第 28 卷第 2 期　　西 安 工 业 大 学 学 报　　Vol. 28 No. 2
2008 年 04 月　　Journal of Xi'an Technological University　　Apr. 2008

文章编号: 1673-9965(2008)02-157-06

蚁群算法的随机 Petri 网分析与路径寻优*

薛淑磊

(西安工业大学 北方信息工程学院,西安 710032)

摘　要: 为解决 Petri 网的最优路径寻找问题,在分析了随机 Petri 网(Stochastic Petri Net, SPN)中各个变迁实施时刻的分布规律之后,提出了一种计算任意网型变迁时间概率分布的方法. 在对 SPN 分析的基础上,基于蚁群算法设计了一种在 SPN 中使用的各个网元素数据结构,提出了一种在 SPN 中更有效率的路径寻优方法. 仿真结果表明,此路径寻优方法对时间延迟具有更高的灵敏度,对路径选择具有更高的准确性.

关键词: 随机 Petri 网;蚁群算法;概率方法;路径寻优

中图号: TP18;TP301.5　　**文献标志码**: A

在 Petri 网中最初引入的时间概念是将一个变迁与一个固定的时间延迟相关联[1-2],它表示系统中某个变迁的发生需要一定的时间. 随着 Petri 网的发展,每个变迁就与一个时间范围相关联,即每个变迁实施的延迟时间有一个最小值和一个最大值,变迁只能在这个时间段内实施. 随后,研究开始向随机延迟方面展开,将一个变迁的可实施到真正实施这段时间与一个符合一定分布的随机延迟相关联,这种类型的 Petri 网称为随机 Petri 网.

Token 以每个选择点选择算法和变迁更新实施度的方式来搜索固定时间延迟 Petri 网中的最短路径,但在 SPN 的使用中表现出较低的搜索效率. 文中提出了一种搜索方法,即放大局部寻优过程中信息素不浓,但是有搜索潜力的路线上的信息素. 当一个 token 找到了另一条路径,它的总延迟与当前蚁路较浓的路径上总的延迟相同或更小,那么则主动在该路径的所有变迁上一次性添加更浓的信息素以引导 token 在这条路径上集中搜索. 使用过程控制和结果反馈控制来引导 token 的搜索路线. 使搜索逐步向全局最优逼近,到达全局最优.

1　算法定义

根据应用需求和算法需要,对基本的 SPN[3] 进行了一定的扩展,为网络中的变迁增加了保留信息素的功能和过滤信息功能,为库所也增加了过滤信息的功能,所以定义的 Petri 网为带有记忆性的、面向对象的连续时间随机 Petri 网(Memorial Object-oriented Stochastic Petri Net,MOSPN).

定义 1　Token = $(r_1, r_2, \cdots, r_v, d_1, d_2, d_3, \cdots, d_v, F)$ 为托肯的数据结构,其中 r_j 的值为一个变迁索引号,$j = 1, 2, \ldots, v$,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

$$F = F(r_1, r_2, \cdots, r_v) = \sum_{i}^{v} d_i \qquad (1)$$

式中:d_j 为 token 通过变迁 r_j 时所用的真实时间延迟,j 的意义与 r_j 中 j 的意义相同;F 值表示本 token 从 r_1 标识的初始库所到达当前库所 r_v 用的时间长短.

定义 2　MOSPN = $\{P, T, Fl, W, K\}$,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 $n > 0$;$p_i = (r_1, r_2, \cdots, r_v, d_1, d_2, d_3, \cdots, d_v, F_i)$ $(i = 1, 2, \cdots, n)$,r_j 的值为一个变迁索引号,$j = 1, 2, \cdots, v$,v 为当前发现由初始库所到达本库最短路径的变迁数;$r_1, r_2, \cdots, r_v$ 序列标识出变迁的前后顺序;$d_1, d_2, d_3, \cdots, d_v$ 是当前通过它的

* 收稿日期:2007-12-17

作者简介:薛淑磊(1964-),女,西安工业大学工程师,主要研究方向为计算机应用与管理. E-mail:xslxait@163.com.

