计算机研究与发展
Journal of Computer Research and Development

ISSN 1000-1239/CN 11-1777/TP
46(9): 1471-1479, 2009

基于规划图的蚁群规划算法

柴啸龙[1,2] 姜云飞[1] 陈蔼祥[2]
[1](中山大学软件研究所 广州 510275)
[2](广东商学院数学与计算科学学院 广州 510320)
(chaixiaolongok@163.com)

Ant Colony Planning Algorithm Based on Planning Graph

Chai Xiaolong[1,2], Jiang Yunfei[1], and Chen Aixiang[2]
[1](*Institute of Software, Sun Yat-sen University, Guangzhou* 510275)
[2](*School of Mathematics and Computing Science, Guangdong University of Business Studies, Guangzhou* 510320)

Abstract Graphplan is an important algorithm of intelligent planning in recent years. It has promoted great development of intelligent planning. Firstly, the Graphplan algorithm will generate a planning graph by action level expanding and proposition level expanding alternatively. Secondly, a valid plan will be extracted from the planning graph by backtracking in exhaustive way. The plan extracting of the algorithm always consume too much time in this way. And the algorithm is apt to plunge into the local searching. In this paper, a way of plan solution searching by using the ant colony algorithm is given. That is the ACP (ant colony planner) algorithm. In the ant colony algorithm, positive feedback and distributed coordination are used to find the solution path. And the ant colony algorithm even has the characteristic of robustness, thus it has been successfully applied in many applications which are NP-hard problems. The searching of the ACP has the characteristic of global and parallel searching. And ACP has the ability of convergence acceleration in the solution searching. The experiments show that ACP is advantage ous especially in solving the large scale planning problems. To absorb the optimizing technique and the learning technique is a rising way in the study of the intelligence planning. Since the ant colony planning algorithm is just based on the optimizing technique, thus the ant colony planning algorithm is promising to make some better progresses in the study of intelligence planning area by using the ant colony planning method.

Key words intelligence planning; planning graph; ant colony algorithm; parallel searching; convergence acceleration

摘 要 图规划是智能规划领域近年来出现的一种重要规划方法,对智能规划的发展起到了很重要的推动作用,图规划算法首先扩展生成规划图,然后通过逐层组合不断回溯的穷举方式进行解提取,这种方式使解提取不仅耗时而且容易陷入局部搜索中.在规划图基础上定义了蚁群智能体,并定义了在规划图上的蚁群搜索方式,提出了蚁群规划算法,使搜索具有较好的全局性和并发性,并具备加速收敛的寻解能力.实验表明,蚁群规划算法在求解一些相对规模较大的规划问题时有更好的优越性.

关键词 智能规划;规划图;蚁群算法;并行搜索;加速收敛

中图法分类号 TP18

收稿日期:2008-10-03;**修回日期**:2009-05-04

基金项目:国家自然科学基金项目(60773201);广东省自然科学基金项目(06301003)

智能规划是人工智能中的一个重要研究领域. 智能规划研究如何用自动化的方式找到规划问题的解,已有人给出证明,领域无关的规划问题是 NP 完全的[1]. 虽然比较困难,智能规划研究近年来仍然取得了较大进展,其中发展比较快的研究方向有在规划中引入各种启发式搜索策略[2];加强控制规则的作用;发现并自动提取各种领域知识,基于模型检测的规划技术等. 在非经典规划领域中发展较快的方向主要有关于时间和资源约束的规划;不确定规划,其中的不确定性主要有动作效果、可观察性以及初始状态和目标状态的不确定性;另外还有规划中的学习技术、规划的在线学习能力等. 2007 年在美国 Rhode Island 召开的智能规划与调度国际会议 ICAPS07 的讨论内容和分组情况也反应了这些新趋势.

近年来智能规化研究所提出的图规划技术为智能规划的研究开辟了一个新道路,1995 年 Blum 等人设计的图规划系统 Graphplan[3-4] 第 1 次采用图的方式来寻找规划解,极大地推动了领域无关的规划技术的发展,后来很多著名的规划系统如 FF, Blackbox, LPG[5] 等都采用了图规划系统的方法,它们从不同角度对图规划算法进行了相应的扩充和改进,都取得了显著的成果.

蚁群算法[6-8] 是 Dorigo 等人于 1991 年提出的具有并行性和全局性的群体智能搜索算法,其主要特点是通过正反馈、分布式协作来寻找解路径,它实质上是一种基于种群寻优的启发式搜索算法,蚁群算法有较好的健壮性和抗干扰能力[9-10],有较广的应用面,它是较早能够获得具有 NP 难度的旅行商问题最优解答的算法,而且蚁群算法在 Job-Shop 调度问题[11]、粗糙集的特征约减问题[12]、路径规划问题[13]、二次指派问题以及其他很多 NP 难问题的应用领域上都取得了优越成果[14-17].

将蚁群算法应用到智能规划领域中的工作,国内外尚少有文献报道,本文试图在规划图上引入蚁群算法,提出了一种蚁群规划算法,使规划解的搜索有较好的全局性和并发性,并具有一定的加速收敛能力,实验表明,蚁群规划算法在较大规模的规划问题求解上有一定优越性.

