对网格地图上的路径连接的在线图形剪枝

丹尼尔·哈拉伯尔和阿尔班·格拉斯汀
NICTA和澳大利亚国立大学
电子邮件：无线网络名称。lastname@ nicta .com .au

摘要

寻路成本不一致的网格环境是在机器人技术和视频游戏等应用领域中常见的一个问题。人工状态由计算机路径查找算法主导，这些算法快速，内存开销小，但通常返回次优路径。本文提出了一种针对网格的新搜索策略，该策略快速、最优且不需要内存开销。我们的算法可以描述一个宏算子，它可以选择性地扩展网格映射中的某些节点。连接两个跳转点的路径上的中间节点不会被扩展。我们证明了该方法总是计算最优溶液，然后进行了彻底的实证分析，并将我们的方法与相关的工作进行了比较。我们发现，使用跳跃点进行搜索可以使A*加快一个数量级和更多，并报告对当前艺术状态的显著改进。

介绍

广泛应用于机器人技术（Lee和Yu2009）、人工智能（王和博2009）和电子游戏(戴维斯2000；（2007），无向统一成本网格图是一种非常流行的表示路径环境的方法。在本质上，该领域通常具有高度的路径对称性（港口和Botea，2010；Pochter等人，2010）。在这种情况下，对称表现为路径共享相同的起点（或路径段），具有相同的终点，相同的长度，除了移动发生的顺序是相同的。除非处理得当，系统测量可以强制研究算法来评估曼耶基瓦贷款的状态，并阻止实现目标的真正进展。

在本文中，我们通过设计一个宏算子来处理这种路径对称性，该节点只包含跳跃点。从一个跳点移动到下一个跳点需要在一个固定的方向上移动，同时重复应用一组简单的修剪规则，直到达到一个死胡同或跳跃点。因为我们不扩展跳转点之间的任何中间节点，所以我们的策略可以对搜索性能产生显著的正确影响。此外，计算

解决方案保证是最佳的。跳点修剪速度快，不需要预处理，并引入内存开销。它也在很大程度上与许多适用于网格地图的现有加速技术相正交。

我们做出了以下贡献：(i)跳跃点算法的详细脚本；（ii）理论上的反驳，表明使用跳跃点搜索具有最优性；（iii）比较我们的方法与两种最先进的搜索空间缩减算法的经验分析。我们从文献中进行了一系列合成的和真实世界的基准测试的实验，发现跳跃点将标准化a*的搜索时间性能提高了一个数量级以上。我们还报告了对沼泽的重大改进（Pochter et al. 2010），一种最近的最优保存修剪技术，以及与许多案例中竞争的性能

HPA*（Botea，M¨勒，谢弗2004）；一个井已知的次优寻径算法。

相关工作

识别和消除搜索空间对称的方法已经被提出，包括规划（Fox和Long 1999）、约束规划（Gentand Smith 2000）和组合优化)（不丹2008）。然而，很少有工作明确地识别和处理寻径领域的对称性，如网格地图。

空的矩形房间（Harabor和Botea2010）是一种离线的对称破坏技术，试图纠正这种监督。它将网格映射分解为无障碍矩形，并用一组促进最优旅行的宏观边替换每个矩形内部的所有节点。针对4连接映射，这种方法比跳跃点剪枝更不一般。它还可以重新查询离线的预处理，而我们的方法是在线的。终端启发式（Bj rnsson和Halld rsson 2006） ö ó

和沼泽（Pochter et al. 2010）是与我们的工作相关的两种类似的修剪技术。两者都将网格映射分解为一系列邻接项。之后，这个十二个集合被用于识别与一个特定的寻径实例无关的区域。这个目标与我们的工作相似，我们希望减少探索任何给定的搜索空间所需的努力。

修剪搜索空间的另一种方法是识别死亡的和冗余的细胞(斯特特凡t，布利特科，和

Bj rnsson2010).该方法是在基于学习的启发式搜索环境下开发起来的，只有在运行多次迭代深化算法后才能加快搜索速度。此外，冗余单元的识别会了跳点没有的额外内存开销。ö

