文章编号:1001 - 9081(2007)04 - 0932 - 04

基于蚁群算法的 Petri 网最优路径序列寻找

黄光球,苏海洋,刘 冠
(西安建筑科技大学 管理学院,陕西 西安 710055)
(huangnan93@sohu.com)

摘 要:根据蚁群算法和时间 Petri 网的特点提出了一种网络元素可以记录少量信息的记忆扩展时间 Petri 网(METPN)。当 METPN 运行时,使用充足量的托肯在网络中行走并在行走过程中留下信息素来调整托肯的路径选择,从而使大量蚂蚁的行走路线不断逼近 Petri 网中时间延迟更短的变迁序列,最终在最短变迁序列上形成清晰的蚁路,从而在一定程度上解决了复杂 Petri 网的最优路径寻找问题。仿真结果表明,托肯可以有效地在最短延时路径上形成蚁路,能够求得从初始库所到网络中任意库所的最短路径。

关键词:蚁群算法;Petri 网;路径序列;优化
中图分类号: TP18;TP301.5 **文献标识码**:A

Optimum route sequence search in Petri net based on ant colony algorithm

HUANG Guang-qiu, SU Hai-yang, LIU Guan
(*School of Management, Xi'an University of Architecture and Technology, Xi'an Shaanxi* 710055, *China*)

Abstract: A Memory Extended Timed Petri Net (METPN) whose elements can record a little information was proposed based on the ant colony optimization algorithm and the features of the Timed Petri Net (TPN). When METPN was running, enough tokens walked and left odor in METPN so that route selections of tokens could be adjusted, in this way it made lots of ant walk routes to approach the transitional sequences with less delay. At last a clear ant walk route could be found on the transitional sequence with the least delay, and the route search problem of complex TPN was solved to certain extent. The result of the simulation shows that the ant walk route is formed along the least delay route effectively by tokens, and the shortest route from initial places to every place of METPN can be gotten.

Key words: ant colony algorithm; Petri net; route sequence; optimization

由于 Petri 网对问题的描述非常全面和细致,以至于在很多应用中显得相当复杂[1]。而复杂 Petri 网的主干路径一直是分析和解决 Petri 网问题的重点和难点。本文不针对特定的网型与应用,设法解决大型复杂 Petri 网的路径寻找问题,可解决如多服务多任务资源调度问题[2]。鉴于 Petri 网与蚁群算法[3]的常用案例相比具有路径并非全连接,且网型固定,网段具有实际意义的特点,本文将蚁群算法的变迁选择概率公式做了必要的修改,从而可以灵活地进行蚁路引导,或者通过一定方式控制 Petri 网只进行有限搜索。因为变迁的实际延迟是可知的,本文也将延时引入路径选择公式,使得选择不同路径的概率差异被加强,有利于加快搜索速度。此外,本文采用两种策略来避免托肯进入局部最小点:1)以概率来调整信息素,当系统陷入局部极小点时可以用小概率变迁的发生找到更短的路径;2)调整变迁的实施度[4],使某条路径的输入输出速度统一,从而在整体上选择出具有短延时的变迁序列。

1 算法定义

常用的赋时变迁 Petri 网(Timed Transition Petri Nets, TTPN)的定义不能够精确表达变迁在任意延迟情况下的 Petri 网特性,因此本文定义了一种记忆扩展的时间 Petri 网(Memory Extended Timed Petri Net, METPN),该网型要求变迁、库所、托肯都记录少量的信息。

定义 1 $Token = (r_1, r_2, \cdots, r_v, F)$ 为托肯的数据结构,其中 r_j 的值为一个变迁索引号,$j = 1,2,\cdots,v$,v 为托肯走过的变迁数;$r_1, r_2, \cdots, r_v$ 序列标识出通过变迁的前后顺序;F 为下列计时函数的值:

$$F = F(r_1, r_2, \cdots, r_v) = \sum_{j=1}^{v} \lambda_{r_j} \tag{1}$$

式中,$\lambda = \{\lambda_1, \lambda_2, \cdots, \lambda_m\}$,为各个变迁的时间延迟集合,$\lambda_{r_j}$ 是索引号为 r_j 的变迁的时间延迟,F 记录本托肯在网中行走从开始到当前花的总时间。

