DOI: 10.11992/tis.202211006
网络出版地址: https://link.cnki.net/urlid/23.1538.TP.20230727.1635.002

基于改进 CBS 算法的多智能体路径规划

王卓然[1], 文家燕[1,2], 谢广明[1,3], 蒋文宇[1]
(1. 广西科技大学 自动化学院, 广西 柳州 545616; 2. 广西科技大学 广西汽车零部件与整车技术重点实验室, 广西 柳州 545006; 3. 北京大学 工学院, 北京 100871)

摘 要: 在基于冲突的搜索 (conflict-based search, CBS) 算法中, 冲突的选择具有随机性, 导致 CBS 算法在多智能体路径规划方面的求解效率不佳。为此, 本文提出一种改进 CBS 算法的多智能体路径规划算法。首先, 基于冲突子节点的相关信息, 提出了一种新的冲突选择策略; 然后, 为发挥新策略的优势和进一步减少算法的运行时间, 采用基于神经网络的 RankNet 算法来学习新策略, 从而得到一个训练好的排序模型; 最后, 利用训练好的排序模型为 CBS 算法选择冲突。通过设计实验对改进 CBS 算法进行仿真验证, 结果表明, 所提改进算法相比于已有的改进算法, 能够有效提高算法的求解效率。

关键词: 多智能体; 全局路径规划; 基于冲突的搜索算法; 改进基于冲突的搜索算法; 机器学习; 排序学习; RankNet 算法; 冲突选择策略
中图分类号: TP23; TP18 **文献标志码**: A **文章编号**: 1673−4785(2023)06−1336−08

中文引用格式: 王卓然, 文家燕, 谢广明, 等. 基于改进 CBS 算法的多智能体路径规划 [J]. 智能系统学报, 2023, 18(6): 1336–1343.
英文引用格式: WANG Zhuoran, WEN Jiayan, XIE Guangming, et al. Multi-agent path planning based on improved CBS algorithm[J]. CAAI transactions on intelligent systems, 2023, 18(6): 1336–1343.

Multi-agent path planning based on improved CBS algorithm

WANG Zhuoran[1], WEN Jiayan[1,2], XIE Guangming[1,3], JIANG Wenyu[1]
(1. School of Automation, Guangxi University of Science and Technology, Liuzhou 545616, China; 2. Guangxi Key Laboratory of Automobile Components and Vehicle technology, Guangxi University of Science and Technology, Liuzhou 545006, China; 3. College of Engineering, Peking University, Beijing 100871, China)

Abstract: In the conflict-based search (CBS) algorithm, random conflict selection leads to poor solution efficiency in multi-agent path planning. Therefore, an improved CBS-based multi-agent path-planning algorithm is proposed in this paper. First, a new conflict selection strategy is introduced according to the information related to the child nodes associated with the conflict. Next, the RankNet algorithm based on a neural network is used to learn the new strategy and further reduce the running time of the algorithm, obtaining a trained ranking model. Finally, this well-trained ranking model is utilized to select conflicts for the CBS algorithm. Simulation verification of the improved CBS algorithm was performed by designing experiments. Results show that the proposed CBS algorithm effectively enhances the efficiency of the algorithm compared with the existing improved algorithm.

Keywords: multi-agent; global-path planning; conflict-based search algorithm; improved conflict-based search algorithm; machine learning; ranking learning; RankNet algorithm; conflict selection policy

随着人工智能技术的不断发展, 多智能体路径规划[1]在自动仓库[2-4]、数字游戏[5]、火车调度[6]、城市道路网络[7]、多机器人系统[8]等方面有广泛的应用前景。多智能体路径规划是指在给定地图下, 为给定的所有智能体, 以最小路径代价或者最小化最大完工时间为目标, 求解无冲突路径。其中, 最大完工时间表示最后一个智能体到达目标点的时间。

基于冲突的搜索 (conflict-based search, CBS) 算法[9]是近年来多智能体路径规划备受关注的算法之一, 许多已有文献在它的基础上进行

收稿日期: 2022−11−07. **网络出版日期**: 2023−07−28.
基金项目: 国家自然科学基金项目(61963006); 广西自然科学基金面上项目(2018GXNSFAA050029); 广西科技重大专项(桂科 AA22068064); 2022 年广西汽车零部件与整车技术重点实验室自主研究课题(2022GKLACVTZZ01).
通信作者: 文家燕. E-mail: wenjiayan2012@126.com.

了一系列研究[10−18], 包括优先解决关键冲突、绕过冲突、向高层添加启发值、对称推理技术等。CBS 算法的中心思想是低层使用解耦路径规划算法为每个智能体规划路径, 高层在约束树 (constraint tree, CT) 上检测并解决路径冲突。冲突的选择会影响 CBS 的运行效率, 选择一个“好”的冲突, 能够快速地增加约束树的下界, 避免陷入相同代价节点的泥潭, 从而更快地扩展到目标节点找到最优解。针对这个问题, Boyarski 等[10] 提出使用多值决策图 (multi-value decision diagram, MDD) 将冲突分类, 并先消解优先级高的冲突, 被广泛应用到其他改进 CBS 算法的文献中。Huang 等[19] 提出了一个新的冲突选择策略, 并首次采用新的机器学习框架来学习该策略, 其效果相较于文献 [10], 表现更佳, 有效地提高了算法的求解效率和成功率。为了进一步提高算法的求解效率, 本文在文献 [19] 的基础上, 针对使用其冲突选择策略, 可能会出现一个节点中所有冲突都应被选择, 即有时冲突的区分度不明显的问题, 提出一个新的冲突选择策略, 并采用 RankNet 算法[20] 学习该策略, 改进算法使用学习得到的排序模型为 CBS 算法选择冲突, 能更快地选择“好”的冲突, 能大幅缩短算法的总运行时间和缩小搜索空间, 有效提高算法的求解效率。仿真验证所提的改进算法在运行时间、搜索空间和成功率上的有效性。