1 蚁群规划算法的平台基础

智能规划研究的问题是给定系统的初始状态、目标状态、动作的操作方式及其效果. 要求自动地给出完成任务所需的有序动作序列或动作集合序列. 下面首先介绍用一阶语言描述的一些相关定义:

定义 1. 动作. 动作 o 是三元组$\langle pre(o), add(o), del(o)\rangle$,其中:$pre(o)$是动作 o 的前提条件集; $add(o)$是动作 o 的添加效果集; $del(o)$是动作 o 的删除效果集.

$pre(o), add(o), del(o)$都是一阶语言表示的原子集合,对操作 o 进行例化就可得到相应的动作 a,状态 s 是用一阶语言表示的逻辑原子集. 在状态 s 下执行动作 a 的结果可表示为

$$\gamma(s,a)=\begin{cases}(s\cup add(a))\backslash del(a), & pre(a)\subseteq s,\\ undecided, & \text{otherwise.}\end{cases} \tag{1}$$

定义 2. 规划问题. 规划问题 P 是一个五元组,$P=\langle S, s_0, g, O, \delta\rangle$,其中:$S$ 是能够描述 P 所在的规划系统中所有可能状态的一阶语言的原子集合; $s_0\subseteq S$ 是规划系统 P 的初始状态; $g\subseteq S$ 是规划系统 P 的目标状态集; O 是规划系统 P 的有限操作集; $\delta: S\times O\to S$ 是规划系统 P 中的状态转换函数.

定义 3. 规划解. 给定规划问题 $P=\langle S, s_0, g, O, \delta\rangle$,规划解是一个例化后的动作序列 $\pi=\langle a_1, a_2, \cdots, a_n\rangle$,其中 $a_i\in A, i=1, \cdots, n$. 在允许动作可并发执行时,规划解也可以是一个动作集有序序列$\pi=\langle A_1', A_2', \cdots, A_n'\rangle$,其中 $A_i'\subseteq A$ 是可并发执行的动作集. 通过有序地执行 π 可以解决规划问题 P. 这里的 A 为操作集 O 的例化集.

下面简要介绍规划图与动作图的的概念,在此基础上将展开蚁群规划算法的设计.

规划图是一种包含两类节点三类边的有向无环分层图. 规划图中命题层和动作层交替出现,命题层包括一些命题节点(由一些命题表示),动作层包含动作节点(表示为一些动作),在第 t 层的动作节点代表时间步 t 所有可能的规划动作,事实节点表示的命题对应于时间步 t 被例化的一个或多个动作的效果. 规划图的第 0 层是命题层,包括规划问题的初始状态的所有命题. 经过充分扩展后的规划图,其最后一层至少需要包含规划问题的所有目标命题.

规划图中假定 a 为第 i 层动作节点,则有下面的 3 类边与 a 相连.

1) 前提条件边(precondition-edges):连接第 i 层的执行 a 的前提条件的事实节点;

2) 添加效果边(add-edges):连接第 $i+1$ 层的执行 a 以后所产生的正效果的事实节点;

3) 删除效果边(delete-edges):连接第 $i+1$ 层

的表示执行 a 后所产生的负效果的事实节点.

当不存在有效规划解使某一层如第 i 层的两个动作能够同时出现时称两动作互斥. 当两动作互斥时至少满足以下 3 种情况之一.

1) 后件(效果)不一致(inconsistent):一个动作的后件是另一个动作后件的否定;

2) 冲突(interference):一个动作的执行删除另一个动作的前提条件;

3) 竞争所需(competing needs):两个动作具有在 $i-1$ 层中发生互斥关系的前件.

当不存在有效规划解,使得两事实节点对应的命题同时成立,则两事实节点互斥,当两事实节点互斥时,满足下列两者中的至少一个:

1) 一个命题是另一个命题的否定;

2) 不一致支持(inconsistent support):获得这两个命题的所有动作对都具有互斥关系.

图 1 展示了同层动作节点的 3 种互斥情形和同层命题节点的不一致支持的互斥情形:

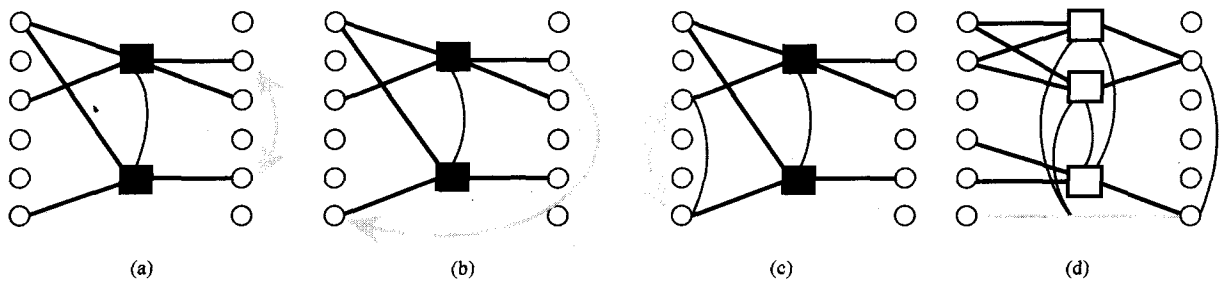

Fig. 1 Action mutex and the literal mutex. (a) Inconsistent effects; (b) Interference; (c) Competing needs; and (d) Inconsistent support.