所有 token 中 F 函数最小的那个 token 的时间序列表；F_i 为一个时间值，该值来自于某个 token 的 F 函数值，表示该 token 从初始库所到达本库所用的时间长短；$T = \{t_1, t_2, \cdots, t_m\}$ 是变迁的有限集合，m 表示变迁的个数，且 $m > 0$；$P \cap T = \Phi$；$t_i = (\tau_i, \lambda_i)$ $(i = 1, 2, \cdots, m)$，$\tau_i \in R$ 为变迁 t_i 上的信息素，是变迁实施时留下的，也是 token 选择路径的主要依据；$\lambda = \{\lambda_1, \lambda_2, \cdots, \lambda_m\}$ 是变迁平均实施速率集合，$\lambda_i \in \lambda$，为变迁 t_i 的平均实施速率，$1/\lambda_i$ 为变迁 t_i 的平均时间延迟；$Fl \subseteq P * T \cup T * P$（* 为笛卡儿积）；$\mathrm{dom}(Fl) \cup \mathrm{cod}(Fl) = P \cup T$，其中 $\mathrm{dom}(Fl) = \{x \mid \exists y : (x, y) \in Fl\}$，$\mathrm{cod}(Fl) = \{y \mid \exists x : (x, y) \in Fl\}$ 即分别是 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；$K = \{k_1, k_2, \cdots, k_n\}$ 定义为 $P \to \cup\{\infty\}$ 为 P 上的容量函数.

2 算法描述

依据蚁群算法的路径寻找原理[4-5]，使用自定义的 token 在网络上行走来寻找最优解. 在 token 的行走过程中 token 一边记录所激发过的变迁序列和实施时间，一边给变迁上留下信息素，以引导后来的 token 选择本变迁. 同一个库所的输出变迁之间互相竞争库所中的 token，那些延时少的变迁在竞争中体现出了较强的优势所以能够引导蚁路在延时少的路径上形成. token 是用来记录问题的解集合，token 中记录的变迁序列便是问题的解集合，而算法并非是寻找问题的解而是寻找问题的最优解. 在文中算法中用的选路策略是 token 选择信息素更浓的路径. 文献[6]企图避开已经走过的路径，在没有涉及的路径上寻找最优解，它以遍历全网为主要搜索思想. 使搜索没有引导性，呈现盲目的遍历特点，在许多无用的路径上耗费了 token 资源. 本算法以有利引导为主要的搜索思想，与基本蚁群算法思想相同，蚁群算法的基本理论已经证明这是一种更有效率的搜索方式.

SPN 中各个变迁的延时不确定，增加了搜索难度. 在搜索技术方面，尝试一种新技术来搜索全局最优. 当搜索进行到一定阶段，发现搜索的值没有太大改进，那么就从搜索阶段进入解的调整阶段，这时认为已经进入了某个局部极小值但并不确定是否是全局最优，即寻到某条路径是当前的一个分支最小而不能够确定是否是由初始某库所出发到终止某库所或目标某库所的全局最小. 此时，使 token 以更大些的概率跳出极小分支，搜索其他分支，并且如果某个 token 寻找到了到达某个库所更短的路径或同样长短的路径，那么它立即较大幅度增加整条路线上的信息素浓度，使该条路径各个变迁上的信息素浓度有更大优势来竞争 token 来对此路径进行重点搜索.

设库所 p_i 的目标函数为

$$F_i^{\mathrm{new}} = \min\{F_{\mathrm{token}}^1, F_{\mathrm{token}}^2, \cdots, F_{\mathrm{token}}^a, F_i^{\mathrm{old}}\}$$
$$F_i^{\mathrm{old}} = F_i^{\mathrm{new}}, i = 1, 2, \cdots, n \qquad (2)$$

式中：F_{token}^a，$a = 1, 2, \cdots$，表示某时刻同时存在于 p_i 中的 token 的 F 函数，a 为 token 数；F_i^{old} 为 p_i 接受 token 前的 F 函数值，初始为 0，F_i^{new} 记录当前发现的，由初始某库所到达本库所 p_i 最短的时间值.