*快速扩展（Sun et al .2009）是另一个加速最佳A*搜索的相关工作。当它找到一个成功的节点与打开列表中最好的节点一样好（或最好）时，它避免了不必要的打开列表操作。Ju mp点是一个相似但从根本上的区别：它们允许我们识别大量的节点，这些节点可以被快速扩展，但可以完全跳过。*

在不需要最优性的情况下，层次结构的寻径方法是普遍存在的。他们通过将通常离线的搜索空间分解成一个更小的近似值来改进其他方法。这种类型的算法，如HPA*（Botea，¨米勒，和谢弗2004），是快速和内存效率，但也次优。

符号和术语

我们使用无定向统一成本的grid地图。每个节点都有≤8个邻居，i可以遍历或不遍历。每一个直线（即水平或垂直）移动，从一个可穿越的节点到它的一个邻居，其代价为1；二角形移动的代价为≈√2。不允许涉及不可遍历（障碍）节点的移动。–八个可移动方向（上、下、左、右等）。我们写y=，当一个对角线移动时，我们将在45度处的两个直线移动表示为d→1和d→2。$\vec{d}$ refers to one $x+k\vec{d}$ when node y can be reached k unit moves from node x in direction $\vec{d}$ $\vec{d}$ is $\vec{d}$

一个pathπ= (n 0，n 1，……，nk>是一个从节点n0开始，从nk开始结束的无循环轨道行走。我们有时会在apath的c上文本中使用这个最小运算符：对于exampleπ\x。这意味着减去的节点x不会出现在路径上（即没有被提到）。我们还将使用函数来引用路径的长度（或cost），函数来引用网格上两个节点之间的距离：例如len(π)或dist（n 0，n k）。

跳点

在本节中，我们将介绍一种搜索策略，通过选择性地只扩展网格地图上的某些节点来加速最优搜索。我们在图1(a).中给出了一个充分的基本思想

这里的搜索是扩展一个节点x，它的p（x）；从p（x）到x的移动方向是一个直接的移动。当扩展时，我们可能会注意到，评估任何突出的灰色，因为这种移动引起的路径总是由（即不比）另一个路径，即p（x），而不是x。我们将在下一节中进一步强调这个想法，但现在观察xl的主要邻居是不合适的。而不是像经典的a*算法那样生成这个邻居并将其添加到开放列表中，

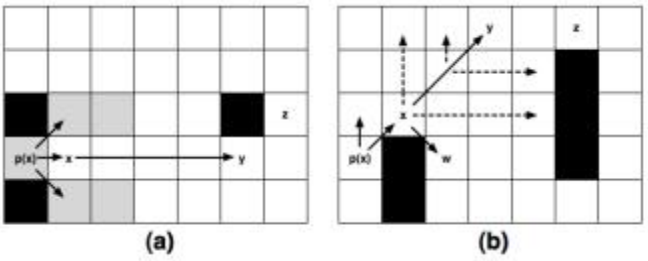

图1：直线(a)和对角线(b)跳跃点的例子。虚线表示到达死角的临时节点评估序列。强线表示最终的后继节点。

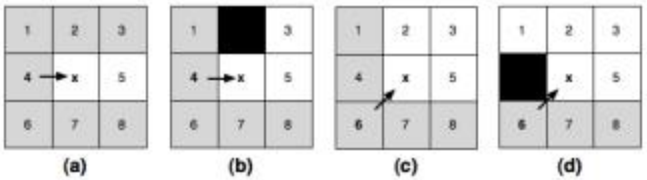

图2：我们展示了几种情况，从它透明的p（x）通过直线或对角线移动到达nodex。当x被展开时，我们可以从考虑中得到标记为灰色的所有节点。

我们建议简单地向右走，并向这个方向继续移动，直到我们遇到一个像y这样的节点；它至少有一个非支配的邻居（这里是z）。如果我们找到了一个节点，如y，即（跳跃点），我们将它作为x的后继节点，并分配它一个g(y)=g(x)+（dist，x，y）的g值（或目前的成本）。或者，如果我们遇到障碍，就认为在这个方向进一步搜索是徒劳的，不会产生任何结果。

在本节的其余部分，我们将开发逐步算子，通过识别直线和对角移动情况下的跳点后继来加速节点扩展。首先，需要定义一系列修剪规则来确定应该生成还是跳过节点。这将使我们能够精确地定位一个跳跃点，并给出一个关于跳跃点测量算法的详细描述。然后，我们证明了“跳转”节点的过程，如xin图1(a)，对搜索的最优性没有影响。