图 1 动作节点的互斥和命题节点的互斥. (a) 效果不一致; (b) 冲突; (c) 竞争所需; (d) 不一致支持

图规划算法主要有两个阶段. 首先在多项式时间内构造出规划图,当扩展的规划图最后一层的命题层中包含所有目标状态命题,且两两不互斥时,停止图扩展,转向解搜索阶段,在规划图上进行解搜索时,经典的图规划算法采用反向的方法,从目标命题层逐层向前搜索,直到不互斥地到达初始命题层.

在这两个阶段中,执行第 2 阶段的耗时要比执行第 1 阶段的耗时大得多. 造成这一现象的主要原因是经典图规划算法是采用逆向逐层搜索,每前进一层都要在不同的动作组合中进行选取,然后不断进行是否不互斥的支持计算,一旦发现不支持则进行重新组合或者回溯处理,当算法终于成功地发现一条完整的从最后一层命题层一直到首层命题层的不互斥路径时,算法有可能快要穷尽整个解空间,这样就造成了效率的低下.

蚁群规划算法吸收了图规划算法中不互斥支持计算的优点,而且通过构造规划图,图规划算法将所有可能的动作例化,它给出了规划解的所有有序搜索的完整空间. 这样很方便地给蚁群规划算法提供了计算的平台空间. 但蚁群规划算法抛弃了图规划算法中的规划解搜索方式,转而采纳蚁群算法具有全局并发搜索的优点,当系统发现更加近似的规划解时,通过信息素的浓度变化可以对其进行迅速强化,从而实现解的快速收敛.

2 蚁群规划算法的设计

以上介绍了蚁群规划算法的平台基础,本节给出蚁群规划算法的相关概念和详细设计.

2.1 蚁群的路线及其不协调度

给定规划问题 P,假定其对应的扩展规划图 G 通过上节所介绍的图规划算法已生成完毕. 设 G 中一共包含 k 个动作层. A_i 是第 i 层动作层中所有动作组成的集合,$i=1,\cdots,k$. 下面是蚁群路线的相关概念.

定义 4. 路节. A_i 中任意一个非空动作子集 π_{ij},如果 π_{ij} 中任意两个动作在 A_i 中都不互斥,则把 π_{ij} 称为是 G 中动作图的第 i 层的一个路节. 把第 i 层所有路节组成的集合记为 Π_i. 如果在路节 π_{ij} 中加入任何 A_i 中别的动作,π_{ij} 都不再成为路节,则称 π_{ij} 为第 i 层的一个极大路节.

路节是蚂蚁行走的最小单位,每个路节对应蚂蚁的一个时间步. 显然第 i 层的极大路节的非空子集也是第 i 层的路节. 在得到扩展规划图 G 后,可通过极大路节的计算来方便地获取所有路节.

定义 5. 路线. G 中从第 1 层到第 k 层的一个有序路节序列$\langle \pi_{1j_1},\cdots,\pi_{kj_k}\rangle$称为一条正向路线,从第 k 层到第 1 层反向的有序路节序列称为反向路线,

正向路线和反向路线统称为路线. 路线用符号 ψ 表示. 规定蚂蚁的运动轨迹集是路线集 Ψ 的子集.

每条正向路线都可看作是规划问题的一个近似解,规划解就是完全符合所有特定要求的正向路线. 在算法初期,为保证全局性和多样性,蚁群的路线选择不受环境约束,在达到设定阈值之后,蚁群的路线选择会越来越受环境的限定和影响,蚁群会对路线轨迹进行不断调整,通过启发信息的不断引导和强化,加速收敛出符合要求的轨迹,这一过程同时也是规划解的逼近取得的过程.

为了能高效简洁地评价路线的优劣,需要考虑动作层之间的前件和效果等约束互动关系,下面给出相关概念.

定义 6. 理想命题集. 路线 $\psi=\langle\pi_{1j_1},\cdots,\pi_{kj_k}\rangle$,定义 ψ 的第 i 层理想命题集为 $Ideal_i(\psi)$,其中 $i=0,\cdots,k-1$,具体如下:

$$Ideal_i(\psi)=\begin{cases}Ideal_{i-1}(\psi)\cup\bigcup\limits_{a\in\pi_{ij_i}}add(a)\setminus\bigcup\limits_{a\in\pi_{ij_i}}del(a),\\ \quad 0<i<k,\\ I,\ i=0,\end{cases}\tag{2}$$

其中 I 是初始状态命题集. 路线 ψ 的第 i 层理想命题集 $Ideal_i(\psi)$ 所表达的含义是:如果把 ψ 当作是完全不冲突可顺利执行的一个动作集有序序列来处理,那么在执行完 ψ 中前 i 层动作集后,用来描述所得状态命题集的就是 $Ideal_i(\psi)$. 虽然 $Ideal_i(\psi)$ 所描述的状态一般并不会出现,但 ψ 中各层的 $Ideal_i(\psi)$ 同 ψ 中各路节之间所引发的各种冲突的次数却提供了一个评价路线 ψ 优劣的量化方法,详见下面定义.