定义 2 $METPN = \{P, S, T, \tau, Fl, W, \lambda, K\}$,其中,$P = \{p_1, p_2, \cdots, p_n\}$ 是库所的有限集合,n 为库所的个数,且 $n > 0$;$S = \{s_1, s_2, \cdots, s_n\}$ 与 P 中各元素一一对应,$s_i (i = 1,2,\cdots,n)$ 与托肯的数据结构相同,存储着通过对应库所 $p_i (i = 1,2,\cdots,n)$ 的所有托肯中,计时函数 F 的值最小的一个托肯元素;F_{s_i} 表示当前 s_i 的计时函数值;$T = \{t_1, t_2, \cdots, t_m\}$ 是变迁的有限集合,m 表示变迁的个数,且 $m > 0$;$P \cap T = \varnothing$;$\tau = \{\tau_1, \tau_2,$

收稿日期:2006 - 10 - 20;修订日期:2006 - 12 - 27 基金项目:陕西省自然科学基金资助项目(2005F38);陕西省教育厅专项基金资助项目(06JK258);西安建筑科技大学基础研究基金项目(JC0616)

作者简介:黄光球(1964 -),男,湖南桃源人,教授,博士生导师,博士,主要研究方向:计算机仿真、电子商务与网络安全; 苏海洋(1981 -),男,陕西延安人,硕士研究生,主要研究方向:计算机仿真、系统工程; 刘冠(1981 -),男,陕西富平人,硕士研究生,主要研究方向:计算机仿真、系统工

$\cdots,\tau_m\}$ 与 T 中元素一一对应,存储对应变迁上的信息素,是变迁实施时留下的,也是托肯选择路径的主要依据;$Fl \subseteq P*T \cup T*P$($*$ 为笛卡儿积);$dom(Fl) \cup cod(Fl) = P \cup T$,其中 $dom(Fl) = \{x \mid \exists y:(x,y) \in Fl\}$,$cod(Fl) = \{y \mid \exists x:(x,y) \in Fl\}$,即它们分别是 Fl 的定义域和值域;$W:Fl \to N$ 称为 N 上的权函数,对 $(x,y) \in Fl$,$W(x,y) = W((x,y))$ 称为 (x,y) 上的权;λ 与 T 中元素一一对应,λ 集合中的数字为事先测定的值,且在 Petri 网的运行过程中保持不变;$K = \{k_1,k_2,\cdots,k_n\}$ 定义为 $P \to N \cup \{\infty\}$ 为 P 上的容量函数。

设 $M_0[\sigma > M_k$ 表示 Petri 网由初始状态 M_0 经过路径序列[5] σ 到达某状态 M_k,在这里假设序列 σ 中的第 k 个路径向量 $\boldsymbol{H}_k$ 是一个仅有 1 个元素不为零的 m 维列向量。Petri 网的状态方程写作[5]:

$$\boldsymbol{M}_k = \boldsymbol{M}_0 + \boldsymbol{C}\sum_{k=1}^{|\sigma|}\boldsymbol{H}_k$$

式中,$\boldsymbol{C}$ 为关联矩阵,$|\sigma|$ 为路径序列长度。令 $U = \sum_{k=1}^{|\sigma|}\boldsymbol{H}_k$,则有:

$$u_i = \sum_{k=1}^{|\sigma|} h_{ki} \tag{2}$$

其中,$U = (u_1,u_2,\cdots,u_m)^T$,$\boldsymbol{H}_k = (h_{k1},h_{k2},\cdots,h_{km})^T$。

路径序列必须满足式(2),序列中每个路径向量必须满足变迁路径条件(3):

$$\boldsymbol{C} = \boldsymbol{C}^+ - \boldsymbol{C}^-,\ \boldsymbol{M}_{k-1} \geqslant \boldsymbol{C}^-\boldsymbol{H}_k \tag{3}$$

