实时环境下的问题求解*

陈 正 张 钹

(清华大学计算机科学与技术系 北京 100084)
(清华大学智能技术与系统国家重点实验室 北京 100084)
E-mail: snowchen@263.net

摘要 实时环境下的问题求解是近年来规划问题研究感兴趣的话题.在讨论了传统规划算法的不足之后,引入了在实时环境下求解问题的方法——任意时间算法.任意时间算法可以合理分配时间资源,保证系统最佳的输出性能;同时,任意时间算法可以在任意时刻中断,并输出当时相对最优可行解.遗传算法具有任意时间算法的特性,在介绍了同其他搜索算法的不同之处后,通过实验得出利用随机搜索技术和知识指导相结合的方法,可以较好地处理实时规划问题.最后给出结论,并且简单地讨论了实时规划算法求解问题的策略,同时讨论了今后的发展方向.

关键词 规划,任意时间算法,遗传算法,算子.
中图法分类号 TP18

传统的规划是从具体行为到输出动作之间的一个映射,它认为外界环境的变化很小,同时外界环境对于规划系统来说是完全可以预知的一个系统.GPS(general problem solve)可以说是最初的规划系统,GPS对规划问题作了一些假设,它假设规划是由动作组成的序列,规划输出的行为都是可以预言的,这种假设现在我们称为"传统规划"(Classical Planning).虽然现在有不少人对这种假设提出了疑问,但是它确实能解决不少问题.[1]

上面的假设虽然能够解决一些规划方面的问题,但是大多情况下只适用于离线规划环境;当机器人处于易变的、不完全已知的环境时,必须不断地感知周围环境,作出相应的反应.在这种条件下,显然不可能再把机器人的行为认为是可以预知的动作序列了.因此出现了不少考虑环境对系统的影响的在线规划算法,如:Agre和Chapman[2],Georgeff和Lansky[3],Rosenschein和kaelbling[4],Sanborn和Hendler[5],Schoppers[6],Firby[7]等等.如果把这些方法简单化,就可以认为它们是在接受外界的刺激(Stimulus)下所作出的反应(Response).[1]

传统规划认为一个规划器具有足够的时间寻找最佳结果.但是,如果这个时间对于一个实际问题来讲太长的话,就应该选择在找到最佳结果前,得到每个时刻的局部最优结果.这就要求我们能在任意时刻中止规划算法,并且给出当时的最优结果,具有这种特点的算法我们称为"任意时间算法"(Anytime Algorithm).[8]

此外,现在规划研究的一种趋势是,把传统的规划方法和某种响应能力(也就是能够处理动态和突发事件的能力)相结合,出现了"分层规划"等算法,这对解决机器人的规划问题很有意义.

1 任意时间算法

依赖于时间的算法就是要研究在给定的时间内如何对一些事件作出最好的响应.在现实的规划问题中,存在着很多资源限制,如时间资源、机器的硬件资源等等因素,如何在给定的资源内对系统作出最佳的规划成为规划问题研究的关键.而且,现实的规划问题处于易变的环境中,这要求算法能够适应环境的变化.

* 本文研究得到国家自然科学基金和国家863高科技项目基金资助.作者陈正,1972年生,博士,主要研究领域为人工智能,实时环境问题求解,规划,调度.张钹,1935年生,教授,博士生导师,中国科学院院士,主要研究领域为计算机应用技术,人工智能.

本文通讯联系人:陈正,北京 100084,清华大学计算机科学与技术系

本文1997-10-15收到原稿,1998-01-09收到修改稿

近年来提出了很多关于实时环境下问题求解的方案,其中任意时间算法具有较突出的特点.给出一定的输入数据,同时分配一定的时间和其他资源,任意时间算法将给出各种性能的输出结果.通过分析给定的输入的类型、给定的时间以及输出结果的质量,可以得到一定具有预计性的算法模型,根据这个模型,可以按照算法各个部分的重要性来分配时间资源,以求得在最短的时间内给出最优输出结果.任意时间算法提出的解决方案能够较好地适应外部以及内部的不确定性因素,满足实时环境规划的需要.

