DOI:10.13413/j.cnki.jdxblxb.2009.04.039
第47卷 第4期 吉林大学学报(理学版) Vol.47 No.4
2009年7月 Journal of Jilin University (Science Edition) July 2009

一种多目标增量启发式搜索算法

魏 唯[1,2], 欧阳丹彤[1,2], 吕 帅[1,2], 殷明浩[3]
(1. 吉林大学 计算机科学与技术学院, 长春 130012;
2. 吉林大学 符号计算与知识工程教育部重点实验室, 长春 130012;
3. 东北师范大学 计算机学院, 长春 130024)

摘要: 提出一种多目标增量启发式搜索算法, 该算法结合启发式搜索与增量搜索的思想, 当多目标问题搜索图的状态格局发生改变时, 该算法并不是对变化后的问题进行完全重新求解, 而是部分利用了先前搜索保留的信息求解新问题的最优解集, 从而提高了问题求解的效率. 通过Gridworld标准测试问题上的实验测试, 验证了算法的效率.

关键词: 启发式搜索; 增量搜索; 多目标问题; 最优解集
中图分类号: TP18 文献标识码: A 文章编号: 1671-5489(2009)04-0752-07

A Multiobjective Incremental Heuristic Search Algorithm

WEI Wei[1,2], OUYANG Dan-tong[1,2], LÜ Shuai[1,2], YIN Ming-hao[3]
(1. College of Computer Science and Tedhnology, Jilin University, Changchun 130012, China; 2. Key Laboratory of Symbolic Computation and Knowledge Engineering of Ministry of Education, Jilin University, Changchun 130012, China; 3. School of Computer Science, Northeast Normal University, Changchun 130024, China)

Abstract: A multiobjective incremental heuristic search algorithm which combines heuristic search with incremental search is put forward. When the state space of the multiobjective problem changes, the algorithm will not resolve the new problem from scratch, but reuse the parts of the information of the previous search to find the set of optimal solutions of the new problem and thus the efficiency of resolution is improved. The experiment results of the Gridworld benchmark problem show that the algorithm can solve a series of similar multiobjective problems very efficiently when the state space changes continuously.

Key words: heuristic search; incremental search; multiobjective problems; set of optimal solutions

启发式搜索方法在智能规划、诊断、自动推理等人工智能领域应用广泛, 但在许多实际问题中, 真实情况可能与最初设想的不同或者系统格局随时间发生改变, 一般的处理方法是对新的格局重新进行完整的搜索, 得到新的规划. 然而, 当系统格局只产生局部变化时, 对整个格局进行重新搜索会付出较大代价. 增量搜索的基本思想是利用先前搜索的信息提高本次搜索效率[1]. Ramalingam等人[2]提出了用于解决动态SWSF不动点问题的增量算法DynamicSWSF-FP; Koenig等人[3,4]提出的LPA*算法是在DynamicSWSF-FP算法的基础上加入了A*算法的思想, D* Lite算法[5]则是D*和LPA*算法相结

收稿日期: 2008-11-13

作者简介: 魏 唯(1984~), 女, 汉族, 硕士研究生, 从事自动推理与智能规划的研究, E-mail: weiweinenu@yahoo.com.cn; 通讯作者: 欧阳丹彤(1968~), 女, 满族, 博士, 教授, 博士生导师, 从事基于模型的诊断、自动推理与智能规划的研究, E-mail: ouyang@jlu.edu.cn

基金项目: 国家自然科学基金(批准号: 60773097, 60873044, 60803102, 60873148)、教育部博士学科点基金(批准号: 20050183065, 20060183044)、吉林省科技发展计划项目基金(批准号: 20060532, 20080107)和吉林省青年科研基金(批准号: 20080617).

合在机器人导航问题上的应用，RTAA* 算法[6]能够实时的调整启发信息，进而提高搜索效率；Mills-Tettey等人[7]在 D* Lite算法的基础上提出了 DD* Lite算法，通过分析状态之间的支配关系对搜索图进行剪枝进而提高搜索效率. 文献[8-13]研究了动态环境下的增量求解方法，在数据挖掘、机器学习、数据库查询和硬件系统验证等问题上提出了采用增量的求解方法，提高了问题的求解效率和处理能力.