定义 7. 不协调度. 路线 $\psi=\langle\pi_{1j_1},\cdots,\pi_{kj_k}\rangle$,把第 i 个路节 π_{ij_i} 在 ψ 中的不协调度定义为

$$disorder_i(\psi)=\left\|\bigcup_{a\in\pi_{ij_i}}pre(a)\setminus Ideal_{i-1}(\psi)\right\|.\tag{3}$$

路线 ψ 的不协调度定义为 $Disorder(\psi)=\sum\limits_{i=1}^{k}disorder_i(\psi)$. 其中 $\setminus$ 是集合之间的差集算子,$\|\cdot\|$ 是集合的阶算子,用来求集合中的元素个数.

蚁群规划算法是在路线空间中搜寻规划解的,搜索的粒度和规划解的粒度完全相同,图规划算法以及基于图规划的 LPG 算法是在动作空间中搜索规划解,相当于在蚁群规划算法中的路节空间进行搜索,其搜索粒度要比规划解本身的粒度要小很多. 因此图规划算法和 LPG 算法不可避免地容易陷入大量的局部搜索,Graphplan 和 LPG 算法都是公开源代码的系统,从它们在执行第 2 阶段时回溯次数较多的情况也可以看出这一事实.

蚁群规划算法有较好的全局性和并发性,但如何保证规划解的快速收敛是一个关键问题,我们通过引入路线的不协调度等启发式评价信息来引导蚁群通过信息素的调节和不断强化来实现规划解的快速收敛,详见下面内容.

2.2 蚁群的信息素控制和选路策略

算法中的蚂蚁是一种简单的智能体,我们设定它只按离散时间步进行运动,每个时间步对应一个路节,这里的时间步和动作的时间步是相对应的. 蚂蚁用 k 个时间步(k 为 G 的动作层的层数)走完一条路线,因此称 k 个时间步为一个周期. 如果蚂蚁用不到 k 个时间步就找到并选择一个第 l 层($l<k$)的路节 π_l,其中 $\bigcup\limits_{a\in\pi_l}add(a)$ 包含了所有的目标命题且在第 l 层中不互斥,则让该蚂蚁在该周期的剩余的 $k-l$ 个时间步中不停地走空路节 π_ϕ. 蚂蚁能够记忆一个周期内走过的路线. 如果蚁群能够发现一条路线 ψ,其中 ψ 的最后一个非空路节能够生成所有目标命题,且满足条件 $Disorder(\psi)=0$,则 ψ 所对应的动作集有序序列就是规划问题 P 的一个解.

设定蚁量为 m,蚂蚁每走完一个周期后给它所经过的 k 个路节增加信息素(空路节除外). 路节 π_{ij} 在第 τ 个周期结束时所拥有的信息素的量值为 $\xi(\pi_{ij},\tau)$,每经历一个完整的周期后,信息素的量值会依据下面的公式重新计算一次:

$$\xi(\pi_{ij},\tau+1)=\rho\times\xi(\pi_{ij},\tau)+\sum_{h=1}^{m}\delta(\pi_{ij},h),\tag{4}$$

其中 ρ 是每经过一个周期后信息素经过挥发后保留下来的程度. $\delta(\pi_{ij},h)$ 是第 h 只蚂蚁在上一周期中经过路节 π_{ij} 后给 π_{ij} 带来的信息素增加值,其计算公式如下:

$$\delta(\pi_{ij},h)=\begin{cases}c_0\times(Max-Disorder(\Psi(h))),\\ \quad \pi_{ij}=\Psi_i(h),\\ 0,\ \pi_{ij}\neq\Psi_i(h),\end{cases}\tag{5}$$

其中 $\Psi(h)$ 是第 h 只蚂蚁在一个周期刚结束时所走出的路线,$\Psi_i(h)$ 是这条路线上第 i 个路节,Max 是一个常数,它是路线不协调度的某一个上限值,$Max-Disorder(\Psi(h))$ 越大表明路线中各路节之间协调得越好,这条路线的质量也就越好. c_0 是一个常系数,它是由于不希望 $\delta(\pi_{ij},h)$ 的数值比较大而引入的一个缩值系数. c_0 的引入能间接地给式(4)的相

关计算提供一些方便,而且它给 ρ 的设定提供了更大的适宜范围. ρ 值设定过高会使旧的信息素挥发较慢,导致当出现更好的近似解时不能迅速给以强化,从而降低了解的收敛速度,而 ρ 值设定过低又会导致信息素挥发较快,使算法容易陷入局部收敛之中.

从上面可以看出,蚁群走出两条不同路线后,质量好的路线能够获得更多的信息素. 为了提高蚁群的选路质量,提高路节之间的连接合理性是非常重要的,下面给出一个相关概念.

定义 8. 禁忌连接集. Π_i 和 Π_{i+1} 分别是 G 中动作图的第 i 层和第 $i+1$ 层路节集($1\leqslant i<k$). $\forall\pi_1\in\Pi_i$, $\forall\pi_2\in\Pi_{i+1}$,称 π_2 是 π_1 的一个禁忌连接,如果满足下面条件:

$$\bigcup_{a\in\pi_2} pre(a)\subset\{\bigcup_{b\in\pi_1} pre(b)\cup\bigcup_{c\in\pi_1} add(c)\setminus\bigcup_{d\in\pi_1} del(d)\}. \tag{6}$$

把 π_1 在 Π_{i+1} 中的所有禁忌连接定义为 $\pi_1\in\Pi_i$ 的禁忌连接集,记为 $\chi(\pi_1\in\Pi_i)$. Π_i 中的 π_1 和 $\chi(\pi_1\in\Pi_i)$ 中的路节是存在冲突的前后相邻的路节对,在路节选择中如果规避了禁忌连接集,那么路节之间连接的合理性将会得到有效提高. 由于蚁群选择每条路线时都会进行 $k-1$ 次规避禁忌连接集,这样每一条所选路线的合理性和正确性都会得到很大提高,而计算和规避禁忌连接集并不会占用太多时间. 因为规避禁忌连接集的操作并不参与算法中的循环,它会在蚁群开始搜索之前就计算完毕.

