基于统计前向规划算法的游戏通用人工智能

LUCAS Simon[1]，沈甜雨[2,3]，王晓[2,4]，张杰[2,4]

（1. 伦敦玛丽女王大学，英国 伦敦 E1 4NS；
2. 中国科学院自动化研究所复杂系统管理与控制国家重点实验室，北京 100190；
3. 中国科学院大学人工智能学院，北京 100049；
4. 青岛智能产业技术研究院，山东 青岛 266109 ）

摘　要：统计前向规划（statistical forward planning，SFP）算法使用仿真模型（也称为前向模型）自适应地搜索有效的动作序列，此类算法提供了一种简单通用的方法，为各种游戏提供快速自适应的 AI 控制。介绍了两种常用的 SFP 算法：蒙特·卡罗树搜索和滚动层进化，并证明了在没有任何事先训练的情况下，SPF 算法就可以在各种视频游戏中出色地运行。

关键词：统计前向规划；游戏智能；滚动层进化
中图分类号：TP391
文献标识码：A
doi: 10.11959/j.issn.2096−6652.201935

General game AI with statistical forward planning algorithms

LUCAS Simon[1], SHEN Tianyu[2,3], WANG Xiao[2,4], ZHANG Jie[2,4]

1. Queen Mary University of London, London E1 4NS, UK
2. The State Key Laboratory of Management and Control for Complex Systems, Institute of Automation, Chinese Academy of Sciences, Beijing 100190, China
3. School of Artificial Intelligence, University of Chinese Academy of Sciences, Beijing 100049, China
4. Qingdao Academy of Intelligent Industries, Qingdao 266109, China

Abstract: Statistical forward planning (SFP) algorithms use a simulation model (also called forward model) to adaptively search for effective sequences of actions. They offer a simple and general way to provide rapidly adaptive AI controllers for a variety of games. The two powerful SFP example algorithms: Monte Carlo tree search and rolling horizon evolution were introduced in this paper and key insights into their working principles were provided. It is demonstrated that the algorithms are able to play a variety of video games surprisingly well without the need for any prior training.

Key words: statistical forward planning, game AI, rolling horizon evolution

1　引言

从简单的石头剪刀布、囚徒困境，到复杂的围棋与暴雪公司的热门游戏星际争霸，游戏最迷人的一点在于它总能提供一些有趣且未知的挑战。游戏的本质在于，在一个复杂的场景中，如何决策下一步采取的行动，以取得游戏定义的最终胜利。

目前，对于通用人工智能（artificial general intelligence，AGI）的研究主要表现在 3 个方面：一是利用深度学习结构（主要使用深度神经网络）捕捉知识和特征[1]；二是利用统计前向规划

收稿日期：2019−06−08；**修回日期**：2019−08−16
通信作者：王晓，x.wang@ia.ac.cn
基金项目：中国科协青年人才托举工程基金资助项目（No.2017QNRC001）；国家自然科学基金资助项目（No.61702519，No.71402178）
Foundation Items: The Young Elite Scientists Sponsorship Program of China Association of Science and Technology (No.2017QNRC001), The National Natural Science Foundation of China (No.61702519, No.71402178)

（statistical forward planning，SFP）算法[2]可以实现动作序列的优化，选出能够引向最好结果的动作，两种主要的实现算法包括蒙特·卡罗树搜索（Monte Carlo tree search，MCTS）[3-4]算法和滚动层进化（rolling horizon evolution，RHE）[5]算法；三是主要用于优化复杂的架构设计的进化算法[6]（evolutionary algorithms，EAs）。在通用人工智能的不同领域中，将这三者在不同尺度下应用或者混合使用都可以取得不错的效果。

游戏智能（game AI）即游戏与通用人工智能的结合。对于游戏而言，一方面，游戏智能不但可以在没有很好的解决思路的情况下提供一种智能决策的方法，还可以提供一种创造性的过程，以此设计更好的游戏；另一方面，因为游戏一般基于虚拟的系统或平台构建，在现实中是无害的，而且容易实现快速仿真，因此成了一个适合通用人工智能研究的良好场景。

游戏智能存在着大量的挑战。例如，如何实现游戏的快速学习？游戏中有大量任务、开放的决策空间以及部分可观测所带来的困难（如有“战争迷雾”的游戏）。这些挑战会影响玩家（或智能体）对游戏世界中状态的观察，也会对其他个体的置信度和目标产生不确定性影响，所有的玩家都无法知道其他的个体会如何做出决策或者行动，这些挑战非常有趣，同时也非常复杂。

传统的常用游戏智能技术包括有限状态机[7]（finite state machine）、脚本语言（scripting language）、模糊逻辑[8-9]（fuzzy logic）、决策树[10]（decision tree）、神经网络[11-12]（neural network）、遗传算法[6,13]（genetic algorithm）等。