对于每个需要作出响应的时间 c,我们假设机器人对 c 作出任何响应都有一个决定过程,每个决定过程我们都分配一些时间,返回的是输出响应的最好的结果 $\varepsilon\in\mathscr{R}$.我们定义 γ 是从 $C\times\mathscr{R}^+$ 到 $\mathscr{R}$ 的映射,因此,对于每个 $c\in C$ 和正实数 δ,$\gamma(c,\delta)=\varepsilon$ 表示机器人在给定的时间 δ 内对 c 作出的最好的响应类型是 ε,在描述机器人对付依赖于时间的规划问题的时候,我们感兴趣的是组合的功能 $utility(c,\gamma(c,\delta))$,它是对给定时间内所输出结果的一个评价.

下面,我们介绍5种类型和时间相关的规划算法的输出性能.我们设 $(x,y)=utilit(x,\gamma(x,y))$.[8]

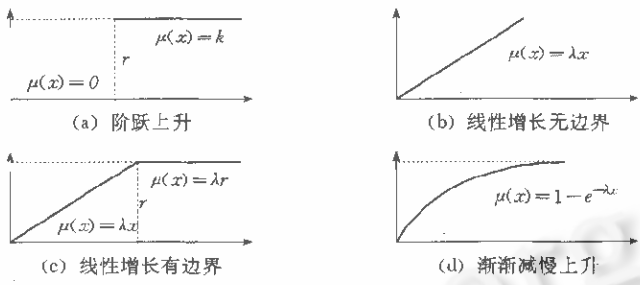
(1) 单调上升性:$\forall\ c\in C,\forall\ \delta,\varepsilon\in R^+,\mu(c,\delta)\leqslant\mu(c,\delta+\varepsilon)$;

(2) 阶跃上升(如图1(a)所示):$\forall\ c\in C,\exists\ r,k\in R^+,\mu(c,\delta)=\begin{cases}0, & \text{for } 0\leqslant\delta<r\\ k, & \text{for } r\leqslant\delta<\infty\end{cases}$;

(3) 线性增长无边界(如图1(b)所示):$\forall\ c\in C,\exists\ \lambda\in R^+,\mu(c,\delta)=\lambda*\delta$;

(4) 线性增长有边界(如图1(c)所示):$\forall\ c\in C,\exists\ r,\lambda\in R^-,\mu(c,\delta)=\begin{cases}\lambda*\delta, & 0\leqslant\delta<r\\ \lambda*r, & r\leqslant\delta<\infty\end{cases}$;

(5) 渐渐减慢上升速度(如图1(d)所示):$\forall\ c\in C,\exists\ f,\mu\in(c,t)=f(t)$,其中 f 是单调增函数,而且 $\forall\ x,y\in R^+$,$f'(x)$ 和 $f'(y)$ 存在,$x<y\Rightarrow(f'(y)\leqslant f'(x))$.



(a) 阶跃上升 (b) 线性增长无边界

(c) 线性增长有边界 (d) 渐渐减慢上升

图1

现在许多规划系统都符合第2种类型,无论提供给多少时间,这些系统都需要固定的时间来完成相同的任务,给的时间如果较多,它们也仅仅是用来规划其他事情或者什么也不做.如果给定的时间不够,那么,系统将执行缺省的动作.如果当规划的事件可以预测的话,或者规划所需要的时间很短,那么,第2种方法可以得到较好的结果.第3、4种方法是第5种方法的特例,它们在实际应用中有较好的性能,可以随着给定时间的增加而输出最优的规划结果.