为了便于对路节及信息素进行相关计算,也为了提高禁忌连接集的表达和存储的方便性,算法中引入了一种 LookUp 表,如图 2 所示. 首先对 $\Pi_i(i=1,\cdots,k)$ 中的所有路节按照层次进行自动编码,把路节和对应编码的关系存入各层路节的 *LookUp_Table_*1 中,然后把每个路节的禁忌连接集转化为相应的编码集,并把对应关系存入 *LookUp_Table_*2 中,按照从前层到后层的顺序逐步生成完毕. 路节上信息素的存放、调取和修改也都在 *LookUp_Table_*2 中进行.

我们限定蚁群在最后一层路节集 Π_k 中的路节选择上,只能选取可以取得所有目标命题的路节,即选取 $\pi\in\Pi_k$,且 $\bigcup_{a\in\pi} add(a)$ 包含所有目标命题,这只需增加 *LookUp_Table_*2 中第 $k-1$ 层路节的禁忌连接集的相关编码即可. 这一限定相当于吸收了逆向搜索时最后几层搜索范围小的优点,进一步减少了搜索空间.

为了让拥有更多信息素的路线能够吸引更多的蚂蚁,更多的蚂蚁会带来更多的信息素,从而取得强化和加速收敛的效果. 在第 $\tau+1$ 个周期中,当第 h 只蚂蚁经过第 i 层的路节 π_{ij} 后,若记 $R(\pi_{ij})=\Pi_{i+1}\setminus\chi(\pi_{ij})$,设定第 h 只蚂蚁选择第 $i+1$ 层的路节 $\pi_{i+1j_{i+1}}$ 的概率为

$$P(\pi_{i+1j_{i+1}})=\begin{cases}0, & \pi_{i+1j_{i+1}}\in\chi(\pi_{ij}),\\ \dfrac{1}{\|R(\pi_{ij})\|}, & \pi_{i+1j_{i+1}}\notin\chi(\pi_{ij}),\\ & \text{且 }\tau<T_0,\\ \dfrac{\xi(\pi_{i+1j_{i+1}},\tau)}{\sum\limits_{\pi_{i+1}\in R(\pi_{ij})}\xi(\pi_{i+1},\tau)}, & \pi_{i+1j_{i+1}}\notin\chi(\pi_{ij}),\\ & \text{且 }\tau\geqslant T_0,\end{cases} \tag{7}$$

其中 T_0 是一个时间周期的分界线常数,设定它的目的是为了避免蚁群过早单纯地因为互相吸引而出现集中,那样会导致算法过早地在局部空间进行收敛搜索,在算法初期($\tau<T_0$),我们让蚁群完全不受信息素的影响,因为初期的时候,信息素的分布只是单纯地反应了蚁群的分布,只有到了后面信息素的分布才会越来越多地反应路线优劣的分布.

Action layer No.	*Path-section* No.	*Path-section* $\pi_{ij}=\{a_{i1},\cdots,a_{iu}\}$

(a)

Action layer No.	*Path-section* No.	*Taboo connection set* No.	*pheromone*

(b)

Fig. 2 *LookUp_Table_*1 and *LookUp_Table_*2. (a) Main contents of *LookUp_Table_*1 and (b) Main contents of *LookUp_Table_*2.

图 2 *LookUp_Table_*1 和 *LookUp_Table_*2 的主要内容. (a) *LookUp_Table_*1 的主要内容;(b) *LookUp_Table_*2 的主要内容

2.3 蚁群规划算法的调校

为了让蚁群规划算法有更好的适应能力,算法在两方面进行了调校:一是进行信息素的定期平滑处理,加强算法在局部搜索时的跳出能力和全局搜索中的解发现能力;另一个是在搜索执行一段时间后,加强优秀近似解附近的局部搜索能力. 下面分别进行阐述.

2.3.1 信息素的定期平滑

在蚁群开始进行路线搜索以后,每过 N 个周期,

将所有路节上的信息素的值通过简单的线性公式转换为闭区间$[\xi_{\min},\xi_{\max}]$中的值. 转换公式如下:

$$f(x)=\frac{\xi_{\max}-\xi_{\min}}{\xi_M}\cdot x+\xi_{\min}, \tag{8}$$

其中,$\xi_{\min}$和$\xi_{\max}$是路节信息素预先设定的最小值和最大值($0<\xi_{\min}<\xi_{\max}$),而ξ_M是当前所有路节上信息素的最大值,x表示当前任意路节上的信息素的值,$f(x)$是转换之后该路节上的新的信息素值. 信息素的数值本身并不会体现太多含义,我们只是通过它们之间的相对差别来体现路线之间的优劣.