- 有限状态机是计算机游戏中应用非常普遍的一个人工智能模型。在游戏中可以构成管理游戏世界的基础、模拟玩家的情绪状态、维持游戏的状态、分析玩家的输入、管理对象的状态[14]。可以用状态转移图的方式描绘状态机，状态转移图中的节点表示不同的状态，不同状态之间由于某种预定义的触发条件而发生转换。
- 脚本语言是一种解释性语言，通常用于控制游戏中的 AI 模式，它在游戏中可以驱动事件、为非玩家角色的智能行为建模、实现某些任务的自动化等功能，集中常见的脚本有 JavaScript、VBScript、Perl、PHP、Python、Ruby、Lua 等。
- 模糊逻辑采用实数值来表示对象属于集合的程度。与传统逻辑相比，它的表达能力更为丰富和细致，因而能够进行更好的推理。模糊逻辑常被用于游戏中的战略决策、输入输出信息的过滤、非玩家角色的健康状态计算以及情绪的状态变化等[15-17]。
- 决策树一般都是自上而下生成的。每个决策或事件（即游戏中的系统状态）都可能引发两个或多个事件，导致不同的结果，将这些决策分支画成图形会形成一棵树的形状，故而称为决策树[18]。决策树类似于一系列 if-then 形式的条件判断，常被用于游戏中的分类、预测和学习[19]。
- 神经网络是一种通过模仿动物神经网络的行为特征，进行分布式并行信息处理的算法数学模型。这种网络依靠系统的复杂程度，通过调整内部大量节点之间相互连接的关系，达到处理信息的目的，在游戏中，神经网络可用于分类、预测、学习、模式识别、行为控制等[20]。
- 遗传算法是近年来发展起来的一种新的全局优化算法，它借用仿真生物遗传学和自然选择机理，通过自然选择、遗传、变异等机制，来提高个体的适应性。从某种程度上说，遗传算法是对生物进化过程进行的数学仿真。遗传算法的通用实现过程包括：对待解决问题进行编码，对群体进行随机初始化，计算群体上每个个体的适应度，评估适应度并计算当前群体中每个个体的适应度，按预定义的选择算子产生后代，对后代进行交叉操作及变异操作，然后判断是否满足停止条件，最终进行循环操作或者输出种群中适应度最优的个体。遗传算法技术试图直接模拟生物进化过程，在一系列的程序、算法和参数之间做出选择，进行杂交以及随机的变异和交叉操作。在游戏中，遗传算法可用于优化、学习、策略形成、行为进化等[21]。

然而，以上这些传统的游戏智能技术通常都依赖于大量的人工干预（例如人工特征的设计），同时对模型的性能以及游戏的结果提升也都有一定的限制。

近年来，深度学习的出现和兴起改变了人工智能的世界。十年前，深度学习还处于萌芽期时，很多人工智能方面的问题难以解决，但是深度学习在一些问题（包括计算机视觉、人脸识别、语音识别等）的研究上还是取得了突破性的进展。深度学习在游戏智能领域取得了令人惊异的成果，例如 DeepMind 利用深度学习解决了很多任务和挑战[1,22]，基于深度学习的游戏智能技术逐步在围棋 AlphaGo[1]、星际争霸[23-24]、德州扑克[25-26]等复杂游戏中战胜了人类

最顶尖的玩家。

但是深度学习的一个显著问题在于：如果要复现他们的结果，需要付出很大的时间代价。如果采用 SFP 算法，这个过程的学习和适应会快很多。此外，由于 SFP 算法是基于前向模型（forward models，FM）的仿真方法，可以更清楚地了解到为什么得到了这样的决策，获得比深度学习更好的可解释性分析。可以预期，SFP 算法也将在人工智能的很多领域做出自己的贡献。

2　统计前向规划

SFP 算法是一种具有强泛化性和快速适应性的 AI 方法。SFP 算法相比深度学习的一个很大的优势在于：针对新的任务，不需要重新对算法进行训练。SFP 算法可以使用同一个算法以及同样的参数，去玩两种不同的游戏，只需要快速地复制系统状态，然后通过前向模型快速地迭代优化。对于每一个决策的迭代优化，通常要经过 1 000 次以上的仿真。

2.1　概念

为了做出决策，SFP 算法首先复制目前世界（游戏系统）的状态，然后对一系列动作进行优化，最后选择一个最有可能带来好的结果的动作，这是 SFP 算法的核心。很多游戏的情境和规则设计非常复杂，难以判断什么是真正的最优决策。对于这种复杂的情况，得到一个确切的最优决策是不可能的，所以优化目标是找到一个可能带来好的结果的动作。

当然，游戏中很多情境都会有一些不确定性，也会存在信息的缺失问题，另外，在游戏中玩家还会面对存在竞争或合作关系的未知个体，玩家不但需要决策自己的动作，还需要猜测或预估其他玩家的动作，使得做出最优决策存在很大的困难。所以，SFP 算法最终的目标并不是找到最优解，而是保证其在某些方面的性能可以超越其他的方法。

2.2　前向模型

SFP 算法使用前向模型（也叫做世界模型），前向模型的辨识是 SFP 算法中最重要的一个问题。这样的模型在游戏中是永远存在的，但不是总能获取到，比如有可能是一段无法访问的计算机代码。对于真实世界的问题，这种模型可能是手动编程得到的，或者通过数据学习得到的。值得注意的是，不一定只有完美的模型才有用，有些实验甚至需要故意破坏模型，通过测试不完美模型条件下的结果来增强泛化性能。

如果已经存在一个前向模型，在该模型里应用 AI 算法时，首先需要设置这个模型的初始状态，接着复制当前游戏的系统状态，然后复制下一步系统采取的动作。注意对于多玩家游戏，这个动作来源于多个个体，算法需知道动作的数量、得到的分数，还有游戏是否已经结束。