邻居修剪规则

在本节中，我们开发了一些规则，用于修剪网格中紧邻某些节点x的节点的设置。射影是从每一组这样的邻居中识别出，即邻居（x），任何不需要评估以达到最佳目标的节点n。我们通过比较两条路径的长度来实现这一点：π，它以节点p（x）访问x开始，以n结束，以及另一个pathπ，它也从节点p（x）开始，以n结束，但不包括男性x。此外，eitherπ orπ提到的每个节点都必须属于邻居（x）。

有两种情况需要考虑，这取决于从父p（x）到x的转换是否涉及直接移动

或对角线移动。注意，如果x是初始节点，则p（x）为null，并且没有修剪。

直线移动：我们修剪任何满足以下优势约束的节点n个∈邻居（∈）：

≤（p(x)，...）≤（≤(x)）(1)图2(a)显示了一个例子。这里是p（x）= 4，我们删除了除n= 5之外的所有邻居。

对角线移动：这种情况类似于我们制定的修剪规则；唯一的区别是，排除x的路径必须是严格主导的：

<（p(x)，...）<（p(x)，<）(2)图2(c)显示了一个例子。这里是p（x）= 6，我们删除了除n= 2，n= 3和n= 5之外的所有邻居。

假设邻居（x）不包含绝对循环，我们将把应用直线或对角线修剪（不适当）后保留的节点称为x的自然边界。这些对应于图2(a)和2(c).中的灰色节点当邻居（x）包含一个障碍循环时，我们可能无法修剪所有非自然的邻居。如果发生这种情况，我们就说，每一次这样的交易所的评估都是被迫的。

Defi nition 1.节点n∈邻居（x）：

1. 我不是x的自然邻居

2. len((p(x), x, n〉)< len((p(x), ... , n〉\x)

在图2(b)中，我们展示了一个直线移动的例子，其中n= 3是强制的。图2(d)显示了一个涉及对角线移动的模拟例子；这里强制对n=1的计算。

算法描述

我们从精确地提出跳跃点的概念开始。定义2。节点y是来自节点x的跳转点，即标题 *in direction* $\vec{d}$, *if*y使值k最小化，使y=和以下条件之一成立：$x+k\vec{d}$

1. 节点也是目标节点。

2. 节点y至少有一个邻居，其邻居根据定义1强制进行评估。

3. z= y+ k ii，由条件1或条件2从y跳点。

$\vec{d}$ *is a diagonal move and there exists a node* $\vec{d}$ *steps in direction* $\vec{d_i} \in \{\vec{d_1}, \vec{d_2}\}$ *such*

图1(b)显示了一个通过条件3进行识别的跳转点的示例。这里我们从x开始，沿对角线移动，直到遇到节点y。从y开始，节点z

算法1识别继任者

要求：x：当前节点，s：开始，g：目标

1: 继任者(x)←∅
2: 邻居（x）←prune（x，邻居(x)）3：对于所有n个∈的邻居（x）都这样做
4:n←跳跃（x、方向（x、n）、s、g）
5: 加n(x)
6: 返回继任者（x）

可以通过k=2移动到达。因此，z是y的一个跳跃点成功（通过条件2），这在旋转中识别y是x的跳跃点后继者

在算法1中给出了识别个体跳跃点成功与否的过程。我们从紧邻该区域出柤节点x（第2行）的修剪过的邻居集开始。然后，我们没有将每个邻域n添加到x的后续集合中，而是尝试"跳转"到一个更远的节点，但该节点与x的相对关系相同（第3：5行）。例如，如果边（x，n）构成从x右移动的直线移动，我们在紧邻x右移动的节点之间寻找一个跳跃点。如果我们找到这个节点，我们就用n而不是n。如果我们知道跳点，我们什么也不加。这个过程继续进行，直到邻居的集合耗尽，我们返回x的后继集合（第6行）。

