

文章编号: 1674-9057(2014)02-0765-06

doi:10.3969/j.issn.1674-9057.2014.02.028

# 基于改进蚁群粒子群算法的移动机器人路径规划

何少佳, 史剑清, 王海坤

(桂林电子科技大学 a. 广西制造系统与先进制造技术重点实验室; b. 机电工程学院, 广西 桂林 541004)

**摘要:** 全局静态地图下, 针对蚁群算法规划机器人移动路径时存在计算时间长、搜索效率低, 并且得到的优化路径转弯次数过多的问题, 提出了一种改进蚁群粒子群算法: 首先利用粒子群算法快速得到蚁群算法初始信息素, 然后进行蚁群算法路径规划, 对得到的路径采用惯性优化, 对每个节点进行遍历, 当2个节点间的路径上无障碍物时, 将中间节点删除, 转换为优化路径。仿真实验表明, 该方法与传统蚁群算法及相关改进算法相比, 能有效减少迭代次数、提高搜索效率、减少转弯次数、缩短路径长度, 从而提高路径质量。

**关键词:** 粒子群算法; 蚁群算法; 路径规划; 移动机器人

**中图分类号:** TP391.9

**文献标志码:** A

## 0 引言

路径规划是移动机器人的关键技术之一, 它在一定程度上标志着移动机器人智能水平的高低。快速找出一条便捷、无碰撞的路径不仅可以保证移动机器人自身的安全, 更体现了机器人的高效性和可靠性。目前, 研究人员采用较多的方法是蚁群算法(ACO)和粒子群算法(PSO)。这两种算法作为进化算法的重要分支, 有着各自的优缺点: 蚁群算法具有良好的分布机制和信息反馈机制, 拥有启发式搜索能力, 但是运算初期信息素缺乏, 求解速度慢<sup>[1-2]</sup>; 粒子群算法概念简单、易于实现, 需调整参数较少, 收敛速度快, 具有较强的全局搜索能力, 但后期局部搜索能力差, 易陷入局部最优<sup>[3-4]</sup>。

本文针对全局静态路径规划问题展开研究。虽然现阶段已经能够较好地利用优化的蚁群算法、粒子群算法找出最优路径, 但是仍然存在迭代次数较多、运算时间过长等问题, 不能很好地满足

移动机器人对实时性的要求; 而且, 在得到的路径中存在转弯次数较多的情况, 这将严重影响移动机器人的工作效率。针对这两个问题, 将蚁群算法和粒子群算法的优点结合起来, 提出一种改进蚁群粒子群优化算法。该方法由3部分组成: ①先利用粒子群算法得到蚁群算法所需要的初始信息分布; ②采用蚁群算法, 但是每次迭代只记录爬行路线, 不释放信息素; ③对每条爬行路线进行惯性优化, 然后进行信息素更新。

## 1 环境描述

移动机器人的路径规划是在工作环境中找出一个从起始点到目标点的点的序列。为不失一般性, 对移动机器人的工作空间作出以下假设: ①移动机器人的活动范围在一个有限的二维空间; ②以移动机器人的尺寸为基准, 将障碍物的尺寸向外扩展, 将机器人看作一个质点; ③障碍物由任意栅格方块组成、数目有限, 并且在机器人移动过程中这些障碍物不会发生变化和移动。

收稿日期: 2013-12-17

基金项目: 广西制造系统与先进制造技术重点实验室项目(0842006-020-Z)

作者简介: 何少佳(1963—), 男, 博士, 研究员, 研究方向: 机电控制与自动化、智能机器人控制技术、电力电子与电力传动, hmhsj@sina.com。

引文格式: 何少佳, 史剑清, 王海坤. 基于改进蚁群粒子群算法的移动机器人路径规划[J]. 桂林理工大学学报, 2014, 34(4): 765-770.