经典的搜索问题通常只考虑单个目标的优化，而单一的目标函数往往很难准确描述现实生活中的问题，例如在运输过程中，希望找寻路程短且路况好的路径，即有多个目标函数需要优化，这就扩展出多目标最短路径问题. Stewart等人[14]提出了 MOA* 算法并证明了其可行性，将 A* 算法扩展到多目标情况；Dasgupta等人[15]将非单调的启发信息用于 MOA*，提出了 MOA** 算法并应用在 VLSI设计问题上；Mandow等人[16]提出的 NAMOA* 算法也是将 A* 算法扩展到多目标的情况，NAMOA* 选择路径进行扩展，而 MOA* 选择节点进行扩展，路径扩展的方法可以避免无用路径的存储以及路径的重复扩展；Harikumar等人[17]提出的 IDMOA* 算法和 Dasgupta等人[15]提出的 MOMA* 0算法分别是将深度优先算法 IDA* 和 RBFS扩展到多目标的情况[15, 17-19].

本文针对已有的多目标搜索算法，提出一种多目标增量启发式搜索算法 MOLPA*，该算法同时考虑了多目标问题与增量搜索方法，对一系列相似的多目标搜索问题，利用增量搜索方法提高重搜索的效率，当多目标问题搜索图的状态格局发生改变时，通过对搜索现场进行实时更新，部分利用先前搜索保留的信息，从更新后的状态开始求解新问题的最优解集，从而提高了重搜索的效率. 本文实现了该算法并通过 Gridworld标准测试问题上的实验验证了该算法的效率.

1　多目标搜索问题的定义

多目标搜索问题有多个目标函数需要优化，边的代价为向量形式，向量的每个元素对应一个目标上的代价值.

定义 1.1[14]　设向量 $f, f' \in R^q$，称 f 支配 f' 当且仅当 $\forall i (f_i \leqslant f'_i)$ 并且 $f \neq f'$，也称 f' 受 f 支配，记作 $f \prec f'$，其中 f_i 表示向量 f 的第 i 个元素.

定义 1.2[14]　对于向量集合 X，称 $nondom(X)$ 为集合 X 中所有不受支配的向量的集合，$nondom(X) = \{x \in X \mid \nexists y \in X,\ y \prec x\}$.

两个向量之间不一定存在支配关系. 例如，给定 3个向量 $x=(1,1)$，$y=(3,2)$，$z=(1,3)$，则 $x \prec y$，$x \prec z$，即 y 和 z 是受 x 支配的，但 y 和 z 之间没有支配关系.

定义 1.3[14]　设 P_{sg} 是顶点 S 与顶点 G 之间所有路径的集合，路径 $p_1, p_2 \in P_{sg}$，称 p_2 受 p_1 支配当且仅当 $g(p_1) \prec g(p_2)$，其中 $g(p_1)$ 与 $g(p_2)$ 分别是路径 p_1 与 p_2 的代价向量.

定义 1.4[14]　设 P_{sg} 是顶点 S 与顶点 G 之间所有路径的集合，路径 $p \in P_{sg}$，称 p 为不受支配路径当且仅当 $\nexists p' \in P_{sg}$，$p' \prec p$，即不存在其他路径在所有目标上都优于该路径.

定义 1.5[14]　多目标搜索问题是一个三元组 $\langle G, s, \Gamma \rangle$，其中 $G=(N, A, c)$ 为有向图，N 为顶点集合，A 为边的集合，c 为代价函数，$c(n, n') \in R^q$ 表示边 (n, n') 的 q 维代价向量；$s \in N$ 为初始节点；$\Gamma \subseteq N$ 为目标节点集合.

多目标的引入使问题的求解与单目标条件下不同，由于各目标之间通常存在冲突，因此针对某个目标最优解，对于另一个目标可能并不是最优的，从而使多目标最短路径问题一般不存在惟一确定的最优解，而是一个最优解集，即初始节点 s 与目标节点集合 Γ 之间所有不受支配路径的集合，其特点是在改进任何目标函数的同时至少削弱一个其他目标函数.

