一种静态优先级保序饱和分配算法

伍 微 倪少杰 刘小汇
(国防科技大学电子科学与工程学院 长沙 410073)

摘 要 在通信、雷达、导航以及各种消费类电子产品等领域,嵌入式实时调度已逐渐成为电子电气系统的控制核心,成本与性价比都是设计者需要考虑的重要内容。实际应用中,系统能够支持的优先级数目是有限的,当任务数目多于系统优先级数目时,RM,DM等优先级非受限最优算法尽管已经不再适用,但是仍然可以作为任务的自然优先级来辅助系统设计。利用自然优先级先验知识,提出一种保序饱和分配算法,用于任意截止期模型的最优保序分配。进一步的研究表明,当所有任务周期不小于其相对截止时间时,DM保序饱和分配是最少优先级分配。本算法复杂度低,可调度的判定总次数等于任务总数,远低于AGP和LNPA。

关键词 实时系统,有限优先级,优先级分配,饱和分配,截止期单调
中图法分类号 TP316 **文献标识码** A

Saturated Assignment Algorithm with Ordered Static Priority

WU Wei NI Shao-jie LIU Xiao-hui
(School of Electronic Science and Engineering, National University of Defense Technology, Changsha 410073, China)

Abstract In the area of communication, radar, navigation and various electronic production, embedded real-time scheduling has became the control kernel of those electronic and electrical systems, where the cost and the performance-price ratio are major concerns for the system designers. In practical applications, those systems only support limited priority levels when the task number is greater than the number of priority levels, those well-known optimal algorithms, such as DM(deadline monotonic) and RM(rate monotonic), are impractical. However, they can still provide natural priority to assist system design. A saturated assignment algorithm with ordered static priority was proposed based on transcendental knowledge of natural priority. It was proved to be the optimal ordered assignment. Further researches show that the saturated assignment with DM ordered priority leads to minimal priority levels, as long as any task is of deadline less than or equal to its period. Our method is of low time complexity, the number of scheduling determination is equal to the total task number, which is much less than the well-known AGP(assignment of priority group) and LNPA(least-number priority assignment).

Keywords Real-time system, Limited priority level, Priority assignment, Saturated assignment, DM(deadline monotonic)

静态优先级调度具有实现简单、调度开销小、系统超载时可预测性好等优点,在实时系统中得到了广泛的应用。已有的静态优先级最优分配算法,比如RM(rate monotonic)[1]、DM(deadline monotonic)[2]等通常不限制系统优先级的数目。当任意任务周期等于其相对截止时间时,RM是静态优先级最优分配算法;当任意任务周期不小于其相对截止时间时,DM是最优静态优先级分配算法[3]。

近年来随着市场需求和技术进步,静态优先级调度广泛应用于使用计算机或嵌入式微处理器的实时场合。实际系统能够支持的优先级数目是有限的,比如实时总线标准MSI STD BUS支持2个优先级[4],DSP/BIOS支持16个优先级[5],实时POSIX支持32个优先级[6],VxWorks支持256个优先级[7]。当任务数目多于系统优先级数目时,就需要给多个任务分配相同的系统优先级,此时已有的最优分配算法不再适用,而是需要考虑有限优先级情况下的优先级分配。最优分配往往意味着优先级数目最少。

目前,对有限优先级情况下最优分配的研究尚不充分。文献[8]提出了常量法,它使用固定数目的优先级分配,实现简单,但仅适用于任务周期与截止期相等的情况,且分配后任务集的可调度性下降很多。文献[9]提出一种AGP(assignment of priority group)算法,即适用截止期范围更广的最少优先级分配算法,即但一般情况下算法复杂度太高。文献[10]提出一种LNPA(least-number priority assignment)算法,即适用于任意截止期的最少优先级分配算法,其算法复杂度较高。AGP和LNPA共同的特点是系统优先级分配不必依照自然优先级,从而能够找到全局最优的分配方案;但是当