图 1 是一个环境地图的例子,其中黑色栅格为障碍栅格,表示障碍物;白色栅格表示自由区域,可以供机器人移动; $S$  表示起始点; $G$  表示目标点。每一个栅格的长宽都为 1 个单位,任意两个栅格间的距离是它们中心点的连线距离,记作  $l$ :

$$l = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}。 \tag{1}$$

总路径长度为  $L$ :

$$L = \sum_{m=2}^n \sqrt{(x_m - x_{m-1})^2 + (y_m - y_{m-1})^2}。 \tag{2}$$

式中:  $(x_m, y_m)$  为路径点坐标信息;  $n$  为路径点个数。 $L$  亦为目标函数。

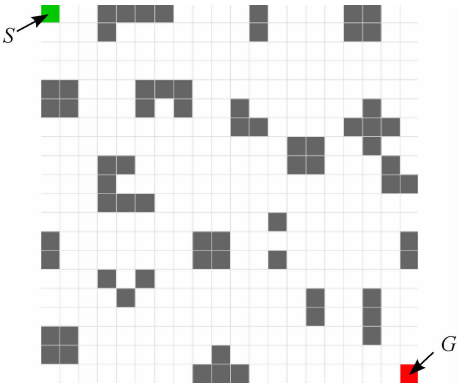


图 1 环境地图  
Fig. 1 Establishing environment map

## 2 粒子群算法的路径规划

利用 PSO 算法完成蚁群算法初始信息素的采集,采用粒子群的建模方式,迅速找出从起始点到目标点的路径,然后在这些路径上散布比周围更多的信息素,能给蚁群的寻优提供导向,使蚂蚁快速搜索出一条从起始点到目标点的路径。

如图 2 所示,首先把起始点和目标点连接起来,然后将其进行等分,图中虚线是在每个等分点作的垂线,虚线经过的白色栅格的编号作为粒子编码,粒子正是在这些编码中随机选择。如果选择出的相邻两个编码所代表的白色栅格之间的连线不穿过黑色栅格,那么就可以将其作为一条待选路径。图 3 为一条经规划得到的路径。

假设粒子总数为  $N$ ,  $\mathbf{Z}_i = (z_{i1}, z_{i2}, z_{i3}, z_{i4}, \dots, z_{iD})$  为第  $i$  个粒子 ( $i = 1, 2, 3, 4, \dots, N$ ) 的  $D$  维位置矢量,在每次迭代中粒子按式(3)更新速度、按式(4)更新位置:

$$v_{id}^{k+1} = wv_{id}^k + c_1r_1(p_{id} - z_{id}^k) + c_2r_2(p_{gd} - z_{id}^k); \tag{3}$$

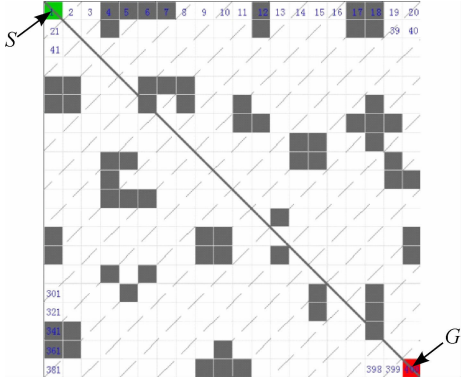


图 2 PSO 建模图  
Fig. 2 PSO modeling figure

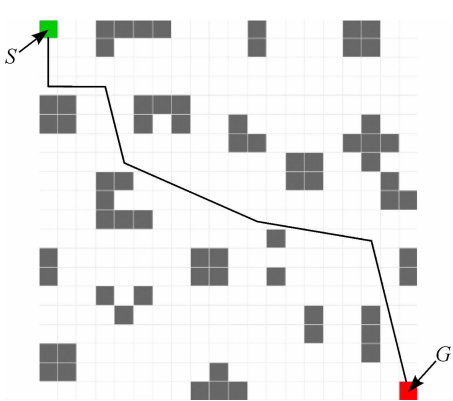


图 3 PSO 快速路径规划  
Fig. 3 PSO fast path planning

$$z_{id}^{k+1} = z_{id}^k + v_{id}^{k+1}。 \tag{4}$$

其中,  $i = 1, 2, \dots, N$ ;  $d = 1, 2, \dots, D$ ;  $k$  为迭代次数;  $w$  为惯性权重;  $c_1$  和  $c_2$  为学习因子;  $r_1$  和  $r_2$  是  $[0, 1]$  之间的相互独立、均匀分布的随机数;  $\mathbf{v}_i = (v_{i1}, v_{i2}, \dots, v_{iD})$  为粒子  $i$  的飞行速度,也是粒子移动的距离;  $\mathbf{p}_{id} = (p_{i1}, p_{i2}, \dots, p_{iD})$  为粒子迄今为止搜索到的最优位置;  $\mathbf{p}_{gd} = (p_{g1}, p_{g2}, \dots, p_{gD})$  为整个粒子群迄今为止搜索到的最优位置,  $\mathbf{p}_g = \min(p_{i1}, p_{i2}, \dots, p_{iD})$ 。