前向模型的定义为：给定一个系统（游戏）状态 S_t，S_t 可通过多个传感器感知得到（S_t^1，S_t^2，…，S_t^n），状态转换函数 f 将系统的当前状态和智能个体的动作 $A_t \in \mathcal{A}$（预定义的动作空间）映射到下一个系统状态 S_{t+1}，即：

$$f:\mathcal{S},\mathcal{A}\to\mathcal{S},\quad S_t,A_t\mapsto S_{t+1} \tag{1}$$

其中，$\mathcal{S}$ 代表系统的状态空间。

如何学习前向模型（状态转换函数 f）是一个很大的挑战，以往大部分的研究通常倾向于运用深度学习的方法，例如采用一种编码解码器的结构来学习模型，但是这种方法有时候比较困难，尤其是在学习小的格子的时候，深度学习需要进行很长时间的训练学习，但也并不能取得很好的效果。如果只考虑本地交互（或者可能主要是本地交互），那么前向模型就会变得更容易实现。这种方法的动机来自参考文献[27]的实验，在实验中，玩家及其周围物体的局部模型被用来预测游戏的未来状态及其获得的奖励。因此本文提出了学习局部模型的方法。在现实的游戏中，学习一个完整的状态转移函数非常困难，因此本文选择聚焦在局部区域，学习局部的模式，比如可以在一个 3×3 的邻域内学习当前元胞的状态转移，这样需要学习的格子数目就大大减少了。模型评估方法很多，可以尝试利用监督学习训练过程评估已经学习到的模型，或者可以做出一个 N 步的预测来评估这个模型，也可以直接验证游戏中的表现。在这些评估方法中，局部模型都能取得非常好的结果，并且相比深度学习的硬件要求，这些方法用笔记本电脑几分钟就能完成局部模型的学习过程。

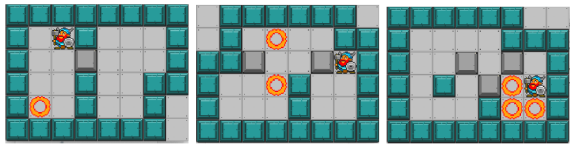
在前向模型定义的基础上，参考文献[28]给出了局部模型的定义：直接学习一个完整的状态转移函数会导致将整个系统的状态转换和所有传感器值的转换都映射到一个单一的转换函数中，而局部模型将游戏的转换函数分为多个组件，每个组件映射一个可观测值子集的转换或一个单一值的转换，即：

$$f_i:\mathcal{S},\mathcal{A}\to\mathcal{S}^i,\quad S_t,A_t\mapsto S_{t+1}^i \tag{2}$$

其中，$\mathcal{S}^i$ 代表某个传感器的观测值。

当每个可观测值的转换函数独立于系统状态的某些组件，从而导致转换函数组件的数量减少时，局部模型非常有用。特别是在基于网格的游戏中，这一特性表现为缺乏全局效应，因为某些单元格的未来状态只需要通过观察单元格的局部邻域来确定。

以推箱子游戏来说明这一学习过程，推箱子属于一个解谜游戏，如图 1 所示，玩家（小矮人）必须将所有的箱子（灰色方块）推入目标（橙色圆圈）才能通过关卡。如果想把箱子推到指定位置，需要先把箱子推到另一侧，让游戏的个体能够把箱子从另一个方向推动到目标位置，这是一个典型的基于网格的游戏模型。这个游戏可以使用局部模型依次建立前向模型，从而确定对应动作。



（a）关卡 1　　（b）关卡 2　　（c）关卡 3

图 1　推箱子游戏示例

在这个游戏中，状态可以表示为一组排列在网格中的瓦片，其中 $T(x,y)$代表处于网格中位置为(x,y)的瓦片。游戏的目标是根据单元格的状态 A_t 以及它在 t 时刻的局部邻域状态 $N(x,y)_t$ 来预测每个瓦片在 t+1 时刻的状态 $T(x,y)_{t+1}$。令局部状态转换函数 $f_{x,y}$ 为：

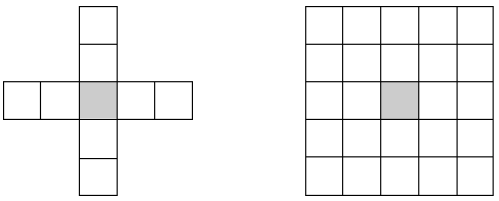
$$f_{x,y}:N(x,y)_t,\quad A_t\mapsto T(x,y)_{t+1} \tag{3}$$

在这个游戏中，考虑两种类型的局部邻域状态，分别为交叉模式（cross pattern）和方形模式（square pattern），如图 2 所示，这两种模式可以表示为：

$$N_{\text{cross}}(x,y)=\{T(x+i,y)\,|\,0\leqslant|i|\leqslant\text{span}\}\cup\{T(x,y+j)\,|\,0\leqslant|j|\leqslant\text{span}\} \tag{4}$$

$$N_{\text{squa}}(x,y)=\left\{T(x+i,y+j)\,\middle|\,\begin{matrix}0\leqslant|i|\leqslant\text{span}\\0\leqslant|j|\leqslant\text{span}\end{matrix}\right\} \tag{5}$$

