نشریه علمی‌ترویجی محاسبات نرم
سال هفتم/ شمارهٔ اول/ بهار و تابستان ۱۳۹۷/ صفحهٔ ۲ـ۲۳

دریافت مقاله: ۱۳۹۸/۰۳/۲۹
پذیرش مقاله: ۱۳۹۸/۱۰/۱۷

ارائهٔ یک ابزار مبتنی بر شبکهٔ پتری به‌منظور تحلیل و شبیه‌سازی سیستم‌های کامپیوتری

حسین صباغیان بیدگلی
استادیار، دانشکده برق و کامپیوتر، دانشگاه کاشان، کاشان، ایران
hsabaghainb@kashanu.ac.ir

چکیده: شبکهٔ پتری به‌دلیل داشتن تنوع و قابلیت‌های متعدد، یکی از پرکاربردترین روش‌های مدل‌سازی و ارزیابی سیستم‌های هم‌روند و مبتنی بر رخداد است. با توجه به تنوع کاربرد و وجود انواع مختلف شبکهٔ پتری ابزارهای مختلفی برای مدل‌سازی، شبیه‌سازی و تحلیل شبکه‌های پتری عرضه شده است. هریک از این ابزارها بخشی از قابلیت‌های مورد نیاز برای تحلیل و ارزیابی سیستم‌ها را فراهم می‌کنند. در کار حاضر، ابزاری برای مدل‌سازی و ارزیابی بر اساس شبکهٔ پتری فراهم شده است که علاوه بر پشتیبانی از انواع مختلف شبکهٔ پتری، برخی از قابلیت‌هایی را که در سایر ابزارها کمتر مورد توجه قرار گرفته فراهم می‌کند. امکان تولید خودکار مدل، امکان تحلیل حالت پایدار انواع شبکهٔ پتری از جمله شبکهٔ پتری زمان ثابت، تغییر ساختار مدل در حین اجرا، امکان یادگیری و تطبیق‌پذیری مدل، امکان ارزیابی و جست‌وجوی خودکار راه حل از جمله قابلیت‌های این ابزار است. در این مقاله، علاوه بر معرفی مدل جامع پیشنهادی امکانات ابزار ساخته‌شده شرح داده می‌شود. علاوه بر آن، با ذکر مثال‌هایی قابلیت‌های انحصاری ابزار مذکور معرفی می‌شود.

واژه‌های کلیدی: شبکهٔ پتری، مدل‌سازی، شبیه‌سازی، مدل مارکوف، تحلیل حالت پایدار، ارزیابی کارایی.

۱. مقدمه

شبکهٔ پتری مدلی دارای پشتوانهٔ ریاضی و نمایش گرافیکی است که برای توصیف، شبیه‌سازی و تحلیل انواع سیستم‌ها به‌خصوص سیستم‌های هم‌روند و مبتنی بر رخداد مناسب است [۱ و ۲]. انواع مختلفی از شبکهٔ پتری برای مدل‌سازی سیستم در حوزه‌های مختلف کاربرد به وجود آمده است. شبکهٔ پتری رنگی [۳]، شبکهٔ پتری زمان ثابت [۴] شبکهٔ پتری زمان تصادفی [۵] و شبکه‌های تصادفی تعمیم‌یافته [۶] نمونه‌هایی از انواع شبکهٔ پتری هستند که از آن‌ها به‌طور گسترده در کاربردهای مختلف [۸، ۹ و ۱۰] استفاده می‌شود.

مدل پتری با مجموعه‌ای از پارامترهای داده‌ای و قوانین معرفی می‌شود. ایجاد، شبیه‌سازی و تحلیل یک مدل پتری نیازمند ابزار مناسب است. ابزار پتری یک نرم‌افزار شامل توابع لازم برای انجام عملیات مختلف روی مدل پتری است. تاکنون ابزارهای مختلفی برای مدل‌سازی و شبیه‌سازی و تحلیل شبکه‌های پتری عرضه شده است. در این میان می‌توان به CPN-Tool [۳]، SPNP [۱۱]، Sharpe [۱۲ و ۱۳] و PIPE [۱۴] به‌عنوان نمونه‌هایی از معروف‌ترین و پرکاربردترین آن‌ها اشاره کرد. در [۱۵] مروری بر ویژگی‌های تعدادی از این ابزارها آورده شده است. تعدادی از ویژگی‌هایی که معمولاً در انتخاب ابزار مدل‌سازی در نظر گرفته می‌شود عبارت‌اند از:

- نوع شبکهٔ پتری که ابزار مربوط از آن پشتیبانی می‌کند.
- داشتن امکان تولید فضای حالت
- داشتن امکان حل حالت پایدار
- داشتن ویراستار گرافیکی
- پشتیبانی از شبکهٔ پتری سلسله‌مراتبی

در مقالهٔ مروری [۱۵] نشان داده شده که هریک از ابزارهای موجود تنها برخی از این قابلیت‌ها را پشتیبانی می‌کنند.

یکی از نیازمندی‌هایی که در این ابزارها به‌خوبی پشتیبانی نمی‌شود، امکان تولید و ایجاد تغییر در ساختار مدل به‌صورت خودکار است. در برخی ابزارها برای ایجاد مدل پتری امکان ترسیم گرافیکی فراهم شده است. این قابلیت اگرچه برای توصیف شبکه‌های پتری کوچک مطلوب است، ترسیم گرافیکی برای شبکه‌های پتری با اندازهٔ متوسط کاری سخت و برای شبکه‌های پتری بزرگ عملاً غیرممکن است. برخی ابزارها امکان دریافت مدل پتری از طریق داده‌های توصیف مدل (مانند ماتریس گراف اتصالی و بردار نشانه‌گذاری اولیه و...) را دارند. در این صورت کاربر باید فرمت مخصوص آن ابزار را بشناسد و از یک ابزار یا یک زبان برنامه‌نویسی دیگر برای تولید آن داده‌ها استفاده کند. این قابلیت نیز اگرچه تا حدودی نیاز به تولید خودکار شبکه‌های پتری بزرگ را برطرف می‌کند، با توجه به یکپارچه نبودن ابزار تولید مدل و ابزار تحلیل و شبیه‌سازی، به‌سادگی امکان تغییر مجدد در ساختار یا ویژگی‌های مدل به‌صورت خودکار در زمان اجرا و هنگام ارزیابی مدل وجود ندارد. این موضوع روند جست‌وجوی پاسخ مناسب را کند نموده و امکان جست‌وجوی خودکار را از بین می‌برد.

در کار حاضر، مدلی جامع برای شبکهٔ پتری پیشنهاد شده که انواع مختلف شبکهٔ پتری را پوشش می‌دهد. توابع لازم برای توصیف، تحلیل، شبیه‌سازی و ارزیابی انواع مدل پتری در محیط متلب نوشته شده و امکان تولید خودکار، ارزیابی و تغییر در ساختار مدل در حین اجرا با استفاده از توابع موجود و برنامه‌نویسی متلب فراهم شده است. امکانی که تاکنون در هیچ‌یک از ابزارهای موجود مشاهده نشده است. یکپارچگی کد منبع مدل، کد منبع توابع مختلف ابزار و امکان‌نویسی برای ارزیابی و جست‌وجو راه حل در همان محیط از مزایای ابزار ساخته شده است. با توجه به قابلیت‌های یادشده ابزار ساخته‌شده گزینهٔ مناسبی برای انجام پروژه‌های تحقیقاتی دانشگاهی و صنعتی خواهد بود.

