doi：10. 12068/j. issn. 1005－3026. 2026. 20250016

面向计算机生成兵力的战术决策行为建模方法

刘 军[1]，梁宇婷[1]，刘向军[2]
（1. 东北大学 计算机科学与工程学院，辽宁 沈阳 110169；2. 中软信息系统工程有限公司，北京 100081）

摘 要：在作战仿真中，建立指挥人员战术行为决策模型至关重要. 然而，计算机生成兵力（CGF）在决策过程中面临状态动作搜索空间大、决策时间有限等挑战. 为此，提出一种基于蒙特卡罗树搜索（MCTS）的战术决策方法，并引入分层任务网（HTN），利用领域知识引导MCTS节点扩展. 此外，传统MCTS在选择对手行动时采用等概率策略，导致对手行为不确定. 针对这一问题，构建隐式对手建模方法，通过粒子滤波识别对手策略，并将预测结果整合至MCTS决策过程中，以增强决策的针对性. 仿真结果表明，在不同时间约束与地图条件下，该方法均能有效预测对手行动并实施针对性决策，从而显著提高胜率.

关 键 词：作战仿真；计算机生成兵力；隐式对手建模；蒙特卡罗树搜索；粒子滤波
中图分类号：TP 391.9 **文献标志码**：A **文章编号**：1005-3026（2026）05-0026-10

Modeling Method of Tactical Decision-Making Behavior for Computer Generated Forces

LIU Jun[1], *LIANG Yu-ting*[1], *LIU Xiang-jun*[2]
（1. School of Computer Science & Engineering, Northeastern University, Shenyang 110169, China; 2. China Software and Information System Engineering Company, Beijing 100081, China. Corresponding author: LIU Jun, E-mail: junliu@mail.neu.edu.cn）

Abstract: In combat simulation, establishing tactical behavior decision-making models for commanders is crucial. However, challenges such as a large state-action search space and limited decision-making time are faced by computer generated forces (CGF) during the decision-making process. To address this issue, a tactical decision-making method based on Monte Carlo tree search (MCTS) was proposed, and a hierarchical task network (HTN) was introduced to guide MCTS node expansion using domain knowledge. Furthermore, an equal-probability strategy was employed by traditional MCTS when selecting opponent actions, resulting in uncertain opponent behaviors. In view of this issue, an implicit opponent modeling method was constructed, in which opponent strategies were identified through particle filtering, and the prediction results were integrated into the MCTS decision-making process to enhance decision-making relevance. Simulation results indicate that under different time constraints and map conditions, opponent actions can be effectively predicted, and targeted decisions can be implemented by this method, thereby significantly improving the winning rate.

Key words: combat simulation; computer generated forces; implicit opponent modeling; Monte Carlo tree search; particle filtering

“从战争中学习战争”自古以来都被视为学习与研究战争规律的重要方法[1]. 然而，旧式的作战理念与理论分析方案已越来越难以适应当下军队信息化建设的需求. 现今作战仿真技术通过计算机平台实现了在虚拟战场上模拟特定的场景与对象，从而满足“在实验室学习战争”[2-3]的要求.

计算机生成兵力（computer generated forces，CGF）是由计算机生成和控制的虚拟作战兵力对

收稿日期：2025-03-06
基金项目：国家自然科学基金青年基金资助项目（62501133）.
作者简介：刘 军（1969—），男，辽宁沈阳人，东北大学教授.
通信作者：刘 军，E-mail：junliu@mail.neu.edu.cn.

象[4-6].随着计算机平台算力性能的不断提升,机器学习算法被应用于行为建模领域,并在作战仿真系统中获得日益增长的关注.2015年,加拿大空军实验室在ITSEC会议上发出利用机器学习进行计算机生成兵力行为建模的倡议,并阐述了引入机器学习的前景与挑战[7-8].2016年,DeepMind公司开发的AlphaGo[9]计算机程序首次在围棋对弈中战胜顶尖职业选手.2017年,AlphaGo被升级为AlphaGo Zero[10],该程序完全基于自我对弈和强化学习,在短时间内就达到了世界顶尖水平,并在围棋领域刷新了一系列记录.2018年,OpenAI Five[11]与世界顶尖DOTA2选手进行了一系列对抗赛,并在多场比赛中取得了胜利,标志着基于学习的任务规划决策由单人对抗转变为多人对抗决策.2020年,DeepMind提出了AlphaStar[12],这是一个在《星际争霸II》中击败人类职业选手的多智能体系统,展示了强化学习在复杂战场环境中的潜力.2021年,美国空军利用边缘AI技术开发了基于无人机的CGF系统,能够在复杂环境中快速响应.2023年,元宇宙概念被应用于军事训练和仿真,CGF在虚拟战场中的行为建模和决策能力得到进一步提升[13].

近年来,随着人工智能技术的发展,隐式对手建模在军事仿真、智能博弈等领域取得了显著进展.徐浩添等[14-15]将基于深度强化学习的对手建模方法分为显式建模和隐式建模两类.其中,隐式建模通过将对手行为特征嵌入智能体的策略网络中,使得模型在训练过程中自动学习对手的行为模式,无需显式构建对手模型,从而提高了模型的泛化能力和适应性.

本文将针对CGF状态动作搜索空间大、决策时间有限等问题,提出一种基于蒙特卡罗树搜索(MCTS)的战术决策方法.同时,针对传统MCTS在决策过程中对手行为不确定的问题,构建隐式对手建模方法,增强决策的针对性,从而在不同时间约束下提升搜索效率与胜率.

1　战术任务规划与决策

1.1　基于分层任务网的战术任务规划

分层任务网利用先验知识将复杂综合问题迭代分解为简单的子任务,迭代过程在所有任务均被分解为子任务或达到算法约束时间时截止.与分层任务网思想类似,战术任务规划过程中,复合任务总是被优先分解为各种子任务,在此基础上继续分解为原子任务以执行战术,两者均含有逐级分层解决问题的思想[16-18].

HTN规划的数学模型如式(1)所示:

$$P=\left(D,s_1,c_1\right). \tag{1}$$

式中:$D=\left(F,C_1,A,M\right)$表示任务规划时的领域知识,其中F是一组状态变量,C_1表示复合任务集合,A表示原子任务集合,M表示任务分解方法集合;$s_1\in F$表示初始状态;$c_1\in C_1$表示要执行的任务.