2　多目标增量搜索算法

在多目标搜索过程中，当系统的状态格局发生改变时，如果完全重新搜索，则效率会非常低. 增量搜索是一种利用先前搜索保留信息提高本次搜索效率的方法. 下面将扩展多目标搜索算法

NAMOA*，提出一种多目标增量启发式搜索算法 MOLPA*，并对该算法进行分析与证明.

2.1 多目标增量搜索算法 MOLPA*

多目标增量搜索的基本思想是当系统的状态格局发生改变时，将受影响的解路径从最优解集中删除，并对图中受影响节点的路径扩展情况和 OPEN表进行更新，再从更新后的 OPEN表开始新的搜索过程.

算法中节点 n的启发函数 $H(n)$是对节点 n到达每个目标节点的实际代价进行估计，因此，每条从 s到 n的路径 g_p都有一个估值函数的集合：

$$F(n, g_p) = nondom\{f \mid f = g_p + h \wedge h \in H(n)\}^{[16]}.$$

OPEN表存储的是待扩展的路径. 对于搜索图 SG中的每个节点 n及其待扩展的路径 $g \in G_{op}(n)$（$G_{op}(n)$存储到达节点 n并且未被扩展的路径），OPEN表中有一条记录$(n, g, F(n, g))$. 算法每次从 OPEN表中选择一条路径进行扩展. 两个集合 GOALN和 COSTS分别保存当前已经到达的目标节点和已找出的解路径代价. 搜索过程反复执行，直到 OPEN表为空.

对于一个多目标搜索问题$\langle G, s, \Gamma\rangle$，MOLPA*先搜索出从 s到 Γ的最优解集 SOLUTIONS及所有解路径的代价集合 COSTS. 之后，问题的状态格局发生改变，例如，$c(u, v)$发生改变，算法首先从 SOLUTIONS和 COSTS中将经过边(u, v)的解路径及其代价删除，未受影响的解路径仍然保留；然后对 v的可达状态集合中的每个节点 n，从 $G_{op}(n) \cup G_{cl}(n)$（$G_{cl}(n)$存储到达节点 n并且已经被扩展的路径）中删除所有受影响的路径，将 OPEN表中的相关记录删除；再将新到达 v的路径插入 OPEN表；最后，从更新后的 OPEN表开始新的搜索过程，找到新的最优解集.

此外，本文算法不对生成的路径进行过滤，即使某条路径受当前已知的解路径支配，当系统状态格局发生改变时，原来受支配的路径可能不再受支配，也可能会构成新的解路径，因此，生成的所有路径都保存在 OPEN表中. 每次扩展前，将 OPEN表中所有不受支配的路径标记为 ND表，作为路径扩展的候选集合，从 ND表中按某种排序方式选择一条路径进行扩展. 算法的终止条件是 ND表为空，即没有可以扩展的路径.

下面给出多目标增量搜索算法 MOLPA*：

Procedure MOLPAstar( ).

(1) 令 OPEN=$\{(s, g_s, F(s, g_s))\}$，GOALN，COSTS，ND和 SOLUTIONS为空集；

(2) 执行 COMPUTESOLUTIONS( )；

(3) 等待系统格局发生改变；

(4) 对所有发生改变的边(u, v)，令 $c_{old} = c(u, v)$，更新 $c(u, v)$，执行 UPDATE(u, v)；

(5) 转(2).

Procedure COMPUTESOLUTIONS( ).

(1) 将 OPEN表中满足如下条件的记录(x, g_x, F_x)标记为 ND表：

$$\exists f_k \in F_x((\forall(n', g_{n'}, F') \in OPEN(\exists / f' \in F'(f' \prec f_k))) \wedge (\exists / f'' \in COSTS(f'' \prec f_k)));$$

(2) 若 ND为空，则从 SG中追溯解路径集合 SOLUTIONS，返回；

(3) 从 ND中选择一条路径(n, g_n, F)进行扩展，将该项从 OPEN表中删除，将 g_n从 $G_{op}(n)$移入 $G_{cl}(n)$；

(4) 若 $n \in \Gamma$，则将 n加入 GOALN，g_n加入 COSTS并过滤所有受支配的路径代价，转(1)；否则，对 n的所有后继 m，令 $g_m = g_n + c(n, m)$并执行 INSERTPATH(m, g_m)，转(1).