2　算法描述

在路径寻找方面蚁群算法表现出了良好的特性。在不知道路径所耗时间的情况下,算法的引导方式只能是纯粹的信息素引导,而在 Petri 网中,变迁的延迟信息是可知的,所以在计算路径选择概率时引入变迁的时间延迟,强化了长时延变迁与短时延变迁间的差异,使 Petri 网从一开始就有了寻找最短序列的倾向。另外,鉴于 Petri 网和蚁群算法的常用案例相比具有路径并非全连接,且网型固定,网段具有实际意义的特点,本算法提供参数使我们可以根据具体问题引导蚁路,这使蚁路与实际问题结合更紧密。此外,为了自动纠正局部最优并非全局最优的路径选择,在初期假设每个变迁的可实施度[4]都为一个小整数(例如 2 或者 3),也就是说可以同时接受两个或三个实施,在网络运行中按照一定规则修改实施度,从而平衡无选择路径上各个变迁的实施速度,确保高速网段的信息素比低速网段的信息素更浓,使系统运行由局部极小点跳入全局最小点。

对每个库所 $p_i(i = 1,2,\cdots,n)$,设其目标函数为:

$$\min_{r_1,r_2,\cdots,r_v} F_i(r_1,r_2,\cdots,r_v) = \sum_{j=1}^{v}\lambda_{r_j},\ i = 1,2,\cdots,n$$

它存在于每个库所 p_i 对应的 s_i 元素中,记录当前发现的到达该库所所用的最小时间。当 Petri 网运行结束时,s_i 保存了托肯当前发现的,由初始某库所到 p_i 的最短路径序列和耗用的最短时间值。

在系统的初始化阶段,各个变迁上的信息素浓度相等且为一自然数 z,即 $\tau_i = z,i = 1,2,\cdots,m$。托肯的个数为 $\boldsymbol{M}_0$ 中所有不为"0"的数字之和 a 的 x 倍,$a*x,x \in N$。我们视 a 为蚂蚁的小组,x 为蚂蚁群中的蚂蚁的小组数目。然后使蚂蚁由 $\boldsymbol{M}_0$ 所表示的库所,按照 $\boldsymbol{M}_0$ 库所中所标识的各个库所中托肯的比例进入 Petri 网。

在蚁群算法中用走完路径才调整信息素的方式并不适合 Petri 网的实际情况,因为 Petri 网运行过程中托肯随时可能消失或增多,所以必须用蚁群算法中的蚁密算法[5] 来调整信息素。但用蚁密算法引导蚁路时,托肯非常容易被个别短延时变迁所吸引,如果短延时变迁后有延迟长的变迁,将会在中间库所中积累大量的托肯,使托肯资源被无端的消耗,甚至不能产生预期结果。通过分析 Petri 网的运行规则,本文运用修改实施度[4] 机制来消除这种过度吸引托肯的现象。

用 $act(t_i)$ 表示变迁 t_i 的实施度,初始状态 $act_{old}(t_i) = z$,$i = 1,2,\cdots,m$。在确定的路径上托肯以 Petri 网的激发规则移动。设向量 $\boldsymbol{H}_h$ 为一个 m 维列向量,向量中第 h 个元素为 1,其他元素为零($1 \leqslant h \leqslant m$),并且 $\boldsymbol{H}_h$ 满足式(3),则有 $M_d[t_h > M_{d+1}$。在变迁 t_h 实施前需要决定本次实施度:

$$act_{new}(t_h) = \begin{cases} act_{old}(t_h) + 1, & \forall p_{in} \in {}^*t_h, M(p_{in}) \geqslant act_{old}(t_h) \times W(p_{in},t_h), \forall p_{out} \in t_h^*, M(p_{out}) = 0 \\ act_{old}(t_h) - 1, & \forall p_{out} \in t_h^*, M(p_{out}) \geqslant act_{old}(t_h) \times W(t_h,p_{out}), \forall p_{ij} \in {}^*t_h, M(p_{in}) = 0 \\ act_{old}(t_h), & \text{其他情况} \end{cases}$$

式中,$M(p)$ 表示库所 p 的托肯数,p_{in} 和 p_{out} 分别表示输入、输出库所集合中的元素。

在进行下次判断实施度时,应令:

$act_{old}(t_h) = act_{new}(t_h)$

这样对于某条无选择的路径来讲,短延时变迁的实施度将有被提高的倾向,长延时变迁的实施度则有被降低的倾向,当大量的托肯在该路径上运行一段时间后,此路径上各个变迁的输入、输出速度将达到一个平衡值。因为此平衡值只由各个变迁延时决定,所以它反映该无选择路径上各变迁的平均延迟信息,蚁路将以较高概率在平衡值高的路径上形成。