当给定需要规划的任务、初始状态和领域知识后,HTN将根据领域知识对初始任务进行分解[19-20].

以HTN算法为基础,结合战术任务规划知识,提出基于HTN的战术任务规划方案如下:

1)根据战术条例与专家经验构建作战任务、环境状态与需要执行的原子任务之间的关系,将此关系作为先验知识进行存储.

2)确定任务网络中任务间的关系和任务分解的前置条件.

3)将制定好的先验知识与环境初始状态输入HTN,对战术任务进行分解规划.

4)针对规划方案利用战场态势预测进行效果评估,对于效果较差的方案进行重新规划.

1.2　基于蒙特卡罗树搜索的战术行为决策

1.2.1　蒙特卡罗树搜索

蒙特卡罗树搜索(MCTS)是一种通过随机抽样构建搜索树以寻找最优决策的算法,其核心在于对特定状态下可行动作的博弈值进行近似[21-23].MCTS是一种典型的马尔可夫决策过程(MDP),其中未来状态只取决于当前状态及所采取的动作[24-25].在有限决策预算下,MCTS通过设置时间或迭代次数限制搜索.在迭代过程中,树策略和推演策略共同影响动作选择,树策略选择估值高的节点,而推演策略则通常采用随机选择.

1.2.2　战术行为决策过程构建

针对作战小队级别的战术任务规划与行为决策,更加关注对抗双方指挥实体进行作战推演,指挥多个实体进行对抗.将前向模型定义为

$$B=\{C,S,A,T_{\mathrm{r}},T_{\mathrm{e}},U\}. \tag{2}$$

其中:$C=\{I,Y\}$表示对抗双方的不同决策者,通常情况下,令I表示我方,Y表示敌方.S表示环境状态集合,通常指对抗双方实体在地图中的位置.$T_{\mathrm{r}}\left(s,a\right):S\times A\rightarrow S$表示状态转移函数,$a$表示状态$s$下双方实体执行的合法动作,执行$a$后状态转移函数返回下一个状态.$T_{\mathrm{e}}(s)=\{\mathrm{true},\mathrm{false}\}$表示状态

s情况下,模型是否停止运行.$U(s)\rightarrow R$表示状态s的价值评估函数.

2 HTN规划引导的蒙特卡罗树搜索

2.1 考虑同时持续行动的UCTCD算法

树搜索是经典对抗游戏常用的方法,其中$\alpha-\beta$搜索和UCT搜索最为流行.针对对抗双方同时行动的问题,常用解决方法是依次执行双方连续两步移动,并延迟动作效果到第二步完成后,以此保证公平性,该方法在小规模战斗脚本决策中表现良好.传统对弈游戏(如围棋)中,双方交替行动且动作时长相同,可用UCT树建模玩家行为.然而在作战仿真环境里,基本MCTS算法难以解决双方同时行动、行动时长不确定的问题.因此,Churchill等[26]将UCT扩展为UCTCD(UCT considering durations),在《星际争霸》中完成战术行为决策.

UCTCD算法中加入了等待节点用于对搜索过程进行控制.在UCT算法的搜索树中,树节点与环境状态相对应,节点之间不存在类型差异.UCTCD将树节点分为3种类型:FIRST,SECOND和SOLO.其中FIRST和SECOND节点在双方同时进行决策的情况下使用,此时将双方行动决策分为两个选择阶段,生成两层树节点.SOLO节点对应只有一方进行行动决策的情况,此时选择出的行动能够直接执行[27-28].

2.2 结合HTN改进UCTCD树搜索

由于任务规划与行为决策的相似性,将UCTCD与HTN相结合,构建复合战术规划行为决策模型HTN-UCTCD.引入HTN后,算法的主体仍是UCTCD,由MCTS完成行动选择.HTN在合成算法中的作用是约束和引导UCTCD的行动选择和扩展过程,该过程基于HTN中包含的规划领域知识.

基于HTN的节点生成算法如表1所示.算法中增加了维护待分解复合任务的任务网络堆栈T和当前正在执行的任务t两种元素.当任务t是不能再分解的原子任务时,能够直接执行该任务对应的行动.对抗环境中包含己方和对手共两个虚拟指挥实体,在对抗开始阶段会将双方各自的任务目标存入任务堆栈中,此时由于输入的复合任务尚未分解,故不存在当前执行的原子任务.在执行算法到扩展阶段时,为对复合任务进行持续分解,会采用GenerateNode对新生成节点的T和t赋值.

表1 基于HTN的节点生成算法

Table 1 Node generation algorithm based on HTN

算法:基于HTN的节点生成算法	
1	**Procedure** GenerateChildNode(Node n, State s)
2	**if** $n.t\neq$ Null $\wedge$!$n.t$. Completed( ) **then**
3	GenerateNode$(n,s,$Replicate$(n.T))$
4	**return**
5	BreakdownTasks$(n,s,$Replicate$(n.T))$
6	**Procedure** BreakdownTasks(Node n, State s, TaskStack T)
7	**if** $T=\varnothing$ **then**
8	**return**
9	$T_0\leftarrow T[0]$
10	**if** T_0 is compound **then**
11	**for each** method $\in T_0$ **do**
12	**if** method. Available(s) **then**
13	$T'\leftarrow$ method. $T_{\text{include}}+T[1:]$
14	BreakdownTasks(n,s,T')
15	**else if** T_0 is primative **then**
16	$n.t\leftarrow T_0$
17	GenerateNode(n,s,T)
18	**Procedure** GenerateNode(Node n, State s, TaskStack T)
19	move$\leftarrow$ GetAction$(p(n.s),s,n.t)$
20	n. AddChild(**new** Node(move, T, $n.t$))

GenerateChildNode函数以当前搜索所在节点n和对应状态s作为输入,首先判断当前节点对应的子任务是否已经完成,若没有完成则调用GenerateNode函数选出相应的动作分支执行以产生新的状态节点.若节点对应子任务已经完成,则转入BreakdownTasks函数,从任务堆栈中获取新复合任务进行分解.分解过程中取出栈顶元素T_0进行判定,若该任务是复合任务则对该任务下包含的所有可用方法进行探索,并生成所有对应的分解子任务.之后递归调用该函数直至所有复合任务全部被分解为原子任务.若该任务为原子任务,则直接执行其对应动作后扩展新的节点.

