一种新的组优先级动态实时调度算法

巴 巍 张大波 李 琦 王 伟
(大连理工大学电子与信息工程学院 大连 116024)

摘 要 传统动态调度算法由于对优先级个数没有限制,在实际应用中往往受制约,达不到很好的调度性能。针对此问题,考虑硬实时抢占任务调度需要,提出一种新的组优先级动态实时调度算法。研究作业执行顺序改变对系统可调度性能的影响,给出作业分组可调度性能测试。新算法将满足分组可调度测试公式的作业作为一个任务组,各任务组之间按照最小截止期优先调度,任务组内按照最短作业优先的原则执行作业。仿真结果表明,与最小截止期优先等传统调度算法相比,新算法不仅能有效降低算法所需优先级个数,还能提高任务调度的成功率,缩短平均响应时间,减少任务切换次数。

关键词 任务调度,最小截止期优先调度算法,成功率,切换次

中图法分类号 TP316.2 **文献标识码** A

New Group Priority Dynamic Real-time Scheduling Algorithm

BA Wei ZHANG Da-bo LI Qi WANG Wei
(School of Electronic and Information Engineering, Dalian University of Technology, Dalian 116024, China)

Abstract There is no restriction for priority levels in priority scheduling algorithms, which limits the application in practice to get good schedulability. Aiming at that question, considering the requirement of preemptive scheduling for hard real-time systems, a new group priority dynamic real-time scheduling algorithm was presented. The influence of the changed order for some jobs on the schedulability of the system was studied. The schedulability test for the job grouping was given. In the new algorithm, the jobs that satisfied the formula of the schedulability test were joined together as a group. The jobs outside the group schedule in darliest-deadline-first, while the jobs in the group schedule in shortest job first. The simulation results show that, comparing with the earliest-deadline-first and other traditional scheduling algorithms, the priority levels decrease deeply, the success ratio increases, the average response time is shorten and the switching number reduces in the new algorithm.

Keywords Task scheduling, Earliest-deadline-first scheduling algorithm, Success ratio, Switching number

1 引言

动态调度算法处理器利用率高,系统资源利用充分,与静态调度算法相比,更适合在计算资源受限的嵌入式系统、多媒体系统等中应用[1,2]。Liu 和 Layland 提出的最小截止期优先(Earliest Deadline First, EDF)[3]调度算法是使用最多的一种动态优先级调度算法,已被证实是一种最优的单处理器调度算法[3]。以 EDF 调度算法为基础,衍生出很多改进算法,如 Locke 提出的尽力服务(Best Effort, BE)[4]调度算法等,它们从多角度丰富了实时调度算法,为实时调度的更广泛应用提供了条件。这些调度算法均假设优先级的个数可以无限扩展。然而,在嵌入式产品中,系统支持的优先级个数通常是有限的[5,6],因此传统动态调度算法存在局限性。

目前,针对降低优先级个数的研究主要是将多个具有不同优先级别的任务映射到一个优先级组。在静态优先级调度算法中,组优先级应用广泛。文献[7]提出了优先级映射的常量法,文献[8]提出了 RM-Least 算法,文献[9]提出了有限优先级静态分配算法(Assignment of Priority Group, APG)。然而,它们解决的都是静态优先级调度算法的分组问题。

针对动态组优先级调度算法的研究,文献[10,11]把作业的截止期分成若干区域,截止期处于某一区域的作业被安排在该区域的优先级组中,组间按 EDF 调度,组内按先进先出顺序执行,算法减少了优先级需求。文献[12]提出一种非抢占组优先级 EDF 调度(group Earliest Deadline First, gEDF)算法,任务按截止期非递减的顺序排列,算法根据组范围参数 Gr 判定与队首作业同组的作业数。在组内,作业按照最小作业先调度(Shortest Job First, SJF)的原则进行调度。与文献[10,11]提出的调度算法相比,gEDF 算法实现了组优先级的动态分配,调度性能增强。但是,上述算法只针对软实时调度,无法保证满足硬实时的要求。

到稿日期:2008-08-20 返修日期:2008-11-19

巴 巍(1980—),女,博士研究生,主要研究方向为实时系统,E-mail:baweidut@gmail.com;张大波(1966—),男,副教授,主要研究方向为实时系统、分布式系统等;李 琦(1979—),男,博士研究生,主要研究方向为实时系统、网络控制;王 伟(1955—),男,教授,博士生导师,主要研究方向为实时系统、网络技术、智能控制等。