为了识别单个跳跃点后继，我们将应用算法2。它需要一个初始节点x，一个direction of travel $\vec{d}$, and the identities of the start node s and the目标节点g。在粗略的透视中，算法试图通过步骤（第1行）确定是否有跳点继，并测试该位置的节点n是否满足定义2。在这种情况下，n被指定为一个跳转点并返回（第5、7和11行）。当没有跳转点时，算法递归和步骤ping in the direction $\vec{d}$
again in direction $\vec{d}$ but this time n is the new initial node (line12).当遇到障碍且无法采取进一步步骤时，递归终止（第3行）。请注意，在每一个对角线步骤之前，算法必须无法检测到任何直线跳跃点（第9：11行）。这个检查对应于定义2的第三个条件和保持最优性的必要条件。

最优性

在本节中，我们证明了对于网格映射中的每一条最优初始路径都存在一个等效长度的路径，该路径只要在搜索过程中只扩展跳点节点就可以找到（定理1）。我们的结果是通过为每一个最优路径确定一个对称的选择，我们将其分割成连续的段。然后我们证明了沿着这个路径的每个转转点也是一个跳跃点。

算法2函数跳跃

要求：1：

n←x: initial node, $\vec{d}$: direction, s: start, g: goal
$step(x, \vec{d})$
2: 如果有障碍或出现在网格之外，那么
3: 如果n= gthen 5
：返回n
6:如果∃n，∈邻居（n）s.t。n，是被强制使用的，然后是7：返回n
8: 如果是9：为所有的我∈{1,2}做$\vec{d}$ is diagonal
10: 如果isg)不是null$jump(n, \vec{d}$
11: 返回n
12: 返回 $jump(n, \vec{d}, s, g)$

定义3。转折点是任意一个节点ni，从n i- 1到ni的移动方向与ni到随后的节点ni+1的移动方向不同。

图3描述了我们在操作时间路径上可能遇到的三种可能的转折点。节点nk的非对角到对角转折点（图3(a)）涉及到其父n k-1的对角步骤，然后是第二个对角步骤，这次是不同的方向，从nk到它的后继者nk+1。类似地，一个直线到对角线（或对角线到直线）的转折点包括一个从n k-1到nk的直线（直）步骤，然后是一个直径（直）步骤到达其后继的nk+1（分别为图3(b)和3(c)）。其他类型的转折点，如直接到直接，通常是次优的，这里没有考虑（它们被我们之前开发的规则修剪；参见图2）。

我们现在准备发展跳跃和出现在某些最优长度对称路径上的转折点之间的等价关系，我们称之为对角线rst。

Defi nition4.如果一个pathπ不包含直接到对角线的转折点<nk-1，nk，nk+1>，则可以用从对角线到s的直接转折点来代替

〈 n k- 1 , n , n k+1〉 s.t.长度ofπ保持不变。

给定任意最优长度pathπ，我们可以通过应用Algorithm 3 toπ，推导出对称对角线第一路径π。请注意，这只是作为一个概念性的装置。引理1。沿着最佳对角线-第一路径π的每个转弯点，也是一个跳跃点。

证明。设nk是一个任意的转折点节点alongπ，。我们将考虑三种情况，每种情况都对应于图3中三种可能的最优l转向点中的一种。

对角线到对角线：Sinceπ，i是最优的，必须有一个障碍nk和n k-1，以绕道。我们知道这一点，因为如果没有问题，我们会有<（n-1，+1）<（n-1，nk）+（nk，+1），这与π是最优的事实相矛盾。我们得出nk+1是nk的强制邻域。这个问题满足定义1的第二个条件，使nk跳跃点。