Procedure INSERTPATH(m, g_m).

(1) 计算 $F_m = F(m, g_m)$，将(m, g_m, F_m)加入 OPEN表，将 g_m加入 $G_{op}(m)$；

(2) 若 F_m中至少存在一个代价估值，其不受 COSTS中任意代价值支配，则设置 m为指向其前驱的指针.

Procedure UPDATE(u, v).

(1)从 SOLUTIONS和 COSTS中删除所有经过 (, u, v)的解路径及其代价值;

(2)对 v的可达状态集合中每个节点 m, 从 $G_{op}(m)\cup G_{cl}(m)$中删除所有经过 (, u, v)的路径 g_m, 从 OPEN表中删除 (m, g_m, F_m);

(3)对每个 $g\in\{g_u+c(u,v),\ g_u\in G_{cl}(u)\}$, 执行 INSERTPATH(, v, g).

2.2 MOLPA* 算法分析与证明

首先分析 MOLPA* 算法的复杂性. 算法使用 OPEN表存储待扩展的路径, 若对到达每个节点的所有路径进行排序, 且只将排在最前面的路径存储在 OPEN表中, 即对于每个节点只存储最有可能被扩展的路径, 则 OPEN表的最坏空间复杂度为 O(|N|). 到达节点的其他路径可以使用二元堆存储[16], 二元堆可以在 O(log|N|)时间内进行插入与删除操作. 由于在系统格局发生变化后, MOLPA* 能重用先前搜索的信息, 从而在新的搜索过程中节省了时间, 另外, MOLPA* 的时间复杂度还与启发函数的定义和计算密切相关.

下面证明该算法的可行性.

定理 2.1 给定启发函数 $h(n)\leqslant h^*(n)$, MOLPA* 算法是可行的, 即若搜索问题存在解, 则该算法一定能找到问题的最优解集, 其中 $h^*(n)$表示节点 n到目标节点实际的代价向量.

证明: 文献[16]证明了 NAMOA* 算法的可行性, 则 MOLPA* 算法在第一次利用 NAMOA* 搜索时, 一定能得到从 s到 Γ的最优解集.

下面证明给定启发函数 $h(n)\leqslant h^*(n)$的情况下, 系统的格局发生改变后, 该算法能够找到新的最优解集.

假设 c(u, v)变为 c′(u, v), 则需要对 $G_{op}(v)\cup G_{cl}(v)$进行更新. 若 $\exists P_v=(s,\cdots,u,v)\in G_{op}(v)\cup G_{cl}(v)$, $g(P_v)=c$, 则将 P更新为 $P_v'=(s,\cdots,u,v)$, $g(P_v')=c'$; 若 $P_v=G_{op}(v)$, 则必 $\exists(v,P_v,F(P_v))\in$ OPEN, 将其更新为 $(v,P_v',F(P_v'))$. 对 v的可达状态集合中的每个节点 m, 将所有满足 $P_m=(s,\cdots,u,v,\cdots,m)$的路径从 $G_{op}(m)\cup G_{cl}(m)$中删去, 若 $P_m\in G_{op}(m)$, 则必 $\exists(m,P_m,F(P_m))\in$ OPEN, 将该项从 OPEN表中删去. 此外, 将所有满足 $P^*=(s,\cdots,u,v,\cdots,\gamma)\in$ SOLUTIONS且 $g(P^*)=c^*\in$ COSTS的解路径及其代价删去, 得到 SOLUTIONS′和 CPSTS′. 至此, 对系统格局的更新完毕, 由更新后的 OPEN表开始新的搜索过程.