任意时间算法的特点在于:(1) 它们可以在任意时刻中断,而不需要附加的负担;(2) 它们可以在任意时刻中断,并返回一些结果;(3) 返回的结果随着时间的增长,性能也在改善.而正是(2)、(3)两个特点使得这个不同于其他传统的规划算法,为了有所区别,我们称之为"任意时间算法"(Anytime Algorithms).[8]

2 遗传算法(GA)

通过上面的描述不难看出,任意时间算法最大的特点就在于它合理地利用了提供给它的资源,使得这个算法可以适应实际环境的需求,在需要的时刻可以输出当时的最优解.任意时间算法不仅可以不断优化算法输出的结果,使性能随着时间的增加而增长;而且它可以对各个子系统合理地分配资源,达到系统的最优输出.不过,任意时间算法给出的是实时算法的一个框架,适用于各种实时计算,但是对于不同的实际问题,还需要在任意时间算法理论指导下采用相应的解决策略.在规划方面出现了许多适用于实时环境的需求的算法,它们都遵循任

意时间算法提出的框架,其中"遗传算法"可以说具有较突出的特点.

2.1 遗传算法介绍

遗传算法是一种基于自然选择和自然遗传学机制的搜索方法.它通过自然选择中的"优胜劣汰"的策略在每次搜索中生成一些新的串结构,淘汰较差的结构,对于最适合的结构加以保留,每次都是利用随机技术进行交叉生成下一代.由于利用了这种随机技术,使得遗传算法不同于其他传统的搜索方法,但它又不仅仅是随机搜索,因为它还有效地利用了许多历史信息,使得在每代生成中都朝着最好的方向前进.

传统的搜索方法与遗传算法比较起来,最大的区别就在于它们的"鲁棒性".传统的搜索算法大多只适用于某些特定的环境,如果环境发生变化,算法就不能很好地适应变化来求出较优解;而对于遗传算法,采用了随机技术,使得它可以运行于任何复杂环境中,而且能够通过不断优化自身的解来逼近最优解.在介绍遗传算法前,我们先来看看传统的搜索算法的特点.

传统的搜索算法大体上可以分为3类:[9]

(1) 基于微分概念的搜索.整个搜索过程就是对搜索空间进行微分,求其导数为0的点,或者利用Hill爬山法求出其局部最优值.这种方法广泛适用于许多场合,但是,如果在搜索空间中存在多个峰值点时,就不能很好地找出整个搜索空间的最优解,只能在找到局部最优解后随机寻找新的起始点,继续开始新的搜索.

(2) 枚举法.这种方法适合于搜索空间是有限的离散数据集合,缺乏有效性,适用范围也比较小.当搜索空间加大时,它就显得不能适应实际环境,也就显示出其较差的"鲁棒性".

(3) 随机搜索.遗传算法是利用"随机选择"作为一个工具来指导算法在一个参数编码串组成的空间中进行搜索.它不是单纯的随机搜索,而仅仅是利用随机选择作为搜索手段,将系统的输出作为评价,使得系统的输出性能不断提高,求得最优解.

评价一个搜索算法的准则在于,算法是否能够在给定的时间与资源范围内给出较优的解决方案.仅仅输出最优解并不能表明这个算法一定是最好的,在实际环境中,我们更重视的是能够在给定的资源(包括时间资源)内给出最好的响应.而遗传算法正是具有这方面的特点.它存在着不同于其他算法的特性,主要表现为以下4个方面:

(1) 搜索空间是参数集合的编码,而不是参数本身.

(2) GAs的搜索空间是一群例子,而不是一个实例.这样可以避免只搜索到局部极值,而丢失全局最优.

(3) GAs利用输出结果的性能来指导算法朝着最优的方向前进,而不利用一些特殊的知识.这样保证了系统的鲁棒性,可以适用于任何环境.

(4) 利用随机工具作为搜索手段.

以上4个特性使得遗传算法可以在搜索中保持良好的鲁棒性,适用于任何实际环境.

2.2 传统GA算法的算子

遗传算法在编码串空间进行搜索,搜索过程中使用算子来对串进行处理,得到新的串集合.在GA算法中先寻找一个基本可行解,然后对这个基本可行解进行进化(优化),进化的方法(算子)基本上来说有以下3种:①selection(选择),②Crossover(交叉),③mutation(突变).

GAs通过以上3个算子在串编码空间进行搜索,根据系统输出的结果性能对编码串进行评价,利用随机技术进行Reproduction,同时利用随机技术选择交叉点,对编码串进行交叉,得到新的下一代编码串,同时在这个过程中允许在突变概率内对编码串进行突变.