1　预备知识

1.1　多智能体路径规划

多智能体路径规划 (multi-agent path finding, MAPF)[21] 是为一组智能体$\{a_1,a_2,\cdots,a_k\}$在指定的无向图$G=(V,E)$下寻找一组无冲突的路径, 其中k为智能体的数量, V是图中顶点的集合, E是图中顶点之间连接边的集合, 用$v\in V$表示智能体可以在图中占据的顶点, 用$e=(v_i,v_j)\in E$表示智能体可以从v_i移动到v_j或相反。每个智能体a_i都有其起始位置$s_i\in V$和终点位置$g_i\in V$。时间被离散化为时间步, 在每一个时间步, 智能体可以移动到非障碍节点的邻节点也可以停在原地等待, 不论移动还是等待, 路径代价都增加 1。单个智能体的路径代价等于智能体从起点到达终点并且之后不再移动所花的总时间步长。本文考虑两种智能体之间发生的冲突: 顶点冲突和边冲突。顶点冲突$<a_i,a_j,u,t>$, 即智能体a_i和a_j在相同的时间t都处在位置u; 边冲突$<a_i,a_j,u,v,t>$, 即智能体a_i和a_j相向而行, a_i在时间t处在位置u, $t+1$时间处在位置v, 而智能体a_j在时间t处在位置v, $t+1$时间处在位置u。本文以最小路径代价为目标, 为所有智能体找到从起点到终点总路径代价最小的一组无冲突路径。

1.2　基于冲突的搜索

CBS 算法是两层算法, 高层检测所有路径中是否存在冲突, 低层使用 space-time A*算法[22] 解耦地为智能体求解有约束下的最优路径。高层对 CT 进行搜索, CT 是一颗二叉树, CT 中的每个节点N由以下信息组成:

1) 约束集N_{cons}: 约束由冲突分裂而来, 且每个约束都只与一个智能体有关。顶点冲突会分裂为$<a_i,u,t>$和$<a_j,u,t>$, 表示为不允许智能体$a_i(a_j)$在时间t处在位置u。边冲突则分裂为$<a_i,u,v,t>$和$<a_j,v,u,t>$, 表示为不允许智能体$a_i(a_j)$在时间$t\to t+1$从位置$u(v)$到位置$v(u)$。CT 中根节点的约束集为空, 其他节点的约束集N_{cons}除继承父节点的所有约束外, 还包括一个新约束, 它来自父节点所选冲突分裂生成的约束。

2) 解决方案N_{sol}: 含有每一个智能体a_i的路径, 且每一条路径都满足N_{cons}中所有与a_i有关的约束条件。

3) 冲突集N_{conf}: N_{sol}中存在的所有冲突。

4) 总路径代价N_{cost}: N_{sol}中所有路径的代价总和。

CBS 算法首先生成根节点, 根节点的N_{cons}为空, N_{sol}由低层算法为每个智能体规划的路径构成, 并计算总路径代价N_{cost}, 而后检测N_{sol}存在的路径冲突并构成N_{conf}。每次都在 CT 中选择最小N_{cost}的节点进行扩展, 并检测当前扩展节点的N_{conf}是否为空, 若N_{conf}为空也即无冲突存在, 则 CBS 算法停止运行并返回解N_{sol}; 相反, CBS 算法默认情况下随机在N_{conf}选择一个冲突, 将其分裂的两个约束分别作为左右子节点的新约束, 并继承父节点的约束集构成两个子节点的约束集N_{cons}。子节点只为新约束所涉及的智能体重新规划满足当前子节点N_{cons}的路径, 其余智能体的路径不变, 以此构成子节点的N_{sol}, 若子节点有解则将其加入 CT 中等待被扩展。CBS 算法具有完整性和最优性, 它通过解决每个冲突的两种情况来保证其完整性, 通过在高层和低层中都使用最佳优先搜索来保证其最优性。

2　改进 CBS 多智能体路径规划算法

对于 CBS 算法, 选择一个“好”的冲突能更快地扩展到目标节点, 从而有效地缩小搜索空间, 提高算法在路径规划中的求解效率。由此提出一种改进冲突选择策略, 并验证其消解冲突的有效性; 然后, 为了发挥改进冲突选择策略的优势和进一步缩短算法的运行时间, 采用基于神经网络