新的搜索过程能找到最优解集. 假设 P^*为一条不受支配的解路径, $g(P^*)=c^*$, 即 $\nexists c^{*\prime}\in$ COSTS′, $c^{*\prime}<c^*$. 由文献[16]可知, 对每个不受控制的解路径 $P^*=(s,n_1,\cdots,n_i,n_{i+1},\cdots,\gamma)$, $g(P^*)=c^*$, 在其被找出之前总有一条子路径 $P_i^*=(s,n_1,\cdots,n_i)\subseteq P^*$, 满足: (1) P_i^*已包含于搜索图 SG; (2) $g(P_i^*)\in G_{op}(n_i)$; (3) $\exists f\in F(P_i^*)|f\leqslant c^*$. 因此, 对于 $(n_i,P_i^*,F(P_i^*))\in$ OPEN, 若该路径在更新前已存在于 OPEN表中, 则更新前对 $\forall f\in F(P_i^*)$, $\exists c\in$ COSTS, $c<f$; 若在更新过程中 c已从 COSTS中删除, 使得 $\exists f\in F(P_i^*)$, $\nexists c'\in$ COSTS′, $c<f<c^*$, 则 $(n_i,P_i^*,F(P_i^*))$会在某一时刻被扩展; 若该路径是在更新过程中加入 OPEN表的, 则 $\exists f\in F(P_i^*)$, $\nexists c'\in$ COSTS′, $c<f<c^*$, 该路径会在某一时刻被扩展. 当 P_i^*被扩展时, 若 $n_i=\gamma$, 则 $P_i^*=P^*$, 该解路径被找出; 若 $n_i\neq\gamma$, 则 $P_{i+1}^*=(s,n_1,\cdots,n_i,n_{i+1})$被生成, $g(P_{i+1}^*)\in G_{op}(n_{i+1})$, 同理 P_{i+1}^*也会被选择进行扩展.

由此可见, 每条不受支配解路径的任意子路径都会被扩展, 直到找出该解路径. 所以, 该算法能够找到所有不受支配的解路径.

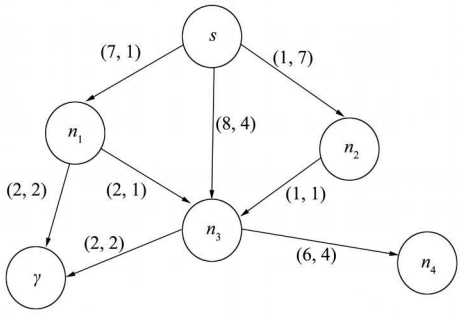
2.3 实例分析

如图 1所示, 本文将扩展文献[16]中的实例, 分析 MOLPA* 算法的执行过程.

首先, MOLPA* 算法调用 COMPUTESOLUTIONS( )搜索出从 s到 Γ的最优解集 SOLUTIONS= $\{(s,n_1,\gamma),(s,n_2,n_3,\gamma)\}$, 代价集合 COSTS= {(9, 3), (4, 10)}. 表 1为第一遍搜索时, 每次迭代过程中 OPEN表中存储的路径.

其次, 等待格局发生变化. 假设 $c(n_2,n_3)$由 (1, 1)变为 (8, 1), 调用 UPDATE(n_2, n_3), 将经过

(n_2, n_3)的解路径(s, n_2, n_3, γ)从 SOLUTIONS中删去，将其代价(4, 10)从 COSTS中删去，n_3的可达状态集合为{n_3, γ, n_4}，从这3个节点的 $G_H \cup G_0$中将经过(n_2, n_3)的路径代价删去. 更新 $c(n_2, n_3)$后，将g_{n_3}的代价更新为(9, 8)，将该代价加入 $G_0(n_3)$，计算 $F(n_3, g_{n_3})$值，并将(n_3, g_{n_3}, $F(n_3, g_{n_3})$)重新加入OPEN表，此时 OPEN表如表2所示.



n	$H(n)$
s	{(3, 3)}
n_1	{(2, 2)}
n_2	{(2, 2)}
n_3	{(1, 1)}
n_4	{(0, 0)}
γ	{(0, 0)}

