DOI：10.13973/j.cnki.robot.2017.0046

单边矩形扩展 A* 算法

李 冲，张 安，毕文豪
（西北工业大学，陕西 西安 710129）

摘 要：提出了一种新的单边矩形扩展 A*（REA*）算法．新算法采用受迫扩展规则，在以矩形单元探索地图的过程中，用单条公共边取代相邻矩形的 2 条冗余独立边，从而提高了算法效率，简化了终止条件，优化了路径质量．在无需对地图进行预处理的情况下，算法速度比传统 A* 算法提高 1 个数量级以上．算法能够保证得到栅格最优的路径点序列，且最终路径（由路径点间直线组成）总是比栅格最优路径更短．典型地图集上的实验结果表明，相比于现有 REA* 算法，新算法提高了对复杂地图的处理能力和算法效率上限．新算法路径长度更短，路径转折次数更少，因此路径质量更优．除了在低复杂且不开阔的地图上外，新算法平均效率也高于 REA* 算法．

关键词：路径搜索；启发式算法；栅格图；A* 算法；路径平衡性
中图分类号：TP18 **文献标识码**：A **文章编号**：1002-0446(2017)-01-0046-11

Single-Boundary Rectangle Expansion A* Algorithm

LI Chong，ZHANG An，BI Wenhao
(*Northwestern Polytechnical University*, *Xi'an* 710129, *China*)

Abstract: A new single-boundary rectangle expansion A* (REA*) algorithm is presented. The new algorithm adopts the forced expansion rule, and the two redundant adjacent boundaries between adjacent rectangles are replaced with a shared boundary during the process of exploring map in rectangular units, which will improve the efficiency, simplify the termination conditions and optimize the quality of the result paths. Without any offline pre-processing, the new algorithm can speed up a highly optimized A* algorithm by an order of magnitude and more. The algorithm guarantees the grid-optimal path point sequence, while the final result path (which consists of the straight lines between path points) is always shorter than grid-optimal path. Experimental results on typical benchmark problem sets show that compared with the existing REA* algorithms, the ability to deal with complex maps and the upper limit of the algorithm efficiency are improved by the proposed algorithm. The shorter result paths and less turning points are achieved with the proposed algorithm, so the quality of result paths are better. The average speed of the proposed algorithm is also better than that of REA*, except in the low complex and low open maps.

Keywords: path search; heuristic algorithms; grid; A* algorithm; breaking path symmetries

1 引言（Introduction）

路径规划是机器人与人工智能领域的一个重要的基本问题，而算法速度、路径质量和预处理代价则是衡量一个路径规划算法优劣的主要指标．A* 算法[1] 由于具有最优性、完备性和高效性，在这一领域受到了较多的关注．目前，许多高效的路径规划算法都是基于 A* 算法的改进．通常，这些算法需要牺牲 1 项或者 2 项性能指标，来获取在剩余指标上的性能提升．例如 RSR（rectangular symmetry reduction）[2-4]、Block A* [5]、JPS+（jump point search plus）[6-7]、SUB（subgoal graphs A*）[8] 和 CPD（compressed path database）[9-12] 等算法，通过牺牲路径最优性或者采用预处理手段来获得算法速度的提高，但前者导致结果路径的质量无法保证，后者导致算法在动态环境中的性能变得很差．而 Theta A* [13-14]、Lazy Theta A* [13,15] 和 A* PS（A* with post-smoothed）[16] 等算法，通过对初始路径进行“拉直”来减少路径转折，缩短路径长度，但这一过程涉及“可视性检测”，其运算代价高昂，降低了算法速度．

矩形扩展 A* 算法（REA*）[17] 是一种新近提出的路径规划算法．REA* 算法以独立的可通行矩形为单元搜索地图，并以矩形边取代传统的单个地

基金项目：国家自然科学基金（61573283）．
通信作者：李冲，lichong@mail.nwpu.edu.cn 收稿/录用/修回：2016-08-29/2016-12-22/2016-12-29

图点作为搜索节点，从而大大提高了算法效率．无需依赖于地图预处理，其算法速度比传统 A* 算法提高 1 个数量级以上，并能够保证得到栅格意义下最优的路径点和优于栅格最优的路径线．但是，受限于所采用的扩展规则，REA* 算法搜索地图的不规则性和破碎化程度会逐渐严重，另外其结果路径点中存在大量无法优化的相邻栅格点，这些缺点影响了算法效率和路径质量的提高．

本文在 REA* 算法基础上，提出了一种改进的单边 REA* 算法．单边 REA* 算法采用新的受迫搜索扩展规则，以 1 条公共边代替矩形扩展时相邻的 2 条独立边，从而提高了算法效率，简化了终止条件，优化了路径质量．典型地图集的实验结果表明，相比于原始 REA* 算法，新算法提高了算法平均效率和效率上限，增强了对复杂地图的处理能力，而且路径长度更短，路径转折次数更少，路径质量更优．

2　基于栅格图的 REA* 算法（REA* algorithm based on grid maps）

栅格化是目前最常用的环境建模方法之一[18]．本文中采用八向栅格图，机器人可以在相邻可通行点间横向、竖向或者斜向 45° 移动，但考虑到机器人实际碰撞体积不可能为 0，为安全起见，机器人斜向移动时，需要保证与运动方向垂直的 2 个相邻节点也必须是可通行点．

A* 算法是一种启发式算法，其启发式代价为

$$p.f = p.g + p.h \tag{1}$$

其中 g 为起始点 S 到栅格点 p 的当前路径长度，h 为栅格点 p 与目标点 G 间的估计代价，f 为栅格点 p 的总启发式代价．如果某一父节点可以使得子节点的 g 值减小，则称子节点被父节点更新．A* 算法总是选取具有最小总启发式代价 f 的待扩展节点，将其移入已扩展队列 Close list，并将被其更新的子节点移入待扩展队列 Open list．如果选取的估计代价 h 值总是不超过栅格点与目标点间的实际路径长度，则 A* 算法保证获得最优路径．八向栅格图中，记相邻栅格点间的横向或竖向移动距离为单位长度 1，斜向移动距离为 1.414，估计代价 h 可以采用 octile 距离，计算方法为

$$p.h = \mathrm{octile}(p,G) = 1.414 \times \min(\mathrm{d}x,\mathrm{d}y) + |\mathrm{d}x - \mathrm{d}y| \tag{2}$$

其中 $\mathrm{d}x = |x_G - x_p|$，$\mathrm{d}y = |y_G - y_p|$，$x_p, y_p, x_G, y_G$ 分别是栅格点 p 和目标点 G 的横纵坐标．

定义 1　一个栅格区间 I 是指栅格图某一行或某一列中，连续的一段栅格点所组成的集合，记为 $I = [a,b]$，其中 a、b 为区间 I 的两个端点．

REA* 算法是 A* 算法的一个改进算法，它采用栅格区间取代传统的单个栅格点作为搜索节点，其基本思路是：组成当前搜索节点的一排栅格点，作为一个整体扫描地图，探索扩展出一个尽可能大的安全可通行矩形．当前搜索节点可以视为进入该矩形的“入口”，而矩形的其它 3 条边则视为“出口”．忽略矩形内部栅格点，直接使用“入口”栅格点去更新“出口”栅格点．然后向矩形外侧延伸一个单位，从当前矩形的相邻栅格点中探索通往新矩形的“入口”，即新的搜索节点，并放入待扩展队列 Open list 中，并以搜索节点内部栅格点 f 的最小值作为该搜索节点的启发式代价．REA* 算法每次总是选取待扩展队列中启发式代价最小的搜索节点进行扩展，重复以上过程，直至找到最优路径．