توجه شود که همهٔ امکانات لازم برای توصیف، شبیه‌سازی، تحلیل و نمایش بصری مدل پتری و انواع گراف‌های دسترسی در محیط متلب فراهم شده است. همهٔ امکانات و ابزارها به‌صورت مجموعه‌ای از توابع به‌طوری که چیزی شبیه یک جعبه ابزار (Tool box) را به وجود آورند پیاده‌سازی شده است. پیاده‌سازی طوری انجام شده که از یک

طرف از انواع شبکهٔ پتری پشتیبانی کرده و از طرف دیگر، قابلیت‌های کلیدی مورد نظر را فراهم کند، همچنین امکان توسعه و سفارشی‌سازی انواع و قابلیت‌های جدیدتر را نیز داشته باشد.

در ادامهٔ این مقاله در بخش ۲، مدل پتری پیشنهادی، اجزای ساختمان دادهٔ نمایش‌دهندهٔ آن و قوانین توانا شدن و شلیک گذرها که از مراحل اساسی شبیه‌سازی است شرح داده می‌شود. در بخش ۳، اهمّ امکانات ابزار پیاده‌سازی شده که KU_PN_Tool نامیده شده، بیان می‌شود. این ابزار به‌صورت متن باز در آدرس [۱۶] قابل دسترسی است. اهمّ امکانات این ابزار شامل ایجاد مدل پتری، رسم گراف پتری، رسم گراف دسترسی و رسم گراف CTMC و DTMC و حل حالت پایدار است. بخش ۴ شامل معرفی امتیازات ابزار ساخته‌شده است. در این بخش، برخی قابلیت‌های انحصاری ابزار با ذکر نمونه‌هایی مانند تولید خودکار مدل با استفاده از حلقهٔ تکرار، تغییر خودکار در ساختار مدل، تطبیق‌پذیری مدل پتری، جست‌وجوی خودکار راه‌حل و بررسی اثر تغییرات در مدل بر روی کارایی بیان خواهد شد. بخش ۵ شامل جمع‌بندی و نتیجه‌گیری است.

۲. معرفی مدل جامع پتری پیشنهادی

از آنجا که قرار است ابزار پیاده‌سازی‌شده به‌طور همزمان از انواع مختلف شبکهٔ پتری مانند شبکهٔ پتری اولیه، زمان ثابت، زمان تصادفی، و انواع ترکیبی مانند GSPN و GTPN پشتیبانی کند، همچنین مواردی مانند کمان بازدارنده، اولویت گذر، وزن احتمال شلیک گذر، تأخیر نشانه، وزن کمان، نشانهٔ رنگی و ساختار سلسله‌مراتبی را نیز پوشش دهد، لازم است از مدلی استفاده کند که از جامعیت کافی برای پیاده‌سازی همهٔ این امکانات برخوردار باشد. طبیعتاً یک چنین مدلی دارای پیچیدگی‌های بیشتری خواهد بود و بیان آن نیازمند تعداد بیشتری اجزای داده‌ای و ساختمان دادهٔ بزرگ‌تر خواهد بود. در ادامهٔ این فصل ابتدا به بیان مدل صوری شبکهٔ پتری پیشنهادی و سپس به معرفی ساختمان دادهٔ در نظر گرفته‌شده برای آن پرداخته می‌شود. سپس قوانین مربوط به توانا شدن و شلیک گذر و اصول اولیهٔ شبیه‌سازی آن بیان خواهد شد.

۱.۲. مدل صوری شبکهٔ پتری پیشنهادی

شبکهٔ پتری پیشنهادی یک ۲۴تایی بدین شرح است:

PN_Model=(Name, P, T, Pre, InhPre, Post, Prd, Cns, InhCns, M0, Cap, Firing_func, Pre_Weight, InhPre_Weight, Post_Weight, Delay, Rate, ProbWeight, TknDly, Priority, Tr_Type, Pl_Type, CountT, low_level_model)

تک‌تک پارامترهای مدل صوری پیشنهادی در جدول (۱) معرفی شده است. این پارامترها اغلب به‌صورت آرایه یا آرایه‌ای از آرایه‌ها هستند.

۲.۲. ساختمان دادهٔ مدل پتری پیشنهادی

مدل پتری پیشنهادی در یک ساختمان داده به شکل ساختار (Structure) که شامل فیلدهای مختلف است ذخیره می‌شود. فیلدهای این ساختار شامل ۲۴ عنصر تشکیل‌دهندهٔ مدل پتری است که در بخش ۱.۳ و در جدول (۱) شرح داده شد. برخی فیلدها به‌صورت آرایه‌ای از اعداد هستند و برخی به‌صورت آرایه‌ای از آرایه‌ها یا آرایه‌ای از رشته‌های کاراکتری که برای پیاده‌سازی دو مورد آخر در متلب از آرایهٔ سلولی استفاده شده است. تابعی به نام null_model_PN(Name) یک ساختار خالی ایجاد می‌کند. این تابع در ابتدای تابع سازندهٔ مدل فراخوانی می‌شود تا ساختمان دادهٔ لازم را آماده کند. سپس با فراخوانی توابع ایجاد مکان و گذر و کمان (که در بخش ۴ شرح داده خواهد شد) اطلاعات مربوط به‌مرور در این ساختمان داده قرار می‌گیرد و مدل پتری به وجود می‌آید. این تابع دارای یک پارامتر ورودی Name است که نام مدل پتری جدید را مشخص می‌کند. طرز استفاده از این تابع را در بخش۳ و شکل (۱ الف) ببینید.

۳.۲. قوانین فعال شدن گذر در مدل پتری پیشنهادی

با توجه به اینکه مدل پیشنهادی از کمان بازدارنده، کمان‌های وزن‌دار و ظرفیت مکان پشتیبانی می‌کند، شرایط فعال شدن یک گذر عبارت‌اند از:

- ✓ هریک از مکان‌های ورودی گذر، حداقل به تعداد وزن

کمان متصل‌کنندهٔ آن مکان به گذر، نشانه داشته باشد.

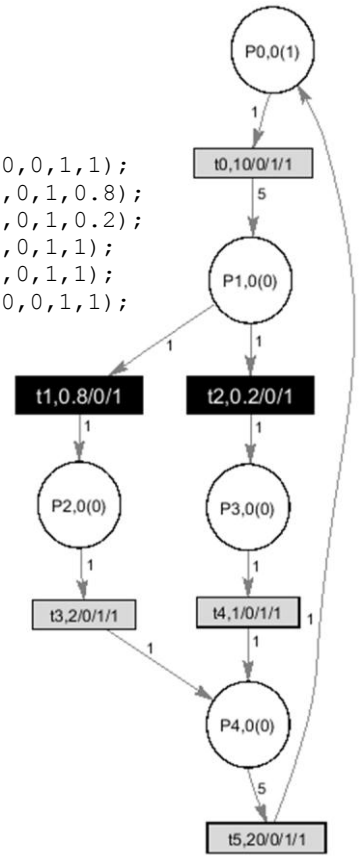
- ✓ در صورت وجود مکان بازدارنده در ورودی یک گذر، تعداد نشانه در آن کمتر از وزن کمان متصل‌کنندهٔ به گذر باشد.
- ✓ هریک از مکان‌های خروجی گذر حداقل به تعداد وزن کمان متصل‌کنندهٔ گذر به آن مکان ظرفیت خالی داشته باشند یا اینکه ظرفیت نامحدود داشته باشد. ظرفیت نامحدود مکان i با Cap(i)=0 مشخص می‌شود.
- ✓ اگر گذر از نوع زمان ثابت یا زمان تصادفی است آنگاه
  - اولاً فیلد زمان تمام نشانه‌های ورودی (که در چهارمین عنصر از آرایهٔ نشانه قرار دارد) صفر باشد.
  - ثانیاً مقدار شمارندهٔ محلی (CountT) گذر صفر باشد.

