arXiv:2304.01724v1 [cs.DC] 4 Apr 2023

具有局部化动力学的多智能体模拟的自适应并行化

Alexandru-Ionut.Bˇabeanu、Tatiana Filatova、Jan H. Kwakkel 和 Neil Yorke-Smith

代尔夫特理工大学，荷兰。{
A.I.Babeanu,T.Filatova,J.H.Kwakkel,N.Yorke-Smith}@tudelft.nl

摘要。 基于智能体的建模是一种用于表示和模拟复杂系统的通用方法。研究大规模系统具有挑战性，因为仿真运行所需的计算时间较长：其扩展性至少与系统规模（智能体数量）呈线性关系。鉴于多智能体仿真（MABS）的内在模块化特性，并行计算是克服这一挑战的自然方法。然而，由于智能体之间存在共享信息和通信，并行化并非易事。我们提出了一种用于多智能体仿真（MABS）共享内存并行执行的协议。该方法适用于能够通过顺序计算来表述的模型，以及涉及局部更新的模型，即更新仅涉及少量智能体。该协议具有自底向上和异步的特性，能够以自适应且平稳的方式处理异构计算。我们通过文化动力学和疾病传播领域的典型多智能体仿真（MABS）模型，展示了该方法在性能提升方面的潜力。

关键词： 基于智能体的模型 · 大规模系统 · 并行计算 · 科学仿真 · 异步计算

1 引言

尽管基于多智能体的模拟是表示和模拟社会、生物及其他类型（复杂）系统的一种灵活方法，但由于所需的计算时间较长，使用 MABS 研究大规模系统颇具挑战。MABS 的扩展性随智能体数量线性增长或更差。解决这一问题的自然方法是并行计算。尽管并行执行一组相互独立的模拟运行十分直接，但使用多个处理器核心加速单次 模拟运行则要困难得多。尽管 MABS 具有模块化特性，智能体（以及其他仿真组件）仍会频繁交互，因此无法以简单的方式并行执行它们的计算 [19]。

为应对这一挑战，我们提出了一种新颖的协议，用于在共享内存环境中并行执行多智能体仿真。我们利用了这样一个事实：MABS 可以表示为一系列计算步骤构成的顺序链，而且 MABS 模型中的更新通常具有局部性，也就是说，只涉及少量智能体。

与大多数关于 MABS 并行化的研究相比，我们的协议以高度异步的方式处理计算：在任意时刻，不同的 CPU 核心可以执行模拟中的不同步骤，并在不同时间处理不同的智能体。这使我们能够根据模拟过程或底层基础设施固有的异质性所导致的波动进行自适应调整。该协议还便于对形式上完全顺序执行的 MABS 模型进行并行化。我们使用两种 MABS（Axelrod 式文化动态和基于网络的疾病传播）展示了潜在的性能提升；这种提升源于计算资源在每个时刻都得到充分利用，但代价是需要执行额外的计算。我们还分析了相关的协议开销，并提出了管理开销的指导建议。

2 相关工作

文献中不断有人尝试解决 MABS 扩展规模的挑战[4, 19]。其中一种思路利用了超级智能体的概念：将具有某些相似性的智能体子集合合并为具有代表性的智能体 [21]。沿着这一思路，Arneth 等人 [2]提出了智能体功能类型的概念，可将其视为对应不同策略或风格的超级智能体。Tregubov 和 Blythe [24] 讨论了自动简化 MABS 的方法，例如自动抽象；这与（手工编码的）超级智能体相关，但并不相同。在空间环境中，超级智能体已被发现存在问题 [19]。

第二种范式采用代理模型来模拟 MABS 的有趣特性 [10, 17]。随着机器学习技术能力的提升，这种方法受到越来越多的关注 [1, 18], ，但它并未直接解决如何加速给定 MABS 的问题。我们的目标是避免任何重新编程或近似处理。

