实时环境问题求解框架*

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)

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.在讨论了实时规划算法同传统离线规划算法之间的不同之后,提出了一个实现实时规划的统一框架,同时具体阐述了框架中的各个模块.通过框架构造出的调度系统可以适应实际环境的需要,在规定时间内解决系统提出的要求,并能较好地解决不确定性.此外,在框架模型指导下构造了智能装配机器人实时调度系统,并给出了实验结果.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以及今后的发展方向.

关键词 规划,任意时间算法,实时调度,不确定性,成组技术,实时监控.

中图法分类号 TP18

1 实时规划与离线规划的区别

求解规划问题的时间、空间复杂度都比较大,因此,传统规划算法对规划问题做了一系列的假设,其中最为关键的一条就是认为外界环境是不变的,即使考虑到外界环境的变化,也认为变化是可以预知的.因此,规划算法可以在给定的外界环境条件下对规划空间进行搜索,得到系统的最优输出.但是,一旦进入现实环境,我们就会发现单纯的传统规划算法并不能很好地解决实际环境中的规划,因为我们所处的实际环境是一个易变的环境,是一个不可预知的世界;而传统的规划算法认为外界环境是一个可预知的模型,这个较大的区别导致传统的规划算法在易变的环境中就显得无法自如运转.因此,构造一个实时环境下的规划算法就成为近年来大家关注的问题.如何合理利用给定的资源,构造有效可行的实时算法成为问题研究的关键.

简单地说,实时规划算法与传统规划算法在数据输入方面最大的区别在于以下两个方面:

(1) 有限的资源;

(2) 不确定性:① 资源不确定性;② 任务不确定性:(a) 任务所需资源改变,(b) 任务约束情况改变,(c) 任务执行过程改变;③ 环境不确定性:(a) 环境意外,(b) 环境精度改变.

在人工智能领域,不确定性、可能性是人们感兴趣的话题.不确定性分为客观不确定性和主观不确定性.客观不确定性是对事件发生的频度进行统计计算后得出的,不受人的主观思想的影响;而主观不确定性则是人们对于事件发生频度的一个估计,这个估计本身就存在很大的误差.因此,引入高阶不确定性来对这种误差进行数值化,$p(p(A)=x|K)=y$.

由于实时算法和离线算法求解问题策略存在很大的差别,因此也决定了所采用的算法的特点也存在很大的差别,主要表现为以下几个方面[1]:① 算法的输出性能上的区别——完全精确解(离线)与大致精确解(在线);② 可中断性的区别——不可中断(离线)与可中断(在线);③ 性能与时间关系——性能不变(离线)与性能随时间增长(在线).

2 实时环境问题求解框架

针对上面提出的实时环境下问题求解的特殊性,传统的离线规划算法已经不能适应它的要求.它要求提出

* 本文研究得到国家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,计算机应用.张钹,1935年生,教授,博士生导师,主要研究领域为人工智能,计算机应用.

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1997-12-11收到原稿,1998-03-02收到修改稿

一个新的解决方案来适应实时环境的需要.下面,我们分别根据上面提出的特殊性给出相应的解决方案.

因为实时环境下的规划受到资源的限制,所以不能再像原先那样对系统的全局进行规划,对于规划的算法我们进行了相应的改变,具体可分为以下几种方法.

(1) 处理资源约束

不再采用离线规划算法(如深度优先或者是宽度优先等算法),取而代之的是"实时环境下的规划算法"(如任意时间算法等).实时环境下的规划算法具有以下特点:① 可以在任意时刻中断并输出当时的较优解;② 能够利用有限的资源进行规划,而且可以在有限资源内给出解;③ 随着资源的增加,系统输出性能在逐渐改进;④ 能够处理各种不确定性,保证系统的正常运转.