变迁 t_h 的信息素[3] τ_h 依据式(4)调整,其初始值 $\tau_h^{old} = z$:

$$\begin{aligned} &\Delta\tau(t_h) = A/\lambda_h \\ &\tau_h^{new} = \rho^w\tau_h^{old} + \Delta\tau(t_h) \times act_{new}(t_h) \end{aligned} \tag{4}$$

在进行下一轮计算时,应令:

$\tau_h^{old} = \tau_h^{new}$

以上各式中,ρ 为每个时间单位(这里是 s)挥发后留下的信息素比例;w 为 Petri 网上次实施变迁到本次实施所经过的时间长度;A 为常数,表示每只蚂蚁在做一次变迁时留给所走过变迁的信息素总量;λ_h 为变迁 t_h 的时间延迟,类似于蚁群算法中的路径长度,用 A/λ_h 表示单位时间释放信息素的量,反映每实施一次该变迁的浓度变化;τ_h^{old} 和 τ_h^{new} 分别是实施前、实施后的变迁浓度。

公式(4)表明,变迁 t_h 上的信息素调整后的浓度是相邻两次实施的间隔时间段内信息素挥发后剩下浓度与新加入信息素浓度的和。这样,那些最后一次实施的变迁的信息素将不会进行调整,也就是说,网络运行时信息素以时间为单位进行调整。当网络运行结束,每个变迁上的信息素保持最后一次调整的结果。

要找的变迁序列是针对一个库所而言的,求得从 $\boldsymbol{M}_0$ 所标识的某个库所到该库所之间连接最短的变迁序列,而关于最

短序列的信息必须由托肯提供，所以托肯数据结构中的各元素也要做相应调整。设变迁 t_h 的输入托肯集合为：

$$Token_{T_{in}} = \{Token_1^{T_{in}}, Token_2^{T_{in}}, \cdots, Token_{in}^{T_{in}}\}$$

式中，x_{in} 为进入变迁的托肯数。每个托肯本身就保存一个序列表和一个由此序列表通过式(1)计算出的计时 F 函数值。当托肯到达变迁后，首先将变迁索引 t_h 加入 $Token_i^{T_{in}}$ $(i=1,2,\cdots,x_{in})$ 的序列表，然后将 $Token_i^{T_{in}}$ $(i=1,2,\cdots,x_{in})$ 序列表第一次出现 t_h 后的序列清空，以此处理走过环状路径而返回的托肯序列，保证每个变迁索引号只出现一次。再由式(1)得到 $Token_i^{T_{in}}(i=1,2,\cdots,x_{in})$ 的 F 元素的值并存入托肯，设 $F_{Token_i^{T_{ij}}}$ 为当前 $Token_i^{T_{ij}}(i=1,2,\cdots,x_{in})$ 的 F 元素值，那么输出给 t_h^* 中各库所的托肯 $Token_{T_{out}}$：

$$Token_{T_{out}} = Token_i^{T_{in}}, \text{ 如果 } F_{Token_i^{T_{in}}} = \max\{F_{Token_1^{T_{in}}}, F_{Token_2^{T_{in}}}, \cdots, F_{Token_{x_{in}}^{T_{in}}}\} \quad (5)$$

式中，$F_{Token_{T_{out}}}$ 表示 $Token_{T_{out}}$ 的 F 函数值。式(5)表明变迁的输出托肯所记录的序列是输入托肯中计时 F 函数值最大的托肯所保存的序列，变迁的输出托肯所记录的 F 值即为此最大值。当托肯在变迁中停留 λ_h 时间后，变迁将 $Token_{Tout}$ 按照权值分发到各个输出库所。