用图 1 来说明 MOSPN 的特殊性，设 p_1 处的 token 为 token_1，p_4 处的 token 为 token_2，p_7 中的 token 为 token_3，如果 t_1 实施，在 t_1 延时后，token_1 将 t_1 的索引号，和真实延时添入自己的序列表并且用式(1)计算 F 函数. 当 t_2 被激发，token_1 再将 t_2 的索引号和真实延时添入自己的序列表，再用式(1)计算 F 函数. t_3，t_4 的变化与 t_1，t_2 相同. 当两个 token 同时到达 t_5 时，t_5 对比两个 token 的 F 函数然后保留 F 函数大的 token 作为当前对象. 假如 token_1 的 F 函数大于 token_2 的 F 函数值，那么 t_5 将 token_1 做为当前 token. 当 t_5 的延时完毕后当前 token 再将 t_5 的索引和真实耗时添入自己的序列表，计算 F 函数，输出当前 token. 当 tokne_1 和 token_3 到达 p_8 时(任意次序)，p_8 保留 F 函数较小的 token 做为当前 token，输出时输出当前 token.

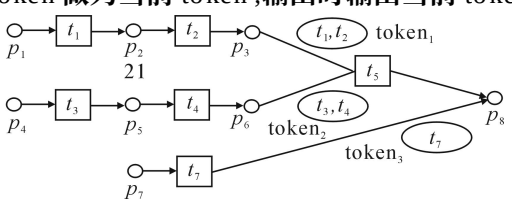

图 1 MOSPN 特殊性示例

Fig. 1 Example of MOSPN particularity

在这个过程中有几个地方需要讨论. t_i 对应的延时 $1/\lambda_i$ $(i = 1, 2, \cdots, 7)$ 都是服从指数分布. 当大量 token 由 p_1，p_4 和 p_7 进入 MOSPN 时，由于变迁延迟，t_1 的输出间隔服从的分布密度函数为

$$f_{t_1}(x) = \lambda_1 \mathrm{e}^{-\lambda_1 x} \qquad (3)$$

当 t_1 走出的 token 进入了 t_2，又因为 t_2 的时间延迟

服从参数为 λ_2 的指数分布,所以 t_2 的输出服从的分布为

$$f(x)=\begin{cases}\dfrac{\lambda_1\lambda_2}{\lambda_2-\lambda_1}(e^{-\lambda_1 x}-e^{-\lambda_2 x}), & \lambda_1\neq\lambda_2, x>0\\ \lambda_1\lambda_2 x e^{-\lambda_2 x}, & \lambda_1=\lambda_2, x>0\\ 0, & x\leq 0\end{cases} \tag{4}$$

如果变迁序列并非只有 λ_1,λ_2,而是多个,那么公式变为

$$f(x)=\begin{cases}(-1)^{n-1}\lambda_1,\cdots,\lambda_n\displaystyle\sum_{i=1}^{n}\frac{e^{-\lambda_i x}}{\prod_{j=1,j\neq i}^{n}(\lambda_i-\lambda_j)}\lambda_k, & (k=1,2,\cdots,n)\ \text{两两不等,且}\ x>0\\ \dfrac{\lambda^r}{(r-1)!}x^{r-1}e^{-\lambda_k}\lambda_k, & (k=1,2,\cdots,n)\ \text{相等,且}\ x>0\\ 0, & x\leq 0\end{cases} \tag{5}$$

如果 $\lambda_k(k=1,2,\cdots,n)$ 中有相等的值,也有不相等的值,那么可以用卷积公式[7] 计算. 因为当随机变量 $\zeta=\varepsilon+\eta$ 时,相当与 $\zeta=\eta+\varepsilon$,所以可以将多个参数相同的随机变量在计算时移动到后边. 即在 λ 集合中,$\lambda_k(k=1,2,\cdots,q)$ $(0<q<n)$ 两两不相等; $\lambda_k(k=q+1,\cdots,n)$ $0<q<n)$ 都相等.