(2) 处理不确定性

在实时规划中,另外一个较大的问题就在于"不确定性".由于系统要处于易变的外界环境中,因此需要系统能够处理各种意外的突发事件.传统的规划认为外界环境是可以预知的,但是这并不符合实时规划的实际需要,因此需要对系统进行相应改变:① 构造实时监控系统;② 分层规划与系统再利用:(a) 对系统进行划分,构造系统拓扑图,(b) 分层规划:从较粗的粒度空间上进行规划,到较细的粒度空间上进行再次规划,(c) 可再用性的利用:系统中可再利用部分的使用.

在实时规划中我们将使用以下技术.

2.1 成组技术(Group Technology)

成组技术的基本出发点就是在规划中把规划系统分解为若干个子系统.从上面的描述中我们可以看出,使用分组技术后,可以对每个子系统分别采用"任意时间算法"进行规划,使得系统更加适用于实时环境.它缩小了问题求解空间,而且可以减少不必要的重复再规划,同时可以使规划系统具有再利用性.

2.2 组织技术

实时环境中的规划系统大都由多个子系统组成,而且出于对有限时间、空间资源的考虑,我们对每个子系统利用任意时间算法对其进行规划,使得每个子系统的性能都随着资源的增加而增加.但是,另外一个问题随之而出现,即如何合理地在各个子系统之间分配资源,以求得最优系统输出,我们称之为任意时间算法的组织问题.我们提到的很多调度问题属于NP问题,同样不难看出,任意时间算法的组织问题同样也是一个NP问题,如何合理地分配系统资源,得到最优的输出性能成为另外一个难题.在离线环境下可以利用传统的方法进行组织问题求解,但一旦应用于实时环境,"大致精确解"就占据了主要位置.

2.3 任意时间算法的监控

利用成组技术对系统进行同类划分,同时对各个部分合理分配系统资源,这些都可以在离线环境下完成,然后就需要把离线规划的结果应用于实际环境中,由于在实际环境中存在着许多的不确定性,因此需要对系统的执行进行监控,以求得系统的最佳执行结果.

在系统监控中,如何决定监控的频度以及如何决定是否采用监控等策略在文献[2]中已有详细的叙述,此处不再赘述.

2.4 实时环境问题求解框架

根据上面提出的各种实时环境问题求解的方法,我们提出一个在实时环境下问题求解的一般框架,它适用于各种实时环境问题求解,利用这个框架可以比较容易地构造出实时调度算法,可以较好地解决问题.在现实生活中,许多规划/调度问题都可以看成是一个实时环境下的问题求解,因此,我们可以采用下面提出的这个框架/工具构造问题求解系统,并进行求解.从图1中可以看出框架主要分为4大模块:离线规划模块、在线调度模块、监控模块和故障恢复模块.具体的数据流关系可以从图1中得到.

对于不同的规划/调度问题,我们将利用上面提出的实时问题求解框架进行构造实时算法,同时利用实时算法数据库以及具体规划/调度的经验知识,对实时算法进行监控,处理各种不确定性问题,以求得系统的最优输出.

下面我们利用实时问题求解框架构造机器人智能装配系统,分析系统具体的运作过程.