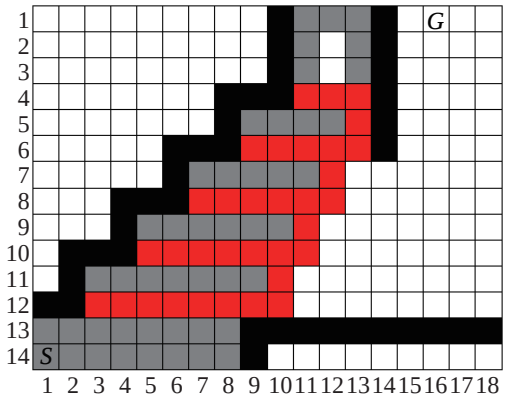
3　单边 REA* 算法（Single-boundary rectangle expansion A* algorithm）

原始 REA* 算法中，每个矩形单元需要试探更新其所有的外侧邻接栅格点，以此保证最优路径总是被包含在新的搜索节点中．子代矩形总是无法与父代矩形边缘对齐，而是会比父代矩形向外凸出一个栅格，矩形单元间的边缘呈阶梯形甚至锯齿形．这导致 REA* 算法分割地图的过程不规整，随着搜索过程进行，矩形单元破碎化越来越严重．图 1(a) 为某算例下 REA* 算法的中间状态，图 1(b) 为其最终结果．图中红色栅格代表 REA* 算法中组成搜索节点的栅格，灰色栅格代表被 REA* 算法访问但从未被包含在搜索节点的栅格．可以看到，前期矩形的不规则边缘导致后期矩形严重破碎，从而影响了搜索效率．当地图本身包含大量斜边或不规则形状障碍物时，这种现象尤为严重．

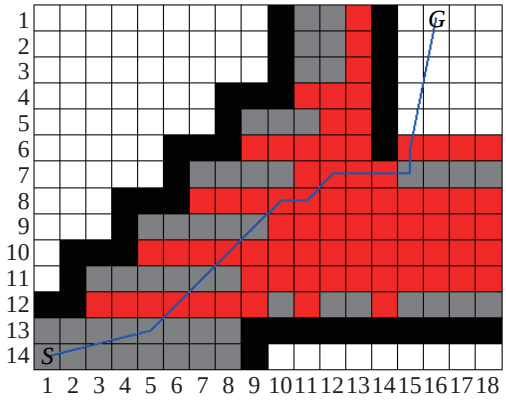
另外，原始 REA* 算法中，新搜索节点并不能通过其父代搜索节点直接生成，而是必须由当前矩形的“出口”作为中介，而很多情况下后者是冗余的（图 1 中不与障碍点或地图边界紧邻的所有灰色栅格），因而增加了不必要的开销．另一方面，REA* 通过拉直不相邻路径点间的 octile 路径来获得优于栅格最优的路径线，但是不同矩形交界处总是会产生大量相邻的、不能进一步拉直的路径点，这限制了路径优化的效果．

针对以上问题，本文提出了一种改进的单边 REA* 算法．矩形扩展时，单边 REA* 算法采用 1

条公共边代替相邻矩形的 2 条独立边，以提高算法效率和路径质量，并引入受迫扩展规则，以保持算法最优性，同时简化了算法终止条件.



(a) 中间状态



(b) 最终结果

图 1 REA* 算法分割地图的不规则性

Fig.1 Irregularity of REA* in map segmentation

3.1 扩展规则

定义 2 单边 REA* 算法中，一个搜索节点 SN（search node）可以表示为元组 $\{I, d, f_{\min}, F_{\text{forced}}\}$. 其中，$I$ 为 SN 所代表的栅格区间，$d \in \{\text{North, South, West, East}\}$，为 SN 的扩展方向，$f_{\min}$ 为 I 中所有栅格点 f 的最小值，$F_{\text{forced}} \in \{\text{False, North, South, West, East}\}$. 如果 SN.$F_{\text{forced}}$ 不为 False，则称 SN 为受迫节点，其 F_{forced} 值为受迫方向.

每个搜索节点根据继承规则（详见 3.3 节）产生并赋值，存储在待扩展队列 Open list 中. 其中，Open list 内具有最小 $f_{\min}$ 值的搜索节点被称为当前最优搜索节点（current best search node，CBN）. 单边 REA* 算法根据起始点 S 产生第一批搜索节点，然后每次从 Open list 中取出当前最优搜索节点 CBN，矩形扩展搜索地图，更新栅格点，产生子代搜索节点，循环往复，直至找到最优路径.

对于起始点，首先从 S 开始沿南、北两个方向搜索地图，直到遇到障碍点或者地图边界停止，从而得到一条竖直的可通行区间. 然后以此为轴线，沿东、西两个方向扫描地图，直到遇到障碍点或者地图边界停止，获得初始可通行矩形单元. 改变竖直搜索和水平搜索的顺序可能产生不同的结果，但并不影响算法最优性.

对于其他当前最优搜索节点 CBN（非起始点），沿其扩展方向扫描地图，直至遇到障碍点或地图边界停止，获得当前可通行矩形单元，记为 UR（unit of rectangle）.

起始点 S 或者当前最优搜索节点 CBN 可以视为 UR 的“入口”，该矩形的其他边为“出口”. 注意，单边 REA* 算法中，CBN 的 2 个端点既是“入口”的一部分，也是侧边“出口”的一部分.

若目标点 G 被包含在当前可通行矩形 UR 中，则算法终止；否则，根据更新规则，使用“入口”栅格点的路径信息更新“出口”栅格点.

3.2 更新规则

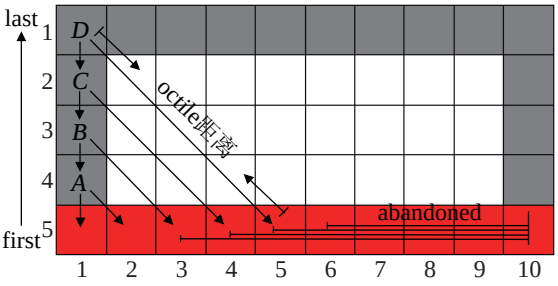
对于起始点 S 形成的初始可通行矩形，直接以边界上各个栅格点与起始点 S 间的 octile 距离作为其 g 值.

如果 UR 不是初始的可通行矩形，假设 CBN.d 为 North，文 [17] 已经证明，此时只需要考虑 CBN 中，位于 UR “出口”栅格东南方向和西南方向 2 条射线夹角之间的那一部分栅格点，即可保证算法最优.