到稿日期:2009-01-15 返修日期:2009-03-25 本文受新世纪优秀人才支持计划(NCET-04-0995)资助。
伍 微(1981—),男,博士研究生,主要研究方向为嵌入式实时调度技术,E-mail:wuwei198106@163.com;倪少杰(1978—),男,博士生,讲师,主要研究方向为嵌入式系统、通信技术;刘小汇(1976—),女,博士生,讲师,主要研究方向为实时系统、导航通信技术等。

任务数目较多时，算法复杂度较高。文献[11]提出一种基于RM自然优先级的最优分配算法(本文沿用文献[10]的说法，称之为RM-Least)，其复杂度低，但仅适用于任务周期与截止期相等的情况。

本文针对现有研究的不足，利用自然优先级先验知识，提出一种保序饱和分配算法。此算法是适用于任意截止期模型的最优保序分配。进一步分析了所有任务周期不小于其相对截止时间的情况，此时DM保序饱和分配是最少优先级分配。本文算法复杂度低，可调度判定总次数等于任务总数。

1 任务模型

静态优先级调度n个周期任务组成的任务集$T=\{\tau_1,\tau_2,\cdots,\tau_n\}$。任务$\tau_i$可以表示为$\{C_i,T_i,D_i,p_i\}$，其中$C_i,T_i,D_i,p_i$分别表示任务$\tau_i$的最大执行时间、任务周期、相对截止时间和优先级。$\tau_i$执行一次称为一个实例，$\tau_i$的第$k$个实例为$\tau_i(k)$，其就绪时刻为$S_i(k)=kT_i(k=0,1,\cdots)$，截止时刻为$d_i(k)=S_i(k)+D_i$(即绝对截止时间)。

模型中，$\{C_i,T_i,D_i\}$是任务集已知参数，p_i则是待分配的优先级。为了便于区分，由RM,DM等优先级非受限系统中最优分配算法得到的优先级称为自然优先级，任务的自然优先级取值为$1,2,\cdots,n$；而在优先级数目限制为$m(m\leqslant n)$的系统中，分配的优先级则称为系统优先级，$p_i=1,2,\cdots,m$。本文以g_u表示系统优先级为u的任务组。优先级数值越大，则优先级越高。实际系统中，通常采用时间片轮转或先来先服务的方式调度同优先级任务。本文主要讨论时间片轮转方式，相关结论可以推广到先来先服务方式。

本文使用以下假设：

1)任务之间相互独立。

2)不存在优先级反转的情况。

3)任务抢占的时间开销很小，可以忽略不计。

如果对于系统优先级数目不受限制的情况，任务集不可调度，那么无论采用哪种静态优先级分配算法，该任务集都不可调度。因此，本文只讨论系统优先级数目无限时任务集可调度的情况。任务的自然优先级可由文献[1,2]中的算法决定，因此本文假设任务自然优先级是已知的。

2 基础理论

2.1 可调度判定准则

对于相对截止时间不大于周期(即$D_i\leqslant T_i$)的情况，任务τ_i可调度的充要条件是[12]：

$$\min_{0<t\leqslant D_i} W_i(t)/t\leqslant 1 \tag{1}$$

其中，

$$W_i(t)=\sum_{\tau_k\in hp(\tau_i)}\lceil t/T_k\rceil C_k+\sum_{\tau_l\in ip(\tau_i)} C_l \tag{2}$$

其中，$hp(\tau_i)$表示系统优先级高于τ_i的任务集合，$ip(\tau_i)$表示系统优先级等于τ_i的任务集合。

对于更一般的任意截止时间的情况，则需要考虑处理器最繁忙的时段。Liu[1]，Bini[13]的分析表明，最繁忙的时刻发生在所有任务同时就绪的临界时刻。在经典的level-i忙周期分析方法[14]中，使用忙周期表示处理器连续繁忙的时间段，系统优先级大于或等于p_i的任务在该时段内连续执行。level-i忙周期起始于临界时刻，即所有优先级大于或等于p_i的任务同时就绪的时刻。