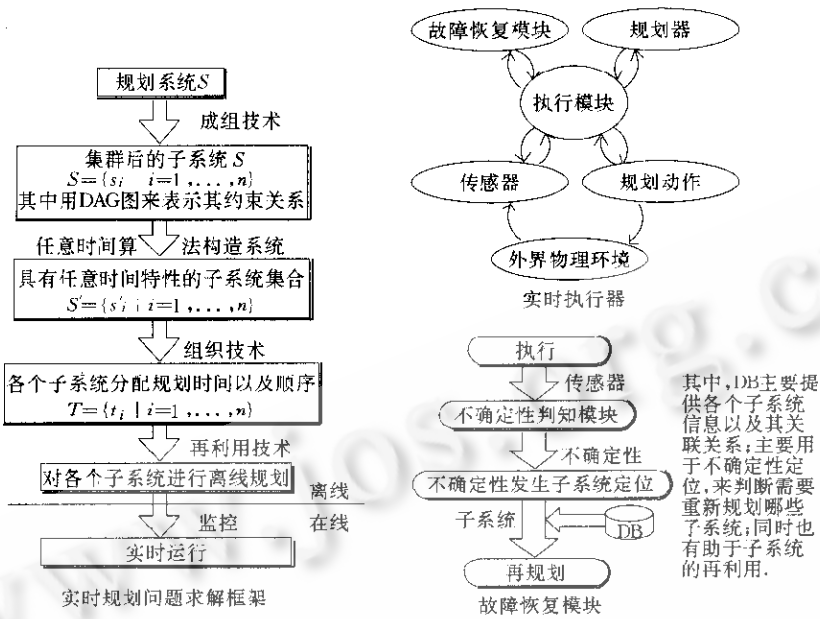


图1　实时问题求解框架

3　具体问题求解

我们利用上面提到的"实时环境下问题求解的框架",结合装配过程的特性,设计并实现了一个装配操作调度系统,同时对一些装配例子进行了相应的测试.

在机器人智能装配中,我们根据装配的特点,设计了快速调度算法,以满足灵活处理各种不确定的发生.机器人装配要解决的问题是:给定 m 台可用的机器 $M=\{M_1,\ldots,M_m\}$,同时给定需要调度的 n 个任务 $T=\{T_1,\ldots,T_n\}$,同时给出这 n 个任务之间的约束关系(以 DAG 图或者关联矩阵表示),需要求解的是如何以较少的时间在这 m 台机器上调度执行完这 n 个任务.

装配调度要解决的就是如何在给定的时间内给出系统较优的规划,同时能够处理在实时调度执行中发生的各种不确定性.我们对装配过程中有可能出现的不确定性进行估计,主要有以下几种:(1) 任务加工时间的变化:主要指当前正在执行的加工任务时间的推延或者提前;(2) 工件不到位:指原先应该可以就绪的任务现在就绪时间推延;(3) 机器故障:在某个时刻机器发生故障,任务需要更换机器执行.

在实际调度中,我们根据装配调度的特性,给出了以下调度策略:(1) 执行时间越长的任务首先被优先执行的可能性越大;(2) 深度优先长度较大的任务被优先执行的可能性较大;(3) 直接后继或者是后继多的任务被优先执行的可能性大.

以上 3 条是我们的快速调度的基本策略,但是基本策略的适用范围是随着调度过程而改变的.在调度中,我们根据实时环境改变每个任务的优先级,对那些已经就绪的任务就根据其优先级来决定调度哪个任务,力求在较短的时间内对所有任务进行调度.之所以采用动态优先级实调度[3],是因为这样可以不断调整任务的优先级来适应易变的外界环境.环境改变,任务的优先级也随之改变,以达到最佳的调度结果.同时,我们在调度中还对未来的调度做了一定的预测,以保证调度的精度.在介绍调度算法前,我们先给出以下几个基本定义.

(1) 就绪任务:如果在 t 时刻任务 i 的所有前驱约束都已经调度结束,那么,称任务 i 为就绪任务,只要有空闲的机器就有可能被调度.记为 $RT(t)$[3];

(2) 空闲机器:如果在 t 时刻机器 j 不在调度任何任务,我们称机器 j 为空闲机器;在调度过程中应尽量减少机器的空闲程度,那样才有希望在最短时间内调度结束.记为 $RM(t)$[3];

(3) 任务执行时间:$ExecT_i$;

(4) 任务执行需要机器数目:UM_i;

(5) 任务的深度优先长度:$Depth(Task_i)$;

(6) 任务的后继深度优先长度:$\sum_{succ} Depth(Task_i)$.