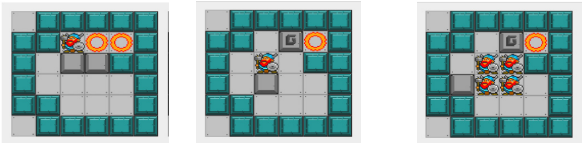
其中，span 代表跨度或者步长。实验中对不同的跨度进行测试评估，可以研究为局部模型学习提供过多或不足的信息对算法造成的影响。



（a）交叉模式　　（b）方形模式

图 2　用于预测中心瓦片下一个状态的本地局部邻域模式

有很多实现局部正向模型学习的算法，例如 Hash 图和决策树模型。不完美的模型也是很有用的，可以提供不错的结果。从图 3 可以看到，3 种状态的结果基本是一致的，但是有的时候也会出现一些错误，不过正如之前说的，不是完美的方法才有用，这个局部模型在很大程度上是适用的。



（a）初始游戏状态　　（b）中间状态　　（c）不完美模型的估计状态

图 3　不完美模型的结果

2.3　可解释性

SFP 算法相对于深度学习的另一个优势在于它具有更强的可解释性。因为 SFP 算法采用基于前向模型的仿真方法，可以通过仿真的结果更清楚地了解到为什么得到了这样的决策。

对于一个 AI 方法，一个有利的特征就是可调节性。在进行某个游戏的时候，玩家常常希望有很多不同能力的人物个体（非玩家角色等），SFP 算法可以通过调节序列的长度来调节个体对未来情况的影响范围，通过调节评估次数来选择决策的精细度，通过调节移位缓冲获得不同决策间的连续性，这是算法很关键的一部分。

利用前向模型，结合 SFP 算法，可以得到一个强大的 AI 方法（这依赖于快速的前向模型），这样就可以实时地进行 1 000 次以上的仿真，模型的表达式如下所示。

$$S_{t+1}=F(S_t,A_t) \tag{6}$$

其中，F 代表模型，S_t 代表当前状态，A_t 代表动作。通过这个模型，可以得到执行当前动作之后的下一个状态 S_{t+1}。

3　蒙特•卡罗树搜索

SFP 算法并不是一个新的算法，其典型案例

可以追溯到 1998 年利用蒙特•卡罗树搜索（MCTS）算法解决拼字游戏，拼字游戏是一个很复杂的游戏，它有很多的不确定因素，玩家不知道会得到哪一张卡片，需要很多的随机仿真来解决，因此效果不是非常好。2006 年，MCTS 算法被用在围棋游戏中，这改变了之前人工制定策略的方法，性能上有了很大的提升。本文将这种算法应用在电脑游戏中，在算法性能上同样带来了显著的提升。

3.1　算法

MCTS 算法利用随机仿真的结果迭代构建搜索树，直到达到一些预定义的计算预算（通常是时间、内存或迭代约束）搜索停止，并返回性能最佳的根操作。搜索树中的每个节点表示系统的状态，指向子节点的链接分别表示指向后续状态的动作。每次搜索迭代应用以下 4 个步骤。

- 选择：从根节点开始，递归应用子选择策略遍历树，直到到达最近的可扩展节点。如果一个节点表示一个非终端状态，并且它有未被访问过的子节点，那么这个节点就是可扩展的节点。
- 扩展：根据有效决策的动作，添加一个（或多个）子节点来扩展树。
- 仿真：根据默认策略（default policy）从新的节点模拟仿真，生成结果。
- 反向传播：将仿真结果向后传递，通过所选节点（即反向传播更新）更新节点的统计信息。

上述步骤采取的策略可以分为以下两种。

- 树策略（tree policy）：从搜索树已经包含的节点中选择或创建叶子节点（选择和展开步骤）。
- 默认策略（default policy）：从给定的非终端状态开始运行，生成一个估计值（仿真步骤）。

参考文献[29]给出了一般的 MCTS 算法流程，如算法 1 所示。v_0 是对应于状态 s_0 的根节点，v_l 是树策略阶段中到达的最后一个节点，对应于状态 s_l，Δ 是从状态 s_l 运行默认策略到达的某个终端状态所获得的奖励。整体搜索 a (BESTCHILD（v_0）) 的结果是指向根节点 v_0 的最佳子节点的操作动作 a，其中“最佳”的确切定义由实际游戏的规则和评分确定。参考文献[30]给出了在一些实时视频游戏中运用 MCTS 算法的实验结果，实验结果展示了 MCTS 算法能够处理非常细粒度的搜索。游戏中每次决策的操作动作 a（即对建立的树进行导向的方式）由式（7）确定。

$$a^* = \arg\max_{a \in A(s)} \left\{ Q(s,a) + C\sqrt{\frac{\ln N(s)}{N(s,a)}} \right\} \qquad (7)$$

其中，C 是超参数。

式（7）的意义是每个节点都对从状态 $N(s,a)$ 开始移动的频率、从状态 $N(s)$ 开始移动的次数和平均奖励分数值 $Q(s,a)$ 三者相关的统计信息进行跟踪，即反向传播步骤中利用状态评估获得的奖励，对遍历的每个节点更新统计量 $N(s,a)$、$N(s)$ 和 $Q(s,a)$，循环执行直到满足终止条件。