针对硬实时抢占任务调度的需要，本文研究作业执行顺序对系统调度性能的影响，提出更为严谨的调度性能测试，保证作业执行顺序的变化不影响可调度性能。在此基础上，提出一种新的组优先级动态实时调度算法。新算法根据调度性能测试决定组内作业数，并允许组内作业按 SJF 调度。仿真结果表明，与传统调度算法对比，新算法不仅能有效降低算法所需优先级个数，还能提高任务调度的成功率，缩短平均响应时间，减少任务切换次数。

2 任务模型

对于具有 N 个独立任务的任务集 $T=(\tau_1,\tau_2,\cdots,\tau_N)$，任务 τ_i 用 (p_i,e_i,D_i) 表示，其中 p_i 为任务的周期，e_i 为任务的最坏执行时间，D_i 为任务的相对截止期，设定为与周期值相同。任务的每次执行被称为作业，任务 τ_i 的第 k 次执行作业用 $\tau_{i,k}$ 表示，描述为 $(a_{i,k},e_i,d_{i,k})$，其中 $a_{i,k}$ 是作业 $\tau_{i,k}$ 的释放时间，$a_{i,k}=(k-1)^* p_i$，$d_{i,k}$ 是作业 $\tau_{i,k}$ 的绝对截止期，$d_{i,k}=(k-1)^* p_i+D_i=k^* p_i$。

在动态实时调度问题中，任务的不同作业往往具有不同的优先级。若每一作业就绪时都要更新作业的优先级并重新排列作业的执行顺序，无疑将给算法执行和优先级数评估带来很大的不便。因此，应提出新的作业假想排序模型。预估一个超周期或某一规定时间内可能出现的作业，并将其按优先级排序，将此过程产生的优先级总和作为系统所需优先级数。设 EDF 调度程序在 $t_{termination}$ 时刻终止，用 T_t 描述 t 时刻 $(0\leqslant t\leqslant t_{termination})$ 由执行状态作业、就绪状态作业和未就绪状态作业组成的作业集，表示成 $T_t=(\tau_{t_1},\tau_{t_2},\cdots,\tau_{tm})$，其中各作业按绝对截止期从小到大排列，分别用 T_{te}，T_{tr}，T_{tn} 表示 T_t 中执行状态作业子集、就绪作业子集和未就绪作业子集。

此外，为方便描述算法行为，给出如下性能参数指标：

ρ 为处理器利用率，$\rho=\sum_{i=1}^{N}\frac{e_i}{p_i}$。

μ 为最坏情况下执行时间按指数分布的期望值。

3 可调度性能测试

文献[13]给出了在有阻塞情况下系统的可调度性能测试，本文借鉴文献[13]中的策略，针对 EDF 调度算法，给出在允许部分任务交换执行顺序的情况下更为严紧的可调度性能测试。

定理 1 对于按 EDF 算法可调度的一组周期任务集 $T=(\tau_1,\tau_2,\cdots,\tau_N)$，当 $T_{te}=\phi$ 时，取 T_{tr} 中第一个作业，假设其为作业 $\tau_{i,k}$，在 T_t 中用 τ_{tu} 表示。若

$\exists\ \tau_{tf}\in T_{tr}$ 满足

$$\sum_{j=1}^{i}\frac{e_j}{p_j}+\sum_{p=1}^{u-1}\frac{e_p}{p_i}+\sum_{q=u+1}^{f}\frac{e_q}{p_i}<1 \tag{1}$$

其中，式(1)左侧第一项针对任务集 T 中前 i 项求和，而后两项则针对作业集 T_t 中各指定项分别求和。令 $T_{t\_change}=(\tau_{t_1},\tau_{t_2},\cdots,\tau_{tf})$，则 T_{t_change} 中就绪作业可随意调整执行顺序而不会改变原有的调度性能。

证明：T_t 中只有就绪作业和未就绪作业，对于 T_t 中作业调度性能分 3 方面考虑：

(1) T_t 中 τ_{tf} 之后的作业。由于 T_t 中作业按其绝对截止期排列，即 $\forall\tau_{tm}\in T_{t\_change}$，$d_{t(f+1)}>d_{tm}$，$T_{t\_change}$ 中作业执行顺序的改变对于 T_t 中其他作业的执行没有影响。