2.3 遗传算法的特点

(1) 遗传算法具有任意时间算法的特点,可以在任意时刻中断,并输出当时的基本较优可行解(当然很多遗传算法输出的解并不符合要求,但是,经过适当改进可以满足实际问题的要求).随着规划时间的增加,系统的性能在增长.

(2) 遗传算法采用随机技术,因此可以避免传统搜索方法的局部最优的限制,找到全局最优解.

(3) 由于其收敛取决于评价以及知识的应用,因此,只有与一些传统方法相结合才能发挥其最大效用.

(4) 传统算法则需要较多的知识支持,当知识不明显或者缺乏时,就无法给出解答方法;而遗传算法只要给出评价方法,就可以进行规划,受外部条件的限制较少.

3 实际问题求解

3.1 实际应用中简单遗传算法(SGA)存在的问题

遗传算法最重要的一个特点就是它不同于其他规划算法,具有很强的鲁棒性,可以适用于各种易变的环境,而且它能够不断优化输出的结构,取得较优的输出解.但是,SGA仅仅依靠系统的输出作为评价,而不利用系统的一些有用的经验知识或其他有效知识,虽然这样提高了系统的鲁棒性,但是却使得整个系统的搜索速度比较缓慢,收敛缓慢,而且还有可能找不到搜索解.尽管如此,遗传算法给出的观点仍值得我们作为指导依据.它采用了自然界中"优胜劣汰"的原则,符合实时算法的需求,可以不断优化系统的输出,能够在任意时刻中断并输出较优解,因此,适当地对遗传算法进行改造,就可以得到一个较好的实时搜索算法.

3.2 实际应用

遗传算法广泛应用于许多搜索领域,下面以其在"路径搜索"算法方面的应用作为例子,来看看遗传算法在实际应用中存在的问题.

传统的搜索方法虽然存在自身的缺陷,不适用于易变的外界环境,但是它也存在着很多优点,它可以很好地利用许多已知的知识作为搜索指导,较快地得到系统的解.因此,可以合理地对遗传算法进行改造,在遗传算法的基础上增加一些知识算子来指导搜索的方向,加快搜索的速度,来弥补简单遗传算法中随机搜索的缺陷.

在规划中,遗传算法的主体都是相同的,还是利用随机技术作为基本依据,但是由于加入了知识的指导,使得系统在每代中都朝着优化的方向前进.在SGA中采用的方法基本上都是先寻找一个基本可行解,然后对这个基本可行解进行进化(优化).进化通过事先选定的算子和评价方法进行操作.进化的方法基本上来说有以下3种:复制、交叉、突变.然而在实际问题解决中,由于不同问题采用的知识表示方法不同,因此在算子的实际应用中也存在较大的差别,而且评价方法也不同,所以对于不同问题,上面3种算子也要进行相应的改变以适应实际需求.下面,我们以"路径规划"算法中的实际解决方案为例,来看看实际知识是如何应用于遗传算法中的.

路径规划要解决的问题是:给定一个具有障碍的二维空间,给定起点和终点,寻找一条无障碍的最短路径.

· 数据表示方法

在路径规划中,实际路径采用的表示方法是采用路径中的一系列关键点来表示实际路径,$(s,d_1,d_2,\ldots,d_n,e)$就代表一条实际路径,其中s为起点,e为终点,d_i为路径中的关键点.

· 评价方法

(1) 与障碍物相交的长度(越短越好);

(2) 与障碍物相交的次数(越少越好);

(3) 路径长度(越短越好).

其中优先级逐渐降低,力求先找出可行解,然后再寻求最优路径.

· 算子

(1) 交叉算子:采用的是随机在两条路径中选择一个交叉点,然后对这两条路径进行交叉,得到两条新的路径,如图2所示(选择3作为交叉点).

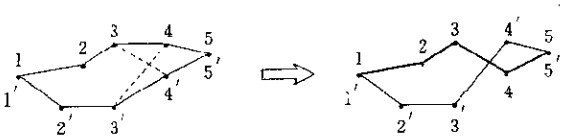


图2 交叉算子

(2) 突变:突变有以下几种:①随机改变关键点的位置,②加点,③减点,也就是随机抽取一个点,在其位置上改变、加入或者删除一个点.改变/加入的点的位置也是随机生成的.如图3所示,改变/加/减点位置选择在3.