算法 1　蒙特•卡罗树搜索算法

function MCTSSEARCH（s_0）
　创建根节点 v_0 及对应状态 s_0
　While 计算预算以内 **do**
　　$v_l \leftarrow$ TREEPOLICY（v_0）
　　$\Delta \leftarrow$ DEFAULTPOLICY（$s(v_l)$）
　　BACKUP（v_l, Δ）
return a (BESTCHILD（v_0）)

3.2　特性

MCTS 算法的主要特性包括非启发式（aheuristic）、随时性（anytime）以及不对称性（asymmetric），这些特性使其成了很多领域的流行算法，并取得了显著的成功。

（1）非启发式

MCTS 算法最重要的优点之一是不需要领域特定的知识，这使得它很容易适用于任何可以使用树进行建模的领域。例如有限深度极大极小（depth-limited minimax）算法的博弈质量在很大程度上取决于用于评估叶节点的启发式算法。在国际象棋这样的游戏中，经过几十年的研究，利用可靠的启发式（heuristics），极大极小算法取得了很好的效果。然而，在像 Go[1]这样的游戏中，分支因子要大几个数量级，而有用的启发式又很难形成，此时极大极小算法的性能就会显著下降。这类问题就需要利用 MCTS 算法来解决。

虽然 MCTS 算法可以在没有启发知识的情况下应用，但是使用特定领域的知识通常可以显著提高性能。目前性能最好的基于 MCTS 算法的围棋程序都使用特定游戏的知识信息，通常利用模式的形式表现。这些知识信息不要求非常完备，只需要在某些模式下提供一个更具偏向性的移动选择（例如对某个移动决策附加偏置项，从而提升其被选择的概率）来保障该模式的利益最大化。在使用领域特

定的知识提供偏向性选择时，还需要考虑一些性能与通用性以及速度的权衡：均匀随机的移动选择允许在给定的时间内执行多次模拟，具有更大的速度优势，而基于特定领域知识的移动选择通常会极大地减少模拟数量，也会导致减少模拟结果的方差。参考文献[31]讨论了游戏中决策选择应该包括特定的知识信息的程度，以及性能与速度的权衡。

（2）随时性

MCTS 算法能够及时地对每一次游戏模拟执行反向传播并输出结果，从而保证所有的参数值在 MCTS 算法执行每次迭代后都是最新的。MCTS 算法在任何时刻都允许从树的根节点返回一个动作决策，同时允许进行额外的迭代来改进决策结果。

（3）不对称性

树搜索和选择允许 MCTS 算法优先选择有更大希望的节点，同时不允许其他节点的选择概率收敛到零，随着时间推移，这种选择策略会导致树呈现不对称性。树的部分构建倾向于更有希望的、也更重要的节点区域，即非对称树生长。最终出现的树的形状可以用来更好地理解它所构造的某个游戏。例如，参考文献[32]证明了应用上限置信区间（upper confidence bound apply to tree，UCT）搜索生成的树的形状的方法，可以区分可玩游戏和不可玩游戏。

3.3　应用分析

以往的对抗性游戏主要有两种决策方法：极大极小搜索算法和基于启发式知识的方法。MCTS 算法提供了第三种有效方法，尤其对于那些很难评估中间游戏状态或很难获取足够详细规则的游戏，MCTS 算法提供了一种有效的途径，并在许多情境中取得了迄今为止最强的游戏效果。同时 MCTS 算法与传统方法的融合提供了一个值得探索研究和开发的丰富领域。虽然极大极小搜索算法已被证明是一种有效的游戏算法，但由于这类算法需要评估游戏的中间状态，因此适用范围较小（如国际象棋和跳棋），MCTS 算法则不需要这样的中间状态评估，并已被证明是一种更稳健和通用的搜索方法。它在广泛的游戏和非游戏的应用中都取得了更好的实践效果，表明了它在许多决策问题上的潜力。

MCTS 算法尤其强大的一点是其可以在不了解游戏规则的情况下应用，即只需要通过对树或仿真策略进行进一步增强，就可以取得有效的游戏效果。增强可能来自于融合人类知识、机器学习或其他启发式方法。MCTS 算法最大的优点是，即使增强所提供的信息存在噪声干扰，或者偶尔会产生误导，MCTS 算法的采样方法通常也足够稳健，能够处理这种噪声并产生有效的结果，这远优于极大极小搜索算法，在极大极小搜索算法中，中间状态的评估函数和搜索策略往往对噪声非常敏感，对于延迟奖励的游戏尤其如此。MCTS 算法的另一个优点是，它采用的前向抽样方法在某些方面与人类游戏玩家使用的方法相似（即可解释性的问题），因为该算法专注于更有希望和前景的路线，而偶尔检查明显较弱的选项。对于某些并不存在战略或启发式知识的游戏，MCTS 算法的这种性质使得其可解释性比某些使用与人类方式截然不同的搜索范例更强。MCTS 算法通常对于少量的模拟是有效的，这些模拟中出现的错误对于人类观察者来说也是合理的。因此，MCTS 算法是一种真正的随时方法，通过不对称地增长树来生成更可信的结果。

