167-170.
Liu Jiaxue, Hu Ping, Zhu Yujuan. Multi-agent Model for Electronic Instrument System Simulation [J]. Computer Measurement & Control, 2016, 24(7): 167-170.

[5] Guo S, Bai F, Hu X. Simulation software as a service and Service-Oriented simulation experiment[C]// IEEE International Conference on Information Reuse and Integration. Las Vegas, Nevada, USA, 2012: 113-116.

[6] Chandy K M, Misra J. Distributed Simulation: A Case Study in Design and Verification of Distributed Programs[J]. IEEE Transactions on Software Engineering (S0098-5589), 1979, SE-5(5): 440-452.

[7] Jefferson D, Beckman B, Wieland F, et al. Time warp operating system[J]. Acm Sigops Operating Systems Review (S0163-5980), 1987, 21(5): 77-93.

[8] 王学慧, 李革, 刘保宏, 等. 分布式集群并行仿真平台中时间同步技术研究[J]. 计算机仿真, 2006, 23(10): 119-123.
Wang Xuehui, Li Ge, Liu Baohong, Huang Kedi. Time Synchronization in Parallel Simulation[J]. Computer Simulation, 2006, 23(10): 119-123.

[9] Qu H, Drummond F. Simulation-based modeling of wild blueberry pollination[J]. Computers and Electronics in Agriculture (S0168-1699), 2018, 144(1): 94-101.

[10] Drogoul A, Amouroux E, Caillou P, et al. GAMA: A Spatially Explicit, Multi-level, Agent-Based Modeling and Simulation Platform[C]//International Conference on Practical Applications of Agents and Multi-Agent Systems. Berlin: Springer Press, 2013: 271-274.

[11] 邢清华, 刘付显. 离散事件系统分布仿真的集中并行控制[J]. 计算机仿真, 2000, 17(2): 43-45.
Xing Qinghua, Liu Fuxian. Central Concurrency Control for Distributed Discrete Event Simulation[J]. Computer Simulation, 2000, 17(2): 43-45.

[12] Thanawin R. Addressing Big Data Time Series: Mining Trillions of Time Series Subsequences Under Dynamic Time Warping[J]. ACM Transactions on Knowledge Discovery from Data (S1556-4681), 2013, 7(3): 1-31.