جدول (۱): معرفی پارامترهای مدل صوری پتری پیشنهادی

نام پارامتر	شرح
Name	یک رشتهٔ کاراکتری شامل نام مدل پتری
P	آرایه‌ای از نام مکان‌هاست. P{i} شامل رشتهٔ کاراکتری نام مکان iام است.
T	آرایه‌ای از نام گذرهاست. T{i} شامل رشتهٔ کاراکتری نام گذر iام است.
Pre	آرایه‌ای شامل کمان‌های ورودی گذرها.Pre{i} شامل اندیس همهٔ مکان‌های ورودی گذر T{i} است.
InhPre	کمان‌های بازدارندهٔ ورودی گذرها. InhPre{i} شامل اندیس مکان‌های بازدارندهٔ ورودی گذر T{i} است.
Post	کمان‌های خروجی گذرها. Post{i} شامل اندیس همهٔ مکان‌های خروجی گذر T{i}
Prd	کمان‌های ورودی مکان‌ها. Prd{i} شامل اندیس همهٔ گذرهای مولد نشانه برای مکان P{i}
Cns	کمان‌های خروجی مکان‌ها. Cns{i} شامل اندیس همهٔ گذرهای مصرف‌کنندهٔ نشانه از مکان P{i}
InhCns	کمان‌های بازدارندهٔ خروجی مکان‌ها. InhCns{i} شامل اندیس همهٔ گذرهای مصرف‌کنندهٔ بازدارندگی مکان P{i} (گذرهایی است که مکان P{i} به‌وسیلهٔ کمان بازدارنده به آن‌ها متصل شده است.)
M0	نشانه‌گذاری مکان‌ها. M0{i} یک آرایهٔ سلولی شامل نشانه‌های P{i} است به‌طوری که M0{i}{j} نشانهٔ jام از مکان P{i} است. هر نشانه خود به‌صورت یک آرایه با طول حد اقل ۴ و حد اکثر دلخواه تعریف شده است.
Cap	ظرفیت مکان‌ها. Cap(i) ظرفیت مکان P{i} را مشخص می‌کند. صفر به معنی ظرفیت نامحدود
Firing_func	توابع شلیک گذرها. Firing_func{i} یک رشتهٔ کاراکتری که نام تابع شلیک گذر iام است.
Pre_Weight	Pre_Weight{i} شامل وزن کمان‌های ورودی گذر T{i} متناظر با مکان‌های Pre{i}
InhPre_Weight	InhPre_Weight{i} وزن کمان‌های بازدارندهٔ T{i} متناظر با مکان‌های InhPre{i}
Post_Weight	Post_Weight{i} شامل وزن کمان‌های خروجی گذر T{i} متناظر با مکان‌های Post{i}
Delay	Delay(i) تأخیر گذر T{i} است. فقط برای گذرهای نوع ۱ (زمان ثابت) معتبر است.
Rate	Rate(i) نرخ شلیک گذر T{i} است که فقط برای گذرهای نوع ۲ (زمان تصادفی) معتبر است. (اگر مقدار نرخ λ باشد زمان شلیک یک عدد تصادفی با تابع توزیع نمایی $\lambda e^{-\lambda t}$ خواهد بود.)
ProbWeight	ProbWeight(i) وزن احتمال گذر T{i}
TknDly	TknDly(i) تأخیری است که گذر T{i} پس از شلیک به تمام نشانه‌های خروجی خود اضافه می‌کند.
Priority	Priority(i) اولویت گذر T{i} را مشخص می‌کند. عدد کوچک‌تر اولویت بالاتر را نشان می‌دهد.
Tr_Type	Tr _Type(i) نوع گذر T{i} است. مقدار ۰ نوع فوری، ۱ زمان ثابت و ۲ زمان تصادفی را مشخص می‌کند.
Pl_Type	Pl_Type(i) نوع مکان P{i} را تعیین می‌کند. مقدار ۰ نشان‌دهندهٔ نوع کنترلی و ۱ نشان‌دهندهٔ نوع داده‌ای. تابع شلیک پیش‌فرض، داده را فقط از مکان‌های داده‌ای ورودی گرفته و فقط به مکان‌های داده‌ای خروجی تحویل می‌دهد. مکان‌های کنترلی (چه ورودی چه خروجی) فقط حامل نشانه‌های سیاه (بدون داده) هستند.
CountT	آرایه‌ای از شمارنده‌های کاهشی محلی که برای اندازه‌گیری زمان تأخیر گذرهای فعال در زمان شبیه‌سازی استفاده می‌شود. CountT(i) زمان لحظه‌ای باقی‌مانده تا شلیک گذر فعال T{i} را نشان می‌دهد.
low_level_model	یک آرایه از زیر مدل‌های پتری متناظر گذرها در مدل سلسله‌مراتبی است. low_level_model{i} زیر مدل پتری متناظر گذر T{i} است که می‌تواند به‌جای تابع شلیک Firing_func{i} اجرا شود.

```
function PN_model=Paper_example1()

%Make null PN model
[PN_model] = null_model_PN('Paper_example1');

%Define Transitions
[PN_model,t0]=New_Transition(PN_model,'t0', 'General_func',2,10,0,1,1);
[PN_model,t1]=New_Transition(PN_model,'t1', 'General_func',0,0,0,1,0.8);
[PN_model,t2]=New_Transition(PN_model,'t2', 'General_func',0,0,0,1,0.2);
[PN_model,t3]=New_Transition(PN_model,'t3', 'General_func',2,2,0,1,1);
[PN_model,t4]=New_Transition(PN_model,'t4', 'General_func',2,1,0,1,1);
[PN_model,t5]=New_Transition(PN_model,'t5', 'General_func',2,20,0,1,1);

%Define Places
[PN_model,p0]=New_Place(PN_model,'P0', 0,1,{[1,1,0,0,0]});
[PN_model,p1]=New_Place(PN_model,'P1', 0,1,[]);
[PN_model,p2]=New_Place(PN_model,'P2', 0,1,[]);
[PN_model,p3]=New_Place(PN_model,'P3', 0,1,[]);
[PN_model,p4]=New_Place(PN_model,'P4', 0,1,[]);

%Add Communication Arcs
PN_model=Arc_P2T(PN_model,p0,t0);
PN_model=Weighted_Arc_T2P(PN_model,t0,p1,5);
PN_model=Arc_P2T(PN_model,p1,t1);
PN_model=Arc_P2T(PN_model,p1,t2);
PN_model=Arc_T2P(PN_model,t1,p2);
PN_model=Arc_T2P(PN_model,t2,p3);
PN_model=Arc_P2T(PN_model,p2,t3);
PN_model=Arc_P2T(PN_model,p3,t4);
PN_model=Arc_T2P(PN_model,t3,p4);
PN_model=Arc_T2P(PN_model,t4,p4);
PN_model=Weighted_Arc_P2T(PN_model,p4,t5,5);
PN_model=Arc_T2P(PN_model,t5,p0);
```

(الف)



(ب)