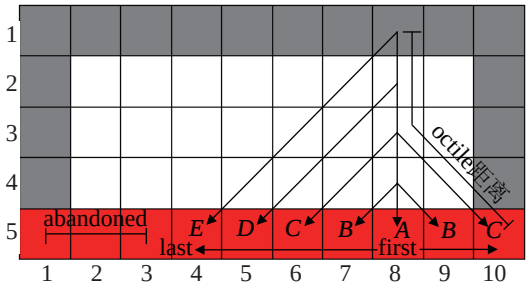
图 2(a) 为单边 REA* 算法更新 UR West 边界上栅格点的过程，红色区域为 CBN. 图中字母表示单边 REA* 算法更新栅格点的顺序，图中的箭头指向每个栅格需要考虑的潜在父代栅格. 单边 REA* 算法首先更新栅格点 $A(1, 4)$，显然，在它的东南方向和西南方向 2 条射线夹角之间，所需要考虑的潜在父代点只有栅格 (1, 5)（相对 octile 距离为 1）和栅格 (2, 5)（相对 octile 距离为 1.414）. 然后向北移动 1 个单位，单边 REA* 算法考虑更新栅格 $B(1, 3)$. 对于 CBN 中位于栅格 (3, 5) 左侧的部分，由于它们通向栅格 $B(1, 3)$ 的 octile 路径（或等效 octile 路径）必然通过栅格 $A(1, 4)$，而栅格 $A(1, 4)$ 已经被 CBN 更新过，因此直接将栅格 $A(1, 4)$ 作为栅格 $B(1, 3)$ 的潜在父代栅格即可，且相对 octile 距离恒为 1. 另外，栅格 $B(1, 3)$ 还需考虑潜在父代栅格 (3, 5)，相对 octile 距离为 $1.414 + 1.414 = 2.828$. 依此类推，单边 REA* 算法由南向北更新栅格点，每个栅格点仅需考虑其南侧相邻栅格点（相对 octile 距离恒为 1）以及其在东南方向的对角点（相对 octile 距离每次增加 1.414）. 此迭代过程可以表示为

$$\begin{cases} p_i^W.g = \min\{p_i^W.g,\ p_{i-1}^W.g+1,\ p_{di}^W.g+\text{octile}_i\} \\ p_i^W = p_{i-1}^W + (0,-1) \\ p_{di}^W = p_{di-1}^W + (1,0) \\ \text{octile}_i = \text{octile}_{i-1} + 1.414 \end{cases} \tag{3}$$

其中，$p_{d0}^W = p_0^W$，$\text{octile}_0 = 1.414$，$i = 1,2,3,4$．注意，只有当 CBN 所代表路径信息能够降低“出口”栅格当前 g 值时，才能更新该栅格点．另外，每当栅格点被更新时，它总是沿着父代节点回溯，找到最后一个位于 UR 中的栅格作为父代节点，从而跳过中间栅格点．例如，图 2(a) 中，假设栅格 $B(1, 3)$ 的父代栅格为 (3, 5)，如果栅格 $C(1, 2)$ 能够被栅格 $B(1, 3)$ 更新，则栅格 $C(1, 2)$ 将跳过 $B(1, 3)$，直接取栅格 (3, 5) 为父节点，这将获得优于栅格最优的结果路径．East 边界更新过程与 West 边界处理方法类似，因而此处省略．



(a) 更新 West 边界



(b) 更新 North 边界

图 2　CBN.d = North 时的单边 REA* 算法更新规则

Fig.2　Update rule of single-boundary REA* algorithm when CBN.d = North

图 2(b) 为更新 UR North 边界栅格的过程．栅格 (8, 1) 是 UR 北边界的一个点，CBN 中的字母表示单边 REA* 算法检测潜在父代栅格点的顺序．首先，单边 REA* 算法检测 CBN 中与栅格 (8, 1) 同一列的栅格点 $A(8, 5)$，初始相对 octile 距离取两栅格点间的行数差，即 $5 - 1 = 4$．然后以栅格点 $A(8, 5)$ 为中心，向东和西各移动 1 个单位，检测标号为 B 的一对栅格点 (7, 5) 和 (9, 5)，此时相对 octile 距离增加 0.414，即 4.414．依此类推，直至超出 CBN 中栅格点 (8, 1) 东南方向和西南方向 2 条射线的夹角范围．此迭代过程可以表示为

$$\begin{cases} p_i^N.g = \min\{p_i^N.g, \cdots, p_{\text{swi}}^N.g+\text{octile}_i,\ p_{\text{sei}}^N.g+\text{octile}_i\} \\ p_{\text{swi}}^N = p_{\text{sw}(i-1)}^N + (-1,0) \\ p_{\text{sei}}^N = p_{\text{se}(i-1)}^N + (1,0) \\ \text{octile}_i = \text{octile}_{i-1} + 0.414 \end{cases} \tag{4}$$

其中，$p_{\text{sw}0}^N = p_{\text{se}0}^N = p_i^N + (0,4)$，$\text{octile}_0 = 4$．

可以看出，单边 REA* 算法通过独特的迭代策略来更新“出口”栅格信息，避免了“出口”栅格与“入口”栅格间的一一遍历，并通过累加操作，规避了 octile 距离计算中的乘法运算，从而大大提高了算法效率．

如果 UR 的某“出口”边上有任一栅格点在以上过程中被更新，则该“出口”边依据继承规则产生子代搜索节点；否则，当且仅当最优搜索节点 CBN 的 F_{forced} 值恰好为该“出口”边的位置时，该“出口”边被允许产生子搜索节点．例如，图 2(a) 中，如果 West 侧边 [(1, 1), (1, 5)] 中没有栅格点被 CBN 更新，则当且仅当 CBN.F_{forced} 值为 West 时，该侧边参与产生子搜索节点．

定义 3　如果 UR 某条侧边上远离 CBN 的端点被更新，则称该侧边为此次更新的受迫侧边，该端点栅格为受迫栅格．

例如，图 2(a) 中，如果栅格 $D(1, 1)$ 被 CBN 更新，则侧边 [(1, 1), (1, 5)] 为此次更新的受迫侧边，栅格 $D(1, 1)$ 为受迫栅格．

3.3　继承规则

定义 4　如果 UR 外侧与某“出口”栅格点水平紧邻或竖直紧邻的栅格点均为不可通行的障碍点，则称该“出口”栅格点为非直通栅格点．

显然，由于障碍点的存在，根据安全通行规则，不存在从某一非直通栅格点不经过其他栅格点而直接通向矩形外侧的合法路径．

UR 的“出口”边总是被其可能存在的非直通成员栅格分割为既不相邻也不重叠的若干区间．如果该“出口”边被允许产生子搜索节点，则单边 REA* 算法根据式 (1) 和 (2) 为直通栅格点更新 f 值，然后每个连续的栅格区间各自产生新搜索节点．新节点范围 I 为该直通栅格区间本身，扩展方向 d 为所在“出口”边在 UR 中的位置，$f_{\min}$ 取 I 中所有栅格点 f 的最小值．

特别地，当且仅当某个直通栅格区间包含受迫

栅格时，其生成的搜索节点的 F_{forced} 值取当前 CBN 的扩展方向 d，否则 F_{forced} 值为 False. 例如，图 2(a) 中，如果栅格 $D(1, 1)$ 为受迫栅格且栅格 $D(1, 1)$ 左侧相邻栅格不是障碍点，则包含栅格 $D(1, 1)$ 的子代搜索节点为受迫节点，其 F_{forced} 值取当前 CBN 的 d 值，即 North.

3.4 终止条件

单边 REA* 算法根据起始点 S 产生第一批搜索节点，然后每次从 Open list 中取出当前最优搜索节点 CBN，矩形扩展搜索地图，更新栅格点，产生继承搜索节点，循环往复.