尽管 MCTS 算法将树搜索的精确性与随机抽样的通用性结合，在大范围的博弈中提供了更强的决策能力，但其应用仍存在着很多挑战和瓶颈。在某些领域，当需要搜索的图的分支因子和深度被限定，使得简单的 MCTS 算法（或者实际上任何其他搜索算法）不可行时，MCTS 算法的应用就会存在很大的挑战。尤其是在视频游戏和实时控制应用程序中，为了限制要搜索的子树，需要强大的系统方法来整合知识，MCTS 算法变得不再适用。应用 MCTS 算法的另外一个问题在于，当计算机模拟非常耗费 CPU 资源时，MCTS 算法必须从相对较少的样本中学习，并进行非常浅层的搜索，这种情况下可以执行的模拟相对较少，MCTS 算法是否仍然为指导模拟的最佳方法是一个有待研究的问题。另外，MCTS 算法的弱点还在于搜索的动态不能被完全理解，参数设置和基本算法增强两方面的决策产生的影响很难预测。

未来的研究可能会对 MCTS 算法的性能有更好的理论理解。如何提高 MCTS 算法的整体性能或提高 MCTS 算法在特定领域的性能，以及理解 MCTS 算法的行为，可能是未来 MCTS 算法的重要研究方向。

4　滚动层进化

滚动层进化（RHE）算法也是一种常用的 SFP 算法。RHE 算法是演化算法的一个子集，它使用种

群的个体来表示行动计划或行动序列，这些个体通过前向模型模拟向前移动来评估。从游戏的当前状态开始，所有的动作（个体的基因）都按顺序执行，直到达到一个终端状态或个体的长度；然后用一个启发式函数和分配给个体的适应度来评估在那一点上达到的状态。

通常，该算法从随机的个体种群开始。在每一个博弈步骤中，首先应用传统的遗传操作（如变异（随机改变序列中的一些动作）、交叉（以不同的方式组合个体））来获得下一代种群的新个体；然后对每一个物种进行评估并分配适应度，再根据适合度对种群进行排序，只有最好的物种才会传给后代；当满足结束条件时（例如达到时间或内存限制，或者执行了一定数量的迭代时），流程结束。

在游戏应用中，RHE 算法是一种实时进化的算法，它在每个回合都为玩家推荐一个动作。进化使用方式与 MCTS 算法使用 roll-outs 和生成模型（模拟器）的方式相同。智能体在一个假想模型中按照某个设计进化计算，通过执行该设计的第一个动作对系统状态进行操作，然后重复地进化计算一个新的设计（同样是基于模拟的方式），直到游戏结束。这种行为被称为滚动层（rolling horizon）、后退层或模型预测规划。滚动层术语起源于只有当游戏存在有限的未来决策时才能够进行设计的规划，因此要求每一步都不断地重新规划。参考文献[5]使用的 RHE 算法标准形式中，RHE 算法进化出包含 L 个动作的序列，每个序列的适应度都应用前向模型向前滚动 L 步骤到达的状态来计算，算法最终评估操作序列结束时达到的状态。评估方法可以使用参考文献[33]中介绍的自定义状态启发式算法。RHE 算法使用 N 个个体的种群，并应用常规的进化算子，如变异、交叉和精英主义。在计算预算之内循环执行后，RHE 算法返回在游戏中找到的最佳序列的第一个动作。

具体的算法过程如算法 2 所示，FM 代表前向模型。首先给定一个最优序列（目前来讲还不算最优，只是一个随机初始化的序列），对于游戏中要做的每一个决策，重复以下步骤：首先，变化并随机增补原有的最优序列，然后，在每次评估中对目前最好的一个序列进行变异操作，如果在目前的前向模型下，变异后的评分大于当前的评分，那么就用变异后的序列替代当前最优序列，最后循环结束后，返回最优的序列的第一个值。

算法 2　滚动层进化算法

初始化　bestSeq **:=**　一个长度为 N 的随机整数序列

For　游戏中的每一个决策：

bestSeq = shiftAndRandomAppend (bestSeq)

While　（计算预算以内）**do**

mutSeq = mutate(bestYet)

if (score(**FM**(mutSeq)) > score (**FM**(bestSeq))

bestSeq = mutSeq

return first(bestSeq)

为了更具体地说明这个算法是如何工作的，假定初始化的可能的动作序列为推进、左转、开火、开火、左转，评分是 250 分，经过变异以后，左转变成了右转。后面也可能有一系列其他变化，结果发现评分增加到了 350 分，那么选择后面的这个序列代替第一个序列，算法只考虑评分，不关心具体动作。算法片段如下所示。

bestSeq : =　[推进、左转、开火、开火、左转，…]

评分 bestSeq　→　+250 points

变异：

mutSeq = [推进、右转、开火、开火、推进，…]

评分　mutSeq　→　+350 points

+350 points > +250 points

bestSeq = mutSeq

本文以星球大战为例来说明这个算法如何处理很复杂的游戏。星球大战游戏的规则是：每个行星都以一定的速度产生飞船，行星是有主人的，可以是游戏中的 1 号玩家、2 号玩家或中立行星，中立行星不产生飞船。动作的搜索空间就是飞船的移动方向，玩家可以选择把一定数量的飞船从一个星球送到另外的星球。更新规则是：当飞船舰队到达一个星球，如果这是盟友玩家拥有的星球，则该星球就会加入相应数量的飞船；如果这是敌对玩家拥有的星球，则减去相应数量；如果玩家所拥有的星球上的飞船数目变为负值，那么这个星球就会易主。游戏最终的目标是把敌对星球全部消灭。这个游戏的规则很经典，也很简单，但是其中的策略非常复杂。参考文献[30]设计了一个星球大战版本，并将其应用在一个新的游戏智能平台上，这个平台比一般平台快 10 倍，非常适合 SFP 算法，也非常灵活，可以设置不同的地图大小。