如果不对路节的信息素进行定期平滑处理,则一些暂时显得比较好的路线会吸引越来越多的信息素,由于其信息素的水平过高而更加容易吸引蚂蚁导致信息素易升不易降. 一旦该路线的附近邻域并不存在解,则容易导致算法陷入大量局部搜索中. 因此定期地(N个周期)通过$f(x)$将路节中信息素差别进行缩小有助于加强算法在局部搜索时的跳出能力. 通过$f(x)$的定期平滑处理并不会伤害那些信息素值高的真正优秀的路线,因为启发式评价信息总会使优秀路线继续保持突出. 相反,一些信息素暂时很高的较差路线在被平滑处理后,在启发式信息评价作用下,其信息素值很难再次得到快速增加.

2.3.2 最优路线的变异

上面所引入的信息素平滑处理主要是为了加强算法的全局性和并发性,这一处理策略在算法的初期和中期是有积极意义的,而在算法进行相当一段时间后,由于有前面全局性的保证,解的附近邻域应该已被搜索过,这时需要加强一些表现较好的近似解的附近邻域的搜索.

当算法执行较长一段时间后($\tau>T_1$)如果仍没找到解,则引入对当前最优路线的变异处理,加强最优近似解附近的局部搜索能力,具体如下所述.

在第T_1+1个周期及以后每个周期结束后,在m只蚂蚁刚走出的m条路线中,选出启发式评价值最高(不协调度值最低)的一条路线ψ_ω. 如果ψ_ω是解,结束循环,否则用数量为m_0的小部分蚂蚁在下一周期中执行下面的新选路策略,其余$m-m_0$只蚂蚁的选路策略不变.

用$(\psi_\omega)^i$表示路线ψ_ω中的第i层路节,$i=1,\cdots,k$,记$R'(\pi_{ij})=\Pi_{i+1}\setminus\chi(\pi_{ij})\cup\psi_\omega^{i+1}$),当第$h_0$($1\leqslant h_0\leqslant m_0$)只蚂蚁经过第$i$层的路节$\pi_{ij}$后,令它选择第$i+1$层路节$\pi_{i+1j_{i+1}}$的概率为

$$P(\pi_{i+1j_{i+1}})=\begin{cases}\lambda, \text{若 } \pi_{i+1j_{i+1}}=(\psi_\omega)^{i+1},\\ 0, \text{若 } \pi_{i+1j_{i+1}}\in\chi(\pi_{ij}),\\ (1-\lambda)\cdot\dfrac{\xi(\pi_{i+1j_{i+1}},\tau)}{\sum\limits_{\pi_{i+1}\in R'(\pi_{ij})}\xi(\pi_{i+1},\tau)}, \text{若}\\ \pi_{i+1j_{i+1}}\neq(\psi_\omega)^{i+1},\pi_{i+1j_{i+1}}\notin\chi(\pi_{ij}).\end{cases} \tag{9}$$

算法把λ设定为区间$[0.5,0.9]$中的实数,它表示仍然选择路线ψ_ω中路节的概率,开始时取$\lambda=0.5$,如果当前周期结束后,新的ψ_ω是由数量只有m_0的小队蚂蚁中的一只走出来的,则加大λ的值. 这样做的作用是,如果在最优近似解的附近邻域能够搜索到更好结果后,则进一步适当加大在新的最优近似解附近邻域的搜索力度.

2.4 算法的总体描述

下面给出蚁群规划算法的一个总体描述:

算法 1. *AntPlan_Based_PlanningGraph*(P, *max_cycle*, *ant_set*).

输入:规划问题P、运行最大周期数*max_cycle*、智能体*ant_set*的相关参数;

输出:规划解P或*fail*.

```
当规划图 G 没有达到不动点时执行循环:
{
  执行 10 次 create_graph_layer(G,P);
  若 G 取得目标命题集,且它们在最后一层中不互斥,则执行:
  {
    基于 G 生成路节集合{π_ij};
    根据{π_ij}和 G 设定 LookUp_Table_1;
    根据{π_ij} 和 G 计算 taboo connection sets;
    根据 LookUp_Table_1 和 taboo connection sets 生成 LookUp_Table_2;
    for τ ←1 to max_cycle do
    {
      if τ <T_0
      {
        屏蔽信息素的影响,基于 LookUp_Table_1 和 LookUp_Table_2 进行蚁群搜索,将路节顺序作为路线存入 routes;
      }
      else if τ ≤T_1
```

```
        {
          在信息素的影响下,基于 LookUp_Table_1 和 LookUp_Table_2 进行蚁群搜索,将路节顺序作为路线存入 routes;
        }
        else
        {
          在信息素的影响下,取其中 m0 只蚂蚁智能体基于 LookUp_Table_1 和 LookUp_Table_2 在基于式(9)的新概率策略下进行搜索,将路节顺序作为路线存入 routes;其余 m- m0 只蚂蚁智能体的搜索概率策略不变,路节顺序结果同样作为路线存入 routes;
        }
        对 routes 中的所有路线 ψ,执行:
        {
        if Disorder(ψ)=0
          把 ψ 作为解,并返回它所对应的规划解;
        else
          根据 Disorder(ψ)和 ψ 中各路节的信息素值重新设定信息素值,存入LookUp_Table_2 中;
        }
        if τ% N=0
        {
          依据 f(x)重新修订 LookUp_Table_2 中个路节上的信息素值;
        }
      }
    }
  }
  return failure;
```