使用level-i忙周期分析，τ_i第q个任务实体的响应时间满足下面的迭代关系式：

$$W_i(q)=(q+1)C_i+\sum_{\tau_j\in hpe(\tau_i)且j\neq i}\left\lceil\frac{W_i(q)}{T_j}\right\rceil C_j \tag{3}$$

其中，$hpe(\tau_i)$表示系统优先级高于或等于τ_i的任务集合。$q=Q$对应于忙周期即将结束的最后一个任务实例，Q的取值满足：

$$W_i(q)-(q+1)T\begin{cases}\leqslant 0, & q=Q\\ >0, & q=0,1,\cdots,Q-1\end{cases} \tag{4}$$

因此忙周期内任务τ_i的最长响应时间为：

$$r_i=\max_{q=0,1,\cdots Q}[W_i(q)-qT_i] \tag{5}$$

任务τ_i可调度的充要条件是：

$$r_i\leqslant D_i \tag{6}$$

2.2 优先级分配的重要性质

优先级分配需要将任务加入到具有特定优先级的任务组中。下面的定理说明了新加入任务对已分配优先级任务的影响。

定理1 已知优先级低于u的任务组$g_1,g_2,\cdots,g_{u-1}$都可调度。若将系统优先级高于u的任务加入g_u，则$g_1,g_2,\cdots,g_{u-1}$仍可调度。

证明：考察低优先级任务组$g_1,g_2,\cdots,g_{u-1}$中任务的响应时间。根据式(3)，高优先级任务加入g_u前后，低优先级任务响应时间表达式保持不变，从而其可调度性保持不变，式(6)仍然满足。因此$g_1,g_2,\cdots,g_{u-1}$仍可调度。得证。

定理2 已知任务组g_u可调度。若将系统优先级高于u的任务加入g_u，则g_u中原有任务仍可调度。

证明：考察任务组g_u中原任务的响应时间。根据式(3)，高优先级任务加入g_u前后，g_u中原任务响应时间表达式保持不变，从而其可调度性保持不变，式(6)仍然满足。因此g_u中原有任务仍可调度。得证。

定理3 已知优先级高于u的任务组$g_{u+1},g_{u+2},\cdots,g_m$都可调度。如果将任意任务$\tau_i$加入到任务组$g_u$，那么$g_{u+1},g_{u+2},\cdots,g_m$仍然是可调度的。

证明：考察高优先级任务组$g_{u+1},g_{u+2},\cdots,g_m$中任务的响应时间。根据式(3)，任意任务加入$g_u$前后，高优先级任务响应时间不会增加，式(6)仍然满足，因此$g_{u+1},g_{u+2},\cdots,g_m$仍可调度。得证。

2.3 优先级从低到高分配策略

AGP和RM-Least均采用了系统优先级从高到低的分配策略，这样每次分配低优先级任务时，都需要重新判断当前优先级其他任务的可调度性。LNPA采用了系统优先级从低到高的分配策略，根据定理1和定理2，每次分配高优先级任务时，不需要重新判断已分配任务的可调度性，从而极大地减少了任务可调度判定次数。本文采用系统优先级从低到高的分配策略。

3 保序饱和分配

3.1 优先级保序与饱和分配

定义保序的概念如下：

定义1 对于两个任务τ_i和τ_j，若τ_i的自然优先级高于τ_j，且τ_i的系统优先级不低于τ_j，则称τ_i,τ_j优先级保序；否则

称 τ_i，τ_j 优先级不保序。如果优先级分配算法使得任意两个任务都是优先级保序的，则称其为保序分配。

根据上述定义，优先级保序分配继承了任务的自然优先级。

性质 1 保序分配之后，若任务组的系统优先级较高，则其任务的自然优先级也较高。

证明：根据定义 1 可以直接得证。

性质 2 保序分配之后，任意任务组中所有任务的自然优先级连续。

证明：使用反证法。不失一般性，假设任务集保序分配后，某任务组 g_u 中存在两个自然优先级为 $i-1$ 和 $i+1$ 的任务 τ_{i-1}，τ_{i+1}，但不存在自然优先级为 i 的任务 τ_i。若 τ_i 的系统优先级高于 g_u，则 τ_i，τ_{i+1} 不保序；若 τ_i 的系统优先级低于 g_u，则 τ_i，τ_{i-1} 不保序。这与保序分配矛盾。得证。