(2) T_t 中 τ_{tu} 之前的作业。τ_{tu} 是 T_t 中第一个就绪作业，T_t 中若 τ_{tu} 之前有作业(即 $\mu>1$)，则其必为绝对截止期小于 d_{tu} 的未就绪作业。它们的执行同样不受就绪作业执行顺序调整的影响。

(3) T_t 中 τ_{tu} 之后、τ_{tf} 之前的作业。这其中包括就绪作业和未就绪作业两部分。分两种情况讨论：

a)就绪作业。假设其原来为可调度，就绪作业改变执行顺序后变为不可调度。因 T_{t_change} 中执行时间小于 e_i 的作业提前到 $\tau_{i,k}$ 之前执行，导致系统不可调度，即 $\tau_{i,k}$ 不可调度。设 $\tau_{i,k}$ 的阻塞时间为 $B_{i,k}$：

$$B_{i,k}=\sum_{p=1}^{u-1}e_p+\sum_{q=u+1}^{f}e_q$$

在 $[(k-1)^* p_i,k^* p_i]$ 内，只有相对截止期小于 D_i 的作业可能产生并完成。在这段时间内，每个作业 τ_j 占用的最大时间为：

$$\left(\left\lfloor\frac{D_i-D_j}{D_j}\right\rfloor+1\right)e_j$$

其中 $\lfloor\ \rfloor$ 为向下取整。

由于 $\tau_{i,k}$ 不可调度，在 $[(k-1)^* p_i,k^* p_i]$ 内，系统的要求必然超过时间约束，即

$$B_{i,k}+\sum_{j=1}^{i}\left(\left\lfloor\frac{D_i-D_j}{D_j}\right\rfloor+1\right)e_j>D_i$$

由 $\lfloor X\rfloor\leqslant X$，上式可变为

$$\frac{B_{i,k}}{D_i}+\sum_{j=1}^{i}\left(1+\frac{p_i-D_j}{D_i}\right)\frac{e_j}{p_j}>1$$

由 $p_j=D_j$，上式变为 $\frac{B_{i,k}}{p_i}+\sum_{j=1}^{i}\frac{e_j}{p_j}>1$，即

$$\sum_{j=1}^{i}\frac{e_j}{p_j}+\sum_{p=1}^{u-1}\frac{e_p}{p_i}+\sum_{q=u+1}^{f}\frac{e_q}{p_i}>1 \tag{2}$$

式(2)与式(1)矛盾，所以此时系统可调度。

b)未就绪作业。设 τ_{tg} 为未就绪作业，由 τ_{tg} 特性知 $a_{tg}\geqslant a_{i,k}$，$d_{i,k}\leqslant d_{tg}\leqslant d_{tf}$。

若 τ_{tg} 到来时仍有就绪作业未执行结束，则 $[a_{tu},a_{tg}]$ 无空闲时间。(1)式两边同乘 D_i，有

$$\sum_{j=1}^{i}e_j\frac{D_i}{D_j}+\sum_{p=1}^{u-1}e_p+\sum_{q=u+1}^{f}e_q<\sum_{j=1}^{i}e_j+\sum_{p=1}^{u-1}e_p+\sum_{q=u+1}^{f}e_q<D_i$$

则

$$a_{i,k}+\sum_{j=1}^{i}e_j+\sum_{p=1}^{u-1}e_p+\sum_{q=u+1}^{f}e_q<a_{i,k}+D_i \tag{3}$$

由于 $a_{i,k}+D_i=d_{i,k}$，$d_{i,k}\leqslant d_{tg}$，式(3)可变为

$$a_{i,k}+\sum_{j=1}^{i}e_j+\sum_{p=1}^{u-1}e_p+\sum_{q=u+1}^{f}e_q<d_{t,g}$$

未就绪作业 τ_{tg} 可调度。

若 τ_{tg} 到来时就绪作业已全部执行结束，则此时作业 τ_{tg} 的执行不受就绪作业执行顺序改变的影响。

证毕。

定理 1 对 EDF 调度算法在任务执行顺序改变状况下给出了更为严紧的可调度性能测试。它不仅对 EDF 调度算法适用，其调度思想对于任务优先级变化但作业优先级不变的其他调度算法同样适用。定理 1 同样适用于最小速率优先(Rate monotonic，RM)静态调度算法，此时由于任务的优先级固定，作业集模型需要通过作业到达时间的判断做相应的调整。组优先级 RM 算法相比组优先级 EDF 算法更为复杂，这里不做讨论。