第三种方法与我们的工作最为接近，旨在利用高性能计算（HPC）技术和硬件运行 MABS。沿着这一思路，Blandin 等人 [6]提出了一种基于分布式内存和消息传递的方法，用于并行化 Python 中某些类型的基于智能体的仿真，并以迁移模型为例进行说明；该方法可处理多达 100 亿个智能体，同时仍采用某些模型简化和抽象，以减少进程间的通信。Blythe 和 Tregubov [8]则是大规模分布式 MABS 执行架构的另一个案例，该架构利用当代多进程处理和负载均衡基础设施，并结合图划分技术。相比之下，Axtell [5] 在单个多核共享内存系统上运行了 1.2 亿个智能体，并指出“事实证明，许多 HPC 架构并无用处”。此外，人们还为生物学中的类基于智能体模拟开发了通用计算框架，例如 Breitwieser 等人 [9] 和 Plesser 等人 [20], 所描述的框架，这些框架支持联合使用分布式内存和共享内存并行化。

尽管高性能计算驱动的工作已展现出显著的扩展能力，但我们注意到，迄今大多数研究都受制于一个根本性局限：并行化与严格按时间步拆分计算密不可分，并且每个时间步都会更新所有智能体（或其依赖于时间步的子集）。不

同时间步内的不同更新子集随后会通过某种工作负载均衡策略分配给可用的 CPU 核心。虽然根据上下文改进分配策略或许能提升一些性能，但最终无事可做的计算核心/节点必须等到当前时间步完成后才能进入下一步。这与许多 MABS 模型的局部特性相悖：只要前者在每一步都拥有所需输入，智能体就可以领先其他智能体许多个时间步进行更新。此外，逐步拆分的方法无法应用于一开始就不采用“每步多次更新”表述的模型，例如 Axelrod 式文化动力学模型 [3] 和马尔可夫链模拟；这类模型采用顺序式的“每步更新一次”表述，但仍可从并行化中受益。我们提出了一种通用方法来共同处理这些问题，其核心是以更灵活的方式处理科学计算中的时间，并将其作为一种用于共享内存自适应并行化的协议加以介绍。

我们提出的协议让人联想到已有的（基于软件的）动态调度研究，其历史至少可以追溯到 Blumofe 等人 [7], 。Ham [16]对相关工具和概念作了很好的综述。广泛用于共享内存环境下并行高性能计算的 openMP 库 [12], 也提供了动态调度选项。不过，动态调度侧重于并行化彼此独立的计算任务，而我们的协议则旨在妥善处理任务之间恰恰存在的这类依赖关系。此外，我们的协议还包含一个简洁的流程，用于与任何给定的 MABS 模型对接。

Scheutz 和 Harris[22]、Scheutz 和 Schermerhorn [23]提出的另一种 MABS 自适应异步并行化方法，也考虑了对依赖关系的敏感性。他们的工作与我们的工作在多个方面互为补充：1. 该方法采用分布式内存范式，并在进程之间使用消息传递，而我们的协议则高度依赖共享内存；2. 该方法通过代理实体和事件视界确保一致性，而我们通过工作线程记录和任务配方来确保一致性；3. 他们的每个 CPU 始终负责处理预先分配给它的一部分智能体，而我们的核心则可能在不同时间处理不同的智能体；4. 该方法配有一套复杂的形式体系（涉及身体、传感器和执行器等构造），模型必须用这套体系来表达，而我们的协议在这方面不那么复杂，也较少“侵入”；5. 该方法专注于包含移动智能体且具有最大移动速度限制的空间模型，而我们的应用目前尚未涉及移动智能体。未来的工作可以逐步融合这两种方法的理念。

3 方法论：计算协议

核心思想是，可以将 MABS 概念化为一条链，其中链中的每个元素都是一个任务。每个任务都是一组连续操作，这些操作会在模拟过程中的某个时刻执行。操作可以是生成一个随机数或递增一个变量等。通常，这些任务会按照它们在