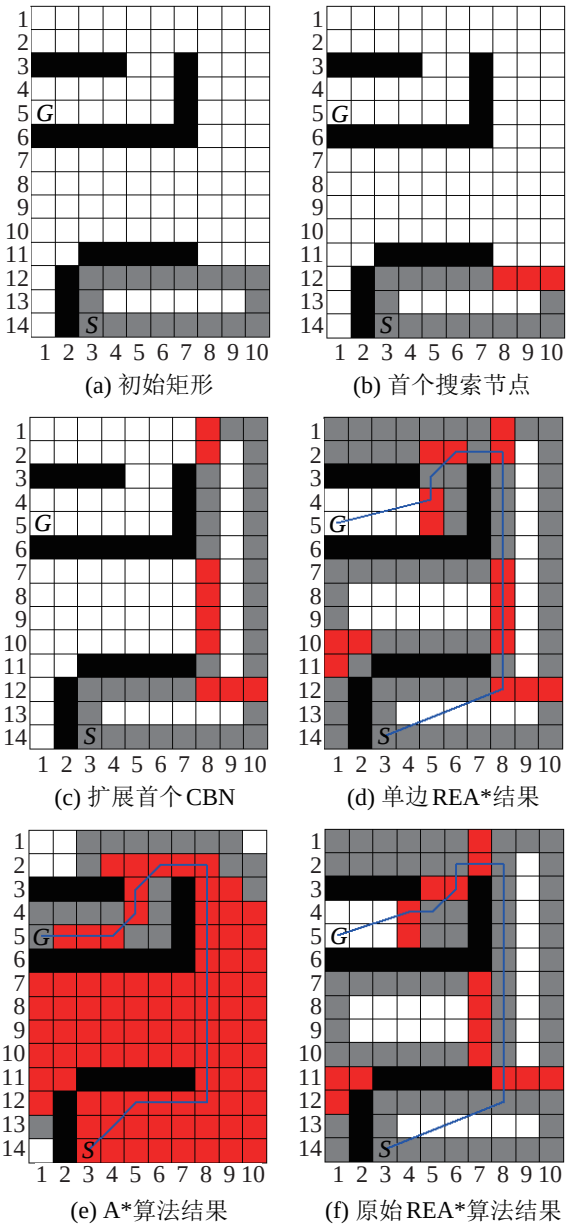


图 3 同一算例下各个算法搜索结果对比

Fig.3 Comparison of different algorithms on the same example

如果 Open list 为空，则起始点 S 与目标点 G 间不存在可通行路径. 否则，当且仅当目标点 G 被包含在当前可通行矩形 UR 中时，算法终止，此时取 CBN 中具有最小 f 的栅格（或者 S，若 UR 为初始可通行矩形）为 G 的父代栅格，从目标点 G 沿父代回溯，即可得到栅格最优路径点，以直线依次连接各个路径点，即为优于栅格最优的结果路径.

图 3 为同一算例下的各个算法搜索结果对比. 图 3(a) ～ (d) 为单边 REA* 算法的各个搜索阶段和最终结果. 图 3(e) 为同一算例下 A* 算法的搜索结果，其中红色栅格代表 A* 算法中的 close 点，灰色栅格代表 open 点. 图 3(f) 为原始单边 REA* 算法的搜索结果. 可以看出，新算法搜索的栅格点更少，且路径更优.

4 最优性证明（Proof of optimality）

设 $\pi = \{p_0, p_1, \cdots, p_n\}$ 为从 S 到 G 的一条栅格最优路径，且路径点顺序排列，其中 $p_0 = S$，$p_n = G$. 搜索过程中，对于最优路径上任一点 p_i，如果 $p_i.g$ 等于 S 到 p_i 的最优路径长度，则称 p_i 是最优点，显然，此后 p_i 不会再被更新. 由于 $p_i.h = \text{octile}(p_i, G)$ 总是不超过 p_i 到 G 的真实最短距离，因此 $p_i.f \leqslant \pi.l$，其中 $\pi.l$ 表示路径 π 的长度，下同. 如果 $i < j$，则称 p_j 是 p_i 在 π 上的下游路径点.

定义 5 如果在某次扩展更新过程中，p_i 由非最优状态点变为最优点，根据更新规则和继承规则，只要 p_i 不是其所在可通行矩形的非直通栅格点，则必然会有一个包含 p_i 的新搜索节点产生，该搜索节点称为 p_i 的伴随搜索节点，根据定义 2，其 $f_{\min}$ 值必然不大于 $p_i.f$.

引理 1 如果在某个搜索节点的扩展过程中，p_i 由非最优状态变为最优点，则扩展结束后，p_i 或其某个下游节点的伴随搜索节点存在于 Open list 中.

证明： 假设当前可通行矩形为 UR，则 $\text{UR} \cap \pi$ 为最优路径处在 UR 中的部分，且 $p_i \in \text{UR} \cap \pi$. p_i 由非最优状态变为最优点，则其父代栅格（包含在 CBN 中）也必然是最优点，文 [17] 已经证明，在 3.2 节的更新规则下，此次更新后 $\text{UR} \cap \pi$ 中所有栅格点均处于最优状态. 设 p_j 为 $\text{UR} \cap \pi$ 中序号最大的栅格点（亦即最优路径 π 处在 UR 中的最后一个点），显然 $j \geqslant i$，p_j 存在两种情况：

1) p_j 在此次更新过程中由非最优状态点变为最优点. 由于最优路径 π 的下一个点 p_{j+1} 必须处在 UR 外侧且能与 p_j 直接连通，p_j 显然是直通栅格点，因此 p_j 的伴随搜索节点必然会被产生并放入 Open list，得证.

2) p_j 在此次更新前已经处于最优状态．显然此时 $j \geqslant i+1$，考虑 p_j 由非最优状态变为最优点的那一次扩展，最优路径 π 在该可通行矩形中的最后一个最优点 p_k（$k \geqslant j$）亦只能有 1) 和 2) 两种情况．由于每次情况 2) 中，最优点都在沿 π 向下游移动，但 π 的长度是有限的，因此情况 2) 不可能无限循环，情况 2) 终将以情况 1) 结束．

综上，引理 1 得证．

引理 2　任意时刻，待扩展队列 Open list 中总存在这样一个搜索节点，它是最优路径上某一点的伴随搜索节点或者它被扩展时必然产生最优路径上某一点的伴随搜索节点．

证明：首先，位于初始可通行矩形边上的最优路径点，必然会直接被起始点 S 赋值为最优状态并产生伴随搜索节点．

其次，当最优路径上某一点 p_i 的伴随搜索节点 SN_i 被扩展时，假设 p_j 为 π 在当前可通行矩形 UR 中的最后一个点，显然 $i \leqslant j$．

如果 $i < j$，根据引理 1，本次扩展后 p_j 一定处于最优状态，且其伴随搜索节点被放入 Open list 中．

如果 $i = j$，则 p_i 为 UR “入口”边与“出口”边的交点，亦即 SN_i 的某个端点．假设 $\mathrm{SN}_i.d =$ “North”且 p_i 为 SN_i 的左端点，SN_i 的父代搜索节点为 PCBN（parent of current best search node），对应父代可通行矩形为 PUR（parent of UR）．由于 p_i 被 PCBN 更新为最优状态，因此 $p_i \notin$ PCBN，PCBN.d 只能是“North”或“West”．如图 4 所示，$\mathrm{SN}_i.I = [(3, 3), (9, 3)]$，$p_i = (3, 3)$，此时 p_{i+1} 只能是栅格 (2, 2) 或栅格 (2, 3)（否则 π 显然不是最优路径），且无论哪种情况，栅格 (2, 3) 必定不是障碍点，故 $p_i(3, 3)$ 总是 PUR 左侧边的可直通栅格．