选择合适的适度值函数可以保证得到最优路径,以路径的最小值作为评价标准,选择适度值函数为

$$f = L。 \tag{5}$$

式中:  $L$  代表该粒子个体找出的路径中相邻序号间的直线距离之和,即式(2)。

- PSO 算法的路径规划按照以下步骤进行:
- ① 按照对图 2 的描述对环境模型进行处理;
  - ② 设置初始参数,种群规模  $N_z$ , 惯性权重  $w$ , 学习因子  $c_1$  和  $c_2$ , 最大迭代次数  $M_z$ , 速度最大值  $V_{\max}$ ;
  - ③ 随机生成粒子  $i$  的位置矢量  $\mathbf{Z}_i = (z_{i1}, z_{i2},$

$\dots, z_{id})$  和速度矢量  $\mathbf{v}_i = (v_{i1}, v_{i2}, \dots, v_{id})$ , 初始化粒子历史最优值  $p_i$  和全局最优值  $p_g$ ;

④ 计算每个粒子当前适应度值  $f(z_i)$ 。如果  $f(z_i) < f(p_i)$ , 则  $p_i = z_i, f(p_i) = f(z_i)$ ; 如果  $f(z_i) < f(p_g)$ , 则  $p_g = z_i$ ;

⑤ 根据式(3)、(4)更新粒子位置信息和速度信息;

⑥ 转到④, 直至达到最大迭代次数或满足精度要求。

使用 PSO 算法的优势在于它相对于蚁群算法来说所需时间很短, 加在蚁群算法之前不仅不会消耗太多时间、影响整体运行速率, 而且能为蚁群算法提供初期信息素反馈, 这样能减少蚁群算法的运行时间, 从而达到既减少搜索时间又能得到优化路径的效果。

### 3 改进蚁群粒子群算法融合

全局静态路径规划的蚁群粒子群算法的基本思想是充分利用这两种算法各自的优点, 优势互补, 快速找出最优路径。蚁群粒子群混合算法首先利用 PSO 找出一条初始路径, 然后将其转化为 ACO 的初始信息素信息, 为蚂蚁提供一定导向作用, 缩短 ACO 的初期搜索时间。

#### 3.1 路径点选择

由于采用的是栅格法建立地图模型, 所以蚂蚁每次搜索的范围是与其当前所在栅格相邻的 8 个栅格, 因此每次搜索计算量不会过大, 由此取消传统蚁群算法中的比例因子  $q_0$ , 直接采用轮盘赌法的方式在周围可选栅格中选取下一路径点。蚂蚁在  $t$  时刻位于栅格  $i$  时选择下一栅格  $j$  的转移概率为

$$P_{ij}(t) = \frac{\tau_{ij}^\alpha(t) \eta_{ij}^\beta(t)}{\sum_{j \in M} \tau_{ij}^\alpha(t) \eta_{ij}^\beta(t)}. \quad (6)$$

式中:  $P$  为选择下个路径点的概率;  $\tau_{ij}$  为信息素浓度;  $\eta_{ij}$  为栅格  $i$  和栅格  $j$  之间距离的倒数;  $\alpha$  为信息素的相对重要性;  $\beta$  为距离的相对重要性;  $M$  为下一步可选栅格集合。

蚂蚁在搜索过程中将走过的路径点记录下来, 在完成一个循环后依据目标函数来确定较优路径。对于陷入死锁的蚂蚁采用后退方式处理, 后退 2 步, 记录所在栅格信息, 将其设置为障碍栅格, 使以后的蚂蚁不再访问该栅格, 有效避免了进入

U 型弯道而无法走出的死循环。

#### 3.2 惯性优化

从仿真实验结果图 4a 可以看出, 混合算法虽然有效提高了时间效率, 但是依然存在路线折线过多、转折次数过多的问题。因此, 笔者采用惯性优化原理, 再次对混合算法进行改进。在简单环境下的效果对比: 图 4a 为普通蚁群粒子算法<sup>[5]</sup>寻优路径图, 路径点数 23, 转折次数 12, 路径长度 28.63; 图 4b 为本文算法寻优路径图, 路径点数 22, 转折次数 7, 路径长度 28.04。相比之下, 本文算法优化后的路径质量更高。