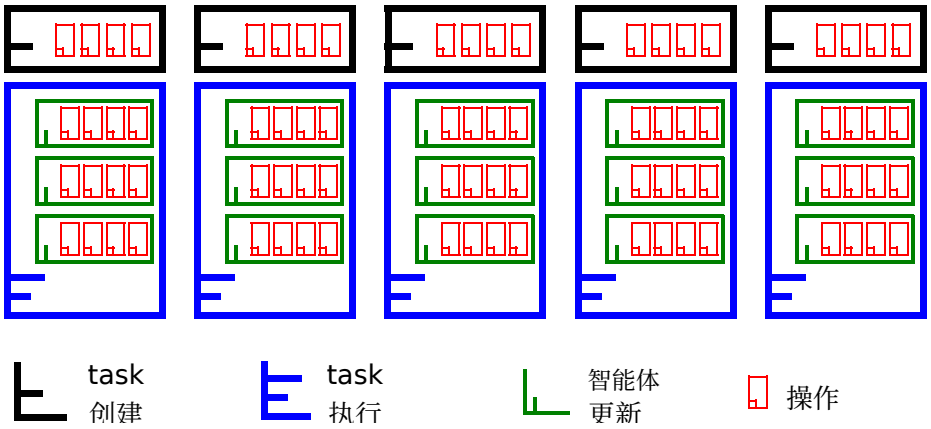


图 1. 模拟的分区。图的底部是图例，通过颜色和（部分）形状说明上方各类对象的含义。每个垂直区块对应一个任务，并分为创建（黑色）和执行（蓝色）两部分。小的（红色）方框表示构成模拟的基本操作。在每个任务的执行过程中，操作子集构成智能体更新（绿色）。

链中的顺序依次评估。然而在许多情况下，可以改变评估顺序而不影响模拟结果：许多任务对彼此完全独立，因此可以按任意顺序评估，也可以并行评估。例如，两个智能体的下一步决策仅受其邻居影响，且它们没有共同邻居，那么在下一步中，这两个智能体彼此独立。

我们的协议旨在充分利用这一特性，在保持系统演化的同时实现显著的并行化。这通过自适应方式实现：多个计算工作线程 自主遍历链，并执行那些不会违反模型依赖约束（表现为任务间的依赖关系）的任务。虽然并行化程度及由此带来的加速最终受可用 CPU 核心数量的限制（即每个工作线程对对应一个核心），但 MABS 模型、系统规模以及协议应用方式的某些选择会带来更严格的限制。

3.1 基于任务的表示

图 1 直观地概述了该协议如何有效地划分构成一次模拟的操作集合。每个生成的操作子集都对应一个任务，并受到重要约束：对于与任务 α 关联的任意两个不同操作 θ 和 ρ，任何依赖于 θ、且 ρ 依赖于它的操作 ϕ，也必须与 α 关联。一次模拟在数学上等价于按顺序拼接所有任务后得到的操作序列，

更新的排序需与它们之间的所有依赖关系保持一致。在更具体的 MABS 语境下，在每个任务的层面上，许多操作通常被归入更新，每次更新都会记录一个或多个智能体状态的变化。

每个任务首先被创建 并添加到链的末尾。随后，它被执行 （执行其中包含的操作），最后从链中删除。创建任务时已经完成了部分相关计算，而执行任务则根据创建期间生成并随任务存储的信息，完成剩余部分。创建与执行之间的划分也受到约束：如果操作 ρ 被包含在创建部分中，那么任何依赖于 ρ 的操作 θ 也必须被包含在其中，前提是 θ 一开始就与同一个任务相关联。不过，在创建阶段和执行阶段分别完成多少计算，仍有很大的决定自由。

3.2 任务之间的依赖关系

我们假设存在一个有限的变量集合 V，这些变量可以在模拟过程中反复改变。每个任务 α 的执行部分依赖于变量的一个输入子集 $I(\alpha)$，并修改变量的另一个输出子集 $O(\alpha)$；这两个 α子集之间的重叠程度可以任意。如果 α 在链中（无论间隔多大）位于另一个任务 β 之前，并且 $O(\alpha) \cap I(\beta) \neq \emptyset$，则二者之间存在依赖关系（或因果链接） $\alpha \prec \beta$：任意 β 的执行只有在所有带有 $\alpha \prec \beta$的 α 执行完毕后才能开始。另一方面，如果两个任务 α 和 β 之间不存在直接或间接的依赖关系，那么α 和 β 是可交换的：它们可以按任意顺序执行，也可以并行执行（并行程度任意）。

准确地说，上文的 $I(\alpha)$ 和 $O(\alpha)$ 是以最保守的方式定义的：它们分别是可能影响由 α 编码的计算，以及可能受其影响的变量子集。在实践中，实际使用或修改的变量子集通常会更小，这是因为在执行 α 的过程中做出了相应决策（这不会导致冗余的因果联系：没有更改某个本可以更改的内容，本身也是一种信息）。请注意，尽管这些概念有助于解释上文内容，但我们的协议在推断依赖关系时，并不会显式计算任何输入或输出子集。相反，它只会动态地利用此前记录在链和工作线程上的必要信息（例如代理 ID）。