4 组优先级 EDF 调度算法

由 EDF 调度算法的可调度性能测试可以得出，当某一时刻系统没有正在执行的作业时，系统可以集中作业于第一个就绪作业 τ_{iu}，按照式(1)由其前作业与其后部分连续作业共同组成一个优先级组，组内的作业可被视为具有同一优先级，其中的就绪作业共同以当前的最高优先级参与系统的 EDF 调度。式(1)左侧 $\sum_{q=u+1}^{f}\frac{e_q}{p_i}$ 项中的任务既有就绪作业也有未就绪作业，其中未就绪作业的绝对截止期大于 d_{iu}，没有必要一定要在 τ_{iu} 之前完成，在选择先于 τ_{iu} 执行的作业时暂不考虑。

据此，本文提出一种新的组优先级 EDF 调度算法(group-priority EDF，简称 gpEDF)。

单处理器系统中，$[0,t_{termination}]$时间段内产生的所有作业按照绝对截止期递增顺序排列，在任一时刻 t，组优先级 EDF 调度算法遵循如下规则：

①作业集 T_t 更新。

查找队列中绝对截止期小于 t 的作业并删除。由于 T_t 中作业按照绝对截止期排列，从队首作业开始，找到第一个绝对截止期大于 t 的作业即可停止。

②优先级组更新。

若系统中没有优先级组，以 T_t 中第一个就绪作业为 τ_{iu}，将满足(1)式的作业组成优先级组。若没有满足式(1)的作业，则将 τ_{iu} 与其前未就绪作业组成特殊优先级组。

③作业调度规则。

(a)选取 t 时刻 T_t 中当前可执行作业

1)如果此时 T_t 中有正在执行的作业，则将其作为当前可执行作业。

2)如果此时 T_t 中没有正在执行的作业，则选取优先级组中具有最小执行时间的就绪作业作为当前可执行作业。

(b)执行选定作业

执行当前可执行作业一个时间周期，若作业完成或丢失，则从 T_t 中删除该作业。若 τ_{iu} 以及排在它前面的作业均完成或已经错过时限，则恢复无优先级组状态。若此时的优先级组为特殊作业组，当 τ_{iu} 之前的作业变为就绪状态时，需要判断其剩余空闲时间是否允许 τ_{iu} 执行结束而不发生抢占。

组优先级 EDF 算法用优先级组优化了 EDF 算法的调度性能。图 1 描述了 3 个周期任务 $\tau_1(4,2,4)$、$\tau_2(8,1,8)$和 $\tau_3(10,2,10)$一个超周期内的 EDF 和 gpEDF 调度表。图 1(a)中，3 个任务按照 EDF 算法调度。在时刻 3，作业 $\tau_{3,1}$ 开始执行，其绝对截止期是 10。下一个时刻，$\tau_{1,2}$ 就绪，$\tau_{1,2}$ 的绝对截止期是 8，小于 $d_{3,1}$，因此 $\tau_{1,2}$ 抢占 $\tau_{3,1}$，$\tau_{3,1}$ 直到时刻 6 作业 $\tau_{1,2}$ 完成时才可以继续执行。同理，$\tau_{3,2}$ 在时刻 12 被 $\tau_{1,4}$ 抢占。图 1(b)中，3 个任务按照组优先级 EDF 算法调度。0 时刻，$\tau_{1,1}$，$\tau_{2,1}$ 和 $\tau_{3,1}$ 就绪，按照(1)式，$\tau_{1,1}$ 和 $\tau_{2,1}$ 可以组成优先级组。$\tau_{2,1}$ 由于执行时间短，先于 $\tau_{1,1}$ 执行。时刻 3，$\tau_{3,1}$ 和作业集中排在其前面的未就绪任务 $\tau_{1,2}$ 不满足(1)式，它们组成特殊优先级组，$\tau_{3,1}$ 执行一个单位时间。在时刻 4，$\tau_{1,2}$ 就绪，它的剩余空闲时间为 2，允许 $\tau_{3,1}$ 继续执行直至结束，因此 $\tau_{1,2}$ 在时刻 5 开始执行并在时刻 7 结束。