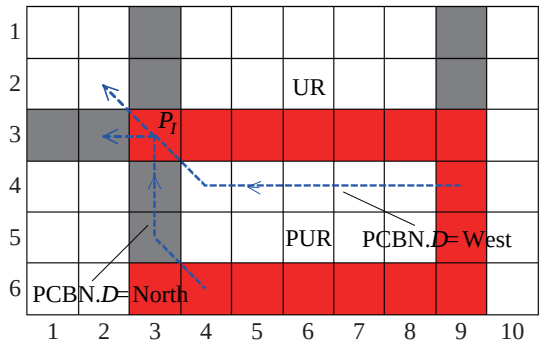


图 4　SN$i.d$ = “North”且 $p_i = (3, 3)$ 时单边 REA* 算法的搜索节点

Fig.4　Search nodes of single-boundary REA* algorithm when SN$i.d$ = “North” and $p_i = (3, 3)$

图 4 中，设 PCBN.d = “North”，则 PCBN.I = [(3, 6), (9, 6)]．由于 p_i 在此次扩展中被 PCBN 更新，根据继承规则，PUR 左侧边 [(3, 3), (3, 6)] 必然产生一个包含 p_i 的受迫搜索节点，且其受迫方向 F_{forced} 值为“North”．根据更新规则，该受迫搜索节点必然产生包含栅格区间 [(2, 3), (3, 3)] 且 d = “North”的子代搜索节点，该子代搜索节点被扩展时，必将更新栅格 (2, 2) 和栅格 (2, 3)，由引理 1，p_{i+1} 或其下游路径点的伴随搜索节点必然会产生．

同理，假设 PCBN.d = “West”，则 PCBN.I = [(9, 3), (9, 6)]．PUR 侧边 [(3, 3), (9, 3)] 必然产生一个包含 p_i 且其受迫方向 F_{forced} 值为“West”的受迫搜索节点，该受迫搜索节点必然产生包含栅格区间 [(3, 2), (3, 3)] 且 d = “West”的子代搜索节点，当其扩展时，p_{i+1} 或其下游路径点的伴随搜索节点必然会产生．

引理 3　假设 π_0 是单边 REA* 算法找到的结果路径，则 $\pi_0.l \leqslant \mathrm{CBN}.f_{\min}$，CBN 是算法终止时正在被扩展的当前最优搜索节点．

证明：根据 3.4 节的终止条件，目标 G 必然处在 CBN 扩展的可通行矩形中，因此 CBN 中所有栅格点均可与 G 以 octile 路径相连，每个栅格点 f 值均代表一条实际可通行路径的长度．根据定义，$\mathrm{CBN}.f_{\min}$ 代表的只是该搜索节点被产生时其内部栅格点的最优路径信息，$\pi_0.l$ 为该搜索节点成为当前最优搜索节点时其内部栅格点最新的最优路径信息，由于任何栅格点的 f 值在更新过程中均不可能增大，显然 $\pi_0.l \leqslant \mathrm{CBN}.f_{\min}$．$\pi_0.l < \mathrm{CBN}.f_{\min}$ 的情况是指，该搜索节点成为当前最优搜索节点而被扩展之前，其内部栅格点仍然可能被其他搜索节点更新，但这种信息更新只会形成新搜索节点，而不会跟踪修改已生成的旧搜索节点的 $f_{\min}$．

定理　单边 REA* 算法总是能找到最优栅格路径，除非它不存在．

证明：只要存在 S 到 G 的可行路径，则根据引理 2，Open list 中总是包含至少 1 个搜索节点，因此 Open list 不会为空．

假设引理 2 中所保证存在的这个搜索节点为 SNX，由于它在产生时总是包含一个最优点，故 $\mathrm{SNX}.f_{\min} \leqslant \pi.l$．假设 π_0 是任意一条非栅格最优的路径，显然 $\pi_0.l > \pi.l$．根据引理 3，当 π_0 被找到时，满足 $\mathrm{CBN}.f_{\min} \geqslant \pi_0.l > \pi.l \geqslant \mathrm{SNX}.f_{\min}$．但是根据扩展规则，既然 Open list 中存在 $\mathrm{SNX}.f_{\min} < \mathrm{CBN}.f_{\min}$，则这个“CBN”显然不是当前最优搜索节点，因此它不会被扩展，非最优路径 π_0 也根本不会被找到．

因此，单边 REA* 算法总是找到最优路径，除非它不存在.

5 算法性能（Efficiency of the algorithm）

单边 REA* 算法继承了原始 REA* 算法的优点，并在算法效率和路径质量上有明显提高. 图 5 给出了图 1 算例的单边 REA* 算法结果.

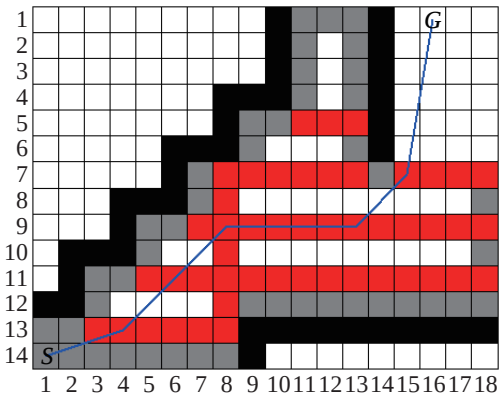


图 5 图 1 中示例的单边 REA* 算法结果

Fig.5 Result of single-boundary REA* algorithm on the same example of Fig.1

矩形剪裁效率：矩形扩展规则能够忽略掉矩形内部不必要的栅格点，算法需要考虑的栅格数量比传统 A* 算法少得多，配合高效的迭代更新策略，可以大大提高算法效率. 原始 REA* 算法中，相邻矩形分别处理自己的边界，而单边 REA* 算法中相邻矩形使用一条公共边，省去了冗余边的处理，剪裁效率更高.

Open list 长度：在忽略内部栅格点的基础上，矩形扩展算法以栅格区间为单位组成搜索节点，相比于以单个栅格为搜索节点的传统 A* 算法，Open list 的长度进一步减小，算法迭代次数和每次迭代的链表操作（包括元素排序、移动和删除等）复杂度都大大减小. 相比于原始 REA* 算法，单边 REA* 算法分割地图更加规整，所需矩形单元数量更少，这有利于进一步缩短 Open list 长度，但另一方面，搜索节点的受迫特性可能会增加一些冗余的搜索节点，最终效果与地图特性有关（详见第 6 节）.

终止条件：矩形扩展规则下，只要目标点 G 处在当前最优搜索节点扩展出的可通行矩形内，算法即可结束，从而避免了很多不必要的处理，最明显的例子是当 G 处在初始可通行矩形内时，矩形扩展算法可直接获得最优路径，A* 算法则不能. 单边 REA* 算法将原始 REA* 算法的 3 个终止条件合并简化成 1 个，更加简洁高效.