为了降低调度过程中整个状态空间的大小,我们把规划时间划分成若干个离散的时间片,假设每个任务将占用1个或多个时间片.因此,我们在调度过程对每个时间片都动态地计算各个任务的优先级,对优先级最高的任务优先分配执行;优先级计算按照上面提出的策略对深度优先长度以及任务执行时间进行相应加权计算而得到.

由上面的策略我们可以得出在 t 时刻任务 i 的优先级为

$$\mathrm{Prior}(Task_i)=\frac{RT(t)}{RT(t)+RM(t)}\times\frac{\sum_j UM_j}{UM_i}\times\frac{ExecT_i}{\sum_j ExecT_j}\times Depth(Task_i).$$

$$+\frac{RM(t)}{RT(t)+RM(t)}\times\frac{UM_i}{\sum_j UM_j}\times\frac{\sum_j ExecT_j}{ExecT_i}\times\sum_{succ} depth(Task_i).$$

上面提到的算法都不存在对将来发生情况的预测,可以对算法进行相应的改变,在 t 时刻计算任务优先级时可以往前预测一段时间 Δt,同时假设在 Δt 时间内不在调度其他任务,则计算出 $t+\Delta t$ 时刻任务之间的优先级,根据这些优先级进行任务调度.同时,可以利用任意时间算法来不断改变 Δt,力求找到此时刻最佳的调度.

3.1 智能机器人任务调度算法

```
t=0;
LeftTask=n;
while (LeftTask>0) {
    计算在t时刻的就绪任务集合
    Δt=0
    BestAlloc=∅
    while (Δt<Max PredictTime) {
        if 就绪任务集合为空,则分配一个等待任务
        else {
            计算t+Δt时刻的就绪任务集合、空闲机器集合,对于就绪任务集合中的所有任务计算其相
            应的动态优先级 Prior(Task_i)
                    选择优先值最高的任务分配执行
                    把分配的任务加入到 BestAlloc
                    LeftTask--;
        }
        求出最优的 BestAlloc(t时刻继续任务分配集合)
        Δt=Δt+step // 优化过程
    }
    按照 BestAlloc 集合调度其中的任务
    t++;
}
```

3.2 复杂度分析

上述的算法在特殊情况下的分析，假设在调度中任务都只需要一台机器完成，同时不存在工具切换问题，则上述的近似算法的精确度如下.

对近似算法的评价：近似算法的完成时间记为 T_n，最佳任务安排的时间记为 T_n^*；定义近似算法的绝对偏差 $\Delta=T_n-T_n^*$，相对偏差 $\delta=\frac{\Delta}{T_n^*}=\frac{T_n-T_n^*}{T_n^*}$.

可以证明，近似算法在任务之间无任何约束的情况下，相对偏差满足 $\delta\leq\frac{1}{3}-\frac{1}{3l}$（其中 l 为机器数目）[4]. 而且从算法分析上可以看出，在两台机器人工作时，算法输出的是系统的最优调度，和 Johnson 的两台机器人调度得到同样的系统输出.

3.3 不确定性处理分析

根据上面提出的优先级计算调度法，我们通过实验发现，它在离线情况下能较好地对装配任务进行调度. 把这个方法应用于实时环境中，发生的不确定性按照我们上面提出的假设来进行，可以发现当不确定性发生后，系统可以根据当时的状况对当时的就绪任务进行重新计算和分配，同时还可以根据给定的时间不断对任务分配进行优化，力求得到较优的系统输出；而且处理不确定性所引起的系统开销也不大，可以在 p 时间内完成.

3.4 实验结果

下面我们以几个特殊的任务调度为例来看算法的输出性能. 算法在 P133+Win95+Vc4.2 下开发完成.

系统输入任务约束关系如图 2～4 所示. 系统调度输出 Gantt 如图 5～12 所示.