3.3 工作线程—链动态

自适应并行化通过多个工作线程实现，它们以同时但异步的方式操作链。每个工作线程都分配到一个专用线程。工作线程可以尽早执行任务，前提是任务之间的依赖关系不被违反、不发生竞争条件（因为所有操作都在同一次程序运行中进行，并使用完全共享的内存池），且（不可避免的）互斥锁不会导致死锁。这需要一套自洽的规则

指定工作线程与链的联合动力学：一种 工作流。我们在此聚焦于一种相对简单的表述，并将其用于第4节中的实验。

该链实现为一个双向任务链表，每个任务都保存指向前一个和后一个任务的指针，因此可以轻松、干净地删除链中的任务。在任何时刻，每个工作线程要么正在等待进入该链，要么位于链中的某个任务处，不同工作线程分别关联不同的任务。每个工作线程都会从头到尾遍历该链。到达任务 β 时，工作线程 w 能够判断 β 是否依赖于已遍历过的任务 α 。此时：

– 如果 β 依赖于前一个任务，或者该任务已由另一个工作线程 u 执行，工作线程 w 就不会执行 β；相反，它会收集描述 β 的信息，并将其与从先前遍历过的任务中已收集的信息整合起来，据此推断后续任务的依赖关系；随后 w 尝试移至下一个任务– 否则， w 将执行 β，从链中删除 β ，并返回链的起点（同时丢弃所有有关先前遍历过的任务的信息）

在任一时刻，最多只会创建一个任务，并由尝试离开最后一个元素的工作线程将其添加到链末端。它通过调用全局的、特定于模型的例程来完成此操作。

至关重要的是，工作线程 w 不能移动到已有另一工作线程 v 所在的任务 γ ，除非后者已在执行 γ。一个工作线程在另一个工作线程后方等待的情况，由附加到链中每个任务的专用互斥锁处理。

另外还有两个互斥锁，即 进入锁 和擦除锁 ，它们附加在链本身上。具体来说，进入锁用于处理链为空时创建新任务的情况（此时不存在任务及其关联的锁）。这种情况在进程开始时必定会出现，之后也很可能多次出现。因此，持有进入锁的第一个进入链的工作线程会创建新任务。其次，擦除锁确保在任意时刻最多只有一个任务正在被擦除（由刚刚执行该任务的工作线程擦除），从而避免同时擦除连续任务所导致的不一致。

我们使用术语周期 来表示某个工作线程对链进行的一轮探索，即两次连续返回到链起点之间的过程。周期会在以下情况之一发生时结束：工作线程刚刚擦除了一个任务，或工作线程已到达链的末尾且无法创建新任务。后一种情况可能是因为：1）给定模拟所需的所有任务都已生成；或 2）工作线程已达到每周期允许生成的最大任务数。后者的每周期任务数限制可避免出现（不太可能的）情况：位于链末尾的工作线程不断创建因依赖先前遇到的任务而无法执行的新任务，从而使链无限延长至不合理的长度，并可能增加所有其他工作线程的探索时间。此上界由一个参数控制。

3.4 应用该协议时的选择

简要说明一下，在将协议应用于给定的 MABS 模型时所做的三项概念性选择。1.链粒度 指分配给每项任务的计算量。2.任务深度 定义了任务创建与执行之间的划分。3. 工作流参数 主要包括 n（工作线程数量）和 C（每个周期创建的任务数上限）。

3.5 特定模型的实现

该协议包含大量与其所应用的 MABS 模型无关的功能。围绕这些与模型无关的功能，我们创建了一个专用软件框架（不久后将以开源形式提供），让用户能够“插入”任何感兴趣的 MABS。为此，需要按照一组约定实现特定模型的接口，以便模型与工作流之间能够按预期进行通信。该框架本身以 C++ 实现。

该接口可以通过两个通用概念来理解：1. 配方：任务在模型侧的对应物；2. 记录：工作线程在模型侧的对应物。这些概念涵盖了特定于模型的数据结构和功能。配方规定了任务创建后所包含的信息类型，以及这些信息如何用于执行任务。记录规定了工作线程需要保存的信息类型，以便识别当前任务是否依赖于此前遇到的任何任务，并包含执行此判断的流程。它还规定了如何将当前任务中的信息与工作线程已持有的信息整合（如有需要），以及如何在新周期开始时初始化和重置工作线程持有的信息。