算法3计算对角线第一路径

要求：π：一个任意的最优长度路径

1: 选择一对出现alongπ的相邻边，其中（n k- 1，n k）是直线移动，（n k，nk+1）是直线移动。

2: 替换（nn-1nk）和（nknk+1）：（n k- 1，n），这是一个对角线移动和（n，nk+1），这是一个直接的举动。该操作已成功

如果（n k- 1、n）和（n，n k+1）都是有效的移动；即，节点不是障碍。

3: 重复第1行和第2行，选择和替换相邻的行边缘，直到没有进一步的更改可以使toπ。

4: returnπ

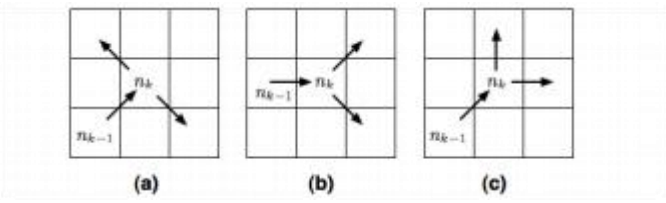

图3：最佳转折点的类型。(a)对角对角(b)直对角(c)对角对直线。

直到对角线：在这种情况下，必须有与nk相邻的支架。如果这不是真的，nk可以通过一个对角线到直线的旋转点来代替，这与thatπ相反，是对角线的。Sinceπ，保证是对角优先的，我们推导出nk+1是nk的强制邻域。这满足定义1的第二个条件，我们得出nk是一个跳跃点。

对角线到直线：在这种情况下有两种可能性，这取决于目标是否可以通过从n kor开始的一系列直线步骤到达，而π是否有额外的转折点。如果目标可以通过直接到达，那么nk有一个满足定义1的第三个条件的跳跃点后继，我们得出nk也是一个跳跃点。如果nk后面跟着另一个转折点nl，那么那个转弯点必须是直到对角线的，根据这种情况的论证，也是跳跃点。我们再次认为nk有一个满足定义1的第三个条件的跳点后继。因此，nk也是一个跳跃点。

定理1。搜索与ju mp点修剪总是返回一个最优的解决方案。

证明。设π为网格andπ上两个节点之间任意选择的最优路径，这是由算法3 toπ导出的对角对称等分。我们将展示每个转折点提到的byπ，在搜索跳跃点剪枝时是最优。我们认为如下：

Divideπ，分解成一系列相邻的节段s.t。π, = π 0, + π+... + π n ,.Eachπi, =<n0, n1, ..., n k- 1, n是一个子路径，所有的移动都涉及在同一个direction（e.g.only“上”或“下”等）。注意，除了开始和目标，一段开始和结束的每一个点头都是一个转折点。

因为eachπ i，只由移动一个指令（直对角线）我们可以使用算法2从n0∈π我，，节点的开始nk∈π我，节点结束，不需要愉快地停止扩展n之间的每个节点。中间展开可能会发生，但我们从n0最优达到nk是保证的。只需要证明n0和n都可以识别为跳点，因此必须存在。每个转折点alongπ，也是一个j个ump点，因此每个转折点节点都必须进行扩展的重复搜索。只剩下开始和目标。开始节点必须在每次搜索开始时展开，而目标节点是定义中的跳转点。因此两者都是

膨胀的 □

实验设置

我们评估了从免费的寻径库层次结构的开放图（猪，http://www.googlecode.com/p/hog2）中提取的跳跃点修剪的性能：

自适应深度是一组12张地图，大小为100×100英寸 which approximately $\frac{1}{3}$ of each map is divided into rect- 不同大小的角度房间和一个大的开放区域，中间有大的随机放置的障碍物。对于这个基准测试，我们为总共1200个实例的每个映射随机生成了100个有效问题。

《巴尔杜之门》是一套120张地图，取自生物软件流行的角色扮演游戏《阿恩的阴影》；它经常作为文献中的标准基准出现（2006；哈拉博和Botea2010；波克特等人，2010年）。我们使用由内森·斯特特凡特造成的变异吨，所有的地图都被缩放到512×512，以更准确地表示现代寻径环境。地图和所有实例都可以从http://movingai.com中获得
ö ó

龙的时代是另一个现实的基准；这次是取自bioware最近的角色扮演游戏《龙的时代：起源》。它由156张地图组成，大小范围从30×21到1104×1260。对于这个基准测试，我们使用了大量随机生成的实例，同样是由于内森·斯特特凡特和romhttp:// movingai .com。

房间是一套300张大小为256×256的地图，它们被分成对称的小矩形图（7×7），由随机放置的入口连接。这个基准测试之前已经出现在（Pochter et al. 2010）中。对于这个基准测试，我们随机生成了100个有效的问题，每个映射总共有30000个实例。