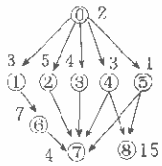


图2

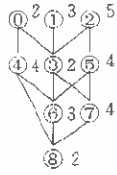


图3

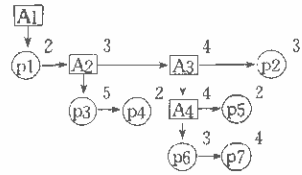
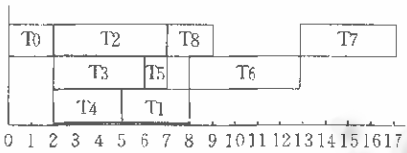


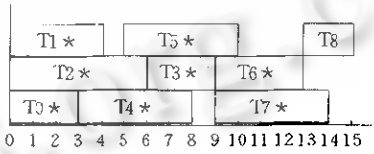
图4



任务关联图：图1；调度策略：动态优先级；

$ExecTime(T_8)=2$

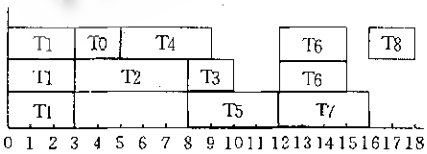
图5　调度实验1



任务关联图：图2；调度约束；工具约束；工具约束条件：每个任务最多使用一个工具；工具切换时间均为1；工具个数为5；图中 * 代表在调度此任务时需要进行工具切换

图6　调度实验2

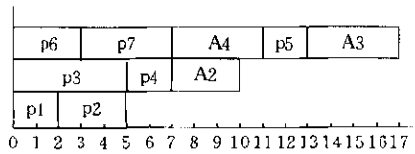
任务	0	1	2	3	4	5	6	7	8
使用工具	1	2	3	2	5	1	4	3	1



任务关联图：图2；调度约束：机器约束；机器约束条件：机器总数目为3；任务在执行时需要1～3台机器同时完成

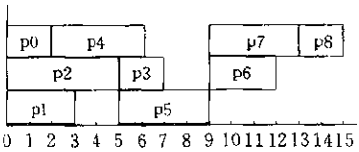
图7　调度实验3

任务	0	1	2	3	4	5	6	7	8
需要机器数目	1	3	1	1	1	1	2	1	1



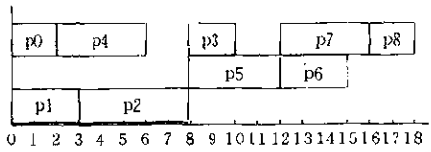
任务关联图：图3；输入数据类型：装配前趋约束关系图，其中方块代表装配部件，圆圈代表装配零件，横向的链代表出于同一层次的节点，纵向的链代表不同层次上的关系. 调度方法：分层规划

图8　调度实验4



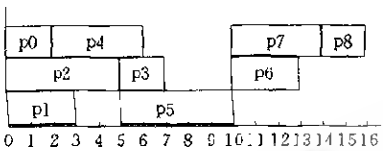
任务关联图:图2;没有任何不确定性

图9 调度实验5



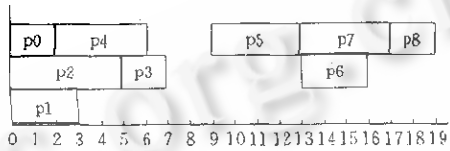
任务关联图:图2;不确定性:$t=4$时刻机器2发生故障,故障到$t=5$时刻解除

图10 调度实验6



任务关联图:图2;不确定性:$t=6$时刻任务p5的执行时间需要延长一个时间

图11 调度实验7



任务关联图:图2;不确定性:工件p5的到位时间延长到$t=9$才能到位

图12 调度实验8

对于在线调度方法,除了采用上面提到的"大致精确解"的方法以外,"分层规划"也是较好的一种解决方案,它可以把复杂问题简单化,同时也能得到较好的解答.

3.5 分层规划方法

为了在装配中减少工具切换所引起的系统开销和满足装配的层次性问题,我们提出分层规划的方法,具体实现方法如下(对于装配序列中的每层进行单独规划):

(1) 求出本层就绪任务和空闲机器;

(2) 对于零件节点,按照原有的策略分配机器执行.对于部件节点,则先估计其执行时间以及需要的执行机器数目,然后分配执行;

(3) 对于部件节点分配进行细化,调用算法继续深入调度.