的 RankNet 算法学习该改进策略，用训练好的排序模型为 CBS 算法选择冲突。

2.1　冲突选择策略

为了与已有冲突选择策略展开比较，也便于后续问题的阐述，事先给出以下 2 种冲突选择策略的定义。定义 1 来自文献 [10]，将冲突划分优先级，并优先选择高优先级冲突进行消解。定义 2 来自文献 [19]，通过假使每个冲突被选择，其生成的相应两个子节点的$g+h$的最小值作为该冲突的得分，优先选择得分高的冲突。

定义 1[10]　对给定的 CT 中节点N，S_0使用 MDD[23] 将N_{conf}中的冲突的优先级从高到低分类为关键冲突，半关键冲突及非关键冲突。存在相同优先级的冲突时，优先选择冲突发生时间更小的冲突。

定义 2[19]　对给定的 CT 中节点N，S_1计算N_{conf}中每个冲突的得分$v_c=\min\{g_c^l+h_c^l, g_c^r+h_c^r\}$，其中$g_c^l+h_c^l$和$g_c^r+h_c^r$分别表示如果该冲突被选中，生成的左右两个子节点的$g+h$值，$g$为$N_{\text{cost}}$，$h$由加权依赖图 (weighted pairwise dependency graph, WDG)[24] 给出。并将得分由高到低进行排序，优先选择得分最高的冲突。

使用S_1作为冲突选择策略求解实例的过程中有时会出现一个节点中所有的冲突得分都相同，而这些冲突对应的子节点的总路径代价不同，包含的冲突数差异较大，未做进一步区分可能选择的冲突会使其子节点包含更多的冲突数，以至于需要扩展更多节点来消解冲突的情况。

为了更直观地展示这一情况，下面给出一个具体实例。如图 1(a) 所示，智能体 0、1、2 分别从起点B_2、D_2、D_1出发，到目标点C_2、B_1、A_2。使用 CBS 的改进算法 CBS+PC[10]+BP[25]+WDG[24] 算法（以下仍简写为 CBS 算法）求解，使用S_1选择冲突，得到的约束树如图 1(b) 所示。在调用低层算法单独为每个智能体规划最优路径后，高层检测到路径间存在两个冲突$<0,1,C_1,2>$和$<0,2,C_2,2>$。分别选择这两个冲突进行消解，得到相应的子节点信息位于冲突集一栏该冲突的右侧。其中，l和r分别表示该冲突的左、右子节点的信息即子节点的$g+h$和子节点中包含的冲突数。可以看出使用S_1计算得到两个冲突的得分均为 10，若不做进一步区分，选择$<0,1,C_1,2>$进行消解，则生成的左右两个子节点中分别含有 2 个和 5 个冲突，需要扩展更多的节点来消解冲突，最终生成 5 个节点才能找到目标节点。