شکل (۱): (الف) کد سازنده و (ب) گراف مدل پتری مثال ۱

۴.۲. قوانین شلیک گذر در مدل پتری پیشنهادی

در مدل پیشنهادی، برای هر گذر می‌توان یک تابع شلیک مشخص کرد؛ این تابع می‌تواند تابع شلیک عمومی به نام General_func باشد که از قبل تعریف شده یا تابعی باشد که توسط کاربر نوشته می‌شود. نام تابع شلیک در هنگام تعریف گذر (توسط تابع New_Transition) به گذر نسبت داده می‌شود. تابع شلیک عمومی General_func عملیات شلیک را بر اساس قوانین اولیه شبکهٔ پتری انجام می‌دهد. مراحل شلیک گذر توسط این تابع به شکل زیر است:

- ✓ از تمام مکان‌های ورودی به تعداد وزن کمان متصل‌کننده نشانه حذف می‌کند.
- ✓ مقدار فیلد تأخیر آن نشانهٔ ورودی به اندازهٔ TknDlyی گذر جاری افزایش می‌یابد.
- ✓ به تمام مکان‌های خروجی به تعداد وزن کمان متصل‌کننده نشانه می‌فرستد.

در تابع شلیک عمومی یعنی General_func ساختار نشانه یک آرایه با حداقل ۴ عنصر در نظر گرفته شده است، ولی با توجه به اینکه تابع شلیک گذر می‌تواند به‌دلخواه نوشته شود، نشانه نیز هر ساختاری می‌تواند داشته باشد.

توجه شود که تابع شلیک عمومی نوع مکان‌های ورودی و خروجی از نظر داده‌ای یا کنترلی بودن را نیز در نظر می‌گیرد. به این ترتیب که نشانهٔ الگو را بر اساس نشانهٔ یکی از مکان‌های داده‌ای ورودی می‌سازد و این نشانه را فقط به مکان‌های داده‌ای خروجی ارسال می‌کند و برای مکان‌های خروجی کنترلی یک نشانهٔ سیاه خالی از داده (که معمولاً یک آرایه با چهار عدد صفر است) ارسال می‌کند. به این ترتیب داده‌ها فقط از طریق مکان‌های داده‌ای در طول شبکهٔ پتری منتشر می‌شوند. این قابلیت می‌تواند برای مدل‌سازی در حوزه‌هایی مانند طراحی

سیستم‌های دیجیتال که در آن سیستم به دو بخش کنترل و داده تقسیم می‌شود مناسب باشد.

تابع شلیک عمومی هیچ پردازشی روی داده‌های نشانهٔ ورودی انجام نمی‌دهد ولی می‌توان با نوشتن یک تابع شلیک دلخواه هر پردازشی روی این داده‌ها انجام داد. این موضوع قابلیت مدل پیشنهادی و ابزار ساخته‌شده را به نحو چشمگیری افزایش می‌دهد به‌طوری که با وجود این قابلیت در کنار قابلیت سلسله‌مراتبی می‌توان مدل پیشنهادی را یک مدل جامع دانست که علاوه بر پشتیبانی از مدل‌های پتری سطح پایین از مدل‌سازی سطح بالا (CPN و HLPN) نیز پشتیبانی می‌کند.

جدول (۲): معرفی توابع سازندهٔ مدل پتری

ایجاد مکان جدید	[PN_model,Pl_num]=New_Place(PN_model,Name,Cap,Type,Marking);
ایجاد گذر جدید	[PN_model,Tr_num]=New_Transition(PN_model, Name, Firing_func, Type, TrValue, TknDly, Priority, ProbWeight);
ایجاد یک کمان از مکان به گذر	PN_model=Arc_P2T(PN_model, Pl_num, Tr_num);
ایجاد یک کمان از گذر به مکان	PN_model=Arc_T2P(PN_model, Tr_num, Pl_num);
کمان بازدارنده از مکان به گذر	PN_model=InhArc_P2T(PN_model, Pl_num, Tr_num);
کمان وزن‌دار از مکان به گذر	PN_model= Weighted_Arc_P2T(PN_model, Pl_num, Tr_num, Weight);
کمان وزن‌دار از گذر به مکان	PN_model= Weighted_Arc_T2P(PN_model, Tr_num, Pl_num, Weight);
کمان بازدارنده وزن‌دار از مکان به گذر	PN_model= Weighted_InhArc_P2T(PN_model, Pl_num, Tr_num, Weight);

۳. امکانات ابزار KU_PN_Tool

در این بخش، امکانات مختلفی که در ابزار ساخته‌شده پیش‌بینی شده معرفی می‌گردد.

۳.۱. ایجاد مدل پتری

مدل پتری با استفاده از یک تابع که به زبان متلب نوشته شده و در یک m فایل ذخیره شده، ایجاد می‌شود. در این تابع که آن را تابع سازندهٔ مدل پتری می نامیم از توابع ایجاد گذر و مکان و کمان های ارتباطی استفاده می‌شود. این توابع اطلاعات شبکهٔ پتری مورد نظر را در ساختمان دادهٔ مربوط با ساختاری که قبلاً معرفی شد قرار می‌دهند. کد شکل (۱ الف) مثالی از تابع سازندهٔ یک شبکهٔ پتری است. گراف شبکهٔ پتری معادل آن در شکل (۱ ب) نشان داده شده است. در طول این مقاله، از این مثال با نام مثال ۱ نام برده خواهد شد و با تغییرات اندکی در آن مثال‌های ۲ و ۳ نیز از روی آن توضیح داده خواهد شد. سطر اول تابع سازنده، نام دلخواه تابع را مشخص می‌کند. هرچند نامگذاری تابع اختیاری است، توصیه می‌شود با نام مدل که در سطر بعد با فراخوانی تابع null_model_PN مشخص می‌شود، یکسان باشد. این فراخوانی ساختمان داده‌ای خالی با نام PN_model ایجاد می‌کند. این ساختمان داده در ادامه با فراخوانی توابع ایجاد مکان، گذر و کمان پر شده و مدل پتری مورد نظر را ایجاد می‌کند. جدول (۲) مجموعه توابع مختلف قابل استفاده در تابع سازنده را نشان می‌دهد. شرح پارامترهای هریک از توابع جدول (۲) در جدول‌های (۳ـ۵) آمده است.

جدول (۳): معرفی پارامترهای تابع New_Place

نام تابع	نوع	پارامتر	شرح
New_Place	ورودی	PN_model	ساختمان دادهٔ مدل قبل از افزودن مکان جدید
		Name	یک رشتهٔ کاراکتری شامل نام مکان جدید
		Cap	ظرفیت مکان (0: ظرفیت نامحدود است)
		Type	نوع مکان (0: کنترلی یا 1: داده‌ای)
		Marking	آرایهٔ سلولی شامل نشانه‌های مکان جدید
	خروجی	PN_model	ساختمان داده مدل بعد از افزودن مکان جدید
		Pl_num	اندیس مکان جدید در مجموعهٔ مکان‌ها