3 实验及分析

3.1 测试运行结果对比

选取智能规划中的基准规划领域 BlocksWorld 中的部分问题作为测试例子. 分别用 Graphplan, LPG 和本文的 ACP(ant colony planner)对上述问题进行求解. 为了减少随机函数和不同的参数设定对算法的性能带来的影响,每个问题分别求解 5 次,用平均求解时间和求解长度作为算法运行的时间和长度. 测试平台为 CPU(Celeron2. 4GHz)+RAM(512MB)+Redhat9. 0+gcc3. 0,实验结果如表 2 所示:

Table 2 Testing Results Comparison of BlocksWorld Examples

表 2 BlocksWorld 例子测试结果对比

Problem	ACP		Graphplan		LPG	
	Time/s	Step	Time/s	Step	Time/s	Step
blocks-4	0.12	8	0.08	10	0.02	10
blocks-5	0.16	10	0.09	10	0.05	22
blocks-6	0.17	12	0.14	12	0.03	12
blocks-7	0.31	20	1.38	22	0.08	42
blocks-8	0.19	18	0.47	20	0.04	38
blocks-9	0.42	26	0.93	28	0.22	96
blocks-10	0.57	32	2.47	34	0.67	138
blocks-11	0.52	30	6.53	34	0.26	98
blocks-12	0.68	34	0.47	36	0.33	198
blocks-13	0.93	46	0.64	52	0.42	184
blocks-14	2.31	38	1.89	43	1.12	108
blocks-15	6.58	42	5.36	57	2.73	180

3.2 分 析

从实验结果来看,对于规模较小的规划问题,比如 blocks-4,blocks-5,blocks-6 等问题,ACP 在求解速度上并没有体现出优势. 而当问题规模逐渐变大时,ACP 的全局搜索能力以及它的加速收敛特点就使求解速度和求解质量逐渐显示出优势来.

从实验可以观察到,在求解一些较大规模的问题时,Graphplan 通过不断回溯来搜索解的弱点就容易暴露出来,这让它容易陷入局部搜索中,因此找到的解往往不是较优的解. LPG 在求解速度上表现得更好一些,这是由于它可以基于动作图对近似规划解做进一步的优化. 而 ACP 由于是从全局的角度进行优化和加速收敛的,因此所取得的规划解质量更好一些.

4 结论与进一步工作

本文的主要工作是在规划图的基础上引入蚁群算法,充分运用蚁群算法的全局并行搜索以及加速收敛能力来实现规划解的搜索. 蚁群规划算法具有较好的求解速度和求解质量,与同类规划算法相比,其在求解稍大一些的规划问题时有更好的求解质量.

展望进一步的工作,利用优化技术和学习技术寻找规划解将是一个比较新兴的智能规划研究方

式,有很好的研究前景.我们将从以下方面对蚁群规划算法进行完善:

1) 引入合理的约束往往能够提高系统的效率,在智能规划领域中最高效的约束就是领域约束,规划问题的领域约束总是有很多,即使看似很简单的BlocksWorld问题也能发掘出至少33个有效的领域约束知识[18],很多潜在的领域约束即使领域专家也未必能全面总结,如果能够通过机器学习技术,尤其是强化学习技术来自动提取领域约束,以此来改进蚁群规划算法的选路策略,则有希望进一步提高算法的效率.

2) 如果已经获得某规划问题的规划解,当规划问题出现一些较小变动而规划环境完全不变时,可否通过对原规划解进行优化来获得新规划问题的解,这也是一项有意义的研究工作,考虑到蚁群规划算法正是基于优化技术进行寻解的,因此蚁群规划算法应该比其他规划算法更有优势来进行这项工作,这方面还有待进一步研究和探讨.

3) 在当前规划领域的研究中,不仅要提高规划算法的求解效率,还要争取扩大规划算法的处理范围,因此进一步的工作还包括加强蚁群规划算法对时间、资源等的处理能力,使算法能够处理综合调度等更复杂的规化应用问题.

参 考 文 献

[1] Helmert M. Complexity results for standard benchmark domains in planning [J]. Artificial Intelligence, 2003, 143(2): 219-262

[2] Bonet B, Geffner H. Planning as heuristic search [J]. Artificial Intelligence, 2001, 129(1-2): 5-33

[3] Blum A, Furst M. Fast planning through planning graph analysis [C] //Proc of the 14th Int Joint Conf on AI. Menlo Park: AAAI Press, 1995: 1636-1642

[4] Blum A, Furst M. Fast planning through planning graph analysis [J]. Artificial Intelligence, 1997, 90(1): 281-300

[5] Gerevini A, Serina I. LPG: A planner based on local search for planning graphs with action costs [C] //Proc of the 6th Int Conference on Artificial Intelligence Planning and Scheduling (AIPS'02). Menlo Park: AAAI, 2002: 169-170

[6] Dorigo M, Maniezzo V, Colorni A. Introduction to natural algorithms [J]. Rivista-di-Informatica, 1994, 24(3): 179-197