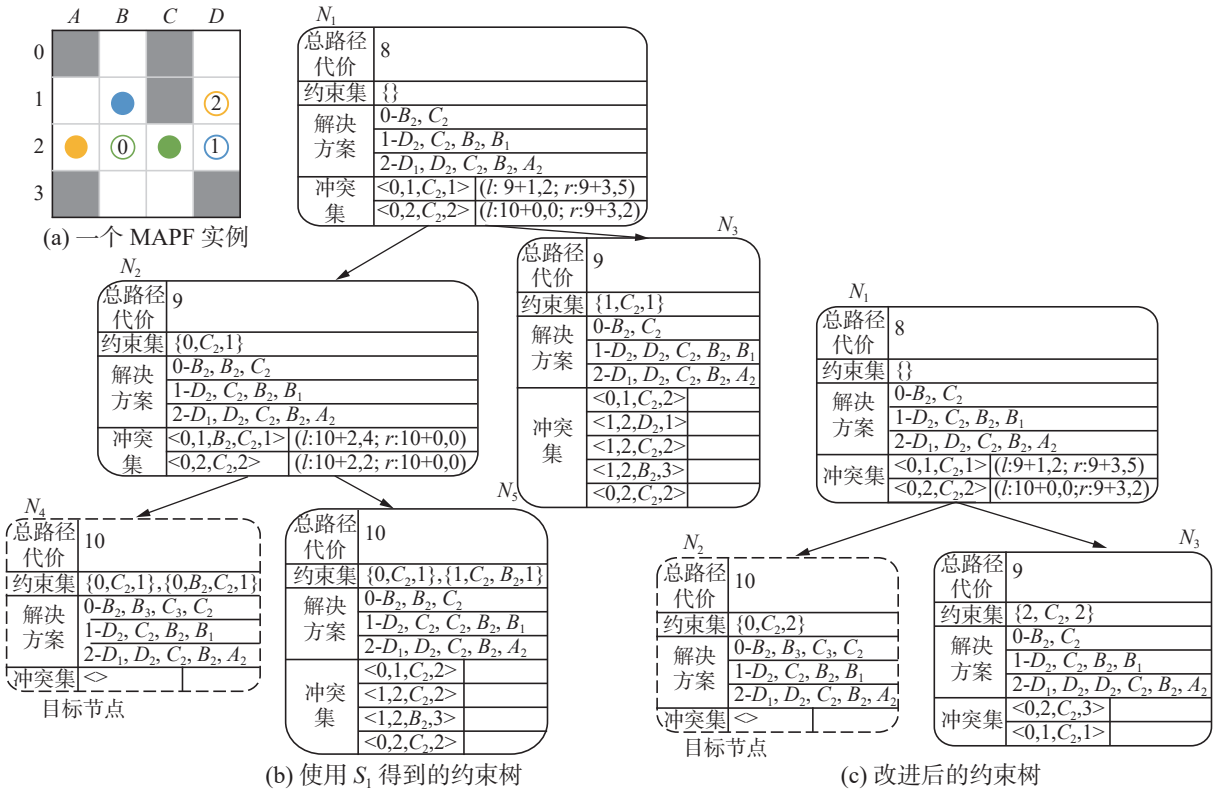

图 1　多智能体路径规划实例及其约束树

Fig. 1　MAPF instance and its constraint tree

若根据冲突的子节点的信息进一步区分，在冲突的得分相同的情况下，优先选择能使子节点更靠近目标节点的冲突，也即选择能生成g更大，冲突数更少的子节点所对应的冲突。在此情况

下, 选择< 0,2,C_2,2 >进行消解, 其约束树如图 1(c)所示, 则只需生成 3 个节点就能够找到目标节点。因此, 从进一步对冲突进行区分的角度出发, 对现有冲突选择策略进行改进使其更具有优越性。

2.2　改进冲突选择策略

接下来具体说明如何改进冲突选择策略。

定义 3 即改进冲突选择策略仍按照定义 2 优先考虑$g+h$, 其次考虑g, g越大h越小则意味着生成的子节点的总代价更接近目标节点的总代价。在两个子节点$g+h$相同的情况下优先扩展g更大的节点, 更可能使整个约束树更快地找到最优解。在此基础上, 为了充分利用子节点的信息, 还考虑另一个子节点的g。其目的是向约束树中添加g更大(更接近目标节点)的 CT 节点, 以加速求解最优解的过程。

定义 3　对给定的 CT 中节点N, S_2不仅计算N_{conf}中每个冲突c的得分v_c(同定义 2), 还要收集该冲突c的得分v_c对应的子节点的g值, 与另一个子节点的g记为g'。首先将每个冲突的得分v_c从高到低排序, 再将相同的v_c按g从高到低排序, 再将相同的g按g'从高到低排序。优先选择v_c最高的冲突, 其次选择g最高的冲突, 最后选择g'最高的冲突。

为了验证改进冲突策略的有效性, 选择 CBS 算法为多智能体规划路径, 使用 Python 语言编程和实现, 仿真实验在处理器为 Intel(R) Core(TM) i5-12500H CPU @ 2.50 GHz, RAM 为 16 GB 的计算机上运行, 分别采用冲突选择策略S_0、S_1和S_2在随机地图上进行测试。随机地图大小为 20×20, 存在有 25% 的随机障碍物, 如图 2 所示。在此随机地图上生成 50 组随机起点和终点的实例, 每个实例的运行时间限制为 20 min, 每个实例中智能体数固定为 17。为了凸显改进冲突策略和排序模型的优势, 大致将总的运行时间分为选择时间和搜索时间。选择时间精准记录了算法在整个求解过程中所有用于运行冲突选择策略的时间, 而搜索时间等于运行时间减去选择时间。搜索空间主要由 CT 大小来衡量, 它表示 CBS算法成功求解实例时生成 CT 节点的数量。仿真实验结果如表 1 所示, 其中运行时间、CT 大小、选择时间和搜索时间每一栏具体数值均为该 50 组相应数据的平均值。可以看出, 改进策略S_2与S_1的搜索时间和 CT 都小于S_0, 均可有效缩小搜索空间和缩短搜索时间, 且S_2在 CT 大小上优于S_1, 体现出改进策略的有效性。总的来说, 改进策略S_2表现更好。

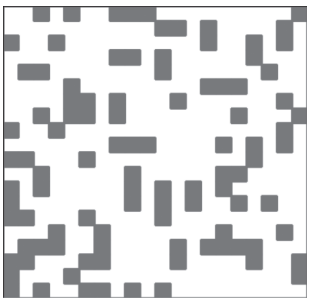

图 2　随机地图
Fig. 2　Random map

表 1　CBS 与不同冲突选择策略的结果
Table 1　Comparison of CBS with different conflict selection strategies

算法	运行时间/s	CT 大小/个	选择时间/s	搜索时间/s
CBS+S_0	4.11	363	0.00	4.11
CBS+S_1	16.56	182	14.52	2.04
CBS+S_2	8.30	118	7.03	1.27

2.3　机器学习算法学习冲突选择策略

使用文献 [19] 中使用的 SVM 方法与本文所用 RankNet 算法两种机器学习算法来分别学习冲突选择策略S_1和S_2。使用机器学习算法学习冲突选择策略主要分为两个部分: 构建数据集和训练排序模型。

2.3.1　构建数据集

需构建两个数据集: 训练集I_{train}和测试集I_{test}。I_{train}和I_{test}是由若干实例I的数据集D_I连接而成, 每个实例均由在指定地图上随机生成的始末点构成。仅在数据收集和模型学习阶段, 固定实例的智能体数。CBS 算法在求解实例I时, 需收集如下数据来构建数据集D_I, 具体步骤如下:

1) 收集 CT 中所有生成的节点$\mathcal{N}$;