4 实验评估

在这项研究的第一阶段，实验的目的是确定：对于给定的 MABS 仿真，随着工作线程数量增加，我们的协议能在多大程度上实现预期的速度提升。为此，我们针对不同的 $n \in \{1 \dots 5\}$ 值，测量总模拟时间 T 随任务规模代理指标 s 的变化情况。对于较小的 s，预计加速效果不明显。我们将 $C = 6$ 固定，因为单独的实验表明其影响可以忽略不计。对于每种 s 和 n 的组合，T 都取 5 个模拟实例的平均值，这些实例使用不同的初始种子（以及初始状态，其生成过程不会计入 T）。

本文报告了两项实验，分别针对依照不同范式构建的两个相关 MABS 模型：第4.1节中的模型采用顺序交互，每次由随机选取的智能体对进行交互；第4.2节中的模型则在每一步更新所有智能体，更新依据是它们最近邻在前一步的状态。前者主要用于展示一种模式形成机制，后者则因其对现实世界的预测潜力而为人所知。两项实验在构建 s 和使用协议的相关细节上也有所不同，详见下文。

4.1 文化动力学

首先，我们按照 B˘abeanu 等人 [11]的规范，将我们的协议应用于一个文化动力学 [3],的 Axelrod 型模型。所有 N 智能体彼此相连，每个智能体都配有一组特征，每项特征对应一种 F 文化特征。在每一步中，随机选取两个智能体进行交互，并预先分配角色：一个源智能体 和一个 目标智能体。随后，目标智能体可能会根据一种概率过程改变一项特征；该过程取决于两个智能体的所有特征，旨在模拟社会影响。

在协议方面，粒度设定为每个任务对应一次成对交互。深度设定为由创建阶段负责随机选择交互双方，而将交互的大部分工作留给执行阶段。因此，配方保存两个智能体的标识符，而记录则说明：如果某个工作线程此前遇到的任何任务中，源智能体或目标智能体曾是目标智能体，那么当前任务就被视为存在依赖关系。

图2展示了实验结果，其中 s 定义为 F，因为一次交互的大部分工作都围绕遍历所有特征展开。我们看到，模拟时间 T 会随任务规模 s增大而增加，与 n无关，这与创建的任务总数固定这一事实相符。此外，对于固定的 s，我们看到 T 会随 n 增加而明显下降，这证实了随着工作线程增加，性能会如预期般提升。虽然这种效应的幅度（以不同 n 曲线的 T 值之比衡量）会随 s增大而增加，但我们也观察到饱和效应：当 $s <$为150时，工作线程超过 $n = 4$ 个并无帮助；而当 s低得多时，工作线程超过 $n = 3$ 个也无济于事（甚至可能适得其反）。这些结果都符合这样的直觉：随着任务规模增大，协议开销的重要性会降低。本实验每次运行步数为 2×10^6，其余模型参数固定为：$N = 10^4$、$q = 3$ （每个特征可能具有的特征数），$\omega = 0.95$ （有界置信阈值）。

4.2 疾病传播

其次，我们将该协议应用于一个简化的 SIR 型流行病模型。该模型涵盖了传染病传播预测型多智能体仿真的关键特征，例如 Gemmetto 等人 [14], Gomez 等人 [15] 在真实场景中使用的模型。具体而言， N 智能体位于固定图的节点上，该图具有恒定的度数 k 和环状结构。每个智能体可能处于三种状态之一：易感（S）、感染（I）或康复（R）。在每个时间步中，每个智能体都可能在两个相邻状态之间转换，转换遵循由参数 p_{SI}，p_{IR}，$p_{\mathrm{RS}} \in (0, 1)$ 控制的简单概率规则。从 S 到 I 的转换还取决于感染连接所占的比例。

与第一次实验相比，使用该协议涉及更多细节。首先，链中有两类任务：第一类任务涉及智能体计算其新状态；第二类任务涉及智能体用新状态替换当前状态。其次，每个任务处理一个子集的