定义 2 已知 g_u 可调度。若将优先级高于 g_u 的任务加入 g_u，新任务组仍可调度，则称 g_u 为不饱和任务组；否则，称 g_u 为饱和任务组。

根据上述定义，饱和任务组中不能再加入更多的任务。

定义 3 任务集优先级保序分配为 $g_1, g_2, \cdots, g_m$。若 $g_1, g_2, \cdots, g_{m-1}$ 都是饱和任务组，则称之为保序饱和分配。

3.2 保序饱和分配算法

保序饱和分配采用优先级从低到高的分配策略。除了最高优先级任务组之外，其余任务组都是饱和的。算法描述如下：

INPUT：Given task set T with natural priority ordered n periodic tasks。

OUTPUT：priority assignment $\{p_i\}$。

```
1. i=n   //i is the current task with lowest priority
2. g=1   //g is the current system priority level
3. while i>0 do   //start from the lowest priority
4.    while schedulable(τ_i, g) do
5.          p_i=g
6.          i--
7.    endwhile   //jump out until g is saturated
8.    g++
9. endwhile   //stop until assignment is finished
```

初始时，任务集按自然优先级从低到高的顺序，各系统优先级任务组置为空，当前系统优先级设为 1。将任务按顺序添加到当前系统优先级任务组，并将其从任务集合中删除，直到不存在当前系统优先级可调度的任务时，将当前系统优先级加 1。重复上述步骤，直至任务分配完毕。

3.3 算法特性

保序饱和分配算法按照优先级从低到高的顺序，每次可调度判定都能确定任务的系统优先级，总判定次数为 n。根据定理 1 和定理 2，每次为任务分配系统优先级时，已经分配优先级的任务仍然是可调度的，因此每次优先级分配都不会影响其他任务的可调度性。

保序饱和分配具有如下性质：

性质 3 保序饱和分配中，除了最高优先级任务组以外，其他任务组都是饱和的。

证明：根据定义 3 可以直接得证。

定理 4 保序饱和分配是优先级数目最少的保序分配。

证明：使用反证法。本文方法得到的可调度任务组集合为 $g_l, g_{l+1}, \cdots, g_m$，根据性质 3，$g_l, \cdots, g_{m-1}$ 都是保序饱和任务组。假设存在优先级数目更少的可行保序分配，那么 g_l，$g_{l+1}, \cdots, g_m$ 中必定有一个任务组（不妨设为 g_u）要拆分到其他任务组中去。由于 $g_l, \cdots, g_{u-1}$ 都是饱和任务组，因此 g_u 中不存在可以加入低优先级任务组的任务，根据保序分配性质 1 和性质 2，只能将 g_u 全部合入优先级较高的任务组 g_{u+1}。由于全部合入 g_u 之后，新的任务集是可调度的，因此 g_u 是不饱和任务组，从而 $u=m$。g_u 全部合入优先级较高的任务组 g_{m+1}，所得优先级数目并没有减少，矛盾。得证。

3.4 DM 保序饱和分配

根据前面的分析，保序饱和分配是最优保序分配，但一般说来，该分配只能得到局部最优解。本节将基于前述分析，考虑相对截止时间小于任务周期的情况，得到全局最优分配。

定义 4 已知任务 τ_i，τ_j 优先级不保序，且 τ_i 的自然优先级低于 τ_j。将 τ_i 从其任务组中删除，并加入 τ_j 所在的任务组，上述操作称为保序降级。若自然优先级是 RM，则称为 RM 保序降级。若自然优先级是 DM，则称为 DM 保序降级。