2) $\forall N \in \mathcal{N}$, 收集节点$N$的冲突集合$N_{\text{conf}}$;

3) $\forall N \in \mathcal{N}$, 对每个冲突$c \in N_{\text{conf}}$收集其特征向量$\boldsymbol{\Phi}_N : N \in \mathcal{N} \to [0,1]^p$, 每个冲突$c$均有$p$个特征值;

4) $\forall N \in \mathcal{N}$, 对每个冲突$c \in N_{\text{conf}}$收集其二值标签$y_c \in \{0,1\}$, 对每个节点$N$, $y_N \in \{0,1\}^{|N_{\text{conf}}|}$。

对步骤 3) 和 4) 中的特征值和标签值的具体说明如下:

特征值　本文使用p维特征向量来表示一个冲突c, 参考文献 [19] 列出的特征指标对每个冲突生成其相应的特征值, 一个冲突包含 67 个特征值。这 67 个特征值主要包括以下几个方面:

1) 冲突的性质。主要有冲突的类型(顶点冲突和边冲突)和优先级(关键冲突、半关键冲突和非关键冲突)。

2) 在当前 CT 节点之前已经扩展的所有 CT 节点

中，统计与冲突有关的智能体或点被选中的频率。

3) 与当前 CT 节点N_{sol}相关的，CT 节点N、与冲突有关的两个智能体a_i和a_j以及顶点u或边(u,v)的信息。

4) 冲突的 MDD 特性和 WDG。

5) 对于当前冲突中的顶点u或边(u,v)，统计地图G中距离u(距离u或v)最短路径为 1~5 的所有顶点的数量。

由上述 5 个方面可知，生成特征值所需要的信息有部分依赖于所处的地图环境，但生成特征值的方法并不依赖于地图环境。因此，可以使用同样的方法在其他地图上收集相应的特征值与标签值，经训练得到与其他地图有关的排序模型。

对于每个冲突c的特征值，需以N_{conf}为一组将特征值归一化为[0,1]，以消除奇异样本数据导致的不好影响。

标签值　标签值由选定的冲突选择策略生成，并将其二值化。本文使用的标签二值化策略参考文献 [19]，将排序后N_{conf}中前 20% 的冲突的标签为 1，其余标签为 0，若其他的冲突和标签已经转化为 1 的冲突都有着相同的得分，则将其他冲突的标签也设置为 1。这样设置使得排序模型能更专注于评分较高的冲突，尽可能少被不相关的冲突干扰。

将收集好的特征数据和标签数据一一对应，以一定的格式构建数据集。

2.3.2　训练排序模型

训练 SVM 排序模型的方法与文献 [19] 类似，这里不做赘述，下面重点介绍使用 RankNet 算法训练排序模型。

RankNet 算法[20]是一个经典的排序学习算法，本文使用该算法预测每个冲突和它所属节点的相关性大小，并依此来对冲突进行排序，进而选择“好”的冲突。训练出较为准确的排序模型具体来说是要确定函数$f(x,w)$，其中x为冲突特征，w为排序模型参数。通过多次训练，使f所计算出的结果尽可能和真实值接近，即让训练集中所有冲突的偏序对P经计算后排序概率误差最小：

$$C=\frac{1}{n}\sum_{i,j\in P}C_{ij}$$

其损失函数C_{ij}为交叉熵损失函数：

$$C_{ij}=-[\bar{y}_{ij}\ln y_{ij}+(1-\bar{y}_{ij})\ln(1-y_{ij})],\ c_i,c_j\in N_{conf}$$

其中，因 RankNet 算法是基于 pairwise 的排序学习方法，故需将数据集中的冲突转换为冲突的偏序对P作为输入。P由P_N连接而成，$P_N=\{(c_i,c_j)|c_i,c_j\in N_{conf}\wedge S_{ij}=1\}$，$S_{ij}=1$表示冲突$c_i$比冲突$c_j$真实更相关。$n$为冲突的偏序对的总数，$\bar{y}_{ij}$为冲突$c_i$比冲突$c_j$排序更靠前的真实概率，$\bar{y}_{ij}=0.5\times(S_{ij}+1)$，$y_{ij}$为经模型计算后冲突$c_i$比冲突$c_j$排序更靠前的预测概率。

2.4　排序模型指导 CBS 算法选择冲突

利用得到的排序模型为 CBS 算法选择冲突的具体步骤如下：对每个当前节点N，当其冲突集N_{conf}中所含冲突数大于 1 时，先计算每个冲突的特征向量，并将其归一化后输入到排序模型中，经排序模型打分得到N_{conf}中所有冲突的得分，因排序模型打分越高代表该冲突与当前节点相关性越高，即分数越高的冲突标签为 1 的可能性越大，故选择排序模型打分最高的那个冲突c。

2.5　机器学习算法的选择

2.5.1　SVM 参数设定

为了与文献 [19] 对照，SVM 参数设定与文献 [19] 相同，即选择线性核，c为 0.01。

2.5.2　RankNet 模型设计

采用 Python 语言以及 PyTorch 框架实现 RankNet 算法。网络参数设置如表 2 设置。使用 Xavier 初始化，使用 momentum 为 0.9 的 SGD 优化器，将初始学习率设为 0.01，使用 L2 正则化和 dropout 减少过拟合。