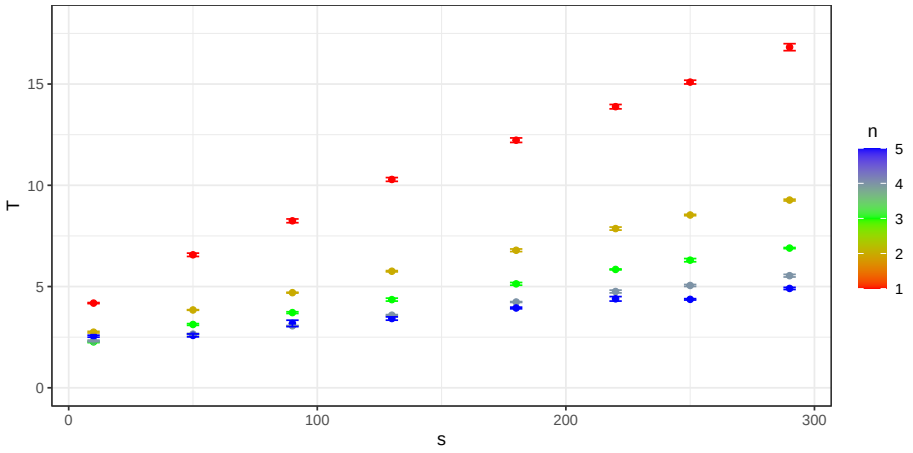


图 2. 使用我们的协议运行文化动力学。显示平均模拟时间（纵轴）随任务规模代理指标（特征数量 F，横轴）的变化，不同工作者数量以图例表示。垂直误差棒显示基于五个模拟实例计算的标准均值误差范围。

智能体，系统被划分为大小相等的子集，并且在整个模拟过程中保持不变。该划分固有的聚合级别决定了链的粒度。就深度而言，每次创建都会生成一个配方，其中包含智能体子集标识符以及一个指示任务类型的二进制标志，而实际计算则留待执行阶段完成。记录规定：处理同一智能体子集的第二类任务，不得在先前遇到的第一类任务之前执行；处理相同或相连智能体子集的第一类任务，不得在先前遇到的第二类任务之前执行。智能体子集之间的连接关系编码在一个聚合图中，该图只计算一次（就在生成初始状态之后）；这项计算计入测得的 T（生成初始状态则不计入）。

图3展示了实验结果。与第4.1节不同，这里没有可作为 s的模型参数。相反，我们将 s 定义为智能体子集的规模，它直接控制任务规模以及链的粒度。由此产生的一个结果在图中一目了然： T 不会随 s 增加（如图2所示），而是急剧下降，随后趋于稳定。对于所有 s，与模型相关的计算量都相同，只是划分方式不同，因此，无论 n如何， $s < 50$ 固有的细粒度划分都会因协议开销而变得非常耗费资源。在稳定区域中，我们可以看到，随着 n增加，性能如预期般提升，并在 $n = 4$时达到饱和。随着我们进一步深入低s 区域，饱和会在更低的 n时出现，而且我们清楚地看到，增加工作线程可能会降低性能。该实验每次运行包含 3×10^3步，其余模型参数固定为： $N = 4 \times 10^3$、 $p_{\mathrm{SI}} = 0.8$、 $p_{\mathrm{IR}} = 0.1$、 $p_{\mathrm{RS}} = 0.3$、 $k = 14$。

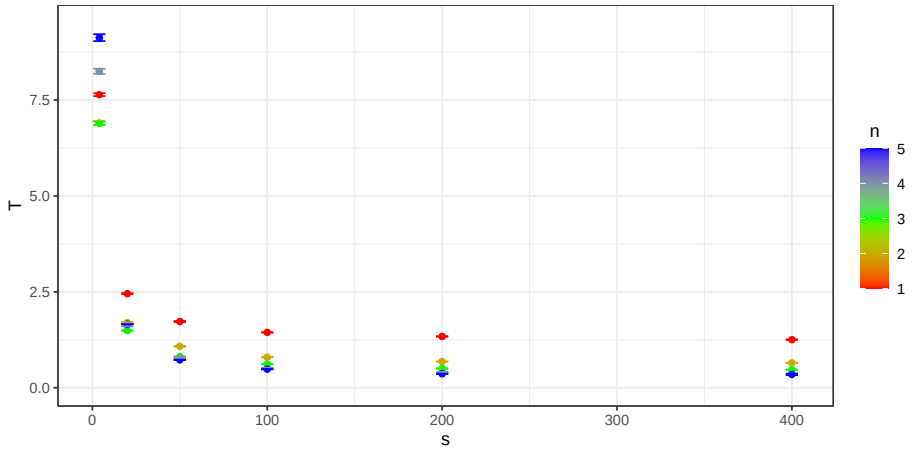


图3. 使用我们的协议运行疾病传播模拟。展示了平均模拟时间（纵轴）随任务规模代理指标（每个任务的智能体更新次数，横轴）的变化，不同工作线程数量见图例。垂直误差棒表示基于五个模拟实例计算的标准均值误差范围。