جدول (۴): معرفی پارامترهای تابع New_Transition

نام تابع	نوع	نام پارامتر	شرح
New_Transition	ورودی	PN_model	ساختمان دادهٔ مدل قبل از افزودن گذر جدید
		Name	یک رشتهٔ کاراکتری شامل نام گذر جدید
		Firing_func	رشتهٔ کاراکتری نام تابع شلیک گذر جدید
		Type	نوع گذر (0: فوری 1: زمان ثابت 2: زمان تصادفی)
		TrValue	بسته به نوع گذر (Type) نقش آن متفاوت است. برای گذر فوری بی‌تأثیر است (0 توصیه می‌شود) برای گذر زمان ثابت تأخیر گذر را معین می‌کند برای زمان تصادفی نرخ شلیک گذر را معین می‌کند. اگر نرخ شلیک گذر λ باشد فاصلهٔ زمانی بین فعال شدن تا شلیک گذر با تابع توزیع $\lambda e^{-\lambda t}$ مشخص می‌شود.
		TknDly	تأخیر نشانه‌های خروجی گذر جدید را مشخص می‌کند؛ این تأخیر توسط تابع شلیک عمومی در محل عنصر چهارم آرایهٔ نشانه قرار می‌گیرد. وجود تأخیر در یک نشانه باعث معطل ماندن نشانه در آستانهٔ گذر زمان ثابت یا زمان تصادفی بعدی می‌شود در حالی که از گذرهای فوری بدون معطلی عبور می‌کند
		Priority	یک عدد که اولویت شلیک گذرهای همزمان فعال را مشخص می‌کند. عدد کمتر اولویت بالاتری دارد. در اولویت برابر انتخاب به‌صورت تصادفی بر اساس پارامتر ProbWeight انجام خواهد شد.
		ProbWeight	یک عدد نا منفی که وزن احتمال شلیک گذر را مشخص می‌کند. در شرایطی که چند گذر همزمان آماده شلیک باشند و این گذرها از نظر اولویت با هم برابر باشند یکی از گذرها به‌صورت تصادفی و با در نظر گرفتن وزن احتمال آن‌ها (به‌طور نسبی) انتخاب و شلیک می‌شود. توجه شود که یک گذر زمانی یا زمان‌تصادفی وقتی آمادهٔ شلیک است که علاوه بر فعال بودن تأخیر گذر و تأخیر نشانه‌های ورودی آن نیز سپری شده باشد.
	خروجی	PN_model	ساختمان دادهٔ مدل پتری بعد از افزودن گذر جدید
		Tr_num	شمارهٔ اندیس گذر جدید در مجموعهٔ گذرها

جدول (۵): معرفی پارامترهای انواع تابع ایجاد کمان

نام تابع	نوع	پارامتر	شرح
Arc_P2T Arc_T2P InhArc_P2T Weighted_Arc_P2T Weighted_Arc_T2P Weighted_InhArc_P2T	ورودی	PN_model	ساختمان داده مدل پتری قبل از اضافه شدن کمان جدید
		Pl_num	شماره اندیس مکان
		Tr_num	شماره اندیس گذر
		Weight	یک عدد صحیح غیر صفر که وزن کمان را مشخص می‌کند
	خروجی	PN_model	ساختمان داده مدل پتری بعد از اضافه شدن کمان جدید

۲.۳. رسم گراف شبکهٔ پتری

در ابزار KU_PN_Tool گزینه‌ای به نام PN graph پیش‌بینی شده است که نمودار گرافیکی شبکهٔ پتری انتخابی را رسم می‌کند. شکل (۱ ب) را ببینید؛ در این شکل گذرها با مستطیل و مکان‌ها با دایره مشخص شده، کمان‌های ورودی و کمان‌های خروجی با سر پیکانی‌شکل رسم شده و وزن آن‌ها نیز در کنار آن‌ها نوشته می‌شود. توجه شود که کمان بازدارنده (که البته در این شکل وجود ندارد) نیز مانند کمان‌های ورودی و خروجی با سر پیکانی‌شکل رسم می‌شود و برای تمایز از کمان معمولی وزن کمان بازدارنده با علامت منفی نشان داده می‌شود. گذرهای فوری با رنگ سیاه و گذرهای زمان تصادفی با رنگ خاکستری مشخص شده است. گذر زمان ثابت (که البته در این شکل وجود ندارد) به رنگ سفید نشان داده می‌شود. از طرفی رنگ دایره هم نشان‌دهندهٔ نوع مکان است. مکان داده‌ای با رنگ سفید و مکان کنترلی با رنگ خاکستری نشان داده می‌شود. در شکل (۱ ب) همهٔ مکان‌ها از نوع داده‌ای هستند. در داخل مکان‌ها و

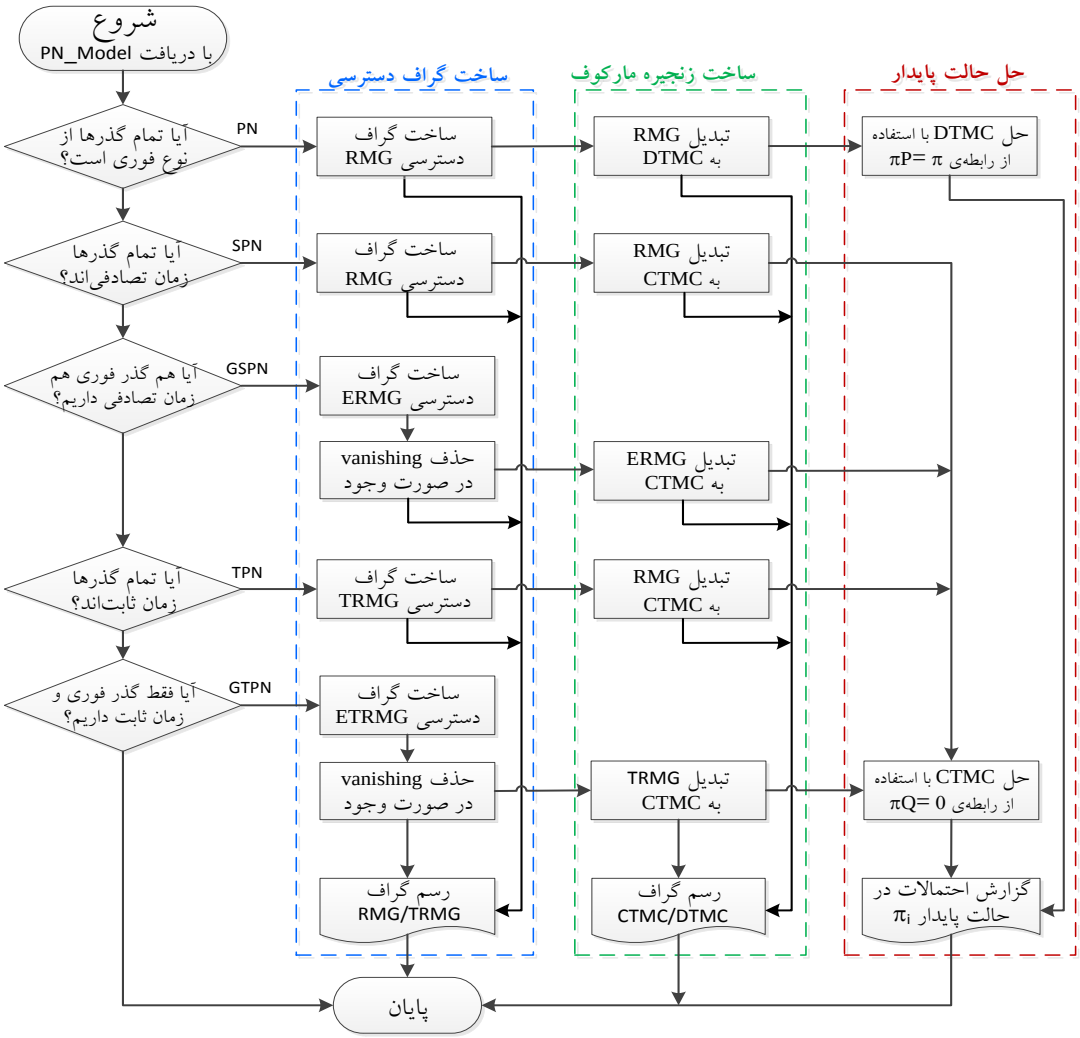
گذرها علاوه بر نام، مشخصات گذر یا مکان هم نوشته شده است. در هر مکان علاوه بر نام، ظرفیت مکان و تعداد نشانهٔ آن مکان در قالب "Name, Cap(#Marking)" نشان داده می‌شود. اطلاعات داخل هر گذر نیز بسته به نوع گذر با توجه به جدول (۶) مشخص می‌شود.