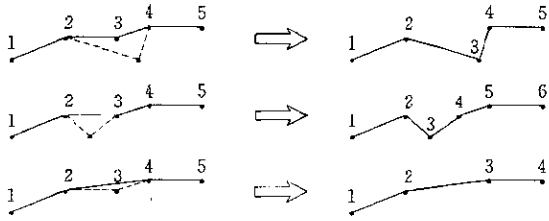


图3　突变算子(改变/加/减点)

综上所述不难看出,传统的GA对解决路径规划问题采用的就是随机的方法与评价策略相结合的方法.这种方法经过多代的选择后可以得到较优的解决方案,但是如果障碍物较多,那么实际搜索时间就会变得很长,甚至找不出解答方案,因为随机搜索寻找最优解毕竟速度比较慢,而且可以说逼近最优的速度完全取决于评价策略.同时,由于缺乏实际知识的指导,因而随机方法存在一定的适用性.

针对以上问题,适当地改进算子,加入实际知识的指导,增加了以下算子.

·躲避障碍的交叉

在路径规划中,需要寻求的是一条无障碍的最短路径.在GA算法中先是生成一些基本可行解,然后对这些基本可行解进行优化,在GA算法的最开始,这个基本可行解是采用随机的方法生成,因此不可避免地存在着与障碍物的交遇,GA在每代生成中试图去除这些不可行部分,同时达到路径长度最短.利用路径规划算法的特点,我们在GA算法的算子中进行适当的改进,使得每次操作后路径都尽可能地避开障碍物,与障碍物的交叉尽量减少,从而加快最优解的寻找.

具体采用的算子操作如下:对一条实际的路径$(s,d_1,d_2,\ldots,d_n,e)$,如果其中d_i-d_j与障碍物相交,那么"躲避障碍物的算子"则会处理这段不合理解,代之以其他的关键点.其主要的指导思想就是利用平时人们搜寻路径时,遇到障碍物就争取避开这条原则,力求使新生成的路径与障碍物的相交点尽可能少,具体采用的方法如图4所示.在增加新关键点策略方面,可以有以下两种:(1)寻找一个新关键点来代替原有的不合理关键点;(2)寻找几个新关键点来代替原有的不合理关键点.

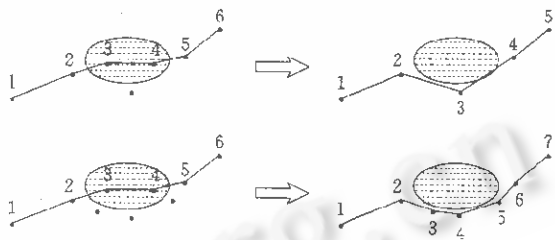


图4　躲避障碍算子

·距离最近交叉

在路径规划中除了需要躲避障碍物之外,另外一个重要的条件就是要求路径长度最短.在交叉中有可能使两条生成的路径的长度都增加,因此,适当地选择交叉点,使交叉后路径的长度尽量变短,也是要采用的一种策略.考虑到路径长度的因素,因此,在交叉时我们选择两点距离较短的点作为交叉选择点.

假设两条路径分别为路径1:$(s,d_1,d_2,\ldots,d_n,e)$与路径2:$(s,d'_1,d'_2,\ldots,d'_m,e)$;计算各个点之间的距离,找出距离最短的点,假设为$i=\{j\mid\min(|d_1,d'_2|-|d_1,d_2|,|d'_1,d_2|-|d'_1,d'_2|,\ldots,|d_j,d'_{i+1}|-|d_j,d_{j+1}|-|d'_j,d_{i+1}|-|d'_j,d'_{j+1}|,\ldots)\}$,则选择$d_i$作为交叉点,交叉路径1和路径2,得到新的路径1':$(s,d_1,\ldots,d_i,d_{i+1},\ldots,d_m,e)$与路径2':$(s,d'_1,\ldots,d'_i,d'_{i+1},\ldots,d_n,e)$,如图5所示.