表 2　RankNet 的网络参数

Table 2　Network parameters for RankNet

网络参数	参数取值
网络层数	3
输入层节点数	67
隐藏层节点数	18
输出层节点数	1
隐藏层激活函数	Relu
输出层激活函数	Sigmoid

2.5.3　训练结果

分别构建使用S_1和S_2策略的数据集，训练集收集 5 000 个节点，训练集与测试集的比例为 3:2。训练结果如表 3 所示。主要从测试集的 P@1 来考虑，P@1 表示经排序模型选取的冲突，标签为 1 的概率。

表 3　SVM 和 RankNet 分别学习S_1、S_2的结果

Table 3　Comparison of SVM and RankNet with conflict selection strategie S_1, S_2

机器学习算法	冲突选择策略	P@1/%
SVM	S_1	85.98
RankNet	S_1	92.60
SVM	S_2	69.84
RankNet	S_2	78.20

从结果上看，在相同机器学习算法不同冲突选择策略的情况下，学习 S_1 的效果整体上优于 S_2；在相同冲突选择策略不同机器学习算法的情况下，使用 RankNet 得到的排序模型，其 P@1 均比 SVM 高。

下面对比分析训练的模型应用到 CBS 算法中表现，分别将训练好的排序模型为 CBS 算法选择冲突。使用NET_1表示利用 RankNet 算法学习冲突选择策略 S_1 得到的排序模型，其余类似。得到的结果如表 4 所示。

表 4　CBS 与不同排序模型的结果

Table 4　Comparison of CBS with different sort models

算法	运行时间/s	CT 大小/个	选择时间/s	搜索时间/s
CBS+S_1	16.56	182	14.52	2.04
CBS+SVM_1	2.82	153	0.99	1.83
CBS+NET_1	2.70	141	1.04	1.66
CBS+S_2	8.30	118	7.03	1.27
CBS+SVM_2	2.23	113	0.72	1.51
CBS+NET_2	2.03	110	0.89	1.14

由表 4 可知，在学习相同的冲突选择策略的情况下，运行 RankNet 排序模型的时间多于运行 SVM 排序模型的时间。这是因为 RankNet 排序模型实质为网络层的权重参数，冲突的特征值要通过三层网络的计算才能得到得分。而 SVM 排序模型实质为 1 个一维权重列表，冲突的特征值只需与其一一对应相乘就能够得到冲突的得分。尽管使用 RankNet 排序模型会产生额外的时间开销，但由于其选择冲突标签为 1 的准确率往往较高，从而在求解实例的过程中表现优于 SVM 排序模型。不论是选择 SVM 排序模型还是 RankNet 排序模型，学习改进冲突选择策略 S_2 的整体表现均优于 S_1。故选择 RankNet 算法学习改进冲突策略 S_2 来为 CBS 算法选择冲突。

3　仿真结果与分析

为验证改进 CBS 算法的有效性，进行多组仿真实验。首先对比分析在指定地图下训练得到的排序模型，在相同地图不同智能体数的情况下，所提改进算法与已有改进算法的表现；其次对比分析在指定地图下训练得到的排序模型，在其他地图上，所提改进算法与已有改进算法的表现。

3.1　相同地图不同智能体数的实验对比

与 2.2 节仿真实验部分设置相同，仍使用相同的 CBS 算法，使用图 2 所示的随机地图。为防止实例过于简单，无法明显地突出所提改进算法的优势，或者过于复杂，每次求解的时间太长，故设置智能体数分别为 18、20、22 和 24，并在该地图上随机生成相应智能体数的 100 组实例，分别使用 CBS+NET_1和 CBS+NET_2求解。每个实例的运行时间均限制为 10 min。表 5 为两个算法在运行时间、CT 大小，成功率方面的数据对比。其中，成功率为算法在指定时间内成功求解的实例总数占总实例数的比例。运行时间和 CT 大小均为共同求解实例数据的平均值。

表 5　两组算法在不同智能体数下的表现

Table 5　Comparison of CBS and improved CBS with different number of agents

算法	智能体数/个	自身成功率/%	共同成功率/%	运行时间/s	CT大小/个
CBS+S_0	18	100	100	6.72	499
CBS+NET_1		100		3.02	158
CBS+NET_2		100		1.34	66
CBS+S_0	20	98	98	5.05	210
CBS+NET_1		100		4.18	103
CBS+NET_2		100		2.98	73
CBS+S_0	22	94	94	9.92	935
CBS+NET_1		98		6.35	288
CBS+NET_2		98		3.70	142
CBS+S_0	24	94	90	15.51	1 360
CBS+NET_1		100		13.53	737
CBS+NET_2		98		5.56	224

由表 5 可知，与 CBS+S_0相比，CBS+NET_2能够使运行时间缩短 61.9%，搜索空间缩小 80.1% 并有效提高算法的成功率；与 CBS+NET_1相比，CBS+NET_2能够使运行时间缩短 46.2%，搜索空间缩小 51.9%。所提改进算法在智能体数不同的实例下，仍能有效地提高了算法在路径规划方面的求解效率。