جدول (۶): معرفی فرمت نمایش اطلاعات گذر

نوع گذر (Tr_Type)	رنگ گذر	فرمت نمایش اطلاعات
فوری (0)	سیاه	Name,ProbWeight/TknDly/Priority
زمان ثابت (1)	سفید	Name,Delay/ProbWeight/TknDly/Priority
زمان‌تصادفی (2)	خاکستری	Name,Rate/ ProbWeight/TknDly/Priority

۳.۳. تحلیل حالت پایدار انواع شبکهٔ پتری

برای تحلیل حالت پایدار شبکهٔ پتری سه مرحله وجود دارد. این مراحل شامل بدست آوردن گراف دسترسی، تبدیل آن به گراف CTMC یا DTMC و حل حالت پایدار است که در بخش‌های بعدی شرح داده می‌شود. در ابزار KU_PN_Tool گزینه‌هایی برای انتخاب هر یک از این مراحل وجود دارد. با انتخاب هر گزینه و با توجه به تنوع شبکهٔ پتری عملکرد سیستم متفاوت خواهد بود. نمودار گردشی شکل (۲) عملکرد این گزینه را در حالات مختلف شرح می‌دهد.



شکل (۲): نمودار گردشی عملکرد KU_PN_Tool برای تحلیل حالت پایدار در پاسخ به انتخاب گزینه‌های مختلف

۴.۳. رسم گراف دسترسی

در گراف دسترسی مجموعهٔ گره‌ها، فضای حالت شبکهٔ پتری را تشکیل می‌دهند هر گره معادل یک نشانه گذاری (Marking) است که به‌صورت دایره رسم شده و شامل شمارهٔ گره و نشانه گذاری حالت مربوط است. هر کمان جهت‌دار در این گراف یک گذر حالت را نشان می‌دهد. چون گذر حالت با شلیک یک گذر اتفاق می‌افتد، شمارهٔ آن گذر در برچسب کمان مربوط نوشته می‌شود تا مشخص کند با شلیک کدام گذر این تغییر حالت اتفاق می‌افتد. شکل (۳ الف) گراف دسترسی مثال ۱ را که قبلاً کد و گراف پتری آن را در شکل (۱) ملاحظه کردید نشان می‌دهد.

توجه شود که امکان تولید و رسم گراف دسترسی برای انواع مختلف شبکهٔ پتری شامل شبکهٔ پتری فوری (PN)، زمان ثابت (TPN)، زمان تصادفی (SPN)، زمان ثابت توسعه‌یافته (GTPN)، زمان تصادفی توسعه‌یافته (GSPN) وجود دارد. به‌طور کلی فضای حالت PN و SPN و GSPN برابر ولی TPN و GTPN از آن ها محدودتر است. از طرفی فضای حالت GTPN و GSPN دارای حالات گذرای ناپدیدشونده (vanishing states) خواهد بود که در تبدیل آن‌ها به زنجیرهٔ مارکوف و حل حالت پایدار این حالات حذف می‌شوند.

در ابزار KU_PN_Tool گزینه‌ای به نام RMG/TRMG برای رسم گراف دسترسی پیش‌بینی شده است. با انتخاب این گزینه برای شبکهٔ پتری که به‌عنوان ورودی انتخاب شده، گراف دسترسی رسم می‌شود. این گزینه بسته به نوع شبکهٔ پتری عملکردهای متفاوتی از خود نشان می‌دهد. با توجه به نمودار گردشی شکل (۲) عملکرد این گزینه در حالات مختلف به شرح زیر است:

- ✓ اگر شبکهٔ پتری از نوع فوری (تمام گذرهای آن فوری) زمان تصادفی (تمام گذرهای آن تصادفی) باشد گراف دسترسی RMG تولید و رسم خواهد شد. در ترسیم گراف RMG روی تمام کمان‌ها فقط نام گذر نشان داده می‌شود.
- ✓ اگر شبکهٔ پتری از نوع GSPN باشد، یعنی شامل مخلوطی از گذرهای فوری و زمان تصادفی باشد، گراف دسترسی گسترش یافتهٔ ERMG که شامل یک سری حالت‌های ناپدیدشونده است، تولید و رسم می‌شود. در ترسیم گراف ERMG روی تمام کمان ها فقط نام گذر نشان داده می‌شود. ضمن اینکه در ترسیم گراف ERMG هر دو نسخهٔ قبل و بعد از حذف حالات ناپدیدشونده رسم می‌شود.
- ✓ اگر شبکهٔ پتری از نوع زمان ثابت یعنی TPN باشد، گراف دسترسی زمان‌دار که ما آن را TRMG می‌نامیم، ساخته و رسم خواهد شد. تفاوت گراف TRMG با RMG این است که با توجه به زمان ثابت بودن گذرها کمتر رقابت پیش می‌آید و معمولاً از میان گذرهای همزمان– فعال، آنکه زمان تأخیر کوتاه‌تری دارد همیشه زودتر شلیک می‌شود. بنابراین در گراف TRMG مگر در مواردی که تأخیر گذرهای همزمان–فعال برابر است هیچ انشعابی نخواهیم داشت. به همین دلیل گراف دسترسی TRMG از این جهت نسبت به RMG ساده‌تر خواهد بود. از طرف دیگر با توجه به اینکه در TRMG علاوه بر نشانه‌گذاری، زمان باقی مانده تا شلیک بعدی نیز تعیین‌کنندهٔ حالت است ممکن است یک نشانه گذاری چندین بار و هر بار با مقدار زمان تا شلیک متفاوت در گراف TRMG ظاهر شود. و این امر گاهی باعث بزرگ‌تر شدن TRMG می‌شود. در ترسیم گراف TRMG روی تمام کمان‌ها فقط نام گذر نشان داده می‌شود.
- ✓ اگر شبکهٔ پتری از نوع زمان GTPN باشد، یعنی مخلوطی از گذرهای فوری و زمان‌دار داشته باشد، گراف دسترسی زمان‌دار آن که ما آن را ETRMG می‌نامیم ساخته و رسم خواهد شد. وجود گذرهای فوری باعث ایجاد حالت‌های ناپدیدشونده از یک طرف و در مواردی که همزمان فعال می‌شوند باعث ایجاد انشعاب در گراف دسترسی می‌شوند. انشعاب‌ها در گراف دسترسی دارای وزن احتمال هستند، ولی در ترسیم گراف ETRMG روی تمام کمان ها فقط نام گذر نشان داده می‌شود. برای مشاهدهٔ وزن احتمال گذرها روی کمان‌ها از گزینهٔ رسم گراف DTMC/CTMC استفاده کنید. ضمن اینکه در ترسیم گراف ETRMG هر دو نسخهٔ قبل و بعد از حذف حالات ناپدیدشونده رسم می‌شود.