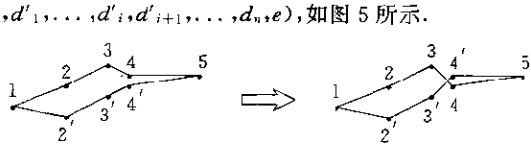


图5　距离最近交叉算子

· 对路径进行优化

在由遗传算法交叉生成的路径中，不可避免地存在着很多自圈路径，消除这些自圈，得到实际的路径才是系统最终的需要. 首先需要解决的是把由关键点组成的路径转换成由图中每个实际点组成的实际路径，对于一条从(x_1,y_1)到(x_2,y_2)的路径(图中虚线表示的路径)可以转换成图中实际表示的路径，也就是把每个小方格看成是一个点，如图6所示.

实现算法如下：

$\Delta x=\dfrac{x_2-x_1}{|y_2-y_1|+|x_2-x_1|}$

$x=x_1$

$y=y_1$

$i=0$

while $(i<|x_2-x_1|+|y_2-y_1|)$ {

$y=y_1+\dfrac{y_2-y_1}{|x_2-x_1|+|y_2-y_1|+1}\cdot i$ if $(x_1==x_2)$

$y=y_1+\dfrac{y_2-y_1}{x_2-x_1}(x-x_1)$ if $(x_1<>x_2)$

则$([x],[y])$为其下一步的坐标，其中$[x]$与$[y]$是x,y四舍五入的结果

$x=x+\Delta x$

}

图6 优化路径

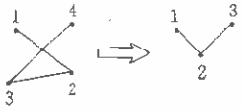


图7 去除重复路径

其次就是对交叉求出的较优解进行优化，优化的方法也就是对整个路径先求出其“精细路径”，去除其中重复的路径. 最后得到最短的一条路径，如图7所示.

除了上面提到的一些利用知识的算子外，我们还引入了一些相关的技术，与遗传算法相结合来共同求解问题. 对于规划空间，我们先在较粗粒度上进行搜索，得到初步的可行解后再对规划空间进行细化，求得准确的较优可行解.

同时，我们也引入了“分区域规划”的概念，利用知识对规划的空间进行区域划分，对划分后的区域分别进行规划(利用遗传算法)，然后合并结果，得到较优的可行解.

在路径规划算法中，我们先较粗地划分规划的整个区域，找出一些基本可行解，然后对区域进行划分，得到哪些区域是路径比较集中的，哪些区域是障碍物较多、路径不集中的，然后再在细的区域粒度下进行规划. 利用分层规划的概念后，可以使系统搜索的速度加快.

3.3 总结知识在实时求解问题中的应用

通过上面的实验不难看出，单纯利用遗传算法对问题进行求解既存在优点，也存在不足之处. 虽然单纯地利用随机技术可以提高系统的“鲁棒性”，但是缺乏相关知识的指导，使得系统搜索的方向比较盲目，单纯的随机搜索使得系统趋于稳定的速度大大降低. 适当地引入经验知识做指导，可以大大加快系统的搜索速度，同时又利用遗传算法的随机选择作为技术，避免了丢失全局最优解，因此，分层规划越来越受到大家的重视.

但是也不容忽视的是，引入的知识是否合适，对系统的性能的影响是很大的，不合适的知识不仅不会对系统的搜索速度起到什么正面作用，而且还有可能把搜索带入歧途，导致系统丢失最优解. 而且引入知识后，系统在处理知识方面不可避免地需要投入适当的开销，因此，需要平衡其利弊来考虑引入哪方面的知识. 我们在实验中发现，在路径规划系统中最花费时间的部分就是对路径的评价. 引入不同的知识评价的时间不同. 只有合理的利用知识和遗传算法相结合，才能更好地取得搜索的结果.

3.4 实 验

根据文章中提到的方法，我们在IMB-PC Pentium133机器上实现了路径规划算法，程序采用Microsoft VC++4.2开发，在Windows95下运行，以下的实验数据均是在此环境下得到的.