根据定义可知，保序降级之后，τ_i，τ_j 优先级相同，变为优先级保序。

性质 4 已知任务集可调度，若对 τ_i，τ_j 进行保序降级，则除任务 τ_i 之外的其他任务仍可调度。

证明：保序降级实际上是将高优先级任务 τ_i 加入到低优先级任务组 g_u（任务 τ_j 所在任务组）的操作。保序降级之后，根据定理 1，所有优先级低于 g_u 的任务仍可调度；根据定理 2，g_u 中的原任务也都可调度；根据定理 3，所有优先级高于 g_u 的任务仍可调度。因此除了任务 τ_i 之外的其他任务仍可调度。得证。

性质 5 保序降级之后，系统优先级数目不会增加。

证明：根据保序降级的定义，将任务加入已经存在的任务组中，这样并不会增加优先级数目，得证。

定理 5 已知可调度任务集满足 $D_i \leqslant T_i (\forall i)$。若任务 τ_i，τ_j 优先级 DM 不保序，则 τ_i，τ_j DM 保序降级之后，任务集仍可调度。

证明：不失一般性，假设任务 τ_i 的截止期不小于任务 τ_j 的截止期，即 $D_i \geqslant D_j$；原任务集中，τ_i 属于任务组 g_i，τ_j 属于任务组 g_j，且 g_i 的优先级高于 g_j。

根据性质 4，DM 保序降级之后，除任务 τ_i 之外的其他任务仍可调度，因此只需考察保序降级后 τ_i 的可调度性。将 τ_i 从任务组 g_i 中删除（得到新任务组 g_i'），加入到任务组 g_j 中（得到新任务组 g_j'），τ_i 的响应时间为：

$$\begin{aligned} W_i'(t) &= \sum_{\tau_k \in hp(g_j')} \lceil t/T_k \rceil C_k + \sum_{\tau_l \in ip(g_j')} C_l \\ &= \left(\sum_{\tau_k \in hp(g_j)} \lceil t/T_k \rceil C_k\right) - \lceil t/T_i \rceil C_i + \left(\sum_{\tau_l \in ip(g_j)} C_l\right) + C_i \\ &= W_j(t) + C_i(1 - \lceil t/T_i \rceil) \leqslant W_j(t) \end{aligned} \tag{7}$$

由于保序降级之前，τ_j 是可调度的，结合式(1)，式(7)以及已知条件 $D_i \geqslant D_j$ 可以得到：

$$\min_{0<t\leqslant D_i} W_i'(t)/t \leqslant \min_{0<t\leqslant D_j} W_i'(t)/t \leqslant \min_{0<t\leqslant D_j} W_j(t)/t \leqslant 1 \tag{8}$$

因此，τ_i 也是可调度的。得证。

定理 6 当任务集满足 $D_i \leqslant T_i (\forall i)$ 时，DM 保序饱和分配是最少优先级分配。

证明：根据定理 5，只要任务集满足 $D_i \leqslant T_i (\forall i)$，那么对

于任意存在 DM 不保序情况的可行调度，都可以通过 DM 保序降级，得到可行分配。根据性质 5，DM 保序降级后优先级数目不会增加，因此任意可行分配都可以通过若干次 DM 保序降级处理得到 DM 保序分配，且优先级数目不增加。

根据定理 4，饱和保序分配是优先级最少的保序分配，因此 DM 保序饱和分配是最少优先级分配。得证。

4 性能比较与分析

4.1 算法分配实例

本节使用实例说明已有算法和本文算法实现最少优先级分配的过程。

实例：表 1 给出一个含有 5 个任务的集合。表中 r_i 表示任务 τ_i 的最大响应时间。

表 1 任务集参数

自然优先级	C_i	$T_i=D_i$	r_i
1	2	5	2
2	1	10	3
3	2	14	5
4	2	14	9
5	2	14	12

表 2 列出了 AGP 算法的执行过程。“当前任务”表示当前加入任务组的任务，“判定次数”表示加入任务所需的可调度判定次数。

表 2 AGP 算法分配优先级

步骤	当前任务	任务组	判定次数	系统优先级
1	1		1	1
2	2	1	2	1
3	3	1,2	3	1
4		1,2,3	4	1
5	4		1	2
6	5	4	2	2
7		4,5	3	2