图 1 实例图和启发函数值

Fig. 1 Sample graph and heuristic function

表 1 每次迭代的 OPEN表

Table 1 OPEN of each iteration

迭代次数	OPEN表
1	(s, (0, 0), {(3, 3)})
2	(n_1, (7, 1), {(9, 3)}), (n_2, (1, 7), {(3, 9)}), (n_3, (8, 4), {(9, 5)})
3	(n_1, (7, 1), {(9, 3)}), (n_3, (8, 4), {(9, 5)}), (n_3, (2, 8), {(3, 9)})
4	(n_1, (7, 1), {(9, 3)}), (n_3, (8, 4), {(9, 5)}), (n_4, (8, 12), {(8, 12)}), (γ, (4, 10), {(4, 10)})
5	(n_1, (7, 1), {(9, 3)}), (n_3, (8, 4), {(9, 5)}), (n_4, (8, 12), {(8, 12)})
6	(n_3, (8, 4), {(9, 5)}), (n_4, (8, 12), {(8, 12)}), (n_3, (9, 2), {(10, 3)}), (γ, (9, 3), {(9, 3)})
7	(n_3, (8, 4), {(9, 5)}), (n_4, (8, 12), {(8, 12)}), (n_3, (9, 2), {(10, 3)})

表 2 $c(n_2, n_3)$发生变化后的 OPEN表

Table 2 OPEN after $c(n_2, n_3)$ changed

迭代次数	OPEN表
7	(n_3, (8, 4), {(9, 5)}), (n_4, (8, 12), {(8, 12)}), (n_3, (9, 2), {(10, 3)}), (n_3, (9, 8), {(10, 9)})

最后，重新计算新的解路径. 经过计算发现，变化后 OPEN表中的路径都受 COSTS支配，无法选出可以扩展的路径，即 ND表为空，算法终止. 发生改变后没有发现新的解路径，SOLUTIONS={(s, n_1, γ)}.

再假设边(n_1, γ)被删除，即 $c(n_1, \gamma)$由(2, 2)变为(∞, ∞)，将解路径(s, n_1, γ)及其代价(9, 3)删去. 此时 OPEN表如表3所示，重新从 OPEN表开始搜索，由于 COSTS中的解路径代价(9, 3)被删去，OPEN表中的记录(n_3, (8, 4), {(9, 5)})和(n_3, (9, 2), {(10, 3)})已经不再受支配. 经过计算，得到两条新的解路径(s, n_1, n_3, γ)和(s, n_3, γ)，代价分别为(11, 4)和(10, 6)，解路径的集合 SOLUTIONS={(s, n_2, n_3, γ), (s, n_1, n_3, γ), (s, n_3, γ)}.

表 3 $c(n_1, \gamma)$发生变化后的 OPEN表

Table 3 OPEN after $c(n_1, \gamma)$ changed

迭代次数	OPEN表
7	(n_3, (8, 4), {(9, 5)}), (n_4, (8, 12), {(8, 12)}), (n_3, (9, 2), {(10, 3)})
8	(n_4, (8, 12), {(8, 12)}), (n_3, (9, 2), {(10, 3)}), (γ, (10, 6), {(10, 6)})
9	(n_4, (8, 12), {(8, 12)}), (n_3, (9, 2), {(10, 3)})
10	(n_4, (8, 12), {(8, 12)}), (γ, (11, 4), {(11, 4)})
11	(n_4, (8, 12), {(8, 12)})

3　实验结果与分析

本文采用 Gridworld问题进行实验测试. 初始位置为 Gridworld左上角的格子，目标位置为右下角的格子；代价向量以二维向量为例，即有两个目标函数需要优化；向量的每个元素值在区间 [1, 10]的整数中随机生成；选择两点间的 Manhattan distance作为每一维的启发信息；选择字典顺序[16]对 ND表进行排列，每次选择排在最前面的路径进行扩展，这种选择机制不会影响算法的可行性.