执行GetAction函数能够在获取下一个行动的同时,将HTN规划的原子任务转换为对应的能够实际执行的实体行动,从而实现环境、任务规划和实体行动之间的闭环连接.

3 面向MCTS的对手建模方法

3.1 基于改进粒子滤波的策略拟合方法

3.1.1 粒子滤波(PF)算法

粒子滤波算法也称为蒙特卡罗滤波(Monte

Carlo filter,MCF),是一种用于状态估计的贝叶斯滤波算法[29-30].它通过使用一组随机样本(粒子)来近似状态空间的后验概率分布,从而对系统状态进行估计.粒子滤波算法的状态预估模型如式(3)和式(4)所示:

$$x_j=f\left(x_{j-1},u_{j-1}\right)+v_j, \tag{3}$$

$$y_j=h\left(x_j,u_j\right)+\eta_j. \tag{4}$$

其中:x_j表示j时刻的系统状态;y_j表示j时刻的环境观测值;u_j表示j时刻的系统输入;f和h分别表示状态转移函数和观测函数;v_j和η_j表示噪声.

通过预测和更新两步来计算x_j的后验概率密度$p(x_j|y_{1:j})$,预测步如式(5)所示:

$$p\left(x_j|y_{1:j-1}\right)=\int p(x_j|x_{j-1})p(x_{j-1}|y_{1:j-1})\mathrm{d}x_{j-1}. \tag{5}$$

式中:$y_{1:j-1}$为从第1时刻到第$j-1$时刻的观测集合;$p(x_j|x_{j-1})$描述系统从$j-1$时刻状态x_{j-1}演化到j时刻状态x_j的概率规律;$p(x_{j-1}|y_{1:j-1})$表示仅基于历史观测$y_{1:j-1}$对x_{j-1}的概率估计.更新步采用贝叶斯滤波,如式(6)所示:

$$p\left(x_j|y_{1:j}\right)=\frac{p(y_j|x_j)p(x_j|y_{1:j-1})}{\int p(y_i|x_j)p(x_j|y_{1:j-1})\mathrm{d}x_j}. \tag{6}$$

为消除积分运算、降低复杂度,采用一组加权随机粒子χ_j对$p(x_j|y_{1:j})$进行近似,加权随机粒子χ_j的表达式如式(7)所示:

$$\chi_j=\{x_j^i,w_j^i\}. \tag{7}$$

故后验概率密度函数为

$$p\left(x_j|y_{1:j}\right)\approx\sum_{i=1}^{H}w_j^i\delta\left(x_j-x_j^i\right). \tag{8}$$

式中:x_j^i表示粒子群中第i个粒子;w_j^i表示j时刻第i个粒子的权重;H表示粒子总数;δ表示狄拉克函数.$p(x_j|y_{1:j})$中难以进行粒子集合抽取,故定义重要性概率密度函数$q(x_j^i)$,并在$q(x_j^i)$中抽取粒子.重要性概率密度函数如式(9)所示:

$$q\left(x_j^i\right)=p(x_j^i|x_{j-1}^i). \tag{9}$$

式中:$p(x_j^i|x_{j-1}^i)$描述从$j-1$时刻的父粒子x_{j-1}^i转移到j时刻粒子x_j^i的概率.故粒子权重w_j^i表达式如式(10)所示:

$$w_j^i\propto w_{j-1}^i p(y_j|x_j^i). \tag{10}$$

式中:w_{j-1}^i表示$j-1$时刻第i个粒子的权重;$p(y_j|x_j^i)$表示在粒子状态x_j^i下,实际观测值y_j的概率.权重归一化表达式如式(11)所示:

$$w_j^i=w_{j-1}^i/\sum_{i=1}^{H}w_j^i. \tag{11}$$

重采样后j时刻的状态预估值由粒子群加权表示,如式(12)所示:

$$\tilde{x}=\sum_{i=1}^{H}w_j^i x_j^i. \tag{12}$$

3.1.2　策略拟合匹配过程

仿真系统中t时刻的环境状态可以表示为式(13)[31-32]:

$$S(t)=f(S(t-1),a_+,a_-). \tag{13}$$

式中:a_+和a_-分别表示$t-1$时刻双方执行的动作;S表示环境状态.

基于改进PF的策略匹配算法执行过程如图1所示.

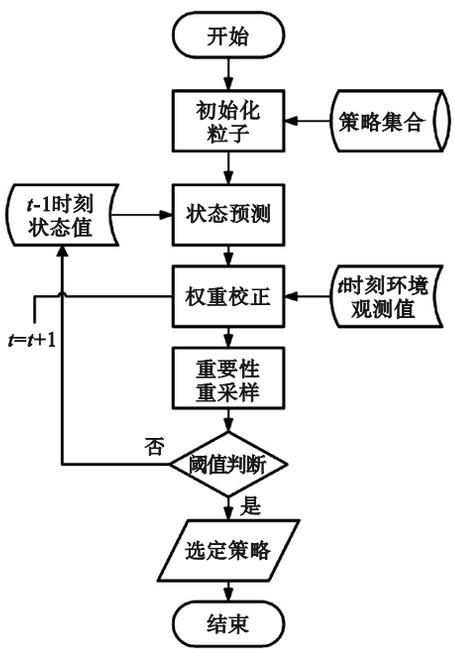


图1　基于改进PF的策略匹配算法执行示意图

Fig. 1　Schematic diagram of strategy matching algorithm execution based on improved PF

1)初始化粒子.初始化阶段给出N个粒子均代表m种已知的对手策略集合,每种策略拥有的粒子数为N/m.对手策略即系统的状态转移函数,策略集合如式(14)所示:

$$f_s=\{f_{s_1},f_{s_2},\cdots,f_{s_m}\}. \tag{14}$$

式中,s_m代表第m种对手策略.

观测值$\boldsymbol{y}(t)$表示为n维向量,如式(15):

$$\boldsymbol{y}(t)=\{y_k(t),k=1,2,\cdots,n\}. \tag{15}$$

式中,$y_k(t)$表示t时刻观测向量的第k个分量.