对于 $\forall p_i \in t_h^*$，s_i 与 p_i 对应，$i=1,2,\cdots,x_{out}$，x_{out} 为 t_h^* 中元素个数，初始状态 $s_i^{old}(i=1,2,\cdots,n)$ 的序列表为空，$F_{S_i^{old}}=0$，则有：

$$s_i^{new} = \begin{cases} s_i^{old}, & \text{如果 } F_{Token_{T_{out}}} > F_{s_i^{old}} \text{ 且 } F_{s_i^{old}} \neq 0 \\ Token_{T_{out}}, & \text{如果 } F_{Token_{T_{out}}} \leqslant F_{s_i^{old}} \\ Token_{T_{out}}, & \text{如果 } F_{s_i^{old}} = 0 \end{cases} \quad (6)$$

$$Token_{p_{out}} = \begin{cases} s_i^{old}, & \text{如果 } F_{Token_{T_{out}}} > F_{s_i^{old}} \text{ 且 } F_{s_i^{old}} \neq 0 \\ Token_{T_{out}}, & \text{如果 } F_{Token_{T_{out}}} \leqslant F_{s_i^{old}} \\ Token_{T_{out}}, & \text{如果 } F_{s_i^{old}} = 0 \end{cases} \quad (7)$$

式中，库所 p_i 是变迁 t_h 输出库所集合中的任意一个元素；s_i^{old} 表示 $Token_{T_{out}}$ 来到 p_i 前 s_i 各元素取值情况；s_i^{new} 表示 $Token_{Tout}$ 来到 p_i 后 s_i 的值；$F_{s_i^{old}}$ 为 s_i^{old} 的 F 元素值；$Token_{p_{out}}$ 表示 p_i 的输出托肯。因为托肯与 s 元素有相同的数据结构，所以 $s_i^{new} = Token_{T_{out}}$ 表示将 $Token_{T_{out}}$ 的内容交给 s_i 保存。通过以上运算确保每个库所 $p_i(i=1,2,\cdots,x_{out})$ 都保存一个非空记录 $s_i(i=1,2,\cdots,x_{out})$，$s_i$ 记录了从某起始库所到本库所当前发现的最短路径序列。

如果，t_h 的一次实施后 p_i 得到多于一个托肯，那么依式(6)，(7)逐个处理每个托肯。注意，在下一轮计算时，应令：

$$s_i^{old} = s_i^{new},\ i=1,2,\cdots,n$$

当实施的变迁并非一个而是很多个时，必须满足式(3)。

在复杂网的路径寻找中，散漫地遍历整个 Petri 网寻找变迁序列虽然可能找到最优路径，但效率相当低。在这里应用蚁群算法的原理，一方面集中托肯在很有可能出现最优序列的路径上寻找局部最优，一方面调整整体的搜索方向确定多个局部最优来对比得到全局最优。

当托肯遇到了“与”路径，即一个变迁需要多个输入，则停止等待，直到其他库所处于就绪状态然后按照 Petri 网变迁规则和上述规则实施变迁。当托肯遇到了“或”路径，即一个库所有多个输出路径可以选择，则托肯可以以一定概率来选择所要激发的变迁，其概率用式(8)来确定：

$$Q_{kij} = \begin{cases} \dfrac{\eta_{ij}^b(t_j)\tau_j^a(1/\lambda_j)^\gamma}{\sum_{s\in allowed_k}\eta_{is}^\beta(t_s)\tau_s^a(1/\lambda_s)^\gamma}, & allowed_k \neq \varnothing \\ 0, & \text{否则} \end{cases} \quad (8)$$

式中，$allowed_k = \{t_1, t_2, \cdots, t_a\}$ 表示托肯 k 下一步允许选择的变迁，a 表示本库所中的托肯允许选择的变迁数，该集合中的数据由基本 Petri 网变迁实施条件而决定。由式(8)可知，托肯 k 在库所 p_i 向变迁 t_j 的转移概率 Q_{kij} 与 $\eta_{ij}^b(t_j)\tau_j^a(1/\lambda_j)^\gamma$ 成正比。τ_j 为当前变迁 t_j 上信息素的量，$\eta_{ij}(t_j)$ 为由 p_i 到 t_j 的引导因数；$1/\lambda_j$ 为可见度，延时 λ_j 越小可见度越大，被选到中概率也越大。α，β 和 γ 分别反映了托肯在运动过程中所积累的信息素，引导因数和路径可见度在托肯选择路径时的相对重要性。

与参考文献[6,7]中的概率公式相比，我们看到常用的蚁群算法在做路径选择时只用信息素和时间延迟作为选择依据。这样，搜索资源会平等地考察每个分支，即遍历所有分支。然而在 Petri 网中，每个变迁都具有实际意义，在应用中有时也会有外加条件，即某个变迁必须被触发，某个变迁一定不能被触发，或者应用者只想进行局部搜索。在这些情况下常用的概率公式可能会找不到所要求的结果，至少要遍历全网来找到所要求的结果。为了进一步使 Petri 网与应用靠近，本文引入了人工引导因子，这样可以根据实际经验和具体应用来引导蚁路，从而提高效率，扩大本网的应用范围。