这个游戏最基本和核心的策略是：尽快得到更

多的星球。当用飞船侵略对手星球的时候，只要使敌对星球的分数变成负数即可，不要浪费太多飞船；当用飞船占领中立星球的时候，会失去一部分飞船，所以自己的星球可能会被对手侵略，因此要非常小心。还需要注意的是不能在一个星球上积攒太多的飞船，需要留足自己持有的飞船，防止对手侵略。还要避免长途迁徙，因为迁徙期间飞船是不能使用的。在这个游戏中应用滚动层进化算法时，序列中的元素表示星球，在游戏引擎中，作为源和目标对飞船进行传递。假设总共有 20 颗行星，序列长度是 100，那么搜索空间是 20^{100}，可以看出，搜索空间是庞大的，因此不必追求绝对的最优。但是实验表明，这个游戏引擎可能会采取一些非法的动作（指游戏规则中没有规定的动作，但可能因为规则漏洞导致玩家可以采用）来获取胜利，因此还需要排除这些情况。

游戏有很多复杂的参数，通过实验研究可以发现很多参数调整的经验，例如使用更长的 roll-outs 长度，并在合适的地方加入衰减因子。一般来说，使用移位缓冲效果会好很多，变异的强度最好大一些。为了应对平缓的奖励区域，可以采用改变评分函数或者加入极大似然估计等方法。比如在推箱子游戏中，不是仅奖励达到最终的结果的情况，而是在将每一个箱子推到相应位置时都给予奖励，实验表明，这样的设计可以防止出现找不到最优解的情况。还有诸如加强探索的方法以及其他的一些备选方案，也同样值得实验研究。

5　结束语

总体来说，统计前向规划方法是一种使用前向模型来模拟可能的未来状态的方法。这些方法是一组稳健的、通用的、随机的人工智能算法，在很多问题上都取得了很好的效果，同时它适应性很强，只需要快速地复制前向模型就可以取得很好的泛化性能。采用错误的模型参数进行的实验也说明，不是完美的模型才有效。

统计前向规划方法的例子有：蒙特•卡罗树搜索算法（它使用随机正向模拟来构建一个搜索树）；滚动层进化算法（它使用相同的随机正向模拟来演进一个计划）。

除了游戏智能以外，统计前向规划在人工智能领域还有很多潜在的应用场景，例如真实世界的决策等。

统计前向规划算法和游戏通用人工智能面临很多开放的挑战和困难，比如如何采用长期的强化学习来提升游戏人工智能的性能，如何更好地学习前向模型。另外，无限的动作搜索空间问题（目前应用的游戏系统都是有限的搜索空间）以及在现实世界应用的问题，都是未来需要研究的内容。

参考文献：

[1] SILVER D, HUANG A, MADDISON C J, et al. Mastering the game of Go with deep neural networks and tree search[J]. Nature, 2016, 529(7587): 484.

[2] GAINA R D, LUCAS S M, PÉREZ-LIÉBANA D. Tackling sparse rewards in real-time games with statistical forward planning methods[C]//Proceedings of the AAAI Conference on Artificial Intelligence. January 27-February 1, 2019, Hawaii, USA. California: AAAI Press, 2019, 33: 1691-1698.

[3] SHEPPARD B. World-championship-caliber scrabble[J]. Artificial Intelligence, 2002, 134(1-2): 241-275.

[4] GELLY S, WANG Y. Exploration exploitation in go: UCT for Monte-Carlo go[C]//NIPS: Neural Information Processing Systems Conference On-line trading of Exploration and Exploitation Workshop, December 4-7, 2006, Vancouver, Canada. Cambridge: MIT Press, 2007.

[5] PEREZ D, SAMOTHRAKIS S, LUCAS S, et al. Rolling horizon evolution versus tree search for navigation in single-player real-time games[C]//Proceedings of the 15th Annual Conference on Genetic and Evolutionary Computation, July 6-10, 2013, Amsterdam, The Netherlands. New York: ACM Press, 2013: 351-358.

[6] BANZHAF W, KOZA J R, RYAN C, et al. Genetic programming[J]. IEEE Intelligent Systems, 2000, 15(3): 74-84.

[7] GILL A. Introduction to the theory of finite-state machines[M]. New York: McGraw-Hi1l, l962.

[8] ZADEH L A. Fuzzy sets *[J]. Information & Control, 1965, 8(3): 338-353.

[9] TAKAGI T, SUGENO M. Fuzzy identification of systems and its applications to modeling and control[J]. IEEE Transactions on Systems Man & Cybernetics, 2012, SMC-15(1): 116-132.

[10] QUINLAN J R. Induction of decision trees[J]. Machine Learning, 1986, 1(1): 81-106.

[11] HECHT-NIELSEN. Theory of the backpropagation neural network[C]// International Joint Conference on Neural Networks, May 12-17, 2002, Honolulu, USA. Piscataway: IEEE Press, 2002.

[12] HAGAN M T, DEMUTH H B, BEALE M. Neural network design[M]. Beijing: China Machine Press, 2002.

[13] GOLDBERG D E. Genetic algorithms in search, optimization and machine learning[J]. Addison-Wesley, 1989.