2)状态预测.在$t-1$时刻,m种策略对应的状态值由式(16)表示:

$$\boldsymbol{X}(t-1)=\{\boldsymbol{X}^e(t-1),e=1,2,\cdots,m\}. \tag{16}$$

式中,第e种策略的状态值$\boldsymbol{X}^e(t-1)$如式(17)所示:

$$\boldsymbol{X}^e(t-1)=\{\boldsymbol{x}_l^e(t-1),l=1,2,\cdots,N/m\}. \tag{17}$$

式中,第e种策略中第l个粒子的状态向量$\boldsymbol{x}_l^e(t-1)$如式(18)所示:

$$\boldsymbol{x}_l^e(t-1)=\{x_{l_k}^e(t-1),k=1,2,\cdots,n\}. \tag{18}$$

式中,$x_{l_k}^e(t-1)$表示第e类策略第l个粒子的第k个分量.

采用m种策略分别对对应的状态值进行预测,求出t时刻的预估状态值$\boldsymbol{X}_{\mathrm{p}}(t)$.相应得出第$e$种策略的状态预测值如式(19)所示:

$$\boldsymbol{X}_{\mathrm{p}}^e(t)=\{\boldsymbol{x}_{\mathrm{p}l}^e(t),l=1,2,\cdots,N/m\}. \tag{19}$$

式中,$\boldsymbol{x}_{\mathrm{p}l}^e(t)$为第$e$类策略预测的第$l$个粒子状态.

3)权重校正.将t时刻的预估状态值输入观测函数f,求出所有粒子在t时刻的预估观测值$\boldsymbol{Y}_{\mathrm{p}}(t)$,相应表达式如式(20)所示:

$$\boldsymbol{Y}_{\mathrm{p}}^e(t)=\{\boldsymbol{y}_{\mathrm{p}l}^e(t),l=1,2,\cdots,m\}. \tag{20}$$

式中:$\boldsymbol{Y}_{\mathrm{p}}^e(t)$为第$e$种策略的预估观测集;$\boldsymbol{y}_{\mathrm{p}l}^e(t)$为第$e$种策略中第$l$个粒子的预估观测值.

4)策略匹配.计算第e种策略中所含粒子的排序值总和如式(21)所示:

$$R_{\mathrm{sum}}^e(t)=\sum_{l=1}^{N/m}r_l^e(t). \tag{21}$$

式中,$r_l^e(t)$表示第e种策略中第l个粒子的排序值.选出粒子权重排序值总和最大的策略即为当前对手策略.

5)重要性重采样.这里采用残差重采样策略,具体步骤如下:

①计算粒子复制子代个数.将每个粒子的权重乘以粒子数量N后取整,得到每个粒子应该被复制的次数.第e种策略中第l个粒子的子代个数如式(22)所示:

$$n_l^e=\mathrm{Int}(N\cdot w_l^e). \tag{22}$$

式中,w_l^e为每个粒子的权重.

将所有粒子产生的后代粒子组成新粒子集合,其总数如式(23)所示:

$$O=\sum_{e=1}^{m}\sum_{l=1}^{N/m}n_l^e. \tag{23}$$

②残差粒子采样.由于$O<N$,故存在O^*个粒子由于权重过低无法进行复制,这里$O^*=N-O>0$.针对这部分粒子,将其权重与粒子数量相乘后的小数部分作为抽样概率,然后在旧粒子集中进行抽样,最终抽样出O^*个残差粒子.

③合成新粒子集.将①和②中得到的新粒子集合成,然后设置每个粒子权重为$1/N$.

6)阈值判断.在粒子重采样后进行判断,若没有任何一种策略的粒子数超过设定阈值,则返回第2)步重新进行迭代.

3.2 基于隐式对手建模的博弈树构建

3.2.1 隐式对手建模

基于策略匹配的隐式对手建模算法具体步骤如下:

1)环境信息与对手动作采集.利用信息数据采集器获取当前时刻的全局环境信息观测值以及对手的动作序列.

2)建立对手策略集.根据对手的历史动作信息和军事经验知识建立对手所有已知的策略集.

3)策略集合初始化.设置粒子总数为N,将粒子均分为m份分别代表不同对手策略,并为每个粒子赋值一个随机的合法开局状态.

4)策略预测状态动作生成.在完成粒子初始化后,通过策略匹配算法预测出对手在下一时刻t可能采取的动作状态集合.

5)不同对手策略权重计算.通过对比前t个时刻环境信息观测预估值与实际观测值的相似度,可以为代表不同策略的粒子赋予不同的权重,相似度越高权重越大.

6)选出当前对手策略.当前时刻进行权重赋值后会按照数值大小对粒子进行排序,排序后进行同类粒子序号求和,将求和数值最大的策略作为当前对手策略.

7)不同策略重要性重采样.为解决粒子退化问题,在策略匹配算法中引入残差重采样算法,对拟合度高的策略粒子进行复制,拟合度低的粒子则按一定概率予以保留.设定占比阈值和迭代次数作为算法终止条件,初始阶段循环步骤4)到7),当满足条件时跳出循环,并在之后的一段时间内将选出的策略作为稳定的对手策略.

3.2.2 博弈对抗蒙特卡罗树搜索

基于提出的MCTS战术规划与行为决策算法,将隐式对手建模加入其中,构建博弈对抗蒙特卡罗树搜索算法,算法流程图如图2所示.

算法可以分为两个部分,即对手模型构建和我方动作决策.在第一部分中,当我方需要进行动作规划时,需要先进行隐式对手建模,通过构建对手策略集合并用不同粒子群分别代表后进行动作预测,得出执行预测动作后的预测环境信息;将预测环境信息与实际环境信息对比得到不同策略与实际对手行为模型的相似度,进行粒子的残差重采样.在多次迭代使得某种策略对应粒子数超过设定阈值或超出最大迭代次数后停止迭代,并将粒子比重最大的策略作为当前对手行为模型.在第二部分中,将对手行为模型和环境

信息输入蒙特卡罗树执行博弈搜索过程,过程中基于确定的对手下一步行为和环境信息在我方合法行为集合中进行搜索评估;在向前进行多步的态势推演达到约束时间或最大深度后,采用评估函数对我方合法动作进行评估,选出评估值最大的动作作为下一步执行动作.