通过图 1 中(a)和(b)的比较可知，在一个超周期内，它们均没有出现不能调度的作业。但在组优先级 EDF 调度算法中，τ_2 的所有作业都得到了最快速的响应。与先响应 τ_1 相比，减小了任务的平均响应时间，同时，τ_3 执行过程中没有发生抢占。在 EDF 调度算法中，就绪作业的绝对截止期不同，其优先级就不一样。而 gpEDF 调度算法中，τ_2 和与它一起产生的 τ_1 均可视为同一优先级，因此它需要的优先级数将小于 EDF。表 1 更为直观地说明了两种算法的性能差异。

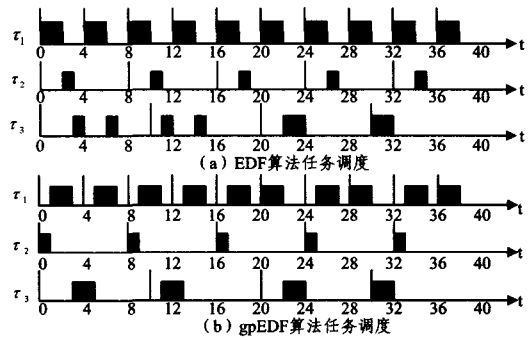


图 1 EDF 算法和 gpEDF 模型任务调度比较

表 1 EDF 和 gpEDF 调度性能比较

	成功率	平均响应时间	切换次数	优先级数
EDF	100%	2.7895	2	19
gpEDF	100%	2.4211	0	12

组优先级 EDF 调度算法在作业 τ_{iu} 所能容忍的最大阻塞范围内接收最多的作业组成优先级组，不仅使具有最小执行时间的作业得到最快速的执行，同时避免了组内各作业间可能发生的任务切换。值得注意的是，当作业 τ_{iu} 无法与其他作业共同组成优先级组时，τ_{iu} 的完成与否与系统的负载有关。当系统超载时，τ_{iu} 可能因为错过截止期而丢失。

gpEDF 与 EDF 调度算法相比稍显复杂，主要体现在生成优先级组和寻找组内具有最小执行时间的作业上。观察式(1)中 $\sum_{q=u+1}^{f}\frac{e_q}{p_i}$ 可知，在验证加入优先级组中的作业时，就绪作业及其之前的未就绪作业可一起参与计算，因此这部分的复杂度是 $O(N-1)$。组内最小执行时间作业的寻找最多需要遍历$(N-1)$个作业，因此 gpEDF 此部分的复杂度是 $O(2(N-1))$。它虽然比 EDF 调度算法复杂，但通过分析以及下一节的仿真可以看到，其调度性能得到了大幅度提升。

5 仿真

为了验证组优先级 EDF 算法调度性能的优越性，gpEDF 在同 EDF 调度算法进行对比的同时，也与 BE 算法以及 gEDF 调度算法比较。由于 BE 调度算法未考虑资源受限情况，不对其进行优先级个数的比较。而 gEDF 是一种非抢占软实时调度算法，因此仅对其进行优先级个数比较即可。

在仿真中，随机产生 5 个周期任务。设定 $\mu=10$，周期取值上限为 100。其中，e_i 是服从 μ 的指数分布随机数，取 e_i 的最大值 $\max(e_i)$为周期的下限，p_i 是$[\max(e_i),100]$之间的随机数，任务的相对截止期与周期相等。假定 $t_{termination}=500$，实验中 BE 算法给定 $V=1$，gEDF 算法给定 $Gr=0.4$。从任务成功率、切换次数、平均响应时间以及优先级个数 4 个方面进行仿真结果比较。

5.1 任务成功率比较

任务成功率是衡量实时算法调度性能的重要指标之一。尽管 EDF 是最优的动态调度算法，但在超载情况下，它的任

务成功率并不是最高的。如图 2 所示，这 3 种调度方案在轻载时成功率均为 100%，但超载时 3 种方案之间的差异较大，BE 算法的成功率最高，组优先级 EDF 调度算法的成功率次之，EDF 调度算法的任务调度成功率最低。