[14] DIANTY R E, AZHARI A M, HAKIM M F A, et al. First aid simulation game with finite state machine model[C]// International Conference on Information & Communication Technology & Systems, October 12, 2016, Surabaya, Indonesia. Piscataway: IEEE Press, 2016.

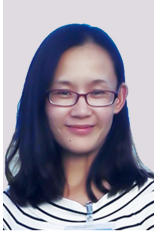
[15] ORLOVSKY S A. Decision-making with a fuzzy preference relation[J]. Fuzzy Sets & Systems, 1978, 1(3): 155-167.

[16] FREELING A N S. Fuzzy sets and decision analysis[J]. IEEE Transactions on Systems Man & Cybernetics, 1984, 10(7): 341-354.
[17] STOICA M, DAN N, ANDREICA A, et al. Fuzzy sets and their applications[C]// Proceedings of the 9th WSEAS International Conference on Mathematics & Computers in Business and Economics (Mcbe '08) , June 24-26, 2008, Bucharest, Romania. [S.l.:s.n.], 2008.
[18] SAFAVIAN S R, LANDGREBE D. A survey of decision tree classifier methodology[J]. IEEE Transactions on Systems Man & Cybernetics, 2002, 21(3): 660-674.
[19] KONISHI M, OKUBO S, NISHINO T, et al. Decision tree analysis in game informatics[J]. Applied Computing & Information Technology, 2017.
[20] LIU D, LI H, DING W. Neural-network-based zero-sum game for discrete-time nonlinear systems via iterative adaptive dynamic programming algorithm [J]. Neurocomputing, 2013, 110(8): 92-100.
[21] REVELLO T E, MCCARTNEY R. Generating war game strategies using a genetic algorithm[C]// Congress on Evolutionary Computation, May 12-17, 2002, Honolulu, Hawaii, USA. Piscataway: IEEE Press, 2002.
[22] KUN S, YUANHENG Z, DONGBIN Z. StarCraft micromanagement with reinforcement learning and curriculum transfer learning[J]. IEEE Transactions on Emerging Topics in Computational Intelligence, 2018: 1-12.
[23] VINYALS O , EWALDS T , BARTUNOV S , et al. StarCraft II: a new challenge for reinforcement learning[J]. Science, 2017, 359(6374): 1733.
[24] MORAVČÍK M, SCHMID M, BURCH N, et al. DeepStack: expert-level artificial intelligence in heads-up no-limit poker[J]. Science, 2017, 356(6337): 508.
[25] BROWN N, SANDHOLM T. Superhuman AI for heads-up no-limit poker: libratus beats top professionals[J]. Science, 2017, 359(6374): 1733.
[26] MNIH V, KAVUKCUOGLU K, SILVER D, et al. Human-level control through deep reinforcement learning[J]. Nature, 2015, 518(7540): 529.
[27] DOCKHORN A, APELDOORN D. forward model approximation for general video game learning[C]//2018 IEEE Conference on Computational Intelligence and Games (CIG), August 14-17, 2018, Maastricht, The Netherlands. Piscataway: IEEE Press, 2018: 1-8.
[28] DOCKHORN A, LUCAS S M, VOLZ V, et al. Learning local forward models on unforgiving games[J]. arXiv Preprint, 2019.
[29] BROWNE C B, POWLEY E, WHITEHOUSE D, et al. A survey of monte carlo tree search methods[J]. IEEE Transactions on Computational Intelligence and AI in games, 2012, 4(1): 1-43.
[30] LUCAS S M. Game AI Research with fast planet wars variants[C]//2018 IEEE Conference on Computational Intelligence and Games (CIG), August 14-17, 2018, Maastricht, The Netherlands. Piscataway: IEEE Press, 2018: 1-4.
[31] DRAKE P, UURTAMO S. Move ordering vs heavy playouts: where should heuristics be applied in Monte Carlo Go[C]//Proceedings of the 3rd North American Conference on Games, 2007: 171-175.
[32] WILLIAMS G M J. Determining game quality through UCT tree shape analysis[D]. London: Imperial College, 2010.
[33] PEREZ-LIEBANA D, GAINA R D, DRAGESET O, et al. Analysis of statistical forward planning methods in Pommerman[J]. Proceedings of the 15th AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment, October 8-12, 2019, Atlanta, Georgia. Palo Alto, California: The AAAI Press, 2019.

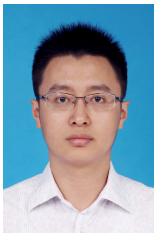
[作者简介]

LUCAS Simon（1972− ），男，博士，伦敦玛丽女王大学教授，主要研究方向为基于计算智能技术的游戏 AI、基于 AI 的游戏设计、通用人工智能。

沈甜雨（1996− ），女，湖北天门人，中国科学院自动化研究所复杂系统管理与控制国家重点实验室博士生，主要研究方向为机器学习、深度学习和计算机视觉。



王晓（1988− ），女，博士，山东滨州人，中国科学院自动化研究所复杂系统管理与控制国家重点实验室副研究员，主要研究方向为社会交通、动态网群组织、人工智能和社交网络分析。



张杰（1982− ），男，博士，中国科学院自动化研究所复架系统管理与控制国家重点实验室副研究员，主要研究方向为最优控制、博弈论与平行控制。