那么公式变为

$$f(z)=\begin{cases}\displaystyle\int_{-\infty}^{+\infty}\Big((-1)^{q-1}\lambda_1\cdots,\lambda_n\sum_{i=1}^{q}\frac{e^{-\lambda_i x}}{\prod_{j=1,j\neq i}^{q}(\lambda_i-\lambda_j)}*\\ \dfrac{\lambda^{n-q}}{(n-q-1)}(z-x)^{n-q-1}e^{-\lambda(z-x)}\Big)dx, & x>0\\ 0, & x\leq 0\end{cases} \tag{6}$$

t_4 输出 token 的时间间隔密度函数与 t_2 相同,类似于式(4). (以下只考虑 $x>0$ 情况)

当 p_3 和 p_6 同时出现 token 时,t_5 变成了使能变迁. 当 t_5 被激发,它比较来自于 t_2 和 t_4 的 F函数,将 F函数值小的 token 作为当前 token. 因为知道来自于两边路径 token 的 F函数的分布,所以当大量 token 进入 t_5 时,它们的延时长短是由数学期望来决定. 来自于 t_2 的 F函数的均值为

$$E(x)=\int_0^{+\infty}x\frac{\lambda_1\lambda_2}{\lambda_2-\lambda_1}(e^{-\lambda_1 x}-e^{-\lambda_2 x})dx=\frac{\lambda_1+\lambda_2}{\lambda_1\lambda_2} \tag{7}$$

如果变迁序列并非只有 λ_1,λ_2,而是多个,那么公式变为

$$E(x)=\frac{1}{\lambda_1}+\frac{1}{\lambda_2}+\cdots+\frac{1}{\lambda_n} \tag{8}$$

对于 t_4 也用同样方法得到 F函数的平均值. t_5 将留下该值大的 token 作为当前 token 并添入 t_5 的索引号和实际延时,计算 F函数,最后输出给 p_8.

在本例中 p_8 同时收到两个 token,一个是来自于 t_2 或 t_4,一个来自于 t_7 它比较两个 token 的 F函数值,将较小值 token 作为当前 token,把它的变迁索引序列和 F函数记录在自己的数据结构中. 当大量 token 运行且比较时,可以用式(8) 数学期望来比较得到结果.

当 Petri 网简单时可以用计算的方法来得到 MOSPN 的主干路径,对复杂 Petri 网,网络节点前后关系模糊,使计算效率不能令人满意. 将蚁群算法引入,通过局部的协作来在全局中产生最优解.

2.1　变迁 t_h 实施时库所的变化

对于 $\forall p_i\in{}^*t_h, i=1,2,\cdots,x_{\text{in}}, x_{\text{in}}$ 为 ${}^*t_h$ 中元素个数(下标 in 是针对 t_h 而言). 像基本 Petri 网一样提取 $W(p_i,t_h)$ 个 token 送入变迁,即在 t_h 中生成 $W(p_i,t_h)$ 个同样数据内容的 token 后,删除 p_i 中的 $W(p_i,t_h)$ 个 token.

对于 $\forall p_i\in t_h{}^*, i=1,2,\cdots,x_{\text{out}}, x_{\text{out}}$ 为 $t_h{}^*$ 中元素个数(下标 out 是针对 t_h 而言). 初始状态 p_j^{old} $(j=1,2,\cdots,n)$ 的序列表为空,它的 F 函数值 $F_{p_j^{\text{old}}}$. p_i 可能同时收到来自于多个变迁的 token. 设某一时刻 token x 进入 p_i, $x=1,2,\cdots,x_{\text{end}}, x_{\text{end}}$ 为 token 个数, F^x_{token} 为该 token 的 F 函数值.

$$p_i^{\text{new}}=\begin{cases}p_i^{\text{old}}, & \text{如果}\ F^x_{\text{token}}>F_{p_i^{\text{old}}}\ \text{且}\ F_{p_i^{\text{old}}}\neq 0\\ \text{token}_x, & \text{如果}\ F^x_{\text{token}}\leq F_{p_i^{\text{old}}}\\ \text{token}_x, & \text{如果}\ F_{s_i^{\text{old}}}=0\end{cases} \tag{9}$$

$$p_i^{\text{old}}=p_i^{\text{new}} \tag{10}$$

2.2　变迁 t_h 实施时 *token* 本身变化

在初始状态,token 中的列表为空, F函数的值为 0. 在网络运行过程中逐渐填充列表和 F 函数,当 t_h 激发后 token 要做一些计算. 首先,每次 token 离开库所时需要

$$\text{token}_x^{\text{new}}=p_i^{\text{old}}, x=1,2,\cdots,x_{\text{end}} \tag{11}$$

然后,当 token 进入了变迁,它对比变迁 t_h 的索引号是否已经出现在记录到的索引序列中了,如果在 token 的列表中找到了变迁 t_h 的索引号,那么就要删除序列中该变迁索引号之后的部分,在延时

结束后确定本次实施的延时时间,然后添入序列表中变迁 t_h 索引对应的延时位置,按照式(1) 重新计算 F 函数值. 如果变迁的索引没有在序列中出现过,那么就将该索引加入 token 的序列表,然后等到变迁实施后,得到本次实施时间添入序列表,再用式(1) 计算 F 函数的值.