表 3 列出了 LNPA 算法的执行过程。“当前任务”表示当前加入任务组的任务，“判定次数”表示加入任务所需的可调度判定次数。

表 3 LNPA 算法分配优先级

步骤	当前任务	任务组	判定次数	系统优先级
1	3		5	2
2	4	3	4	2
3	5	3,4	3	2
4	2	3,4,5	2	2
5		2,3,4,5	1	2
6	1		1	1
7		1	0	1

表 4 列出了本文算法的执行过程。

表 4 本文算法分配优先级

步骤	当前任务	调度判定	系统优先级
1	3	schedulable(τ_3,1)? Yes	1
2	4	schedulable(τ_4,1)? Yes	1
3	5	schedulable(τ_5,1)? Yes	1
4	2	schedulable(τ_2,1)? Yes	1
5	1	schedulable(τ_1,1)? No	2

4.2 算法性能分析

RM-Least 复杂度是 $O(n)$，但其最优性仅适于 $D_i=T_i$ $(\forall i)$的情况。AGP 适合受限截止期模型以及任意截止期模型的一种特例，且在最坏情况下，AGP 的复杂度高达 $O((n-1)!/((m-1)!(n-m)!))$。LNPA 适合任意截止期模型，其算法复杂度为 $O(n^2)$。本文提出的保序饱和分配算法，适合在任意截止期模型中寻找最优保序分配，算法复杂度低，仅为 $O(n)$；进一步地，当 $D_i \leqslant T_i$ $(\forall i)$时，DM 保序饱和分配是最少优先级分配。

本文通过下面的实验，比较 DM 保序饱和分配算法、LNPA 和 AGP 的时间性能。图 1 给出了实验结果。实验条件如下：按任务集中任务数分组，任务数从 5 依次递增到 50。共生成 50 组任务集。每组任务集中各有 100 个任务集。其中的任务满足以下条件：

(1) 任务周期在[10,10000]上均匀分布，任务的相对截止时间等于周期。

(2) 系统优先级数目无限时，任务集静态优先级可调度。

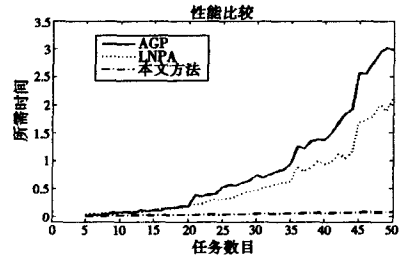

图 1 本文算法与 LNPA，AGP 算法的时间性能比较

由图 1 可以看出，由于使用了 DM 自然优先级的先验知识，本文算法的时间复杂度远优于 LNPA 和 AGP。

结束语 尽管在优先级非受限系统中，已经存在 RM，DM 等最优分配算法，但实际系统能够支持的优先级数目总是有限的，最优优先级分配往往意味着优先级数目最少。本文利用自然优先级先验知识，提出一种保序饱和分配算法，是适用于任意截止期模型的最优保序分配。进一步的研究表明，当所有任务周期不小于其相对截止时间时，本文提出的 DM 保序饱和分配是最少优先级分配。

对于任务周期大于其相对截止时间的情况，本文算法虽然无法得到全局最优的优先级分配方案，但是本文算法给出了局部最优的分配方案，且与 AGP，LNPA 等算法相比，可调度判定次数仅为任务数目，大大降低了算法复杂度。此外，还提出并证明了 DM 保序饱和分配是全局最优分配的适用条件，对于实时系统设计与实现来说，具有重要指导意义。