路径质量：REA* 算法和单边 REA* 算法都保证栅格最优的路径点序列，顺序路径点间不一定直接紧邻，但不紧邻的路径点一定处在同一个可通行矩形. 在该矩形保护下，路径点之间可以用任意 octile 路径[8,17] 连接（此时获得栅格最优路径），但更好的方案是直接以直线连接路径点，即可获得优于栅格最优的路径. 但原始 REA* 算法中，如果路径点分别处于相邻矩形单元的一对邻边上，则它们直接紧邻，该路径段无法被拉直或缩短. 单边 REA* 算法将 2 条独立邻边合并成 1 条公共边，移除了这种情况，路径点个数更少，进一步增强了路径优化的潜力. 例如，图 3 中，A* 算法的结果路径长度为 24.656（栅格最优距离），路径转折次数为 7，原始 REA* 算法的路径长度为 23.962，路径转折次数为 6，同样的算例，单边 REA* 算法的路径长度为 23.922，路径转折次数为 5，显然路径质量更高.

另外，文 [17] 中提出的针对 REA* 算法的优化措施同样适用于单边 REA* 算法，且不会影响算法最优性. 例如，可以为可通行矩形每个边界栅格标记一个扩展方向，使晚扩展的搜索节点能够检测并避开已经被扩展的区域，从而避免不必要的重复扩展；如果目标点 G 能扩展出比初始点 S 更大的初始矩形，则可以交换 S 和 G；可以通过检测和回避半封闭区域来增强对复杂地图的处理能力等.

6 实验结果（Experimental results）

为验证和对比算法效果，采用与文 [17] 相同的实验设置和对比算法，对单边 REA* 算法进行了仿真测试. 所有测试地图和算例均取自于丹佛大学举办的 Grid Based Path Planning Competition 比赛[18-19]，实验中共选取了来自 4 个地图集的 331 张测试地图和 917 835 个算例. 图 6 为来自各个地图集的示例地图.

Dragon Age Origin（DAO）：共包含 156 张地图和 159 465 个算例，均来源于同名电脑游戏，地图大小从 22×28 到 1260×1104（单位为栅格）.

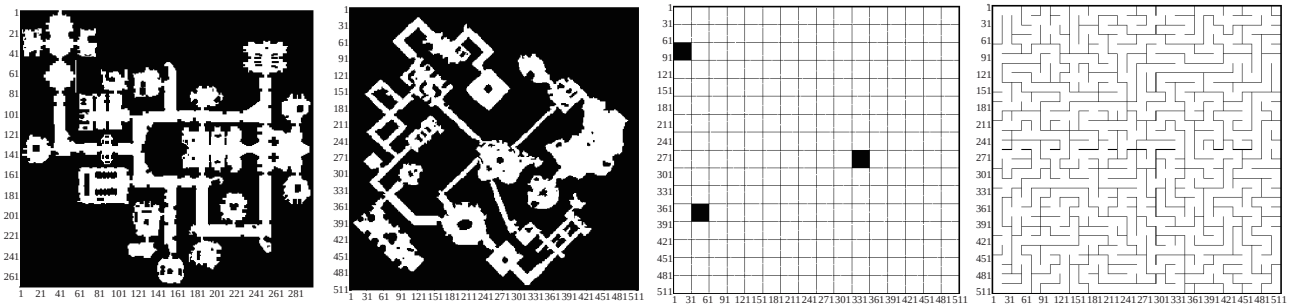
Baldur's Gate II（BG2）：共包含 75 张地图和 93 160 个算例，均来源于同名电脑游戏，地图大小均为 512×512. 该地图集的特点是含有大量 45° 倾斜的场景.

Room：共包含 40 张 512×512 大小的人工地图和 78 840 个算例，整个地图被分割成小方块，相邻方块间有随机生成的连通口，方块大小为 8×8、16×16、32×32 或 64×64.

Maze：共包含 60 张迷宫地图和 586 370 个算例，迷宫大小为 512×512，走廊宽度为 1、2、4、8、16 或 32.

实验硬件为 CPU 型号 Core i3，主频 3.2 GHz，内存 2 GB 的 PC 机，实验结果如表 1 所示.

表 1 中列出了实验中平均每解决 1 个算例，A* 算法与单边 REA* 算法分别需要的平均搜索节点总数、平均路径长度、平均路径转折次数和平均运行时间. 对于单边 REA* 算法，路径长度表示为栅格长度/实际长度，其中栅格长度为各个结果路径点间 octile 距离之和，栅格长度为各个结果路径点间直线距离之和.



(a) DAO　(b) BG2　(c) Room　(d) Maze

图 6　测试地图集示例

Fig.6　Sample maps in the experiment

表 1　单边 REA* 算法及 A* 算法性能测试对比结果

Tab.1　Results of single-boundary REA* and A* algorithms

地图集	平均搜索节点总数		平均路径长度		平均路径转折次数		平均搜索时间 /ms	
	A*	单边 REA*	A*	单边 REA*	A*	单边 REA*	A*	单边 REA*
DAO	19662.2	1717	418.57	418.57/412.76	49.2	23.0	14.43	4.03
BG2	16079.5	399	249.69	249.69/244.86	18.4	8.8	13.10	1.96
Room8	37953.5	4599	389.85	389.85/386.72	119.9	89.8	33.44	6.98
Room16	42649.9	1112	385.66	385.66/381.08	80.5	49.6	37.19	3.22
Room32	56094.5	300	392.20	392.20/386.30	53.4	26.1	48.70	2.54
Room64	79885.6	84	426.37	426.37/418.71	41.2	13.5	68.50	2.60
Maze1	64132.1	20214	2986.14	2986.14/2986.14	887.0	887.0	37.61	18.75
Maze2	83425	8368	2193.1	2193.1/2119.67	822.8	510.3	54.86	10.56
Maze4	104524	3107	1953.39	1953.39/1901.54	432.5	278.2	71.58	6.89
Maze8	126648	1016	1927.86	1927.86/1877.95	315.9	161.3	87.05	4.77
Maze16	145337	317	1730.85	1730.85/1689.79	199.7	79.3	102.24	3.98
Maze32	149669	85	1246.84	1246.84/1217.69	95.3	31.8	110.28	3.86

表 1 数据表明，单边 REA* 算法继承了 REA* 算法原有的优点，通过采用矩形扩展规则，单边 REA* 算法找到最优路径所需的平均搜索节点总数仅为 A* 算法的 0.06% ～ 31.25%，这意味着单边 REA* 算法的迭代次数和链表操作的复杂度均远小于 A* 算法，同时单边 REA* 算法采用了高效的迭代更新规则，这些都使得单边 REA* 算法运算时间远小于 A* 算法，在表现最好的地图子集 Room64 和 Maze32 上，单边 REA* 算法的效率比 A* 算法分别高 26.48 和 28.55 倍. 另一方面，在保证最优栅格路径的同时，除了 Maze1 地图子集（走廊宽度为 1 的迷宫地图，此时矩形扩展无法缩短路径长度），单边 REA* 算法总是能给出路径长度和转折次数均小于栅格最优路径的实际结果路径.