2.3 变迁 t_h 实施时变迁的变化

蚁群算法中,蚂蚁是靠信息素的能度对比来决定路径选择,在 MOSPN 中,信息素确定方法是由下式确定. 变迁 t_h 的信息素[4] τ_h,其初始值为 $\tau_h^{\text{old}} = z$(z 为常数). 在 SPN 中延迟时间是服从一定分布的随机变量,那么信息素的修改公式变为

$$\Delta\tau(t_h) = \lambda_h \mathrm{e}^{-\lambda_h A t_h^{-1}} * A t_h^{-2} \tag{12}$$

$$\tau_h^{\text{new}} = \lambda_h \mathrm{e}^{-\lambda_h \frac{A * \text{act}_{\text{new}}(t_h)}{x - \rho^w \tau_h^{\text{old}}}} * \frac{A * \text{act}_{\text{new}}(t_h)}{(x - \rho^w \tau_h^{\text{old}})^2} \tag{13}$$

在进行下一轮计算时,应令

$$\tau_h^{\text{old}} = \tau_h^{\text{new}} \tag{14}$$

以上各式中,ρ为每个时间单位(这里用秒) 挥发后留下的信息素比例; w 为 Petri 网上次实施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; 变迁 t_h 的时间延迟服从以 λ_h 为参数的指数分布; 由式(12) 可以得到每次变迁实施而留下的信息素,由式(13) 可以得到更新信息素后新的信息素浓度; τ_h^{old} 和 τ_h^{new} 分别是实施前,实施后的浓度.

式(12)、(13) 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的信息素浓度的和. 这样对于那些最后一次实施的变迁来说它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.

在变迁 t_h 实施前需要决定本次实施度.

$$\text{act}_{\text{new}}(t_h) = \begin{cases} \text{act}_{\text{old}}(t_h) + 1, \forall p_{\text{in}} \in {}^*t_h, \\ \quad M(p_{\text{in}}) \geq \text{act}_{\text{old}}(t_h) \times W(p_{\text{in}}, t_h), \\ \quad \forall p_{\text{out}} \in t_h^*, M(p_{\text{out}}) = 0 \\ \text{act}_{\text{old}}(t_h) - 1, \forall p_{\text{out}} \in t_h^*, \\ \quad M(p_{\text{out}}) \geq \text{act}_{\text{old}}(t_h) \times W(t_h, p_{\text{out}}), \\ \quad \forall p_{\text{in}} \in {}^*t_h, M(p_{\text{in}}) = 0 \\ \text{act}_{\text{old}}(t_h), \text{其他情况} \end{cases} \tag{15}$$

式中: $M(p)$ 为库所 p 的托肯数, p_{in} 和 p_{out} 分别为输入,输出库所集合中的元素.

2.4 路径选择

面对的 SPN 是复杂网时,token 散漫的遍历整个 Petri 网而寻找变迁序列虽可能找到最优路径,但效率低. 应用蚁群算法的原理,一方面集中 token 在很有发觉潜力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[8].

在系统运行中,当 token 在某库所时有多个输出路径可以选择的话,token 以一定的概率来选择所要激发的变迁,其概率用各个变迁的延迟,当前的信息素浓度来决定,但是,变迁的延迟是随机变量,不能确定某一次的延迟时间,但是当多数 token 来到时可以用平均时间延迟来确定路径选择.

$$Q_{kij} = \begin{cases} \dfrac{\eta_{ij}^{\beta}(t_j)\,\tau_j^{\alpha}(\lambda_j)^{\gamma}}{\sum_{s \in \text{allowed}_k} \eta_{is}^{\beta}(t_s)\,\tau_s^{\alpha}(\lambda_s)^{\gamma}}, & \text{allowed}_k \neq \varnothing \\ 0, & \text{否则} \end{cases} \tag{16}$$