参 考 文 献

[1] Liu C L, Layland J W. Scheduling algorithms for multi - programming in a hard real-time environment[J]. Journal of the ACM, 1973, 20(1): 40-61

[2] Leung J Y T, Whitehead J. On the Complexity of Fixed-priority Scheduling of Periodic Real-time Tasks[J]. Performance Evaluation (Netherlands), 1982, 2(4): 237-250

[3] Audsley N, Burns A, Richardson M, et al. Applying New Scheduling Theory to Static Priority Pre-emptive Scheduling[J]. Software Eng. J., 1993, 8(5): 284-292

[4] MSI-C851 STD BUS 80C51 Microcontroller Card. Microcomputer systems [EB/OL]. http: // www. microcomputersystems. com

[5] Oshana R. DSP Software Development Techniques for Embedded and Real-time Systems[M]. Newnes, 2005

[6] Harbour M G. Real-time POSIX: An overview[EB/OL]. Proc. of the VVConex'93 Int'l Conf. 1993. http: // www. ctr. unican.

es/ publications/mgh-19931. pdf
[7] VxWorks Programmers guide [EB/OL]. http : // www . slac . stanford. edu/exp/glast /flight/sw/vxdocs/vxworks/guide/c-basic. html # 100061
[8] Lehoczky J P, Sha L. Performance of real-time bus scheduling algorithms[C]// Proc. of the '86 ACM SIGMETRICS Joint Int'l Conf. on Computer Performance Modeling, Measurement and Evaluation. New York: ACM Press, 1986: 44-53
[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] 邢建生，王永吉，刘军祥，等. 一种静态最少优先级分配算法[J]. 软件学报，2007，18(7)：1844-1854
[11] Cayssials 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
[12] Katcher D I, Sathaye S S, Strosnider J K. Fixed priority scheduling with limited priority levels[J]. IEEE Transactions on Computers, 1995: 1140-1144
[13] Bini E, Buttazzo G C. Schedulability Analysis of Periodic Fixed Priority Systems[J]. ACM, 2004: 1-12
[14] Tindell K W, Burns A, Wellings A. An extendible approach for analyzing fixed priority hard real-time tasks[J]. The Journal of Real-time Systems, 1994, 6(2): 133-152

（上接第 40 页）

着检测器的生命周期 L 的增加而增加。L 从 10 增至 20，FP 增幅较小，且变化平稳；在 L 从 20 增至 30，FP 有了明显增加；当 $L=50$ 时，FP 变化剧烈，检测效果就比较差。

图 3 是检测器的 TP，FP 与生命周期 L 的关系图。其中，纵坐标为检测百分比(单位是 100%)，横坐标为进化代数(单位是 100)。蓝色曲线标示 TP 的变化，绿色曲线显示 FP 的走向。

这是因为在生命周期 L 增大的同时，成熟检测器的规模随之增加，就有更多的候选检测器有机会被激活成为记忆检测器，成熟检测器整体的数量和比例在一次迭代中都提高了，所以带来了较高的检测率 TP。较长的生存周期里，成熟检测器保留的时间也增长，一定程度上也延缓了检测器更新换代的频率，影响了种群的多样性。所以可以看到，L 增加到 50 时，由于检测器对异常模式的评估会出现偏差，即不能准确地区别自我和非自我，从而导致 FP 的猛烈增加。L 缩短的情况与 T 增大类似，引起了 FP 和 TP 的下降。

综上，根据试验中检测率 TP 和误报率 FP 之间的情况，选择生存周期 $L=20$，$T=10$。

4.2 实验结果及分析

选定实验中的参数 $T=10$，$L=20$ 后，嵌入否定算子前后的两个克隆选择算法的不同之处就只是检测器的演化过程。因此，为避免影响算法间的性能比较，选取相同样本数据集，采用实数编码，选取相同交叉、变异操作，将没有嵌入否定选择算子的克隆选择算法的实验结果与嵌入否定选择算子后算法的实验结果相对比，可以看到检测器的性能变化，如图 4 所示。