第一次进行完整的搜索结束后，随机选择一定比例的边（1%，3%，5%，6%，8%和 10%），更改其代价向量的值，然后进行增量搜索，处理所有受影响的路径，再重新计算最优解集. 每次变化至少测试 100次，取路径扩展数、更新时间以及搜索时间的平均值，实验结果列于表 4～表 6. 表 4列出了不用启发式的 NAMOA*算法（NAMOA* 0）和使用启发式的 NAMOA*算法的平均路径扩展数与平均搜索时间，在此搜索基础上对矩阵中一些边的代价进行更改；表 5列出了不用启发信息的 MOLPA*算法（MOLPA* 0）在不同更改比例下的平均路径扩展数、平均更新时间及平均搜索时间；表 6列出了使用启发信息的 MOLPA*算法在不同更改比例下的平均路径扩展数、平均更新时间及平均搜索时间.

表 4　完整的 NAMOA*的路径扩展数与搜索时间

Table 4　Expanded paths and time of complete NAMOA*

算法	路径扩展数	搜索时间 / s
NAMOA* 0	2 701	11. 960 2
NAMOA*	2 044	8. 521 05

表 5　不用启发信息的 MOLPA*的路径扩展数、更新时间与搜索时间

Table 5　Expanded paths, update time and search time of MOLPA* without heurist

变化比例（%）	路径扩展数	更新时间 / s	搜索时间 / s
1	76	0. 011 713	2 256 94
3	183	0. 030 591	5 977 01
5	201	0. 037 775	7 397 36
6	298	0. 069 124	12 556 4
8	320	0. 079 433	13 955 1
10	364	0. 138 965	16 337 1

表 6　用启发信息的 MOLPA*的路径扩展数、更新时间与搜索时间

Table 6　Expanded paths, update time and search time of MOLPA* with heuristic

变化比例（%）	路径扩展数	更新时间 / s	搜索时间 / s
1	56	0. 009 07	1 215 01
3	143	0. 012 265	3 575 1
5	151	0. 022 799	4 046 06
6	225	0. 043 021	6 976 61
8	237	0. 048 92	7 489 47
10	271	0. 086 23	8 957 27

由表 5可见，当发生变化的比例较小时，MOLPA*在搜索时间上明显优于完整的 NAMOA*算法的搜索，通过重用先前搜索中保留的信息，能更快找到新的最优解集，时间效率明显提高，扩展的路径数更少，但当变化的比例大于 5%时，MOLPA*开始丧失优势，由于 OPEN表大量的插入和删除操作，整体搜索时间甚至长于重新执行 NAMOA*算法的搜索时间.

由表 6可见，使用启发信息引导搜索过程能压缩搜索空间，使路径扩展数更少，进一步缩短搜索时间，但当变化比例达到 10%时，MOLPA*也开始丧失优势，时间效率不如重新执行 NAMOA*算法. 因此，MOLPA*算法适用于解决一系列变化较小的多目标问题.

综上所述，本文提出了一种多目标增量启发式搜索算法 MOLPA*，当多目标问题搜索图的状态格局发生改变时，该算法通过对搜索现场进行实时更新，部分利用先前搜索保留的信息，从更新后的状态开始求解新问题的最优解集，从而提高了重搜索的效率. 实验结果表明算法效率较好. MOLPA*算

法可应用于智能交通、通信网络等动态线路规划问题.

参 考 文 献

[1] Frigioni D, Marchetti-Spaccamela A, Nanni U. Fully Dynamic Algorithms for Maintaining Shortest Paths Trees[J]. Journal of Algorithms, 2000, 34(2): 251-281.

[2] Ramalingam G, Reps T W. An Incremental Algorithm for a Generalization of the Shortest-path Problem[J]. Journal of Algorithms, 1996, 21(2): 267-305.

[3] Koenig S, Likhachev M, LIU Yaxin, et al. Incremental Heuristic Search in Artificial Intelligence[J]. Artificial Intelligence Magazine, 2004, 25(2): 99-112.

[4] Koenig S, Likhachev M, Furcy D. Lifelong Planning A*[J]. Artificial Intelligence, 2004, 155(1/2): 93-146.