在路径规划程序中,我们在 300 * 200(pixel)的地图上进行规划,路径障碍物可随机或人工生成,初始化随机路径数设为 100,路径中关键点个数设为 10,随机突变概率为 5%.评价方法采用"路径与障碍物相交长度"以及"路径实际长度"作为评价.在规划的初期以"与障碍物相交长度"评价为主,到了规划后期,路径趋于合理,与障碍物相交的长度渐渐变小,因此评价以"路径最短"为主.我们分别通过以下几种途径对遗传算法进行测试:

(1) 采用基本随机算法,不加入任何多余的知识信息;

(2) 改变初始路径个数以及关键点个数;

(3) 引入相应的知识信息作为指导.

从实验中不难看出,基本的遗传算法虽然具有很强的鲁棒性,可以适应外界环境,但是由于缺乏相应的知识作为指导,容易陷入迷途,导致需要较长的时间才能找到较优解,而且如果系统初始的人口规模不大,还容易使系统稳定在某些解上,而这些解并不满足系统的要求.在路径规划算法中,如果在障碍物的数目相对来说并不多的情况下,基本的遗传算法可以较好地找出可行最优解,但是,随着障碍物的增加,基本遗传算法的寻找速度就大大减低.下面的例子就稳定在一个不适宜的解上,但是如果加大人口规模,系统就可跳出这个相对稳定解,有可能寻找到较优可行解.系统运行了 78 代后,基本稳定在这个解上,与障碍物相交长度最短为 37,如图 8 所示.

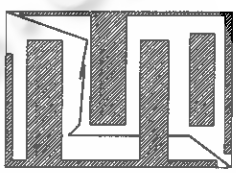
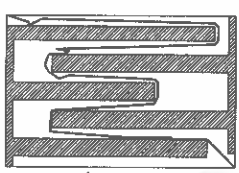


图8　基本遗传算法

在引入了相应知识指导后,系统性能大大提高.我们引入的知识基本观点在于两个方面:一方面是尽量避免与障碍物相交,也就是上面提到的避免碰撞算子,另一方面也引入了相应知识使得路径长度尽量减短,例如:交叉中有意识选择交叉点、优化相应路径等方法.从实验中我们看出,并不是所有知识对系统都能起到效果,而且具体对系统的影响也不同.不同知识适用的方面也各不相同,在系统开始时,避免碰撞算子起到了显著的作用,到了系统运行的后期,路径较短算子开始起作用.同时,还需注意到的是加入知识算子后加大了系统的负载,无论从时间还是空间复杂度上来说都比基本遗传算法大.因此不可盲目地引入知识,只有选择适合系统的知识,才能发挥其最大的效果.如何适当选择知识,处理知识与复杂度之间的平衡关系也是需要进一步研究的问题.采用以上方法,我们进行了相应实验,实验结果如图 9(a)、(b)所示.实验 1 数据见表 1.



(a) 实验1

(a) 实验2

图9　加入知识(避障算子)后规划结果

表 1

n	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
N	5	71	80	77	83	46	73	19	66	80	20	59	49	69	36	73	43	31	41	30
C	599	562	109	107	79	81	81	81	85	85	64	93	94	74	49	97	96	101	99	115
n	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40
N	28	18	32	40	55	80	60	75	34	86	25	29	44	23	86	0	44	16	57	61
C	102	114	110	114	110	63	90	90	100	85	71	59	72	72	61	73	75	55	59	59
n	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59	60
N	74	81	20	8	2	23	86	20	32	28	11	39	28	3	6	0	0	11	31	82
C	59	59	59	59	50	41	62	62	26	26	25	25	25	24	27	19	13	0	0	0

其中 n 代表第几代,N 代表第 n 代中评价值最高的路径编号,C 代表第 n 代中评价值最高的路径与障碍物相交的长度.

在实验 1 中,系统运行了 60 代左右后基本稳定,而且寻找出较优可行解.在其他实验中,我们发现引入不同的避障算子,对系统的性能的影响也各不相同,在这里就不给出具体的实验数据了.