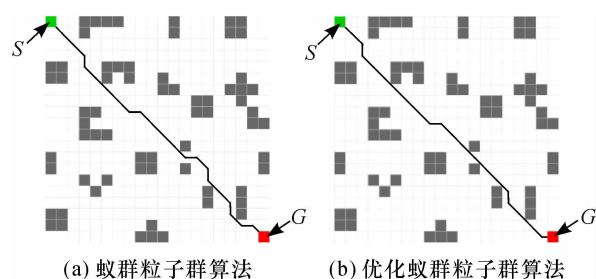


图 4 同一环境下两种算法比较

Fig. 4 Contrast between two algorithms under the same environment

惯性优化<sup>[6]</sup>的基本原理是: 从起始点开始, 遍历已获得路径上的每个路径点, 当路径点  $i$  与任意大于  $i$  的路径点满足惯性条件时, 将中间多余路径点删除, 添加优化后的路径点, 然后继续搜索满足惯性条件的路径点, 直至目标点。惯性条件是路径点  $i$  到路径点  $j$  朝  $\eta_{ij}'$  方向, 中间没有障碍物, 如果有障碍物则停止, 进入下一路径点寻找。假设  $G_i(x_i, y_i)$  为当前路径点,  $G_{i-1}(x_{i-1}, y_{i-1})$  为上一个路径点,  $G_{i+1}(x_{i+1}, y_{i+1})$  为下一个路径点(是  $G_i$  周围待选取的 8 个路径点之一),  $G_j(x_j, y_j)$  为目标点。由式(7)确定:

$$\eta_{ij}' = \begin{cases} \sqrt{(x_{i+1}' - x_j)^2 + (y_{i+1}' - y_j)^2}, & i+1 \in \text{惯性点}; \\ \sqrt{(x_{i+1} - x_j)^2 + (y_{i+1} - y_j)^2} + C, & i+1 \notin \text{惯性点}. \end{cases} \quad (7)$$

式中:  $C$  为常数, 惯性点是指与  $G_i, G_{i-1}$  在同一直线上的  $G_{i+1}$ , 需满足:

$$\begin{cases} x_{i+1} - x_i = x_i - x_{i-1}; \\ y_{i+1} - y_i = y_i - y_{i-1}. \end{cases} \quad (8)$$

如果  $G_{i+1}$  也属于惯性点, 那么用于计算  $s$  的  $(x_{i+1}', y_{i+1}')$  需要调整为

$$\begin{cases} x_{i+1}' = 3x_i - 2x_{i-1}; \\ y_{i+1}' = 3y_i - 2y_{i-1}. \end{cases} \quad (9)$$

调整的目的是为了突出惯性点,在一定范围内使惯性点的  $\eta_{ij}'$  最小。式(7)是经过优化的距离公式,用以寻找从路径点  $i$  到路径点  $j$  的最短路径,式(8)、(9)是惯性公式,用以删除不必要的转折点。惯性优化的步骤为:

- ①假设起始点为  $i$ , 终止点为  $j$ ;
  - ②按式(7)、(8)、(9)寻找一条到终止点  $j$  的最短路径;
  - ③判断是否有障碍物,如有障碍物返回②,如果没有障碍物,删除从  $i$  到  $j$  之间的路径点,加入优化后的路径点;
  - ④循环执行②、③步,直至满足终止条件。
- 具体过程如图 5 所示。