5 总结与未来工作

为了释放多核在加速单次 MABS 运行方面的潜力，我们提出了一种用于多智能体仿真的共享内存并行执行新协议。在 Axelrod 式文化动态和基于网络的疾病传播仿真的实验中，可以看到该协议带来的益处。

原则上，我们的协议可以应用于任何形式的计算，但只有当计算受 CPU 限制（而非受输入/输出限制）时，使用该协议才有意义；例如，在包含大量智能体的 MABS 中（如生态学领域）[13]。此外，如果选择合适的链粒度和任务深度，使任务之间依赖关系构成的有向无环图足够稀疏，我们预计计算能够获得实际加速。换言之，典型任务 α 只应依赖于系统的一小部分，并且只影响系统的一小部分。在 MABS 的背景下，“小”通常意味着相关输入变量和输出变量的比例 $I(\alpha)/V$ 和 $O(\alpha)/V$ 按 $1/N$ 的比例缩放，其中 N 是智能体的数量。这就是我们所说的局部化动力学——对于更具体的 MABS，局部一词更合适，因为其中的信息传播受到底层几何结构的限制。例如，涉及格点上智能体且智能体仅与最近邻交互的模型，就很适合采用我们的协议。即使模型涉及与整个系统耦合的全局实体，如果相关交互相对少见，和/或耦合仅为单向（仅影响系统的其余部分，或仅受系统其余部分影响），该协议仍可能有益。

未来的工作将展示我们的协议在具有非平稳智能体或动态空间几何结构的模拟中的应用。其次，我们会进一步考虑

明确利用计算的有向无环图特性，从而降低协议在内存和 CPU 使用方面的开销。第三，我们计划借鉴 Scheutz 和 Harris [22] 的方法，并结合这些方法的优势。除了 MABS 之外，我们预计其他类型的（非基于智能体的）模拟也能从这条研究路线中受益，其中包括某些类型的马尔可夫链模拟。

致谢 本文得益于与 J. Decouchant 和 M. Mircea 的交流，并获得了代尔夫特理工大学计算科学与工程研究所的部分支持，同时也得到 TAILOR 项目的资助。该项目由欧盟地平线2020计划资助，资助协议编号为 952215。我们还感谢两位匿名审稿人对本文早期版本提出的宝贵意见。

参考文献

1. Angione, C., Silverman, E., Yaneske, E.：使用机器学习作为基于智能体的仿真的代理模型。PLOS ONE **17** （2022），https://doi.org/10.1371/journal.pone.0263150 2. Arneth, A., Brown, C., Rounsevell, M.：用于基于土地的减缓与适应评估的人类决策全球模型。《自然气候变化》**4**，550–557（2014），https://doi.org/10.1038/nclimate2250 3. Axelrod, R.：文化的传播：一个具有局部趋同和全球极化特征的模型。《冲突解决杂志》**41**（2），203–226（1997），https://doi.org/10.1177/0022002797041002001 4. Axtell, R., Farmer, J.：经济学与金融中的基于主体的建模：过去、现在与未来。技术报告 2022-10，牛津马丁学院新经济思维研究所（2022） 5. Axtell, R.L.：1.2亿个智能体自组织成600万家企业：美国私营部门模型。载于：AAMAS’16 论文集，第806–816页（2016） 6. Blandin, N., Colglazier, C., O’Hare, J., Brenner, P.：面向全球规模基于主体建模的并行 Python。载于：CSSSA’17 论文集（2017），https://doi.org/10.1145/3145574.3145588 7. Blumofe, R.D., Joerg, C.F., Kuszmaul, B.C., Leiserson, C.E., Randall, K.H., Zhou, Y.：Cilk：一种高效的多线程运行时系统。载于：PPOPP’95 论文集，第207–216页（1995），https://doi.org/10.1145/209936.209958 8. Blythe, J., Tregubov, A.：Farm：分布式基于智能体社会模拟的架构。载于：第二届大规模多智能体系统国际研讨会论文集，第96–107页（2018） 9. Breitwieser, L., Hesam, A., de Montigny, J., Vavourakis, V., Iosif, A., Jen- nings, J., Kaiser, M., Manca, M., Meglio, A.D., Al-Ars, Z., Rademak- ers, F., Mutlu, O., Bauer, R.：BioDynaMo：用于高性能基于智能体仿真的模块化平台。《生物信息学》**38**，453–460（2021），https://doi.org/10.1093/bioinformatics/btab649 10. ten Broeke, G., van Voorn, G., Ligtenberg, A., Molenaar, J.：使用代理模型分析基于智能体的模型。J. Artif. Soc. Soc. Simul.**24**（2）（2021），https://doi.org/10.18564/jasss.4530