从实验数据可以看出,引入规划知识,对搜索起到了很大的指导作用,加快了搜索速度.但是,我们也注意

到,随着知识的引入,加大了系统的复杂度,每代为了躲避障碍物需要花费较多的时间,而且随着障碍物的增加,系统增加的关键点也逐渐增加,系统的空间和时间复杂度都明显加大.因此,并不是引入知识就一定对规划起到正作用,只有达到一定程度,才能发挥其最大作用.可以确认的一点是,随着知识的引入,可以减小初始的人口;如果不引入知识,当障碍物数量达到一定程度时,系统搜索会稳定在一些不可行解上,这时候只有加大初始人口才能求出可行解,不过随着知识的引入,可以较好地解决这个问题.因此,如何处理好障碍物数量、初始人口、知识类型以及规划的时间、空间复杂度之间的关系成为今后研究的关键.

4 总 结

实时规划算法要求系统能在制定的资源内给出相对最优的解决方案,同时,系统可以处理各种突发的意外事件,可以在任意时刻中止规划算法,输出当时认为最优的解决方案;给定的时间越多,系统的输出性能就越好.任意时间算法提出的框架符合这些特性,也决定了它适用于实时环境下问题规划求解.利用任意时间算法,可以合理分配系统资源,以求得最优系统性能;同时,在实际应用中适当地引入经验知识作为指导,可以大大加快系统的规划速度,以求得最优解决方案.

参考文献

1 McDermott D, Hendler J. Planning: what it is, what it could be, an introduction to the special issue on planning and scheduling. Artificial Intelligence, 1995,76:1~16

2 Agre P E, Chapman D. Pengi: an implement of a theory of activity. In: Proceedings of American Association for Artificial Intelligence'87. 1987. 268~272. http://www.ncstrl.org

3 Georgeff M P, Lansky A. Procedural knowledge. IEEE Proceedings, Special Issue on Knowledge Representation, 1986, 74(10):1383~1398

4 Kaelbling L P, Rosenschein S J. Action and planning in embedded agents. In: Patti Maes ed. New Architectures for Autonomous Agents: Task-level Decomposition and Emergent Functionality. Cambridge, MA: MIT Press, 1990. 35~48

5 Sanborn J, Hendler J. A model of reaction for planning in dynamic domains. In: International Joint of AI Engineering. 1988. http://www.ncstrl.org

6 Schoppers M. Universal plans for reactive robots in unpredictable environments. In: Proceedings of the International Joint Conference'87 on Artificial Intelligence. 1987. 1039~1046. http://www.ncstrl.org

7 Firby R J. An investigation into reactive planning in complex domains. In: Proceedings of American Association for Artificial Intelligence'87. 1987. http://www.ncstrl.org

8 Dean T L, Boddy M. An analysis of time-dependent planning. In: Proceedings of American Association for Artificial Intelligence'88. 1988. 49~54. http://www.ncstrl.org

9 Goldberg D E. Genetic Algorithms in Search, Optimization, and Machine Learning. Reading, MA: Addison-Wesley Publishing Company, Inc., 1989. 3~6

Real-Time Problem Solving

CHEN Zheng ZHANG Bo

(*Department of Computer Science and Technology Tsinghua University Beijing* 100084)
(*Laboratory of AI Technology and Systems Tsinghua University Beijing* 100084)

Abstract Real-time problem solving is an interesting topic in planning in recent years. Besides discussing the deficiency of traditional planning algorithm, the authors imported the anytime algorithm, which can solve the real-time problems in this thesis. Anytime algorithm could allocate time resource reasonably to ensure the best system output performance. Anytime algorithm could be interrupted at any time and output the relatively best probable solution in that time. Genetic algorithm has the properties of the anytime algorithm. After introducing the differences between this and other search algorithms, through the experiments, the authors found that the method, which combines the random search technology and knowledge based method, could solve real-time planning problems relatively better that other methods. At last, the authors gave out the conclusion, discussed the policy of real-time planning problem solving algorithm simply, and discussed the possible developments in the future.

Key words Planning, anytime algorithm, genetic algorithm, operator.