[7] Dorigo M, Maniezzo V. Ant system: Optimization by a colony of cooperating agents [J]. IEEE Trans on Systems, Mans, and Cybernetics, Part B: Cybernetics, 1996, 26(1): 29-41

[8] Dorigo M, Bonabeau E. Ant algorithm and stigmergy [J]. Future Generation Computer Systems, 2000, 16(8): 851-871

[9] Keinprasit R, Chonqstitvatana P. High-level synthesis by dynamic ant [J]. International Journal of Intelligent Systems, 2004, 19(1-2): 25-38

[10] Silvaa C A, Sousa J M C, Runkler T A. Rescheduling and optimization of logistic processes using GA and ACO [J]. Engineering Applications of Artificial Intelligence, 2008, 21(3): 343-352

[11] Serap Ulusam Sec _ kiner, Mustafa Kurt. Ant colony optimization for the job rotation scheduling problem [J]. Applied Mathematics and Computation, 2008, 201(1-2): 149-160

[12] Liangjun Ke, Zuren Feng, Zhigang Ren. An efficient ant colony optimization approach to attribute reduction in rough set theory [J]. Pattern Recognition Letters, 2008, 29(9): 1351-1357

[13] Kwee Kim Lim, Yew-Soon Ong, Meng Hiot Lim, et al. Hybrid ant colony algorithms for path planning in sparse graphs [J]. Soft Computing, 2008, 12(10): 981-994

[14] Kwang Mong Sim, Weng Hong Sun. Ant colony optimization for routing and load-balancing: Survey and new directions [J]. IEEE Trans on Systems, Man, and Cybernetics, Part A: Systems and Humans, 2003, 33(5): 560-572

[15] Wu Bin, Shi Zhong Zhi. An ant colony algorithm based partition algorithm for TSP [J]. Chinese Journal of Computers, 2001, 24(12): 1328-1333 (in Chinese)
(吴斌, 史忠植. 一种基于蚁群算法的TSP问题分段求解算法[J]. 计算机学报, 2001, 24(12): 1328-1333)

[16] Chen Ling, Shen Jie, Qin Ling. A method for solving optimization problem in continuous space by using ant colony algorithm [J]. Jounal of Software, 2002, 13(12): 2317-2323 (in Chinese)
(陈峻, 沈洁, 秦玲. 蚁群算法求解连续空间优化问题的一种方法[J]. 软件学报, 2002, 13(12): 2317-2323)

[17] Du Ronghua, Yao Gang, Wu Quanyuan. Application of an ant colony algorithm in migration of mobile agent [J]. Journal of Computer Research and Development, 2007, 44(2): 282-287 (in Chinese)
(杜荣华, 姚刚, 吴泉源. 蚁群算法在移动Agent迁移中的应用研究[J]. 计算机研究与发展, 2007, 44(2): 282-287)

[18] Wu Xiang Jun, Jiang Yun Fei, Ling Ying Biao. Strategy of extracting domain knowledge for STRIPS world [J]. Journal of Software, 2007, 18(3): 490-504 (in Chinese)
(吴向军, 姜云飞, 凌应标. 基于STRIPS的领域知识提取策略[J]. 软件学报, 2007, 18(3): 490-504)

Chai Xiaolong, born in 1980. Lecturer. Since 2006, he has been a PhD candidate in computing science from the Sun Yat-sen University, Guangzhou, China. His current research interests include intelligence planning and knowledge engineering.

柴啸龙, 1980年生, 博士研究生, 讲师, 主要研究方向为智能规划、知识工程.

Jiang Yunfei, born in 1945. He is a professor and PhD supervisor of Sun Yat-sen University. His current research interests include intelligence planning, automatic reasoning, model checking and knowledge engineering.

姜云飞,1945 年生,教授,博士生导师,主要研究方向为自动推理、智能规划、基于模型的诊断、知识工程.

Chen Aixiang, born in 1978. Received PhD degree in computing science from Sun Yat-sen University in 2007. Since 2007, he has been a lecturer in Guangdong University of Business Studies, Guangzhou, China. His current research interests include intelligence planning, model checking, algorithm and complexity.

陈蔼祥,1978 年生,博士,主要研究方向为智能规划、模型诊断、算法及复杂性.

Research Background

Intelligence planning is an important study area in AI. It has been proved that the planning problems without domain-dependent constraints are NP-complete. Despite the difficulty, the study of the intelligence planning has made great progress in recent years. In this paper, an ant colony planning algorithm based on the planning graph is presented, that is the ACP (ant colony planner) algorithm. The searching of the algorithm has the characteristic of global and parallel. And ACP has the ability of convergence acceleration in the solution searching. Ant colony algorithm is a kind of searching algorithm with swarm intelligence which has the characteristic of global and parallel. In the ant colony algorithm, positive feedback and distributed coordination are used to find the solution path. And the ant colony algorithm even has the character of robustness, thus it has been successfully applied in many applications which are NP-hard problems such as the travelling salesman problem, the job rotation scheduling problem, etc. To absorb the optimizing technique and the learning technique is a rising way in the study of the intelligence planning. Since the ant colony planning algorithm is just based on the optimizing technique, thus it is promising to make better progress in the study of intelligence planning area by using the ant colony planning method. Our work is supported by the National Natural Science Foundation of China under Grant No. 60773201 and the National Natural Science Foundation of Guangdong Province under grant No. 06301003.