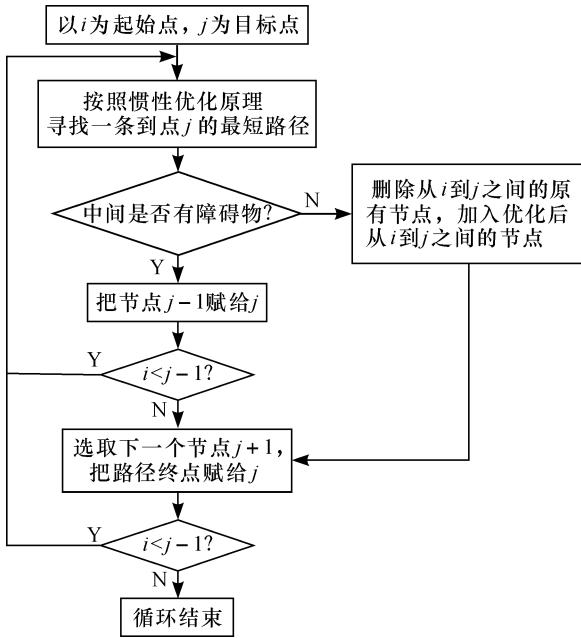


图5 惯性优化流程

Fig. 5 Inertia optimization flow chart

### 3.3 信息素的更新

传统的蚁群算法是对所有蚂蚁走过的路径都进行信息素更新,这种机制有一定的缺陷,会减慢算法的收敛速度、降低搜索效率。笔者采取的机制是:对蚁群中的周游最优蚂蚁和全局最优蚂蚁的路径信息进行信息素更新,从而加强了反馈效果、增加解的多样性,减小陷入局部最优的概率。更新的机制为

$$\tau_{ij}(t+1) = \rho\tau_{ij}(t) + \Delta\tau_{ij}. \quad (10)$$

式中:  $\rho$  表示信息素的挥发系数;  $\Delta\tau_{ij}$  表示本次循环的信息素增量期,其表达式为

$$\Delta\tau_{ij} = (1 - \rho) \times \left( \frac{a_k}{L_c} + \frac{b_k}{L_w} \right) \times \frac{Q}{a_k + b_k}. \quad (11)$$

其中:  $L_c$  为周游最优蚂蚁走过的路径长度;  $L_w$  为全局最优蚂蚁走过的路径长度;  $a_k$  和  $b_k$  为整型变量,分别代表周游最优蚂蚁和全局最优蚂蚁更新信息素的权重;  $Q$  为常数。

信息素的挥发系数  $\rho$  对在复杂地图中搜索有着重要作用:如果  $\rho$  过大,会降低算法收敛速度;如果  $\rho$  过小,则易陷入局部最优。笔者采用动态调整方法确定该参数:

$$\rho(t) = \frac{L_{mid} - L_{min}}{L_{max} - L_{min}}. \quad (12)$$

式中,  $L_{max}$  表示最长优化路径;  $L_{mid}$  表示平均优化路径;  $L_{min}$  表示最短优化路径。

### 3.4 改进蚁群算法与粒子群算法的融合

本文提出的改进混合算法是利用 PSO 算法产生蚁群算法的初始信息素信息。首先根据环境地图信息得到信息素的初始值  $\tau_{ij}^s$ ,然后将 PSO 算法得到的最优路径转换为信息素的增强变量  $\Delta\tau_{ij}$ ,最后对初始信息素进行再分配,计算得到新的初始信息素信息  $\tau_{ij}$ :

$$\tau_{ij} = \tau_{ij}^s + \Delta\tau_{ij}. \quad (13)$$

基于改进蚁群粒子群算法的全局静态路径规划的主要步骤可描述如下:

- ① 设置蚂蚁数量  $N_a$ 、最大迭代次数  $M_a$ ,利用 PSO 算法得到的最优路径,根据式(13)初始化信息素信息,并将全部蚂蚁置于起始点;
- ② 启动蚁群,每只蚂蚁按式(6)的概率,使用轮盘赌法进行路径点选择,并记录路径点信息;
- ③ 重复②,直至所有蚂蚁达到目标点;
- ④ 对得到的路径进行惯性优化,保存优化后的路径信息;
- ⑤ 对周游最优蚂蚁和全局最优蚂蚁的路径信息根据式(10)、(11)更新各个路径上的信息素;
- ⑥ 转跳至②,直至搜索路径不再变化或达到最大迭代次数;
- ⑦ 输出最优路径。

4 仿真分析

为了验证算法的正确性和可行性，本文在 PC 平台上进行仿真实验，硬件环境为主频是 2.3 GHz 的 CORE I5 CPU，内存 2 GB，软件环境为 Windows 7 操作系统和 Visual Studio 2010 编译器。粒子群算法参数为：种群规模  $N_z = 25$ ；最大迭代次数  $M_z = 50$ ；学习因子  $c_1 = c_2 = 1.45$ ；惯性权重  $w$  从 0.8 随迭代次数线性递减到 0.4；速度最大值  $V_{\max} = 10$ 。蚁群算法参数为：蚂蚁数量  $N_a = 20$ ；最大迭代次数  $M_a = 100$ ； $\rho = 0.5$ ； $\alpha = 1.5$ ； $\beta = 2.0$ ；常数  $Q = 100$ 。