与上面提到的算法相比,分层规划算法主要在以下几个部分有所改变:

(1) 在计算每个任务的执行时间时算法需要改变,对于零件的执行时间则直接得到;而对于部件的执行时间则需要进行估计.如果在一个机器上完成则需要$\sum_i t_i$的时间,然后估计执行需要的机器数目n,则对于不同的执行机器数目,存在不同的时间估计$ExecT=\frac{\sum_i t_i}{n}$.

(2) 根据上面的估算,求出就绪任务集合以及空闲机器集合.

(3) 计算优先级.计算优先级时需要考虑到因对部件的执行机器的数目有不同的估计而产生的影响.

(4) 对于优先级较高的任务调度执行.

(5) 需要不断地对部件的调度进行细化调度.

分层规划算法在处理不确定性方面的优点:① 把原有的一个大系统分成许多相对独立的小系统.② 在不同层次上对任务调度进行处理.③ 如果时间允许,可以对每个部件局部最优求解;降低了算法的复杂度.④ 如果发生不确定性,可以对不确定性发生定性、定位,仅仅重规划不确定性发生的层次,而不影响其他层次的规划;同时,使得规划具有再利用的价值.

此外,除了上面提到的利用装配过程操作方法对系统进行分层外,我们还可以利用装配中常用的一些经验知识对系统进行分层规划,例如在调度中,我们可以把一些操作合并成一组操作,然后在较高的层次进行调度,当调度到这组任务时,再调度较低层次的调度进行规划.

4 总 结

随着现代化工业的发展,实时规划在现实中的应用也越来越广泛,传统的离线规划已渐渐不能适应实际的需求,新的规划模式逐渐占主导地位.实时规划算法可以获得系统输出性能与给定资源之间的动态平衡.随着应

用的推广,合理地构造实时规划算法,有效地利用规划中每步得到的结果,不断优化系统的输出性能,最优地监控系统的执行,这些都成为研究实时规划问题的关键.

参考文献

1 Joshua Grass, Shlomo Zilberstein. Anytime algorithm development tools. Department of Computer Science, University of Masschusetts. URL: http: //anytime.cs.umass.edu/~sclomo

2 Hansen E A, Zilberstein Shlomo. Monitoring the progress of anytime problem-solving. Department of Computer Science, University of Massachusetts. URL: http: //anytime.cs.umass.edu/~sclomo

3 黄必清.关于智能装配机器人系统的任务调度研究[博士学位论文].清华大学,1994
(Huang Bi-qing. Research on task scheduling in intelligent assembly robot systems[Ph. D. Thesis]. Tsinghua University, 1994)

4 卢开澄.算法复杂度分析.北京:清华大学出版社,1983
(Lu Kai-cheng. Analysis of Algorithm Complexity. Beijing: Tsinghua University Press, 1983)

Real-time Problem Solving Framework

CHEN Zheng ZHANG Bo

(*Department of Computer Science and Technology Tsinghua University Beijing* 100084)
(*State Key Laboratory of Intelligent Technology and Systems Tsinghua University Beijing* 100084)

Abstract Real-time problem solving is an interesting topic in planning in the recent years. Besides discussing the differences between real-time scheduling and off-line scheduling, a real-time scheduling framework is proposed and some pivotal technologies are discussed in this paper. This framework can deal with uncertainty in real-time environment and output approximate solution in given time. Based on this framework, an intelligent assembly robot system is built, and through some experiments this framework is proved to do well in real-time environment. After all, the conclusion is given out, and the policy of real-time planning problem solving algorithm simply and the possible developments in the future are discussed in this paper.

Key words Planning, anytime algorithm, real-time scheduling, uncertainty, group technology, real-time monitoring.