式中: $\text{allowed}_k = \{t_1, t_2, \cdots, t_a\}$ 为托肯 k 下一步允许选择的变迁; a 为本库所中的托肯允许选择的变迁数. 由式(16) 可知,托肯 k 在库所 p_i 向变迁 t_j 的转移概率 Q_{kij} 与 $\eta_{ij}^{\beta}(t_j)\,\tau_j^{\alpha}(\lambda_j)^{\gamma}$ 成正比. τ_j 为当前变迁 t_j 上的信息素的量, $\eta_{ij}(t_j)$ 为由 p_i 到 t_j 的引导因数; λ_j 平均时间延迟,即变迁多次实施时,它延迟的数学期望的倒数. α,β和γ分别反映了 token 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.

2.5 反馈型信息素调整方法

SPN 中变迁延时是随机变量,增加了搜索难度,提出新的搜索技术来协助搜索. 取 token 走出 MOSPN 时所带的 F 函数值作为反馈信息. 当系统运行到一定阶段,各个输出库所出来 token 的序列表和 F 函数值已经趋于稳定,即 F 函数的方差小于一个接受值时,若收到一个 token 带有不同于其他 token 的序列表,并且 F 函数小于当前发现路径的 F 函数,将给该新路线上涂上当前路线上大于平均浓度且小于最大浓度的信息素,使搜索向该方向进行,进一步寻找该路段的潜力.

3 仿真实现

为了体现新算法对时间延迟的灵敏度和对路径选择的准确,采用路径选择更复杂些的 MOSPN

网形,如图 2 所示.

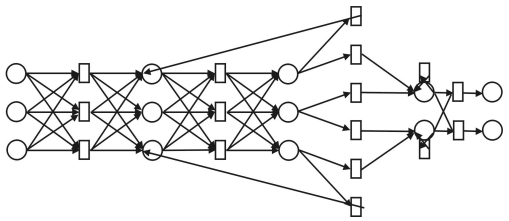

图 2　Petri 网仿真用例

Fig. 2　Example of Petri net simulation

所采用的参数根据文献[3]的建议设定参数为信息素挥发系数 $\rho = 0.6$. 每次涂上的信息素总量 $A = 40$. 在"或"变迁的路径选择公式中 $\alpha = 0.5$, $\beta = 0.5$, $\gamma = 1$, $\eta_{ij}(t_j) = 1$, 初值 $\tau_j = 10^{[9]}$, $i = 1,2,\cdots,n$, $j = 1,2,\cdots,m$.

程序实例中图 3～5 的截图中比图 2 多出的库所是用来控制实现实施度的库所,里边的 token 数显示出对应变迁可用的实施度,可用实施度为当前还可以接受的任务数,小于零时表示不能再接受任务. 图 3～5 中,库所中的点为 token. 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.

本网的运行过程大致描述如表 1,2 所示:(其他各个时刻的信息素以及实施度数字不再赘述).

表 1　变迁延时表

Tab. 1　Transition delay table

变迁索引	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
延时/s	5	5	4	3	3	8	6	4	3	6	4	6	9	9	4	2

表 2　在 0 s 时的信息素和 token 分布状况

Tab. 2　Pheromone and token distribution in 0 s

变　迁	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
信息素/个	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10
可用实施度	3	3	3	3	3	3	3	3	3	3	3	3	1	1	3	3