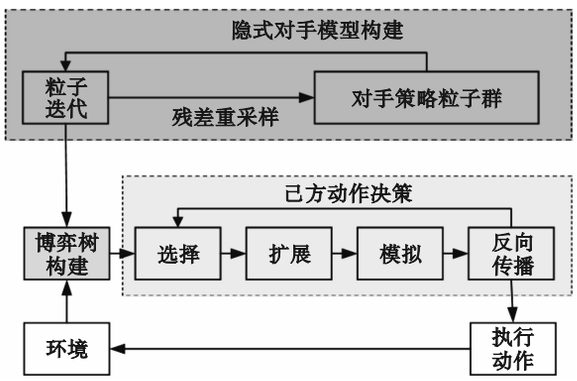


图2　博弈对抗MCTS战术行为决策过程

Fig. 2　Tactical behavior decision-making process of adversarial game MCTS

4　仿真实验与性能分析

4.1　仿真平台

为专注于相关问题的研究,选择μRTS作为仿真平台.μRTS是一个开源的简化RTS游戏,该平台具备挑战性的基本功能,包括同时和持续的动作、组合分支因素和实时决策[33-36].图3显示了在8×8地图上进行比赛的μRTS状态截图.图中带有短虚线或长虚线轮廓的方形或圆形实体分别表示可以由玩家I(我方)和玩家Y(敌方)控制的实体.斜线方块表示资源,方块中的数字表示剩余资源的数量.资源默认分布在地图的左上角和右下角,未着色的方块是地图上可遍历的区域.

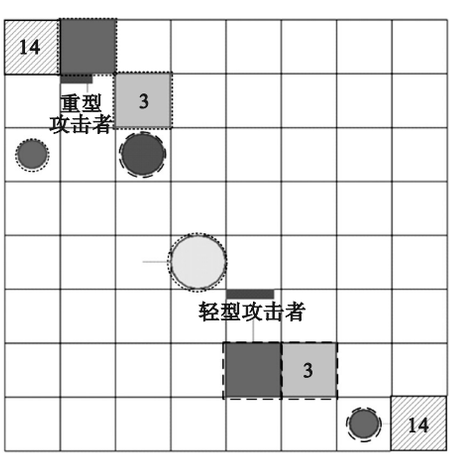


图3　μRTS对局状态截图

Fig. 3　Screenshot of game state in μRTS

游戏中所有实体单位的定义如下[37-38]:

1) 工人(worker):最小的深灰色圆形单位,负责采集资源到特定位置建造基地和兵营,之后采集资源运向基地进行存储;

2) 轻型攻击者(light):中等大小深色圆形单位,攻击力和防御力中等,速度最快的近战单位;

3) 重型攻击者(heavy):浅色圆形单位,攻击力和防御力最高,速度最慢的近战单位;

4) 远程攻击者(range):短虚线圆形单位,攻击范围最大,防御最弱的远程消耗单位;

5) 基地(base):浅灰色正方形单位,基地利用工人运回的资源生产新的工人,方块中的数字表示当前玩家可用的资源数量;

6) 兵营(barrack):深灰色正方形单位,能够训练轻型、重型和远程攻击者,训练需要消耗基地中存储的资源.

4.2　HTN领域知识仿真实验

4.2.1　仿真设置

为验证HTN中领域知识对HTN-UCTCD算法性能的影响,设置了4种知识复杂度不同的HTN模型,并与UCTCD结合,生成4个指挥型实体AI模型.4种指挥策略模型的运行方式如下[39]:

低级策略模型(low level, LL):仅包含游戏中的低级原子任务,如移动(任意方向)、攻击(射程内敌方单位)、基地和工人产生新单位、采集资源和闲置.

低级寻路策略模型(low level with pathfinding, LLPF):与低级策略相似,但单位移动采用路径查找算法确定最短路径.该模型包含4个不同任务和12种方法.

灵活模型(Flexible):一种更复杂的策略模型,具有非原子任务,用于获取资源、训练不同单位和攻击敌人.该模型包含19种任务和49种方法.

灵活单一目标模型(flexible single target, FST):类似于灵活模型,但命令所有攻击单位针对同一目标,从而降低分支因子.该模型包含21种任务和51种方法.

这4种策略所包含的原子任务均为12种,为便于描述,统一将上述4种策略的HTN-UCTCD算法简写为LL,LLPF,Flexible和FST.

为评估HTN-UCTCD算法的性能,将其与11种AI模型进行循环对抗比赛.每个算法在3个不同大小的地图(8×8,12×12和16×16)中各进行20场比赛,总计进行60场比赛,胜者得1分,败者得0分.若达到最大对抗时间(4 000个游戏帧),则

视为平局,双方各获0.5分.所有对局均从一个基地和两名工人开始.实验结果如表2所示,RTS-AI代表对手AI模型.

表2 对手AI模型及相应策略

Table 2 Opponent AI models and corresponding strategies

RTS-AI	AI策略
Random	随机策略,随机生成合法动作并执行
重型冲击(HR)	采用1个工人收集资源,将资源全部用于训练重型攻击者并发起攻击
轻型冲击(LR)	与HR类似,该策略只训练轻型攻击者
ABCD	考虑动作持续时间的$\alpha-\beta$搜索算法
NaiveMCTS	采用CMAB建模MCTS,具有特定采样策略
AHTN	对抗分层任务网,结合最大最小搜索树与HTN规划
AHTN-FST	采用灵活单一目标模型的对抗分层任务网

4.2.2 仿真结果与分析

如图4所示,胜率最高的两种算法是采用灵活策略(Flexible)和灵活单一目标策略(FST)的HTN-UCTCD算法,随后是灵活单一目标策略的AHTN算法.这3种算法在不同尺寸地图中保持稳定的优势,因其采用了复杂的领域知识生成策略,能够执行更高级的任务.在小尺寸地图中,节奏较快,算法优势不明显,但随着地图增大,优势逐渐显现.尽管HTN-UCTCD和AHTN都使用FST策略,HTN-UCTCD在胜率上均优于AHTN,因为AHTN的搜索过程依赖于固定策略顺序,未能充分利用领域知识.HTN-UCTCD算法能并行搜索同一任务下的不同分解方法,决策更加灵活.