实验中，重点是把本文提出的算法与文献 [7] 提出的算法以及最大 - 最小蚁群算法进行对比。为方便起见，称本文算法为 IAC + PSO (improved ant colony + particle swarm optimization)，称文献 [7] 算法为 ACO + PSO (ant colony algorithm + particle swarm optimization)，称最大 - 最小蚁群算法为 MAS (Max-min Ant System)。3 种算法在同一个  $40 \times 40$  的地图上进行仿真比较，仿真对比

结果如图 6、图 7 所示。

由图 6 可知，IAC + PSO 算法路径长度 58.08，ACO + PSO 算法路径长度 62.01，MAS 算法路径长度 72.91。首先，本文算法在长度上比 ACO + PSO 算法减少了 6.3%，比 MAS 算法减少了 20.3%；其次，ACO + PSO 算法和 MAS 算法明显存在较多弯路，转折点分别达到 27 个和 35 个，而本文算法只有 11 个，在获取的路径质量上明显优于 ACO + PSO 算法和 MAS 算法。同样，在图 4 中也能得出类似结论。

图 7 为 3 种算法的收敛图：横轴代表迭代次数；纵轴代表路径长度；线 1 代表每次迭代的平均路径；线 2 代表每次迭代的最小路径。MAS 算法在 85 代左右开始收敛，ACO + PSO 算法在 60 代左右开始收敛，而本文算法在 35 代左右就开始收敛。这并不表示本文算法收敛过早，而是在蚁群算法开始前就由粒子群算法提供了相当可靠的信息素信息，使得前期搜索加快，当迭代后的最优路径与平均路径相差不大时，通过调整挥发系数而加速收敛，达到了缩短运行时间的目的。