[5] Koenig S, Likhachev M. D* Lite[C]//Proceedings of the Eighteenth National Conference on Artificial Intelligence and Fourteenth Conference on Innovative Applications of Artificial Intelligence. Menlo Park: AAAI Press, 2002: 476-483.

[6] Koenig S, Likhachev M. Real-time Adaptive A*[C]//5th International Joint Conference on Autonomous Agents and Multiagent Systems(AAMAS 2006). New York: ACM, 2006: 281-288.

[7] Mills-Tettey G A, Stentz A, Dias M B. DD* Lite: Efficient Incremental Search with State Dominance[C]//Tech Rep. Robotics Institute, Pennsylvania: Carnegie Mellon University, 2007: 1032-1038.

[8] JIN Yang, ZUO Wan-li. Multi-dimensional Concept Lattice and Incremental Discovery of Multi-dimensional Sequential Patterns[J]. Journal of Computer Research and Development, 2007, 44(11): 1816-1824. (金 阳, 左万利. 多维概念格与多维序列模式的增量挖掘[J]. 计算机研究与发展, 2007, 44(11): 1816-1824.)

[9] WANG Dong, ZUO Wan-li, HE Feng-ling, et al. Design and Implementation of an Incremental Inverted Index Framework[J]. Journal of Jilin University: Science Edition, 2007, 45(6): 953-958. (王 冬, 左万利, 赫枫龄, 等. 一种增量倒排索引结构的设计与实现[J]. 吉林大学学报: 理学版, 2007, 45(6): 953-958.)

[10] WANG Xiu-jun, SHEN Hong. Improved Growing Learning Vector Quantification for Text Classification[J]. Chinese Journal of Computers, 2007, 30(8): 1277-1285. (王修君, 沈 鸿. 一种基于增量学习型矢量量化的有效文本分类算法[J]. 计算机学报, 2007, 30(8): 1277-1285.)

[11] YANG Xiao-wei, OUYANG Bai-ping, YU Shu, et al. Study on Features of Support Vector Set of Adaptive and Iterative Algorithm[J]. Journal of Jilin University: Information Science Edition, 2006, 24(2): 153-157. (杨晓伟, 欧阳柏平, 余 舒, 等. 自适应迭代算法支持向量集的特性研究[J]. 吉林大学学报: 信息科学版, 2006, 24(2): 153-157.)

[12] LIAO Wei, XIONG Wei, WANG Jun, et al. Scalable Processing of Incremental Continuous k-Nearest Neighbor Queries[J]. Journal of Software, 2007, 18(2): 268-278. (廖 巍, 熊 伟, 王 钧, 等. 可伸缩的增量连续 k近邻查询处理[J]. 软件学报, 2007, 18(2): 268-278.)

[13] SHEN Sheng-yu, LI Si-kun. A Fast Counterexample Minimization Algorithm with Refutation Analysis and Incremental SAT[J]. Journal of Software, 2006, 17(5): 1034-1041. (沈胜宇, 李思昆. 基于悖论分析和增量求解的快速反例压缩算法[J]. 软件学报, 2006, 17(5): 1034-1041.)

[14] Stewart B S, White C C. Multiobjective A*[J]. Journal of the ACM, 1991, 38(4): 775-814.

[15] Dasgupta P, Chakrabarti P P, De Sakar S C. Multiobjective Heuristic Search[M]. Vieweg: Braunschweig/Wiebaden, 1999.

[16] Mandow L, De La Cruz J L P. Multicriteria Heuristic Search[J]. European Journal of Operational Research, 2003, 150(2): 253-280.

[17] Harikumar S, Kumar S. Iterative Deepening Multiobjective A*[J]. Information Processing Letters, 1996, 58(1): 11-15.

[18] Korf R E. Iterative-deepening A*: an Optimal Admissible Tree Search[C]//Proc of the IX Int Joint Conf on Artificial Intelligence(IJCAI-85). Los Angeles: [s.n.], 1985: 1034-1036.

[19] Korf R E. Linear-space Best-first Search[J]. Artificial Intelligence, 1993, 62(1): 41-78.