11. B˘abeanu, A.I., van de Vis, J., Garlaschelli, D.：超度量性提高了文化动力学的可预测性。New Journal of Physics **20**（10），103026（2018），https://doi.org/10.1088/1367-2630/aae566 12. Dagum, L., Menon, R.：OpenMP：共享内存编程的行业标准 API。Computational Science & Engineering, IEEE**5**（1），46–55（1998） 13. Filatova, T., Voinov, A.A., van der Veen, A.：保护沿海生态系统空间的土地市场机制：基于主体的分析。Environ. Model. Softw.**26**（2），179–190（2011），https://doi.org/10.1016/j.envsoft.2010.08.001 14. Gemmetto, V., Barrat, A., Cattuto, C.：学校传染病缓解：针对性班级停课与学校停课的比较。BMC Infectious Diseases**14**（1），695（2014），https://doi.org/10.1186/s12879-014-0695-9 15. Gomez, J., Prieto, J., Leon, E., Rodr´ıguez, A.：Infekta——一种用于传染病传播的基于主体的模型：哥伦比亚波哥大的 COVID-19 案例。PLOS ONE**16**（2），1–16（2021），https://doi.org/10.1371/journal.pone.0245787 16. Ham, D.R.：多核处理器中的动态调度。博士学位论文，曼彻斯特大学（英国）（2012） 17. La mperti, F., Roventini, A., Sani, A.：使用机器学习代理模型进行基于主体的模型校准。Journal of Economic Dynamics and Control**90**，366–389（2018），https://doi.org/10.1016/j.jedc.2018.03.011 18. Leeuw, B.D., Ziabari, S.S. M., Sharpanskykh, A.：机场航站楼运营的代理建模。载于：MABS'22 论文集，第82–94页（2022），https://doi.org/10.1007/978-3-031-22947-3 7 19. Parry, H.R.：基于主体的建模与大规模模拟。载于：《复杂社会与行为系统：博弈论与基于主体的模型》，第913–926页，施普林格（2020），https://doi.org/10.1007/978-1-0716-0368-0 9 20. Plesser, H.E., Eppler, J. M., Morrison, A., Diesmann, M., Gewaltig, M.O.：多处理器计算机集群上大规模神经元网络的高效并行模拟。载于：Euro-Par'07 论文集，第672–681页（2007） 21. Scheffer, M., Baveco, J., DeAngelis, D., Rose, K., van Nes, E.：超级个体：以个体为基础对大型种群进行建模的简单解决方案。Ecological Modelling **80**（2–3），161–170（1995） 22. Scheutz, M., Harris, J.：空间基于主体模型的动态分配与并行执行的自适应调度算法。载于：Parallel and Distributed Computational Intelligence，第207–233页，施普林格（2010），https://doi.org/10.1007/978-3-642-10675-0 10 23. Scheutz, M., Schermerhorn, P.W.：基于主体模型动态分配与并行执行的自适应算法。J. Parallel Distributed Comput. **66**（8），1037–1051（2006），https://doi.org/10.1016/j.jpdc.2005.09.004 24. Tregubov, A., Blythe, J.：通过自动抽象与简化优化大规模基于主体的仿真。载于：MABS'20 论文集，第81–93页（2020），https://doi.org/10.1007/978-3-030-66888-4 7