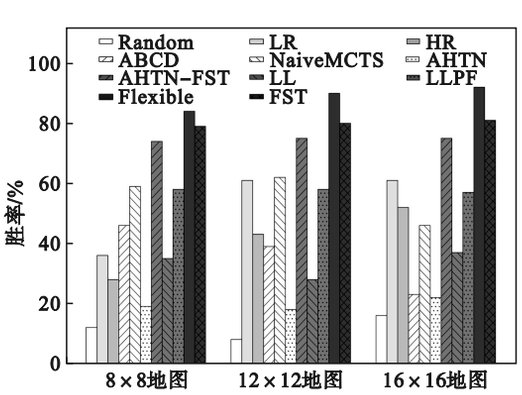


图4 3种地图中不同AI的胜率对比

Fig. 4 Comparison of winning rates of different AI in three maps

为深入了解HTN-UCTCD的性能,固定每个游戏帧的时间预算为100 ms,采用LL,LLPF,Flexible和FST策略的HTN-UCTCD算法与所有地图上的其他AI进行对抗,并记录生成的游戏树.表3显示了采用不同策略的HTN-UCTCD所能达到的平均深度、叶节点数量和前瞻度.前瞻度是指根节点状态与叶节点状态之间的差值.结果表明,策略中方法和任务数量越多,复合任务越抽象,HTN对UCTCD的搜索约束程度越高,因此FST策略在较小决策空间中能够搜索到更大的深度和前瞻度.

4.3 隐式对手建模算法仿真实验

4.3.1 仿真设置

为验证隐式对手建模算法与MCTS算法结合的可行性,在μRTS游戏平台上进行仿真对比.为减少无关变量的影响,所有对局开局时双方均拥有一个工人、一个基地和相同的资源数量.比赛在3种尺寸的地图上进行,分别为8×8,12×12和16×16,其中前两种地图的最大对局时间设置为3 000帧,16×16地图的最大对局时间设置为4 000帧.每局比赛结束后记录双方得分,获胜一方得1分,失败方不得分,平局双方各得0.5分.

表3 不同策略HTN-UCTCD玩家的搜索树统计数据

Table 3 Statistical data of search tree for HTN-UCTCD players with different strategies

不同策略	深度	叶节点数量	前瞻度
LL	1.69(6)	3 171.23(12 020)	14.58(40)
LLPF	2.48(28)	3 056.14(9 415)	36.94(1 001)
Flexible	4.63(13)	1 245.63(11 140)	49.38(210)
FST	5.92(20)	476.65(2 355)	59.17(200)

注:括号内表示最大值,括号外表示平均值.

为测试隐式对手建模方案与UCTCD算法结合后的AI性能,采用硬编码脚本机器人作为对手,同时,极大极小树搜索与HTN规划结合生成的AHTN也使用领域知识作为规划策略,可作为对手进行测试.初始粒子数设置为50,迭代阈值为粒子总数的某一比例.

4.3.2 仿真结果与分析

为了验证组合隐式对手建模方法后,UCTCD算法能够通过对手策略匹配算法选出对手策略并做出针对性行动决策,从而实现决策算

法性能的提升.采用加入隐式对手建模博弈的UCTCD(G-UCTCD)算法和传统UCTCD算法作为AI,在μRTS平台的3种不同尺寸地图上,与表4中的所有AI进行循环对抗.每个算法都与其余所有算法进行20场比赛.在所有比赛结束后,将总得分除以对局数得到相应胜率,通过比较胜率分析算法性能.

表4　对手AI模型及相应策略

Table 4　Opponent AI model and corresponding strategies

RTS-AI	AI策略
工人冲击(WR)	只训练工人并发起攻击
重型冲击(HR)	采用1个工人收集资源,将资源全部用于训练重型攻击者并发起攻击
远程冲击(RR)	与HR类似,该策略只训练远程攻击者
PortfolioAI	对WR,LR,RR和Random策略评估后选择一种最佳策略
AHTN	对抗分层任务网,结合最大最小搜索树与HTN规划
G-AHTN	结合隐式对手建模的AHTN
Random	随机策略,随机生成合法动作并执行

1)不同决策时间约束.为探究一个游戏帧内不同决策时间约束对提出算法性能的影响,在每张地图上设置约束时间从50 ms至400 ms,间隔为50 ms,共8组仿真.图5~图7分别展示了在3种尺寸地图下进行8组仿真后各AI获得的胜率.