3.2　不同地图不同智能体数的实验对比

使用学到的排序模型在不同地图不同智能体数的情况下为 CBS 算法选择冲突，验证所训练的模型是否依赖于地图。地图选择标准地图的 Maze 地图，其大小为 32×32，如图 3 所示。选择由图 1 学习到的排序模型，智能体数分别设置为 6、8、10 和 12 在上随机生成 50 组实例，分别使用 CBS+S_0、CBS+NET_1和 CBS+NET_2求解。每个实例的运行时间均限制为 10 min。表 6 为两个算法在运行时间、CT 大小、成功率方面的数据对比。

由表 6 可知，与 CBS+S_0相比，CBS+NET_2能够使运行时间缩短 39.2%，搜索空间缩小 59.6% 并能够有效地提高算法的成功率；与 CBS+NET_1相

比，CBS+NET$_2$能够使运行时间缩短 15.2%，搜索空间缩小 25.4%。由此可知，所学模型在未学习过的地图上，仍能有效提高路径规划的求解效率。

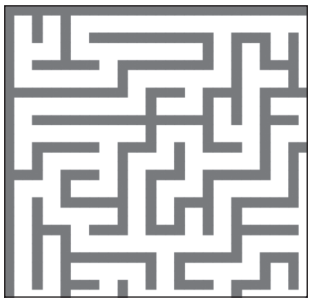

图 3　迷宫地图
Fig. 3　Maze map

表 6　两组算法在不同智能体数下的表现
Table 6　Comparison of CBS and improved CBS with different map and different number of agents

算法	智能体数/个	自身成功率/%	共同成功率/%	运行时间/min	CT大小/个
CBS+S_0		96		0.09	143
CBS+NET$_1$	6	100	96	0.10	43
CBS+NET$_2$		100		0.09	46
CBS+S_0		96		0.06	102
CBS+NET$_1$	8	98	96	0.03	52
CBS+NET$_2$		98		0.03	69
CBS+S_0		80		0.40	310
CBS+NET$_1$	10	82	80	0.14	155
CBS+NET$_2$		82		0.08	69
CBS+S_0		56		0.46	997
CBS+NET$_1$	12	64	54	0.37	438
CBS+NET$_2$		64		0.34	396

此外，本文所提的方法均可应用到其他地图中，因为排序模型与冲突的特征值和冲突的得分密切相关。而获取特征值和得分的方法均为独立的，因此该方法在不同地图中也具有很好的适应性。

4　结束语

本文主要聚焦于如何优化 CBS 算法中冲突选择的问题，提出一种改进 CBS 算法的多智能体路径规划算法。首先提出一种新的冲突选择策略，缩小了搜索空间。其次使用 RankNet 算法学习该策略，得到一个既快速又准确的排序模型，该模型能选到“好”的冲突。最后改进算法使用该排序模型为 CBS 算法选择冲突。仿真实验结果表明，当在同一地图下，所提改进算法在不同智能体数的情况下，总体表现比之前的改进算法好，求解效率和成功率得到有效提升。当地图变化时，相较于之前的改进算法，本文所提改进算法仍有很高的求解效率和成功率。未来工作考虑实际场景的动态性，开展在动态环境下提高多智能体路径规划效率的研究。

参考文献：

[1] 刘庆周, 吴锋. 多智能体路径规划研究进展 [J]. 计算机工程, 2020, 46(4): 1–10.
LIU Qingzhou, WU Feng. Research progress of multi-agent path planning[J]. Computer engineering, 2020, 46(4): 1–10.

[2] 张飞, 白伟, 乔耀华, 等. 基于改进 D*算法的无人机室内路径规划 [J]. 智能系统学报, 2019, 14(4): 662–669.
ZHANG Fei, BAI Wei, QIAO Yaohua, et al. UAV indoor path planning based on improved D* algorithm[J]. CAAI transactions on intelligent systems, 2019, 14(4): 662–669.

[3] LI J, TINKA A, KIESEL S, et al. Lifelong multi-agent path finding in large-scale warehouses[C]//Proceedings of the 19th International Conference on Autonomous Agents and Multiagent Systems. Auckland: International Foundation for Autonomous Agents and Multiagent Systems, 2020: 1898−1900.

[4] ZHANG Zheng, GUO Qing, CHEN Juan, et al. Collision-free route planning for multiple AGVs in an automated warehouse based on collision classification[J]. IEEE access, 2018, 6: 26022–26035.

[5] MA Hang, YANG Jingxing, COHEN L, et al. Feasibility study: moving non-homogeneous teams in congested video game environments[C]//Proceedings of the 13th AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment. Arizona: AAAI Press, 2017, 13(1): 270−272.

[6] MOHANTY S, NYGREN E, LAURENT F, et al. Flatland-RL: multi-agent reinforcement learning on trains[EB/OL]. (2020−10−10)[2022−11−10]. https://arxiv.org/abs/2012.05893.pdf.

[7] CHOUDHURY S, SOLOVEY K, KOCHENDERFER M, et al. Coordinated multi-agent pathfinding for drones and trucks over road networks[C]//Proceedings of the 21st International Conference on Autonomous Agents and Multiagent Systems. New Zealand: International Foundation for Autonomous Agents and Multiagent Systems, 2022: 272−280.