我们的测试机器是一台2.93 GHz的英特尔酷睿2双处理器，4 GB内存，运行OSX 10.6.4。

结果

为了评估跳点，我们使用了a*的通用实现，我们调整了它来促进在线邻居的预输入和跳点识别。我们讨论了加速领域的性能：即在解决给定问题的时间上，相对的改进，以及不扩展des的数量。使用这个度量的搜索时间加速2.0是速度，而节点扩展加速的两倍
2.0表明节点的数量被扩展了。在每种情况下，越高越好。图4显示了我们四个基准测试中的平均搜索时间加速。表1显示了平均节点扩展加速；每列的最佳结果是粗体。

与沼泽的比较：我们首先比较跳跃点和沼泽（Pochter et al. 2010）：一种加速路径的最优保持修剪技术。我们使用了作者的源代码和他们对A*的组合，并使用相应的运行参数运行了所有实验：沼泽种子半径为6，“无变化限制”为2。

	A.深	B.门	D. 年龄	房间
跳点	**20.37**	**215.36**	**35.95**	**13.41**
沼泽	1.89	2.44	2.99	4.70
人权事务部*	4.14	9.37	9.63	5.11

表1：平均A*节点扩展速度。

如图4所示，跳点修剪显示了对所有基准标记的平均搜索时间的改进改进。最大的差异是在巴尔杜尔门和龙时代，用单p点搜索达到目标快25-30倍，而用沼泽搜索只加速3-5倍。当看表1时，也观察到类似的趋势，其中对扩展的节点总数的证明甚至更明显。

基于这些结果，我们得出结论，虽然沼泽对于识别与达到目标无关的地图区域是有效的，但那些仍然存在的区域仍然需要大量的努力来搜索。跳跃点使用一个更强的正交策略来证明许多被沼泽扩展的节点可以被忽略。由于这两种想法看起来很复杂，所以我们假设它可以很容易地组合起来：首先，应用基于沼泽的分解来修剪当前搜索以外的区域。然后，使用跳转点来搜索到地图上的剩余部分。

与HPA*的比较：接下来，我们比较跳跃点修剪到hpa*算法(¨，¨，和
谢弗出版社，2004年)。HPA*虽然次优，但速度非常快，广泛应用于电子游戏。为了评估HPA*，我们测量了插入和层次搜索的总成本。我们没有引用任何抽象路径，而是假设有一个预计算的集群路径的数据库可用。这种配置代表了HPA*最快的通用组合，但需要额外的记忆头。在搜索时，我们使用了分类级抽象层次结构，集群大小为10。这些设置是由原始作者推荐的，他们注意到更大的集群和更多的级别往往没有什么好处。

如图4所示，跳跃点与HPA*具有很高的竞争力。在自适应技术，巴尔杜尔的门和房间，跳跃点有一个明显的优势——提高HPA*搜索时间的几个因素。在龙的年龄基准上，跳点对于长度<500的问题有一个小的优势，但对于更长的实例，两种算法之间很少。Ta- ble1提供了进一步的见解：尽管使用跳点搜索比HPA*少，但每次这样的操作需要更长的时间。

我们的结论是，跳点修剪是HPA*的竞争性替代品，对于各种各样的问题，可以帮助解决方案。HPA*仍然是有利的，特别是如果内存开销是可接受的，最优性不重要，但只有如果将起始节点和目标节点插入抽象图的成本（如果路径数据库不可细化），可以在找到抽象路径所需的时间内有效地摊销。进一步工作的一个方向可能是使用