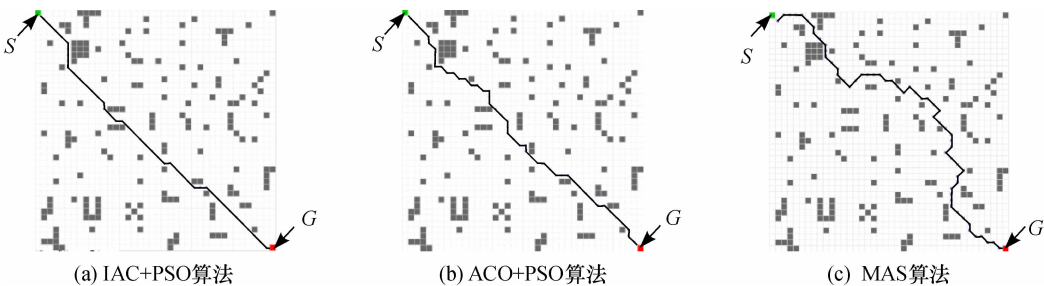


图 6 算法效果对比  
Fig.6 Contrast of algorithm effect

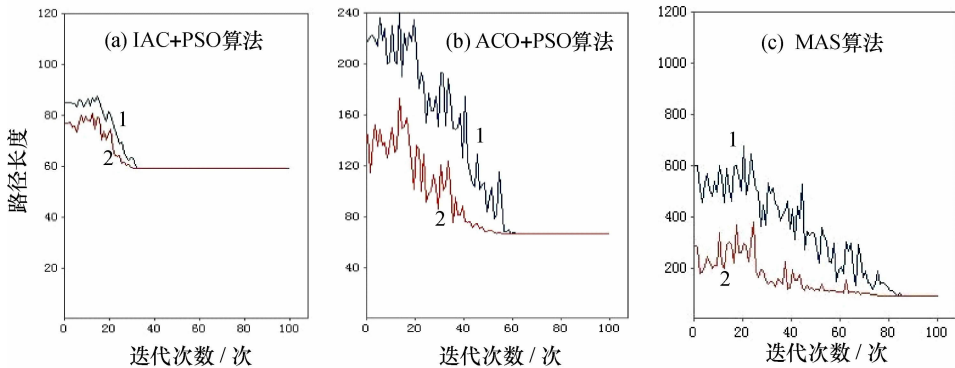


图 7 算法收敛对比  
Fig.7 Contrast of algorithm convergence

表 1 为 IAC + PSO、ACO + PSO、MAS、ACO 算法在 3 种栅格地图下分别进行 10 次实验数据的平均值。可以看出,本文算法相对于其他 3 种算法在路径长度、转折次数以及运算时间上都有优势;对于一般环境下的移动机器人路径规划,能以很大概率搜索到全局最优路径,并且收敛速度快;对于复杂环境下的路径规划,算法收敛速度有所减慢,但仍然优于其他算法,得到的最优路径长度及路径质量明显提高,并且地图越大优势越明显。

表 1 不同环境下各种算法对比

Table 1 Contrast of various algorithms under different environments

地图大小	算法	路径长度	转折次数/次	时间/s
20 × 20	IAC + PSO	28.46	7	0.125
	ACO + PSO	28.78	13	0.125
	MAS	31.64	17	0.160
	ACO	33.01	23	0.360
40 × 40	IAC + PSO	58.08	11	0.319
	ACO + PSO	61.01	27	0.314
	MAS	68.5	36	1.030
	ACO	70.58	41	1.256
60 × 60	IAC + PSO	90.23	17	1.027
	ACO + PSO	94.81	38	1.006
	MAS	105.74	54	3.476
	ACO	111.64	67	4.734

5 结束语

本文针对移动机器人的路径规划问题,将粒子群算法与蚁群算法进行了有效融合,利用粒子群算法为蚁群算法提供初始信息素信息,缩短蚁群算法前期搜索时间,提高搜索效率,并采用惯性原理加以改进,明显改善路径质量,减少移动机器人在运行过程中的转折次数。在减少路径长度、提高路径质量方面,仿真结果证明了本文算法的有效性和可行性。

参考文献:

[1] 任春明, 张建勋. 基于优化蚁群算法的机器人路径规划 [J]. 计算机工程, 2008, 34 (15): 1-3.

[2] 邓皓, 谢晓兰, 程小辉. 多处理机调度问题的蚁群优化算法 [J]. 桂林理工大学学报, 2013, 33 (2): 329-332.

[3] 周驰, 高海兵, 高亮, 等. 粒子群优化算法 [J]. 计算机应用研究, 2003 (12): 7-11.

[4] 蒙正中. 一种改进的混合粒子群优化算法 [J]. 桂林工学院学报, 2009, 29 (3): 411-415.

[5] 王宪, 王伟, 宋书林, 等. 基于蚁群粒子群融合的机器人路径规划算法 [J]. 计算机系统应用, 2011, 20(9): 98-102.

[6] 何少佳, 刘子扬. 基于惯性蚁群算法的机器人路径规划 [J]. 计算机工程与应用, 2012, 48 (15): 245-248.

[7] 邓高峰, 张雪萍, 刘彦萍. 一种障碍环境下机器人路径规划的蚁群粒子群算法 [J]. 控制理论与应用, 2009, 26 (8): 879-883.

Path planning for mobile robot based on improved ant colony and particle swarm optimization

HE Shao-jia, SHI Jian-qing, WANG Hai-kun

(a. Guangxi Key Laboratory of Manufacturing System & Advanced Manufacturing Technology;

b. College of Electromechanical Engineering, Guilin University of Electronic Technology, Guilin 541004, China)

**Abstract:** Ant colony optimization for mobile robot under global static grid environment is defective in long calculation time and low search efficiency, at the same time, the optimal path has many bends. In order to solve these problems, an improved ant colony and particle swarm optimization algorithm is presentsed. First, particle swarm optimization is used to get the initial pheromone of ant colony algorithm. Second, the optimal path is searched by ant colony algorithm. Then inertia principle is used to tra-verse all the nodes on the path, deleting intermediate node when there is no obstacle existing between the two nodes and changing optimum path. The simulation results show that this method is better than traditional ant colony algorithm and other algorithms. This method can reduce the number of iterations, improve the search efficiency, reduce the number of turning, shorten path length and improve the quality of the path effectively.

**Key words:** particle swarm optimization; ant colony optimization; path planning; mobile robot