$\eta_{ij}(t_j)$ 为从库所 p_i 到变迁 t_j 的人工引导因子，是实际应用时所用的一个变量，默认为 1，表示花费同样的代价来搜索和评比每个分支。实际工作中，并非所有的分支都有必要平等地占用搜索资源。为避免不必要的托肯浪费，式(8)提供了可以灵活进行网络引导的方法，即在分支路径处将通往复杂网段的人工引导因子值设高，使托肯有高概率进入该网段；将通往简单网段的人工引导因子值设低，使托肯有低概率进入简单网段；或者通过修改此因子将不愿意涉及的分支路径设为 0，将通往目标网段分支设为 1，从而将托肯引导入某个具体局部网段进行搜索。以上策略可提高本算法的搜索效率和适用范围。

统运行的结束条件是网络各个库所对应的 s 元素中计时函数 F 的值在一段较长的时间内不变化，即 $F_{s_i}=c, i=1,2,\cdots,n$，c 为一个常数。

3 仿真实例

以上算法可以应用到各种类型的 Petri 网，本文用图 1 所示的 Petri 网为例说明该算法。

图 1 中变迁 1，2，…，14 对应延迟 λ_j 分别为：5、5、4、3、3、8、3、9、16、6、4、6、2、8。本文参数根据文献[3]的建议设定参数为：库所 1，2，3，4，5 中初始托肯数各为 40，信息素挥发系数 $\rho=0.6$，每次涂上的信息素总量 $A=40$。在或变迁的路径选择公式中 $\alpha=0.5$，$\beta=0.5$，$\gamma=1$，$\eta_{ij}=1$，初值 $\tau_j=10$[6]，$i=1,2,\cdots,14$，$j=1,2,\cdots,m$。

图 2、图 3 分别表示了在时刻 0s、31s 时仿真计算情况。图 2

中变迁 1 ~ 14 的信息素均为 10,可用实施度均为 3;图 3 中变迁 1 ~ 14 的信息数分别为:158.44, 148.78, 113.62, 382.96, 230.93, 71.00, 218.00, 23.33, 13.28,10.00,36.00,10.00, 10.00,10.00;可用实施度分别为 - 98, - 96, - 49,11,11,5, 4,17,32,6,3,3,0,3。其他时刻的仿真计算情况类似,此处不再赘述。

顺便指出,图 2 和图 3 中比图 1 多出的库所是用来在程序运行中实现实施度的库所。库所中的点为托肯,变迁中的点表示信息素浓度,变迁中的数字为延迟,可用实施度为当前还可以接受任务的实施度,小于零时表示不能再接受实施。