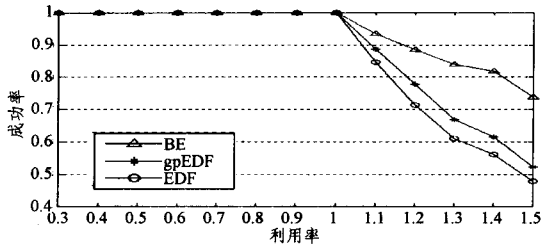


图 2　成功率性能比较

5.2　平均响应时间比较

平均响应时间标志着任务完成的速度。任务的平均响应时间越小，系统的实时性也就越高。在 BE，EDF 和 gpEDF 3 种调度方案的仿真实验中，它们的平均响应时间差异较小，这里用表格形式加以说明，如表 2 所列。

从表 2 中可以看出，与 EDF 调度算法相比，BE 算法只在超载较大时才减少了任务的平均响应时间，而组优先级 EDF 调度算法的平均响应时间在各采样点均小于 EDF，不受负载限制。

表 2　BE，EDF 和 gpEDF 平均响应时间比较

ρ	BE	EDF	gpEDF
0.3	5.475	5.475	4.875
0.4	7.3158	7.3158	7.0789
0.5	8.0364	8.0364	7.5818
0.6	8.3585	8.3585	8
0.7	10.533	10.533	9.3833
0.8	12.759	12.759	11.466
0.9	15.918	15.918	15.689
1.0	20.6	20.6	19.703
1.1	28.563	33.675	28.167
1.2	29.609	38.057	37.795

5.3　切换次数比较

在实时系统的抢占调度中，正在执行的低优先级作业不得不让出处理器资源给就绪的高优先级作业，而在高优先级作业结束时恢复其执行。在处理器资源有限时，抢占引起的额外开销将严重影响系统的性能。任务切换次数记录了系统的抢占行为。3 种方案的切换次数比较如图 3 所示。

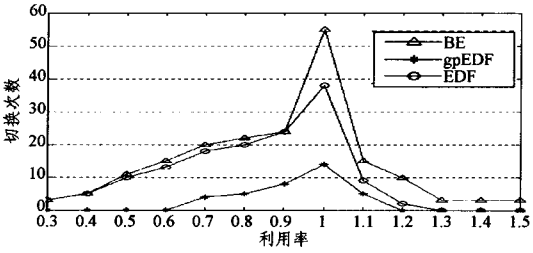


图 3　切换次数性能比较

从图 3 中我们可以看到，当处理器利用率在 1 附近时，调度算法的切换次数都达到最高值。在相同利用率下，BE 算法的任务切换次数要稍高于 EDF 算法，但组优先级 EDF 调度算法的切换次数要远远小于 BE 和 EDF 算法。

5.4　优先级数比较

组优先级 EDF 调度算法的主要目的是降低 EDF 调度算法的优先级个数，使其更能适应优先级有限系统的需要。图 4 比较了 gpEDF，EDF 算法和 gEDF 调度算法在相同条件下需要的优先级个数。

从图 4 中可以看到，当处理器利用率在 1 附近时，算法需要的优先级个数最多。对比 3 种算法的优先级个数曲线，在相同负载下，EDF 需要的优先级个数最多；gEDF 由于采用了分组算法，对优先级个数的需求略有下降，而本文提出的组优先级 EDF 调度算法需要的优先级个数则最少。

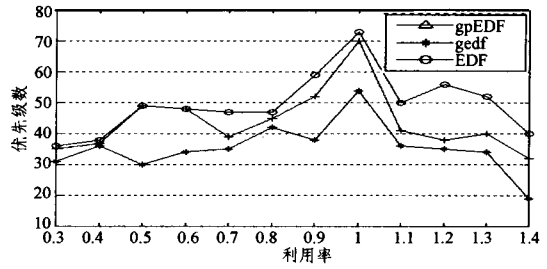


图 4　优先级数性能比较

从仿真结果可以看出，尽管 BE 算法具有最高的任务成功率，但它同时具有最多的任务切换次数。综合比较，组优先级 EDF 调度算法具有最佳的实时性能。

结束语　本文给出作业交换执行顺序后系统的可调度性能测试，按可调度性能测试划分优先级组，并提出一种组优先级 EDF 调度算法。仿真结果表明，本算法能有效降低所需优先级个数。与此同时，它还提升了任务调度的成功率，减少了平均响应时间，极大地降低了切换次数，大大地提升了系统调度性能。