[8] CARTUCHO J, VENTURA R, VELOSO M. Robust object recognition through symbiotic deep learning in mobile robots[C]//2018 IEEE/RSJ International Conference on Intelligent Robots and Systems. Madrid: IEEE, 2019: 2336−2341.

[9] SHARON G, STERN R, FELNER A, et al. Conflict-based search for optimal multi-agent pathfinding[J]. Artificial intelligence, 2015, 219: 40–66.

[10] BOYARSKI E, FELNER A, STERN R, et al. ICBS: Improved conflict-based search algorithm for multi-agent pathfinding[C]//Twenty-Fourth International Joint Conference on Artificial Intelligence. Buenos Aires: AAAI Press, 2015: 740−746.

[11] FELNER A, LI Jiaoyang, BOYARSKI E, et al. Adding heuristics to conflict-based search for multi-agent path finding[C]//Proceedings of the 28th International Conference on Automated Planning and Scheduling. California: AAAI Press, 2018, 28: 83−87.

[12] LI Jiaoyang, GANGE G, HARABOR D, et al. New techniques for pairwise symmetry breaking in multi-agent path finding[C]//Proceedings of the Thirtieth International Conference on Automated Planning and Scheduling. Nancy: AAAI Press, 2020, 30: 193−201.

[13] LI Jiaoyang, HARABOR D, STUCKEY P J, et al. Disjoint splitting for multi-agent path finding with conflict-based search[C]//Proceedings of the Twenty-Ninth International Conference on Automated Planning and Scheduling. California: AAAI Press, 2021, 29: 279−283.

[14] BOYARSKI E, FELNER A, LE BODIC P, et al. Further improved heuristics for conflict-based search[C]//Proceedings of the Fourteenth International Symposium on Combinatorial Search. Guangzhou: AAAI Press, 2021, 12(1): 213−215.

[15] BOYARSKI E, FELNER A, LE BODIC P, et al. Faware conflict prioritization & improved heuristics for conflict-based search[C]//Proceedings of the 35th AAAI Conference on Artificial Intelligence. Vancouver: AAAI Press, 2021, 35(14): 12241−12248.

[16] RAHMAN M, ALAM M A, ISLAM M M, et al. An adaptive agent-specific sub-optimal bounding approach for multi-agent path finding[J]. IEEE access, 2022, 10: 22226–22237.

[17] BOYARSKI E, FELNER A, HARABOR D, et al. Iterative-deepening conflict-based search[C]//Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence. Yokohama: International Joint Conferences on Artificial Intelligence Organization, 2020: 4084−4090.

[18] LI Jiaoyang, RUML W, KOENIG S. EECBS: a bounded-suboptimal search for multi-agent path finding[C]//Proceedings of the AAAI Conference on Artificial Intelligence. Virtual: AAAI Press, 2021, 35(14): 12353−12362.

[19] HUANG Taoan, KOENIG S, DILKINA B. Learning to resolve conflicts for multi-agent path finding with conflict-based search[C]//Proceedings of the AAAI Conference on Artificial Intelligence. Virtual: AAAI Press, 2021, 35(13): 11246−11253.

[20] BURGES C, SHAKED T, RENSHAW E, et al. Learning to rank using gradient descent[C]//Proceedings of the 22nd international conference on Machine learning. New York: ACM, 2005: 89−96.

[21] STERN R, STURTEVANT N, FELNER A, et al. Multi-agent pathfinding: definitions, variants, and benchmarks[C]//Proceedings of the international symposium on combinatorial search. California: AAAI Press, 2021, 10(1): 151−158.

[22] SILVER D. Cooperative pathfinding[C]//Proceedings of the First AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment. California: AAAI Press, 2021, 1(1): 117−122.

[23] SHARON G, STERN R, GOLDENBERG M, et al. The increasing cost tree search for optimal multi-agent pathfinding[J]. Artificial intelligence, 2013, 195: 470–495.

[24] LI Jiaoyang, FELNER A, BOYARSKI E, et al. Improved heuristics for multi-agent path finding with conflict-based search[C]//Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence. Macao: International Joint Conferences on Artificial Intelligence Organization, 2019: 442−449.

[25] BOYRASKY E, FELNER A, SHARON G, et al. Don't split, try to work it out: bypassing conflicts in multi-agent pathfinding[C]//Proceedings of the international conference on automated planning and scheduling. Prague: PKP, 2015, 25: 47−51.

作者简介：

王卓然，硕士研究生，主要研究方向为多智能体路径规划。

文家燕，教授，博士，中国自动化学会青年工作委员会委员，广西自动化学会理事。主要研究方向为多智能体系统协同控制、多机器人编队控制。现主持国家自然科学基金及省部级基金项目 5 项，获专利授权 8 项，发表学术论文 2 余篇。

谢广明，教授，博士生导师，中国自动学会机器人竞赛工作委员会副主任，国际水中机器人联盟创始人，中国仿真学会机器人系统仿真专委会主任委员，主要研究方向为复杂系统动力学与控制、智能仿生机器人多机器人系统与控制。现主持国家自然科学基金重点项目等 8 项，获发明专利授权 20 余项。曾荣获国家自然科学奖二等奖、教育部自然科学奖一等奖、吴文俊人工智能科学技术奖创新奖二等奖，发表学术论文 200 余篇。