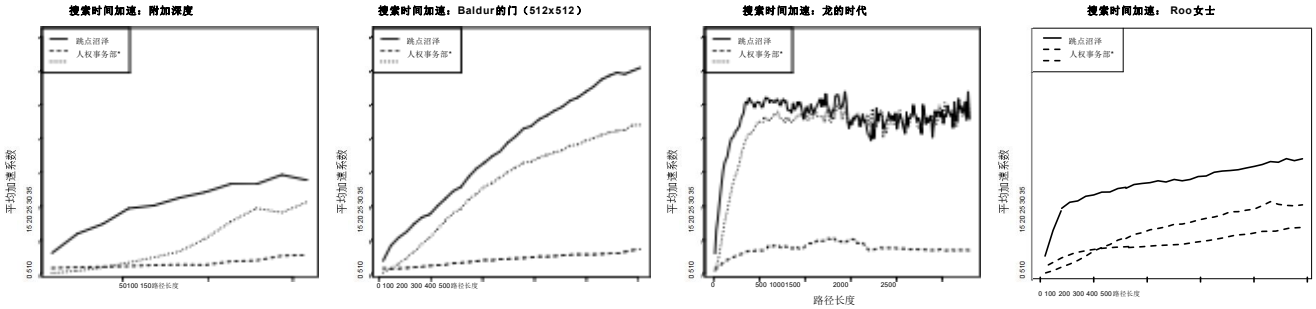

图4：在我们的四个基准测试中，平均A*搜索时间的加速。

例如，在插入和细化期间。

结论

本文介绍了一种新的在线节点修剪策略，用于加快无向均匀代价网格映射的速度查找。我们的算法只识别并选择性地扩展了网格映射中的某些节点。在跳跃点之间的移动只包括直线或对角线的直线运动。我们认为，在两个跳点之间的路径上的中间节点不需要扩展，“跳”它们不会影响搜索的最优性。

我们的方法在寻路文献中是独一无二的，因为缺点很少：非常简单，非常有效；它保持了最优性，但不需要外用；但速度非常快，但不需要预处理。此外，炎在很大程度上正交和容易与竞争加速技术相结合。我们不知道还有任何其他的算法具有所有这些特性。

该新算法与现有的新算法具有很强的竞争力。与沼泽（Pochter et al. 2010），最近最先进的操作时间保存修剪技术相比，我们发现跳跃点速度快一个数量级。我们还表明，跳点可以与HPA*竞争，而且许多实例明显比HPA*快；一种流行的次最优寻径技术通常使用内部敏感的应用程序，如电子游戏。

进一步工作的一个有趣的方向是将端跳转点转移到其他类型的网格，如六边形或文本（Yap2002）。我们建议通过开发一系列类似于平方网格的剪枝规则来实现这一点。由于这些dom ains上的分支因子小于正方形网格，我们假设跳跃点可能比本文中观察到的更有效。另一个有趣的方向是将跳跃点与其他加速技术相结合：例如沼泽或HPA*。

致谢

我们要感谢阿迪·博蒂亚、菲利普·基尔布和帕里克哈斯勒姆在本项目开发过程中给予的鼓励、支持和许多有益的建议。我们也

感谢尼尔·波克特为我们提供了沼泽事件的源代码。

NICTA由澳大利亚政府资助，并由宽带、通信部、数字经济部和澳大利亚研究中心通过ICT卓越中心计划进行资助。

参考文献

，和，2006。改进了游戏地图上的最优路径识别算法。在AIIDE，9-14岁。波¨；和谢弗，J。2004.接近最优的分层路径保护的新定义。J.第1场(1)：7-28。öó

戴维斯，我的律师事务所，2000年。扭曲速度：星际迷航无敌舰队的路径计划。在AAAI春季研讨会（AIIDE）上，18-21日。

福克斯，m.，和朗，D. 1999年。规划问题中对称性的检测与开发。在IJCAI，956-961。

国中山，A. S.2008。在多个解决器中整合对称、优势和束缚。在CPAIOR，82-96。

金特，ip和史密斯，2000。对称性打破了约束编程。在ECAI中，599-603。

Harabor, D., 和Botea， A.2010.破坏4个连接的网格图中的路径对称性。在AIIDE，33-38。

李，j.y。和于，W. 2009。移动机器人快速路径识别的粗糙方法。在IROS，5414-5419。

波克特尔，n。；佐哈尔王朝，A.；罗森谢恩，J.S.；和费尔纳， A. 2010.使用沼泽层次结构搜索空间。在AAAI。

斯特凡特；布利特科，v；Bj rnsson，Y。2010.关于在以代理为中心的搜索中的学习。在AAMAS，333-340。ö

斯特特万特，2007年。路径的记忆抽象。在AIIDE，31-36。

孙x；杨，w；陈p。；和凯尼格，S.2009。基于*的搜索的模拟优化技术。在AAMAS
(2), 931–936.

王K。 C., 和波在网格地图上的可处理的多智能体路径规划。在IJCAI，1870-1875年。

Yap ,P. 2002.基于网格的路径识别。在加拿大人工智能中，44-55。