组优先级调度策略不仅适用于 EDF 调度算法，而且对于很多作业级优先级固定的调度算法均适用，经修改后也可应用在 RM 等静态调度算法中。研究组优先级调度策略在更多实时调度算法中的应用，是我们下一步的工作重点。

参 考 文 献

[1] Buttazzo G C. Rate Monotonic vs. EDF: Judgment Day [J]. Real-Time Systems(S0922-6443), 2005, 29(1): 5-26

[2] 陈英革，王小英，赵海，等. 任务调度过程中就绪队列的优化研究 [J]. 系统仿真学报，2006，18(4)：877-882

[3] Liu C L, Layland J W. Scheduling Algorithms for Multiprogramming in a Hard Real-time Environment [J]. Journal of ACM (S0004-5411), 1973, 20(1): 40-61

[4] Locke C D. Best-effort Decision Making for Real-Time Scheduling [D]. Dept. of Computer Science, Carnegie-Mellon University, 1986

[5] The Concise Handbook of Linux for Embedded Real-Time Systems Version 1.0 [M]. Timesys Corporation, 2002

[6] Harbour M G. Real-time POSIX: An overview [C]// Proc. of the VVConex'93 Int'l Conf. 1993

[7] Lehoczky J P, Sha L. Performance of Real-time Bus Scheduling Algorithm [C]// Proc. of the '86 ACM SIGMETRICS Joint Int'l Conf. on Computer Performance Modeling. New York, 1986: 44-53

[8] Cayssial R, Orozco J, Santos J, et al. Rate Monotonic Scheduling of Real-time Control Systems with the Minimum Number of Priority Levels [C]// Proc. of the 11th Euromicro Conf. on Real Time Systems. Oakland: IEEE Computer Society Press, 1999: 54-59

（下转第 239 页）

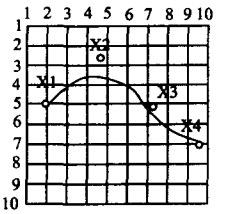
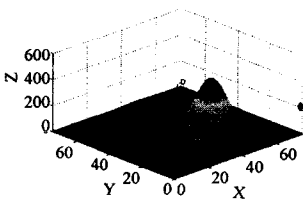


图 6　局部地图规划平面表示



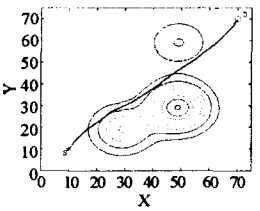
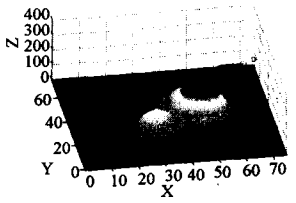


图 7　S(10,10,5),D(70,70,5);约束项 f_1 的权重 α_1 为 0.38,约束项 f_2 的权重 α_2 为 0.1,约束项 f_3 的权重 α_3 为 0.52 时,在地图 1 上产生的路径

图 8　对应于图 7 的等高线图



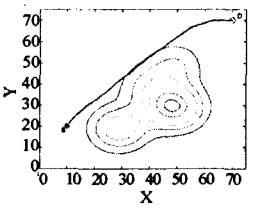
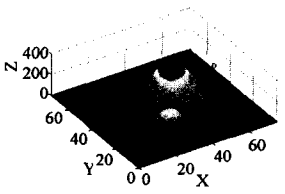


图 9　S(10,20,5),D(70,70,5);约束项 f_1 的权重 α_1 为 0.38,约束项 f_2 的权重 α_2 为 0.1,约束项 f_3 的权重 α_3 为 0.52 时,在地图 2 上产生的路径

图 10　对应于图 9 的等高线图



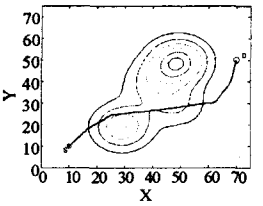
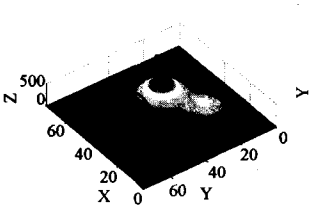