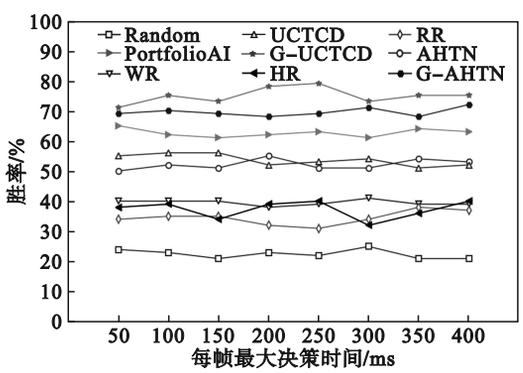


图5　8×8地图中不同算法的胜率对比

Fig. 5　Comparison of winning rates of different algorithms in 8×8 map

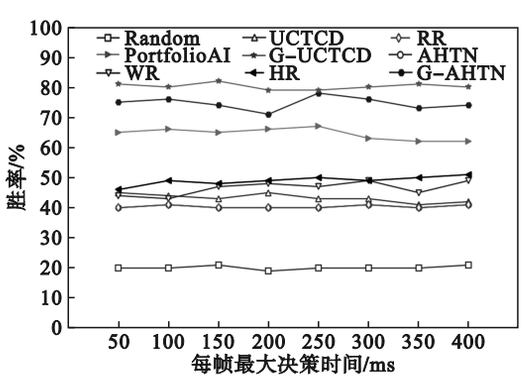


图6　12×12地图中不同算法的胜率对比

Fig. 6　Comparison of winning rates of different algorithms in 12×12 map

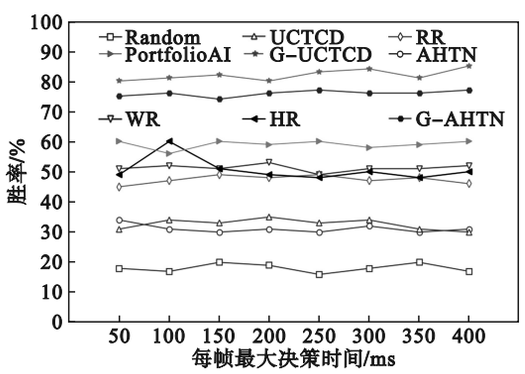


图7　16×16地图中不同算法的胜率对比

Fig. 7　Comparison of winning rates of different algorithms in 16×16 map

综合来看,G-UCTCD算法和G-AHTN算法在3种地图上的胜率均位于前两名,特别是G-UCTCD算法的胜率总体上均优于G-AHTN算法.在前两种小尺寸地图中,胜率处于50%左右的是基础AHTN算法和UCTCD算法,相比于WR,HR和RR算法存在明显优势,在16×16地图上情况发生反转.在全部的仿真设置状态下Random算法均取得最低胜率,在小尺寸地图中由于双方距离较近,其获胜概率相对较高.

在各地图中,从横向上看,不同决策时间约束下各算法胜率保持相对稳定.与G-UCTCD算法相比,没有隐式对手建模方法辅助决策的UCTCD算法在所有地图上均与改进算法存在较大差距,这很好地表明了改进算法在辅助决策方面的有效性.

2)相同决策时间约束.在不同决策时间约束条件下,所有算法的胜率保持相对稳定,为进一

步探究不同算法在不同地图上的综合表现,选取每帧决策时间限定为200 ms,计算所有AI在3种地图下的胜率.

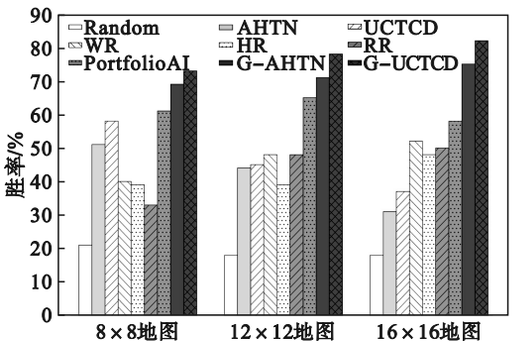


图8 限定不同算法决策时间为200 ms的胜率对比

Fig. 8 Comparison of winning rates of different algorithms with decision-making time limited to 200 ms

在图8中可以更直观地看出,采用隐式对手建模方案的G-UCTCD算法和G-AHTN算法在3种地图中始终处于胜率的前两位.在3种地图中,UCTCD的综合胜率约为45%,G-UCTCD算法综合胜率约为78%,加入隐式对手建模方法后,行为决策算法的性能获得很大提升.这同样能通过另一组G-AHTN算法和AHTN算法的对比体现出来.

5 结 论

1)针对状态动作搜索空间大,决策时间有限等问题,采用HTN将复合任务分解为原子任务引导MCTS搜索过程,基于HTN中蕴含的领域知识限制其搜索空间,提高决策效率.在不同尺寸地图对抗仿真环境中,不同领域知识复杂度下的结果表明,HTN能够利用领域知识约束MCTS搜索空间,提高搜索效率.

2)针对传统MCTS在决策过程中等概率选择对手可能行动,导致对手行为不确定的问题,构建隐式对手模型以识别对手当前策略并纳入己方行为决策考虑中.首先提出基于粒子滤波的策略识别方法,在明确对手可选策略集合后,根据不同策略预测值与对手实际动作的拟合度更新相应策略表征粒子的数量.在数次迭代识别出对手策略后,将该策略作为对手模型进行对手动作预测.之后将预测对手动作加入MCTS博弈树,用于辅助己方针对性行为决策.仿真结果表明,在不同时间约束下的所有地图中,结合隐式对手模型的MCTS算法均能通过预测对手行动进行针对性行为决策,从而获得最高胜率.

参考文献:

[1] 胡晓峰.战争复杂系统仿真分析与实验[M].北京:国防大学出版社,2008:1-3.
(Hu Xiao-feng. War complex system simulation analysis & experimentation[M]. Beijing: National Defense University Press, 2008: 1-3.)

[2] 黄柯棣,刘宝宏,黄健,等.作战仿真技术综述[J].系统仿真学报,2004,16(9):1887-1895.
(Huang Ke-di, Liu Bao-hong, Huang Jian, et al. A survey of military simulation technologies[J]. *Journal of System Simulation*, 2004, 16(9): 1887-1895.)

[3] Kang K, Cheng K, Shao T H, et al. Planning, monitoring and replanning techniques for handling abnormity in HTN-based planning and execution [J]. *Journal of Systems Engineering and Electronics*, 2024, 35(5): 1264-1275.

[4] 杨伟龙.CGF战术任务规划行为建模关键技术研究[D].长沙:国防科技大学,2019.
(Yang Wei-long. Research on key technologies of CGF tactical task planning behavior modelling[D]. Changsha: National University of Defense Technology, 2019.)

[5] 许霄.面向CGF战术决策的蒙特卡罗树搜索方法研究[D].长沙:国防科技大学,2018.
(Xu Xiao. Modeling CGF tactical decision making through Monte Carlo tree search [D]. Changsha: National University of Defense Technology, 2018.)

[6] Frutos-Pascual M, Zapirain B G. Review of the use of AI techniques in serious games: decision making and machine learning [J]. *IEEE Transactions on Computational Intelligence and AI in Games*, 2017, 9(2): 133-152.

[7] Muñoz-Avila H, Bauckhage C, Bida M, et al. Learning and game AI[C]//Artificial and Computational Intelligence in Games. Dagstuhl: Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2013: 33-43.

[8] Toubman A, Roessingh J J, van Oijen J, et al. Modeling behavior of computer generated forces with machine learning techniques, the NATO task group approach[C]// 2016 IEEE International Conference on Systems, Man, and Cybernetics (SMC). Budapest, 2017: 1906-1911.

[9] Wang F Y, Zhang J J, Zheng X H, et al. Where does AlphaGo go: from church-turing thesis to AlphaGo thesis and beyond[J]. *IEEE/CAA Journal of Automatica Sinica*, 2016, 3(2): 113-120.