表 2 给出了单边 REA* 算法与目前几种路径搜索算法的性能对比. 为了排除不同算法运行时的软/硬件环境差异，表 2 中采用了各个算法搜索速度和路径长度与标准 A* 算法之间的比值. 例如，时间加速系数如果为 2，则表示算法搜索时间为 A* 算法的 1/2；长度优化系数如果为 0.9，则表示算法实际结果路径长度为 A* 算法的 90%. 表 2 中各个对比算法的性能数据均取自于该算法的原始发表文

献[2-7].

可以看出，在衡量算法的 3 个指标（搜索时间、路径长度和预处理代价）上，单边 REA* 算法都有良好的表现．作为一个在线算法，其搜索速度甚至优于部分预处理算法（例如 RSR、Block A*）．与另一在线算法 JPS[6] 相比，在不同地图上两种算法的搜索速度各有胜出，但 JPS 算法仅能保证结果路径是栅格最优的，而单边 REA* 算法可以获得优于栅格最优的更短结果路径．

表 3 给出了单边 REA* 算法相对于原始 REA* 算法的改进效果．为了消除不同地图和不同量纲造成的参数数量级差异，表 3 中以 2 种算法相对于 A* 算法的性能提高程度进行对比．例如，节点压缩系数如果为 2，则表示表示改进算法搜索节点总数为 A* 算法的 1/2；转折优化系数如果为 0.9，则表示改进算法实际结果路径所需转折次数为 A* 算法路径的 90%．

路径质量：通过合并相邻边，单边 REA* 算法减少了所需路径点数目．从表 3 可以看出，在各个地图上，单边 REA* 算法的路径转折次数明显少于原始 REA* 算法，路径更加平滑，有利于实际执行，且路径长度比原始 REA* 算法更短，因此单边 REA* 算法的路径整体质量明显优于原始 REA* 算法．

表 2 各主要算法性能对比（以标准 A* 算法为基准）

Tab.2 Comparison of the algorithms (with respect to A*)

	地图集	时间加速系数	路径长度系数	是否预处理
单边 REA*	DAO	3.62	＜1	否
	BG2	6.81		
	Room	4.87 ～ 26.48		
	Maze	2.01 ～ 28.55		
RSR	BG2	2.0 ～ 3.0	1	是
	Room	5.0 ～ 9.0		
Block A*	DAO	2.1	＞1	是
	BG2	2.3		
JPS	BG2	2.0 ～ 30.0	1	否
	Room	3.0 ～ 16.0		
JPS+	DAO	20.8 ～ 180.4	1	是

表 3 REA* 算法和单边 REA* 算法相对于 A* 算法的性能对比

Tab.3 Comparison of REA* and single-boundary REA* algorithms over A* algorithm

地图集	节点压缩系数		时间加速系数		. 路径长度系数		转折优化系数	
	REA*	单边 REA*	REA*	单边 REA*	REA*	单边 REA*	REA*	单边 REA*
DAO	20.6	11.5	4.45	3.62	0.987	0.986	0.624	0.466
BG2	38.2	42.3	4.67	6.81	0.982	0.981	0.723	0.478
Room8	11.0	8.3	5.54	4.87	0.997	0.992	0.808	0.749
Room16	43.2	38.4	12.01	11.77	0.990	0.988	0.694	0.616
Room32	166.4	187.0	18.55	19.51	0.986	0.985	0.569	0.489
Room64	609.8	951.2	23.22	26.48	0.983	0.982	0.386	0.328
Maze1	3.2	3.2	2.04	2.01	1.000	1.000	1.000	1.000
Maze2	10.0	10.0	5.01	5.20	0.983	0.967	0.997	0.620
Maze4	33.7	33.6	9，64	10.39	0.980	0.974	0.772	0.643
Maze8	124.8	124.7	16.34	18.25	0.975	0.974	0.634	0.511
Maze16	456.0	458.5	23.82	25.68	0.976	0.976	0.490	0.397
Maze32	1760.8	1760.8	26.27	28.55	0.977	0.977	0.407	0.334

运算时间：单边 REA* 算法对运算时间的影响与地图本身特性有关．一幅地图可以从复杂度和开阔度两方面衡量．地图复杂度指地形不规则的严重程度，地图内障碍物边缘越凹凸不平，斜边越多，地图复杂度越高．地图开阔度指可通行矩形的平均大小，地图内可通行区域越狭窄，地图开阔度越低．图 7 给出了测试地图集的地图特性分布．

相对于原始 REA* 算法，单边 REA* 算法对运算时间的影响主要包括以下 3 个方面：

1) 单边 REA* 算法的矩形单元处理效率更高，能够缩短搜索节点平均处理时间．一般地，地图开阔度越高，矩形单元平均尺寸越大，该效果越明显．

2) 单边 REA* 算法能够通过使矩形单元整齐化

来减少矩形单元总数，进而减少搜索节点总数．一般地，地图复杂度越高，原始 REA* 算法的矩形单元碎片化和交叉重叠越严重，新算法的规整效果越明显．

3) 单边 REA* 算法会产生冗余的受迫节点，进而增大搜索节点总数．由于受迫情况总是发生在矩形单元的 4 个顶点栅格，因此冗余的受迫节点与矩形单元的总数正相关，与矩形单元平均尺寸负相关．一般地，地图开阔度越高，该效果越弱．

其中，效果 1) 和 2) 有利于提高算法效率，效果 3) 不利于提高算法效率．显然，地图开阔度越高，效果 1) 越强，效果 3) 越弱，新算法整体效率越高；同样，地图复杂度越高，效果 2) 越强，新算法的整体效率也越高．

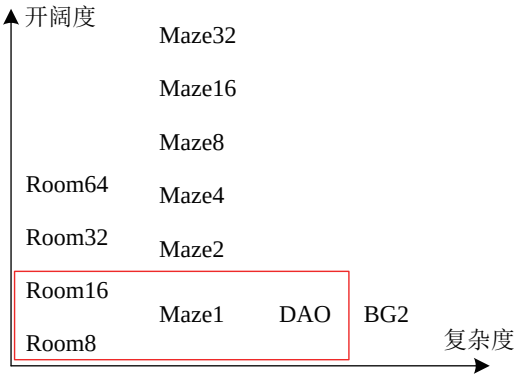


图 7　测试地图特性分布

Fig.7　Distribution characteristics of sample maps

根据表 3 实验数据，图 7 中红色框内为 REA* 算法速度占优的地图子集，其余为单边 REA* 算法占优的地图子集．可以看出，原始 REA* 算法仅在低复杂度且不开阔地图上（图 7 左下角）有优势，而当地图复杂度高或开阔度高时，单边 REA* 算法搜索效率明显高于原始 REA* 算法．显然，这与理论分析的结果相吻合．

具体而言，在低复杂度且不开阔地图上（例如，DAO、Room8、Room16、Maze1），单边 REA* 算法的主要影响效果是增加了冗余受迫节点，因此算法速度低于原始 REA* 算法．当地图开阔度增大时（例如从 Maze1 到 Maze32），单边 REA* 算法增加冗余受迫节点的程度减弱，而对矩形单元处理效率的提高效果增强并成为主导因素，因此算法效率逐渐高于原始 REA* 算法；或者当地图复杂度增大时（例如从 DAO 到 BG2，地图集斜边数量增多），单边 REA* 算法规整矩形单元的效果变得明显，搜索节点总数降低，因此算法效率高于原始 REA* 算法．