图 11　S(10,10,5),D(70,50,5);约束项 f_1 的权重 α_1 为 0.4,约束项 f_2 的权重 α_2 为 0.1,约束项 f_3 的权重 α_3 为 0.5 时,在地图 2 上产生的路径

图 12　对应于图 11 的等高线图



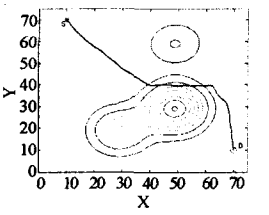


图 13　S(10,70,5),D(70,10,5);约束项 f_1 的权重 α_1 为 0.4,约束项 f_2 的权重 α_2 为 0.1,约束项 f_3 的权重 α_3 为 0.5 时,在地图 1 上产生的路径

图 14　对应于图 13 的等高线图

结束语　针对三维航迹在线规划问题,本文基于最小威胁曲面的概念,利用 B-Spline 函数逼近航迹在水平面内的投影,给出了一种基于量子粒子群优化的在线航迹规划方法。本方法克服了简单粒子群算法在高维空间寻优过程中易陷入局部最优的缺点,提高了搜索效率。采用上述方法,通过减少局部控制点、在代价函数中引入势能函数等来实现对飞行器航迹的实时规划,在线得到了符合各种约束条件的三维航迹。用这种方法得到的航迹是一种在线的次优解。

参 考 文 献

[1] Menon P K A, Kim E, Cheng V H L. Optimal Trajectory Synthesis for Terrain-Following Flight [J]. Journal of Guidance, Control and Dynamics, 1991, 14(4): 807-813

[2] 李思海,白存儒. 基于遗传算法的飞行器航迹规划研究[J]. 华东交通大学学报,2007,24(4):147-151

[3] 高晖,陈欣,夏云程. 无人机航迹规划综述[J]. 南京航空航天大学学报,2001,33(2):35-38

[4] 刘砚菊,杨青川,辜吟吟. 蚁群算法在机器人路径规划中的应用研究[J]. 计算机科学,2008,35(5):263-265

[5] 唐强,朱志强,王建元. 一种用于低空飞行的在线航迹重规划方法[J]. 西北工业大学学报,2005,23(2):271-275

[6] 李士勇,李盼池. 求解连续空间优化问题的量子粒子群算法[J]. 量子电子学报,2007(5):569-574

[7] 谢景权,须文波,孙俊. 基于量子粒子群算法的医学图像配准[J]. 计算机工程与设计,2008(2):430-432

[8] Nikolos I K, et al. Evolutionary Algorithm Based Offline/Online Path Planner for UAV Navigation [J]. IEEE Transactions on System, 2003(33): 898-912

[9] 郑昌文,李磊徐,帆江,等. 基于进化计算的无人飞行器多航迹规划[J]. 宇航学报,2005,26(2):223-227

[10] Sun J, Feng B, Xu W B. Particle swarm optimization with particles having quantum behavior [C]// USA: Proceedings of 2004 Congress on Evolutionary Computation. 2004: 325-331

(上接第 233 页)

[9] 宾雪莲,杨玉海,金士尧. 一种有限优先级的静态优先级分配算法[J]. 软件学报,2004,15(6):815-822

[10] Doytchinov B D, Lehoczky J P, Shreve S E. Real-time queues in heavy traffic with earliest-deadline-first queue discipline [J]. Annals of Applied Probability (S1050-5164), 2001, 11(2): 332-378

[11] Hansen J P, Zhu H, Lehoczky J P, et al. Quantized EDF Scheduling in a Stochastic Environment [C]// Proceedings of the International Parallel and Distributed Processing Symposium. 2002: 94

[12] Li Wenming, Kavi K, Akl R. A Non-preemptive Scheduling Algorithm for Soft Real-time Systems [J]. Computers & Electrical Engineering (S0045-7906), 2007, 33(1): 12-29

[13] Baker T P. Stack-based Scheduling of Realtime Processes [J]. Real-Time Systems (S0922-6443), 1991, 3(1): 67-99