[10] Holcomb S D, Porter W K, Ault S V, et al. Overview on DeepMind and its AlphaGo zero AI[C]//Proceedings of the 2018 International Conference on Big Data and Education. Honolulu, 2018: 67-71.

[11] Løvlid R A, Bruvoll S, Brathen K, et al. Modeling the behavior of a hierarchy of command agents with context-based reasoning[J]. *The Journal of Defense Modeling and Simulation: Applications, Methodology, Technology*, 2018, 15(4): 369-381.

[12] Vinyals O, Babuschkin I, Czarnecki W M, et al. Grandmaster level in StarCraft II using multi-agent reinforcement learning[J]. *Nature*, 2019, 575(7782): 350-354.

[13] Doumanas D, Soularidis A, Kotis K. Causal reasoning and

large language models for military decision-making: rethinking the command structures in the era of generative AI[J]. *AI*, 2026, 7(1): 14.

[14] 徐浩添, 秦龙, 曾俊杰, 等. 基于深度强化学习的对手建模方法研究综述[J]. 系统仿真学报, 2023, 35(4): 671-694.
(Xu Hao-tian, Qin Long, Zeng Jun-jie, et al. Research progress of opponent modeling based on deep reinforcement learning[J]. *Journal of System Simulation*, 2023, 35(4): 671-694.)

[15] Zhang Y, et al. Decision-making with speculative opponent models [C]// Proceedings of the 22nd International Conference on Autonomous Agents and Multiagent Systems. London, 2023: 1332-1341.

[16] Pomerleau F, Krüsi P, Colas F, et al. Long-term 3D map maintenance in dynamic environments [C]//2014 IEEE International Conference on Robotics and Automation (ICRA). Hong Kong, 2014: 3712-3719.

[17] Du H B, Wen G H, Wu D, et al. Distributed fixed-time consensus for nonlinear heterogeneous multi-agent systems [J]. *Automatica*, 2020, 113: 108797.

[18] Wu J P, Lu Y J, Li D Z, et al. Modeling and solution for course of action planning driven by operational task network [J]. *IEEE Access*, 2025, 13: 41169-41193.

[19] Sun H L, Liu J G, Han Z Q, et al. Stochastic Petri net based modeling of emergency medical rescue processes during earthquakes [J]. *Journal of Systems Science and Complexity*, 2021, 34(3): 1063-1086.

[20] Tanui D J. Application of Boyd's OODA loop to emergency response[J]. *Journal of Emergency Management*, 2021, 19(5): 461-468.

[21] Choudhury S, Gupta J K, Morales P, et al. Scalable online planning for multi-agent MDPs [J]. *Journal of Artificial Intelligence Research*, 2022, 73: 821-846.

[22] Du B, Yuan C, Yu Z J. Command and control modeling method for federated intelligence [C]//Proceedings of International Conference on Artificial Intelligence and Communication Technologies (ICAICT 2023). Singapore: Springer, 2024: 3-16.

[23] Smith M, Robson D. A Concept for the Modelling and Simulation of Complex Urban Environments [C]// STO-MP-MSG-207. Brussels, 2023: 21.

[24] Shao T H, Zhang H J, Cheng K, et al. The hierarchical task network planning method based on Monte Carlo Tree Search[J]. *Knowledge-Based Systems*, 2021, 225: 107067.

[25] Wu I C, Wu T R, Liu A J, et al. On strength adjustment for MCTS-based programs [C]//Proceedings of the AAAI Conference on Artificial Intelligence. Hawaii, 2019: 1222-1229.

[26] Churchill D, Buro M. Portfolio greedy search and simulation for large-scale combat in starcraft [C]//2013 IEEE Conference on Computational Inteligence in Games (CIG). Niagara Falls, 2013: 1-8.

[27] Fauzi R, Hariadi M, Lubis M, et al. Defense behavior of real time strategy games: comparison between HFSM and FSM[J]. *Indonesian Journal of Electrical Engineering and Computer Science*, 2019, 13(2): 634-642.

[28] Sarker I H, Colman A, Han J, et al. BehavDT: a behavioral decision tree learning to build user-centric context-aware predictive model [J]. *Mobile Networks and Applications*, 2020, 25(3): 1151-1161.

[29] Floyd M. A general-purpose framework for learning by observation[D]. Ottawa: Carleton University, 2013.

[30] de Weerd H, Verbrugge R, Verheij B. Negotiating with other minds: the role of recursive theory of mind in negotiation with incomplete information [J]. *Autonomous Agents and Multi-Agent Systems*, 2017, 31(2): 250-287.

[31] Gronauer S, Diepold K. Multi-agent deep reinforcement learning: a survey[J]. *Artificial Intelligence Review*, 2022, 55(2): 895-943.

[32] Polceanu M, Buche C. Computational mental simulation: a review[J]. *Computer Animation and Virtual Worlds*, 2017, 28(5): e1732.

[33] Ontañón S. Combinatorial multi-armed bandits for real-time strategy games [J]. *Journal of Artificial Intelligence Research*, 2017, 58(1): 665-702.

[34] Løvlid R A, Bruvoll S, Brathen K, et al. Modeling the behavior of a hierarchy of command agents with context-based reasoning[J]. *The Journal of Defense Modeling and Simulation: Applications, Methodology, Technology*, 2018, 15(4): 369-381.

[35] Wang C, Chen P, Li Y D, et al. Portfolio online evolution in StarCraft [C]//Proceedings of the Twelfth AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment. Burlingame, 2016: 114-121.

[36] Manandhar S, Banerjee B. Reinforcement actor-critic learning as a rehearsal in MicroRTS [J]. *The Knowledge Engineering Review*, 2024, 39: e6. 1-15.

[37] Ontanon S. Experiments on learning unit-action models from replay data from RTS games[J]. *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, 2016, 12(2): 9-14.

[38] Lorentz R. Using evaluation functions in Monte-Carlo tree search[J]. *Theoretical Computer Science*, 2016, 644(C): 106-113.

[39] Lanctot M, Winands M H M, Pepels T, et al. Monte Carlo Tree Search with heuristic evaluations using implicit minimax backups [C]//2014 IEEE Conference on Computational Intelligence and Games. Dortmund, 2014: 1-8.