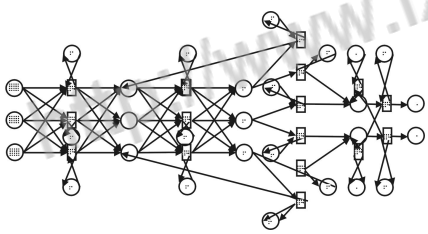

图 3　初始时刻网络状态

Fig. 3　Initial network state

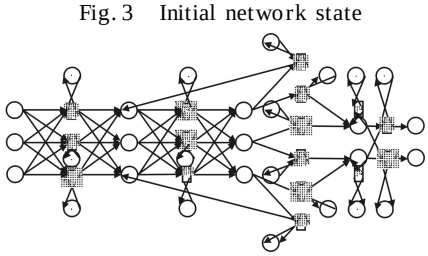

图 4　在 32 s 时刻网络状态

Fig. 4　Network state in 32 s

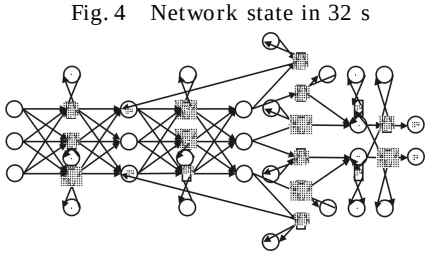

图 5　在 56 s(结束)时刻网络状态

Fig. 5　Network state in 56 s(finish time)

4　结 论

描述了在 SPN 中每个变迁真正实施时间的概率特性,提出一种可以计算任意一种网型的变迁时间概率分布的方法. 在分析了时间特性后,基于蚁群算法提出了在 SPN 环境中使用的各个网元素的数据结构,然后重新解决环路消解等问题,最后提出在 SPN 环境中寻找最优路径的新算法. 以路径选择复杂的 MOSPN 网仿真表明本算法可分辨出变迁延迟之间的细微差别,有很高的灵敏度;在路径选择方面有更高的准确性.

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.
YUAN Chong-yi. Petri Nets Principles and Application[M]. Beijing: Publishing House of Electronics Industry, 2005. (in Chinese)

[2] René K. Boel, Petri Nets Models of Timed Discrete Event Plants[M]. Heidelberg, 2001, 163.

[3] 林闯. 随机 Petri 网和系统性能评价[M]. 2 版. 北京:清华大学出版社,2005.
LIN Chuang. Stochastic Petri Net and Performance Evaluation of Systems[M]. 2nd ed. Beijing: Tsinghua

University Press ,2005. (in Chinese)

[4] 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.
LI Shi-yong. Research on Ant Colony Algorithm and Its Application [M]. Harbin: Harbin Institute of Technology Press ,2004. (in Chinese)

[5] 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 .
YUE Xiao-bo ,LI Jing-jing ,TANG Xian-ying. An Ant Colony Optimization Algorithm of Resource Scheduling Based on Petri Net[J]. Computer Technology and Development ,2006 ,16(1) :44. (in Chinese)

[6] 邵志芳 ,刘重英 ,钱省三. 整合 petri 网和蚁群优化算法用于柔性制造系统调度优化研究[J]. 计算机应用 ,2006(11) :27.
SHAO Zhi-fang ,LIU Zhong-ying ,QIAN Xing-san. Integrated Petri Net and Ant Colony Optimization Algorithm for the Scheduling of FMS[J]. Journal of Computer Applications ,2006(11) :27. (in Chinese)

[7] 赵彦晖. 概率论[M]. 西安:陕西科学技术出版社 ,1999.
ZHAO Yan-hui. Probability Theory[M]. Xi'an: Shaanxi Science and Technology Publishing House , 1999. (in Chinese)

[8] Lefebvre , Demitri; El Moudni , Abdellah , Firing and Enabling Sequences Estimation for Timed Petri Nets [J]. IEEE Transactions on Systems , Man & Cybernetics: A , 2001 , 31 ,3 , 153

[9] 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(1) :93.
HUANG Yong-qing , LIANG Chang-yong , Zhang Xiang-de. Parameter Establishment of an Ant System Based on Uniform Design[J]. Control and Decision , 2006(1) :93. (in Chinese)

Analysis and Route Optimization in Stochastic Petri Net Based on Ant Colony Algorithm

XUE Shu-lei

(North College of Information Engineering ,Xi'an Technological University ,Xi'an 710032 ,China)

Abstract: For solve the problem that Petri net's route optimization search , the distribution rule of each transition implementation moment has been analyzed in the stochastic Petri net (SPN) , a method for the probability distribution of transition time in arbitrary network is proposed. Based on the analysis of SPN and ant colony algorithm , a data structure for each network elements in SPN is designed , and a more efficient route optimization method in SPN is proposed. The result of the simulation shows that this method has the features of more sensitivity of the time-delay and higher accuracy to the route optimization.

Key words: stochastic Petri net ;ant colony algorithm ;probability method ;route optimization

(责任编辑、校对 苗 静)