需要注意的是，无论原始 REA* 算法还是单边 REA* 算法，最高的搜索效率总是发生在低复杂且高开阔的地图（Room64 和 Maze32），此时单边 REA* 算法的时间加速系数最大值达到了 26.48 和 28.55，超过了 REA* 算法的 23.22 和 26.27．图 8 给出了 Room64 和 Maze32 中，2 种算法相对于 A* 算法的时间加速系数随路径长度的变化曲线，可以看出，单边 REA* 算法的效率上限更高．

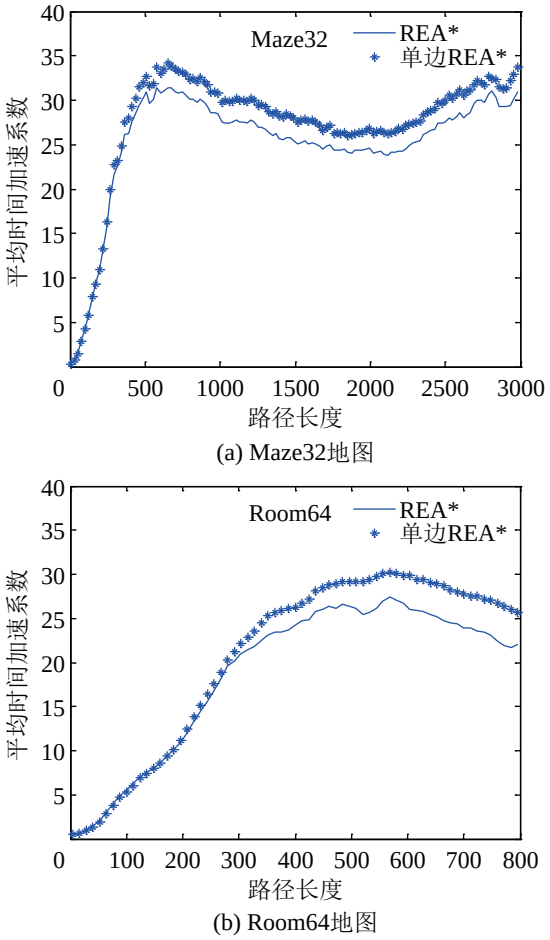


图 8　低复杂、高开阔地图上 2 种算法的平均时间加速系数

Fig.8　Average speed-up of two algorithms in low complex and highly open maps

7　结论（Conclusion）

本文提出了一种新的栅格路径搜索算法——单边 REA*算法．单边 REA* 算法继承了 REA* 算法的优点，能够极大地提高 A* 算法的在线搜索效率，并提高路径质量．理论分析和实验结果表明，相比于 REA* 算法，新算法不仅路径长度更短，路径转折次数更少，且在大部分地图上的搜索速度均高于 REA* 算法，具有更高的平均效率和效率上限，因此综合性能具有明显优势．

下一步工作包括设计更高效的地图搜索策略，

以进一步提高算法效率，特别是提高对低复杂度不开阔地图的处理能力.

参考文献（References）

[1] Hart P E, Nilsson N J, Raphael B. A formal basis for the heuristic determination of minimum cost paths[J]. IEEE Transactions on Systems Science and Cybernetics, 1968, 4(2): 100-107.

[2] Harabor D, Botea A, Kilby P. Path symmetries in undirected uniform-cost grids[C]//Proceedings of the 9th Symposium on Abstraction, Reformulation, and Approximation. Menlo Park, USA: AAAI Press, 2011: 58-61.

[3] Harabor D, Botea A. Breaking path symmetries on 4-connected grid maps[C]//Proceedings of the 6th Artificial Intelligence and Interactive Digital Entertainment Conference. Menlo Park, USA: AAAI Press, 2010: 33-38.

[4] Harabor D. Graph pruning and symmetry breaking on grid maps [C]//Proceedings of the 22nd International Joint Conference on Artificial Intelligence. Menlo Park, USA: AAAI Press, 2011: 2816-2817.

[5] Yap P, Burch N, Holte R, et al. Block A*: Database-driven search with applications in any-angle path-planning[C]//Proceedings of the 25th AAAI Conference on Artificial Intelligence. Menlo Park, USA: AAAI Press, 2011: 120-125.

[6] Harabor D, Grastien A. Online graph pruning for pathfinding on grid maps[C]//Proceedings of the 25th AAAI Conference on Artificial Intelligence. Menlo Park, USA: AAAI Press, 2011: 1114-1119.

[7] Harabor D, Grastien A. Improving jump point search[C]//Proceedings of the 24th International Conference on Automated Planning and Scheduling. Menlo Park, USA: AAAI Press, 2014: 128-135.

[8] Uras T, Koenig S, Hernández C. Subgoal graphs for optimal pathfinding in eight-neighbor grids[C]//Proceedings of the 23rd International Conference on Automated Planning and Scheduling. Menlo Park, USA: AAAI Press, 2013: 24-32.

[9] Botea A. Ultra-fast optimal pathfinding without runtime search [C]//Proceedings of the 7th AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment. Menlo Park, USA: AAAI Press, 2011: 122-127.

[10] Botea A. Fast, optimal pathfinding with compressed path databases[C]//Proceedings of the 5th Annual Symposium on Combinatorial Search. Menlo Park, USA: AAAI Press, 2012: 204-205.

[11] Botea A, Harabor D. Path planning with compressed all-pairs shortest paths data[C]//Proceedings of the 23rd International Conference on Automated Planning and Scheduling. Menlo Park, USA: AAAI Press, 2013: 293-297.

[12] Strasser B, Harabor D, Botea A. Fast first-move queries through run-length encoding[C]//Proceedings of the 7th Annual Symposium on Combinatorial Search. Menlo Park, USA: AAAI Press, 2014: 157-165.

[13] Nash A, Koenig S. Any-angle path planning[J]. AI Magazine, 2013, 34(4): 85-107.

[14] Daniel K, Nash A, Koenig S, et al. Theta*: Any-angle path planning on grids[J]. Journal of Artificial Intelligence Research, 2010, 39(1): 533-579.

[15] Nash A, Koenig S, Tovey C. Lazy theta*: Any-angle path planning and path length analysis in 3D[C]//Proceedings of the 3rd Annual Symposium on Combinatorial Search. Menlo Park, USA: AAAI Press, 2010: 153-154.

[16] Thorpe C E, Matthies L H. Path relaxation: Path planning for a mobile robot[C]//Proceedings of the 4th National Conference on Artificial Intelligence. Menlo Park, USA: AAAI Press, 1983: 318-321.

[17] Zhang A, Li C, Bi W. Rectangle expansion A* pathfinding for grid maps[J]. Chinese Journal of Aeronautics, 2016, 29(5): 1385-1396.

[18] Sturtevant N R. Benchmarks for grid-based pathfinding[J]. IEEE Transactions on Computational Intelligence and AI in Games, 2012, 4(2): 144-148.

[19] Sturtevant N R. The grid-based path planning competition[J]. AI Magazine, 2014, 35(3): 66-69.

作者简介：

李 冲（1989 –），男，博士生．研究领域：无人机自主控制，人工智能，任务规划.

张 安（1962 –），男，博士，教授．研究领域：航空武器火力控制技术，复杂系统建模与仿真，智能指挥与控制工程.

毕文豪（1986 –），男，博士生．研究领域：航空火力控制技术，先进航空电子集成技术.