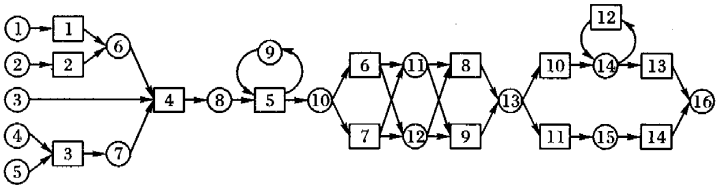

图 1　Petri 网仿真用例

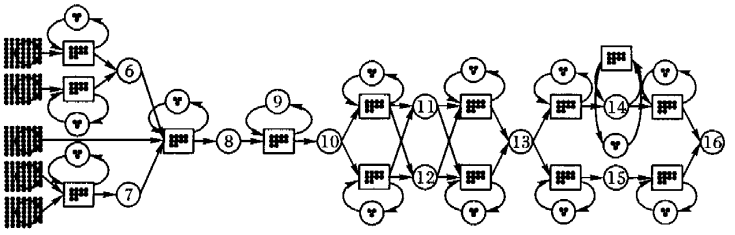

图 2　运行变迁 0 秒时库所可用实施度与变迁信息素情况

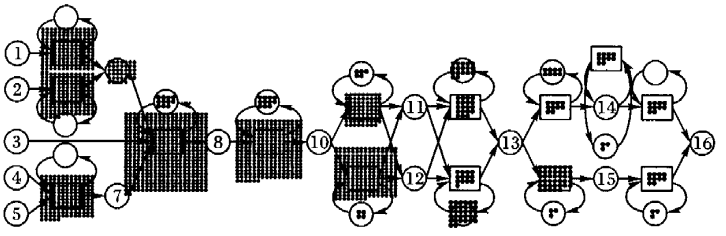

图 3　运行变迁 31 秒时库所可用实施度与变迁信息素情况

在仿真实例中可以看到,在 31s 时变迁 11 上的信息素明显浓于变迁 10 上的信息素,使托肯很容易走变迁 11、14 这条路径,但是由于 11、14 两个变迁的处理速度是前边快后边慢,所以在库所 15 上产生了托肯积累而降低变迁 11 的实施度,从而在路径寻找中给予变迁 10 更多的机会提高信息素;又由于变迁 14 的处理速度高于变迁 10,所以变迁 10、14 路径的整条路径的处理速度为变迁 10 的速度,变迁 10 的实施度有被提高的倾向,直到变迁 10、14 两者速度一致,从而进一步提高这一条分支路径的速度,这样保证该路径变迁的实施度不会降低而使大量托肯进入该路径。从仿真实例中可以看到,该路上的信息素增加很迅速,最终蚁路在全局延时最短的变迁上形成。

当 Petri 网运行结束时库所 8 中的 s 元素 $s_8 = 3、4, F = 7$;库所 10 中的 s 元素 $s_{10} = 3、4、5, F = 10$;库所 13 中的 s 元素 $s_{13} = 3、4、5、7、8, F = 22$;库所 16 中的 s 元素 $s_{16} = 3、4、5、7、8、10、13, F = 30$。以上结果表明到达库所 8 最短变迁序列为 3、4,耗时 7s;到达库所 10 的最短变迁序列为 3,4,5,耗时 10s;到达库所 13 的最短变迁序列为 3、4、5、7、8,耗时 22s;到达库所 16 的最短序列为 3、4、5、7、8、10、13,耗时 30s。

4　结语

在时间 Petri 网和蚁群算法的基础上提出了一种记忆扩展时间 Petri 网(METPN)模型。METPN 的运行与基本 Petri 网相比增加了信息素功能。每次变迁实施时,托肯都会在变迁上留下信息素来影响后来托肯的路径选择,使蚁路最终逼近于全局最短的时间变迁序列。本文通过具体的算法解决了蚁群算法在 Petri 网中应用的具体问题,从而在一定程度上解决了 Petri 网路径寻找。仿真结果证明采用 METPN 能够快速地找到全局最优路径,但本文的一个重要假设是各个变迁的延时确定且可知,这就限制了本网型的使用范围,随后还需要针对随机网型进行更深入的研究。

参考文献:

[1] 袁崇义. Petri 网原理与应用[M]. 北京: 电子工业出版社, 2005.

[2] 乐晓波, 李京京, 唐贤瑛. 基于 Petri net 建模的资源调度的蚁群算法[J]. 计算机技术与发展, 2006, 16(1): 44 - 46.

[3] 李士勇. 蚁群算法及其应用[M]. 哈尔滨: 哈尔滨工业大学出版社, 2004.

[4] 林闯. 随机 Petri 网和系统性能评价[M]. 第 2 版. 北京: 清华大学出版社, 2005. 22.

[5] 李勇, 曹广益, 朱新坚. 一种基于单亲遗传算法的 Petri 网发射路径求解算法[J]. 系统仿真学报, 2005, 17(1): 203 - 206.

[6] 黄永青, 梁昌勇, 张祥德. 基于均匀设计的蚁群算法参数设定[J]. 控制与决策, 2006, 21(1): 93 - 96.

[7] DORIGO M, MANIEZZO V, COLORNI A. The Ant System Optimization by a Colony of Cooperation Agents[J]. IEEE Transactions on Systems, Man, and Cybernetics-Part B, 1996, 26(1): 29 - 41.

[8] TIEHUA CA, ARTHUR CS. Variable Reasoning and Analysis About Uncertainty with Fuzzy Petri Nets[M]. Application and Theory of Petri Nets, New York: Spring-Verlag, 1993. 126 - 145.