۵.۳. رسم گراف CTMC و DTMC

گزینهٔ DTMC/CTMC در ابزار KU_PN_Tool برای رسم زنجیرهٔ مارکوف گسسته/پیوسته زمان پیش‌بینی شده است. اگرچه این گزینه بسته به نوع شبکهٔ پتری عملکردهای متفاوتی از خود نشان می‌دهد، به طور کلی گراف CTMC یا DTMC به‌نوعی همان گراف دسترسی است که تنها برچسب کمان‌های گذر حالت آن‌ها عوض شده است. هر گره در این گراف‌ها با یک دایره نشان داده شده و شامل یک شمارهٔ گره و نشانه‌گذاری حالت مربوطه است. با توجه به نمودار گردشی شکل (۲) عملکرد این گزینه در حالات مختلف به شرح زیر است:

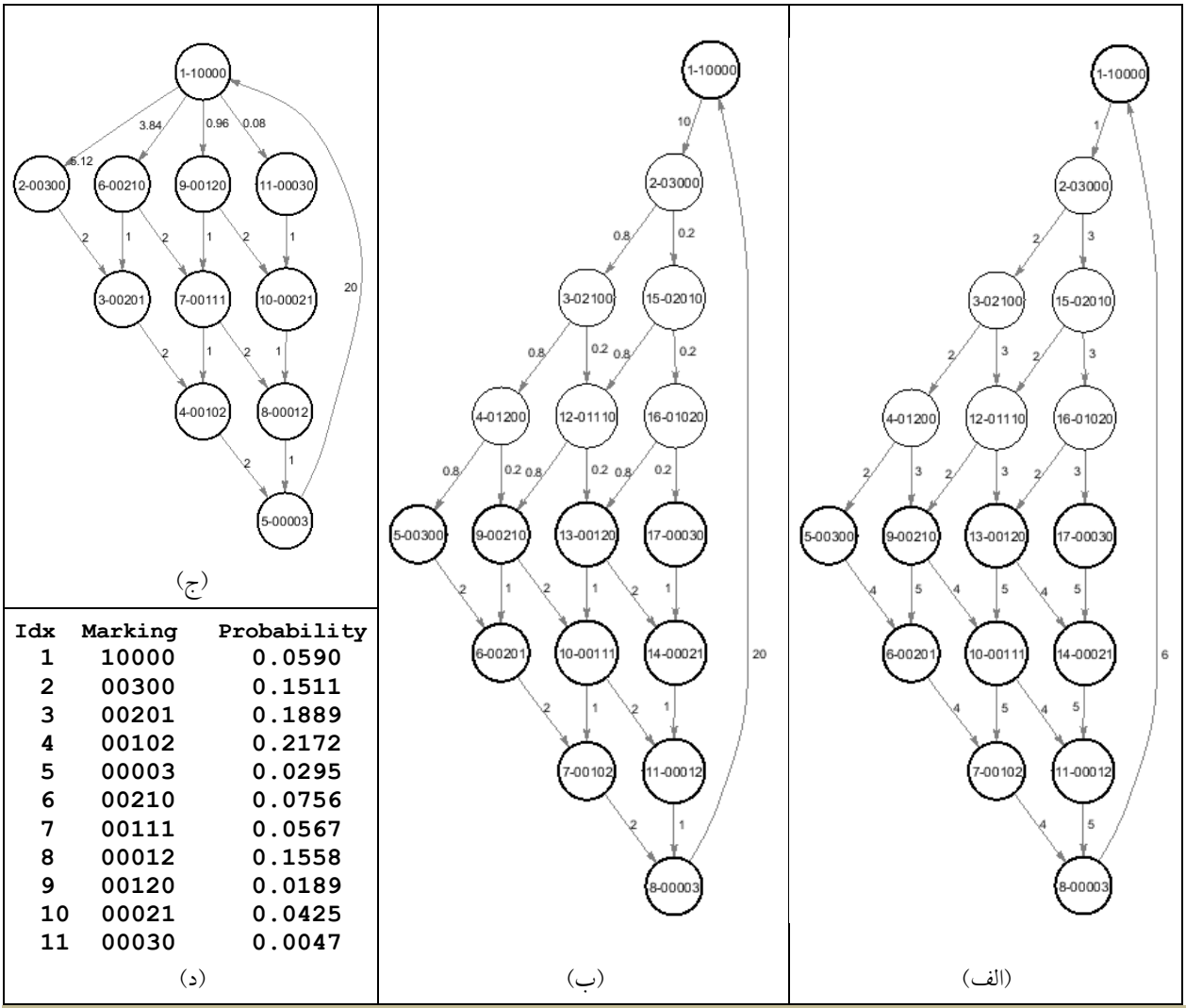
- ✓ اگر شبکهٔ پتری از نوع فوری (تمام گذرها فوری) باشد گراف DTMC رسم خواهد شد که شکل آن کاملاً مشابه RMG است با این تفاوت که روی کمان‌ها به‌جای نام گذر احتمال شلیک گذر که یک عدد بین صفر و یک است نوشته می‌شود.
- ✓ اگر شبکهٔ پتری از نوع زمان تصادفی (SPN) یعنی تمام گذرهای آن زمان تصادفی باشد، CTMC رسم خواهد شد. شکل CTMC کاملاً مشابه RMG است با این تفاوت روی کمان‌ها به‌جای نام گذر نرخ شلیک گذر نوشته می‌شود.
- ✓ اگر شبکهٔ پتری از نوع GSPN باشد، یعنی شامل مخلوطی از گذرهای فوری و زمان تصادفی باشد، CTMC مشابه گراف دسترسی GSPN به دو شکل رسم می‌شود: اولی مشابه گراف دسترسی قبل از حذف حالت‌های ناپدیدشونده است با این تفاوت که در برچسب کمان‌ها به‌جای نام گذر، نرخ شلیک گذر (اگر گذر زمان تصادفی باشد) یا وزن احتمال شلیک گذر (اگر گذر فوری باشد) گذاشته می‌شود. گراف دوم مشابه گراف دسترسی بعد از حذف حالت‌های ناپدیدشونده است. در این نسخه که از آن برای حل حالت پایدار استفاده خواهد شد، برچسب همهٔ کمان‌ها نرخ شلیک گذر است.
- ✓ اگر شبکهٔ پتری از نوع زمان ثابت یعنی TPN باشد، گرافی کاملاً مشابه TRMG رسم خواهد شد. با این تفاوت روی کمان‌ها به‌جای نام گذر زمان باقی‌مانده تا شلیک گذر نوشته می‌شود. در صورتی که انشعاب وجود داشته باشد، علاوه بر زمان تا شلیک، وزن احتمال هریک از انشعاب‌ها هم در تحلیل حالت پایدار مدل اهمیت پیدا می‌کند. به همین دلیل نسخهٔ دیگری از گراف رسم می‌شود که در آن فقط وزن احتمال شلیک گذر به‌عنوان برچسب روی تمام کمان‌ها قرار گرفته است. برای حل حالت پایدار از نسخه سومی که از ترکیب این دو گراف به دست می‌آید و برچست کمان‌های آن از تقسیم وزن احتمال بر زمان تا شلیک به دست می‌آید.
- ✓ اگر شبکهٔ پتری از نوع GTPN باشد، یعنی شامل مخلوطی از گذرهای فوری و زمان ثابت باشد، دو گراف مشابه گراف‌های دسترسی ETRMG مربوط به قبل و بعد از حذف حالت های ناپدیدشونده رسم می‌شود. تفاوت گراف‌های رسم‌شده در اینجا این است که برچسب کمان ها به‌جای نام گذر، زمان تا شلیک گذر گذاشته می‌شود. البته با توجه به اینکه در تحلیل حالت پایدار، وزن احتمال هریک از انشعاب‌ها هم اهمیت دارد برای هر گراف یک تکرار دیگر نیز رسم می‌شود که در آن فقط وزن احتمال شلیک گذر به‌عنوان برچسب روی تمام کمان‌ها قرار گرفته است. بنابراین در این حالت ۴ گراف رسم خواهد شد. دو گراف اولیه و دو گراف کاهش یافته که هرکدام شامل یک گراف با کمان زمان‌دار و یک گراف با کمان احتمالی است.