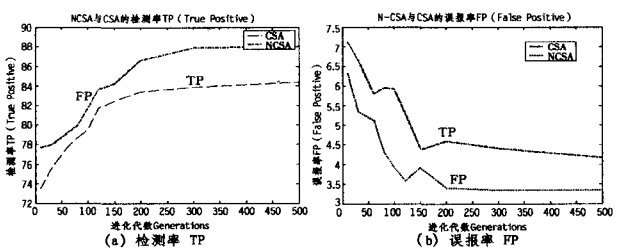

图 4 否定选择算子嵌入前后算法的 TP 与 FP 变化(检测器大小不变)

图 4(a)和图 4(b)是否定选择算子嵌入前后算法的 TP 与 FP 变化图。在检测器规模大小不变的情况下，NCSA 具有较高的非自我检测率和较低的误报率，NCSA 整体的检测性能优于 CSA，证明了嵌入否定选择算子是有效的。同时检测的准确性能也表明，NCSA 能通过检测器的更新识别未知攻击模式，减少了误报率，能更好检测未知入侵，适应网格复杂动态变化的环境。

结束语 由于基于人工免疫机制的网格入侵检测研究还处于起步阶段，在免疫算法、网格入侵检测器的实际推广、性能优化方面还要进一步研究。同时，免疫计算的群体性，要求检测器的数目更多，随着网格节点增加，网格入侵检测器的部署和管理问题也需要进一步解决。

目前解决的都是网格节点上的离线数据的检测，下一步将收集在线数据，应用该算法进行检测以适应入侵检测的实时性，保障网格环境安全性。

参 考 文 献

[1] Foster I, Kesselman C. The Grid 2: Blueprint for a Future Computing Infrastructure[M]. Morgan Kaufmann, San Francisco, 2005
[2] 褚永刚. 大规模分布式入侵检测系统关键技术研究[D]. 北京：北京邮电大学，2005
[3] 李涛. 计算机免疫[M]. 北京：电子工业出版社，2004
[4] Forrest S, Peterlson A, Allen L, et al. Self-nonself discrimination in a Computer[C]// Proceedings of the 1994 IEEE Symposium on Research in Security and Privacy. IEEE Computer Society Press, 1994
[5] Hofmeyr S, Forrest S. Immunity by Design: An Artificial Immune System[C]// Proceedings of the Genetic and Evolutionary Computation Conference. Morgan Kaufmann, San Francisco, 1999
[6] Kim J, Bentley P. Immune Memory in the Dynamic Clonal Selection Algorithm[C]// Proceedings of ICARIS 2002. 2002
[7] Kim J. Integrating Artificial Immune Algorithms for Intrusion Detection[D]. Department of Computer Science, University of London, 2002
[8] Dasgupta D. Immunity-Based intrusion detection system: A general framework[C]// Proc. of the 22nd NISSC. 1999
[9] Dasgupta D, Yu S, Majumdar N S. MILA: Multilevel Immune Learning Algorithm[C]// Proceedings of the Genetic and Evolutionary Computation Conference. Chicago, July 2004
[10] Dasgupta D, Rodriguez J, Balachandran S. Mining Security Events in Cougaar Agent Society[C]// Proceedings of SPIE Defense and Security Symposium'2006. Orlando, Florida, 2006
[11] 陈荣，高济，郭航. 面向网格计算的按需入侵检测模型[J]. 浙江大学学报：工学版，2006，40(3)
[12] 魏宇欣，武穆清. 智能网格入侵检测系统[J]. 软件学报，2006，17
[13] 倪建成，李志蜀，孙飞显，等. 基于免疫 Multi-agent 的网格入侵检测模型[J]. 计算机工程，2007，33(8)
[14] http: //kdd. ics. uci. edu/databases/kddcup99/kddcup99. html