مثال ۱: در شکل (۲) گراف دسترسی ERMG و CTMC مربوط به مثال ۱ نشان داده شده است. در شکل (۲ الف و ب) این دو گراف را قبل از حذف حالات ناپدیدشونده نشان می‌دهد که در آن‌ها حالات ناپدیدشونده با دایره‌های نازک‌تر رسم شده است. در ERMG شکل (۲ الف) برچسب کمان‌ها، شمارهٔ گذر است درحالی‌که در CTMC شکل (۲ ب) برچسب کمان‌ها نرخ شلیک گذر (برای گذرهای زمان تصادفی) و وزن احتمال شلیک گذر (برای گذرهای فوری) است.

شکل (۲ ج) CTMC نهایی مربوط به مثال ۱ را نشان می‌دهند. این شکل از روی شکل (۲ ب) و با حذف حالات ناپدیدشونده به وجود آمده است. در CTMC نهایی (شکل ج) با حذف هر حالت ناپدیدشونده، وزن احتمال ورود به آن در

نرخ کمان خروجی از آن ضرب شده و حاصل روی کمان‌های جایگزین آن ها قرار گرفته است. از CTMC به‌دست‌آمده برای حل حالت پایدار استفاده می‌شود.

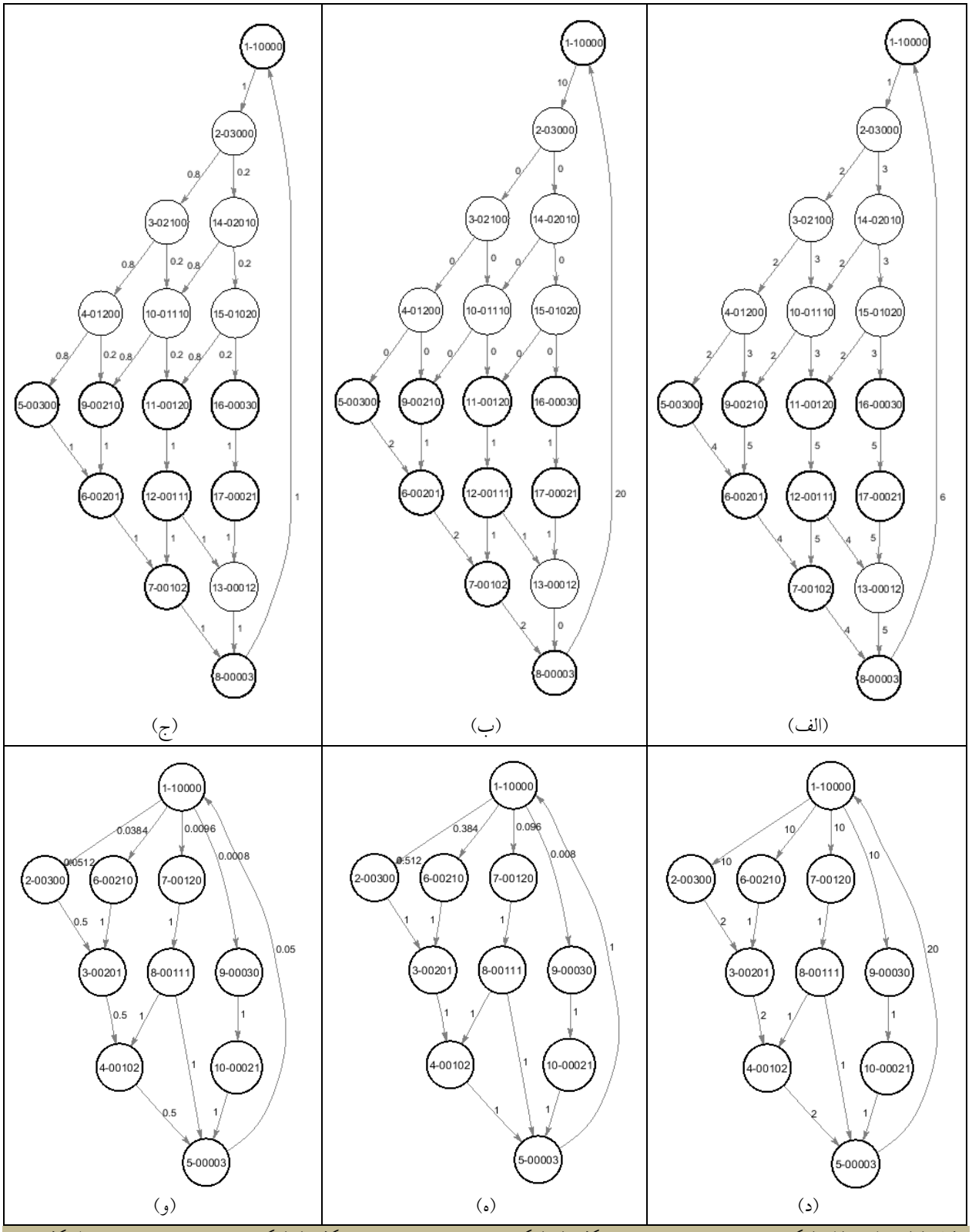
مثال ۲: شکل (۴ الف) گراف‌های دسترسی برای GTPN مثال۲ را نشان می‌دهد. شبکهٔ پتری مثال ۲ همان شبکهٔ پتری مثال ۱ است با این تفاوت که نوع تمام گذرهای زمان تصادفی (نوع ۱) آن (t0، t3، t4 و t5) به زمان ثابت (نوع ۱) تغییر کرده است. سه گراف الف تا ج در این شکل ETRMG را قبل از حذف حالت‌های ناپدیدشونده نشان می‌دهد که به‌ترتیب با شمارهٔ گذر، زمان تا شلیک گذر و وزن احتمال شلیک گذر به عنوان برچسب کمان رسم شده است. گراف شکل (۴ د) همان گراف شکل (۴ ب) با حذف حالات ناپدیدشونده و گراف شکل (۴ ه) همان گراف شکل (۴ ج) با حذف حالات ناپدیدشونده است. آنچه برای حل حالت پایدار استفاده می‌شود، ترکیبی از دو گراف آخر است به این ترتیب که برچسب کمان آن حاصل‌ضرب وزن احتمال (وزن کمان‌ها در شکل ۴ د) در عکس زمان شلیک گذر (وزن کمان‌ها در شکل ۴ ه) خواهد بود که نرخ گذر حالت را ایجاد می‌کند. از گراف حاصل را که در شکل (۴ و) نشان داده شده، می‌توانیم به‌عنوان CTMC استفاده کرد و حل تقریبی حالت پایدار را برای GTPN به دست آورد.



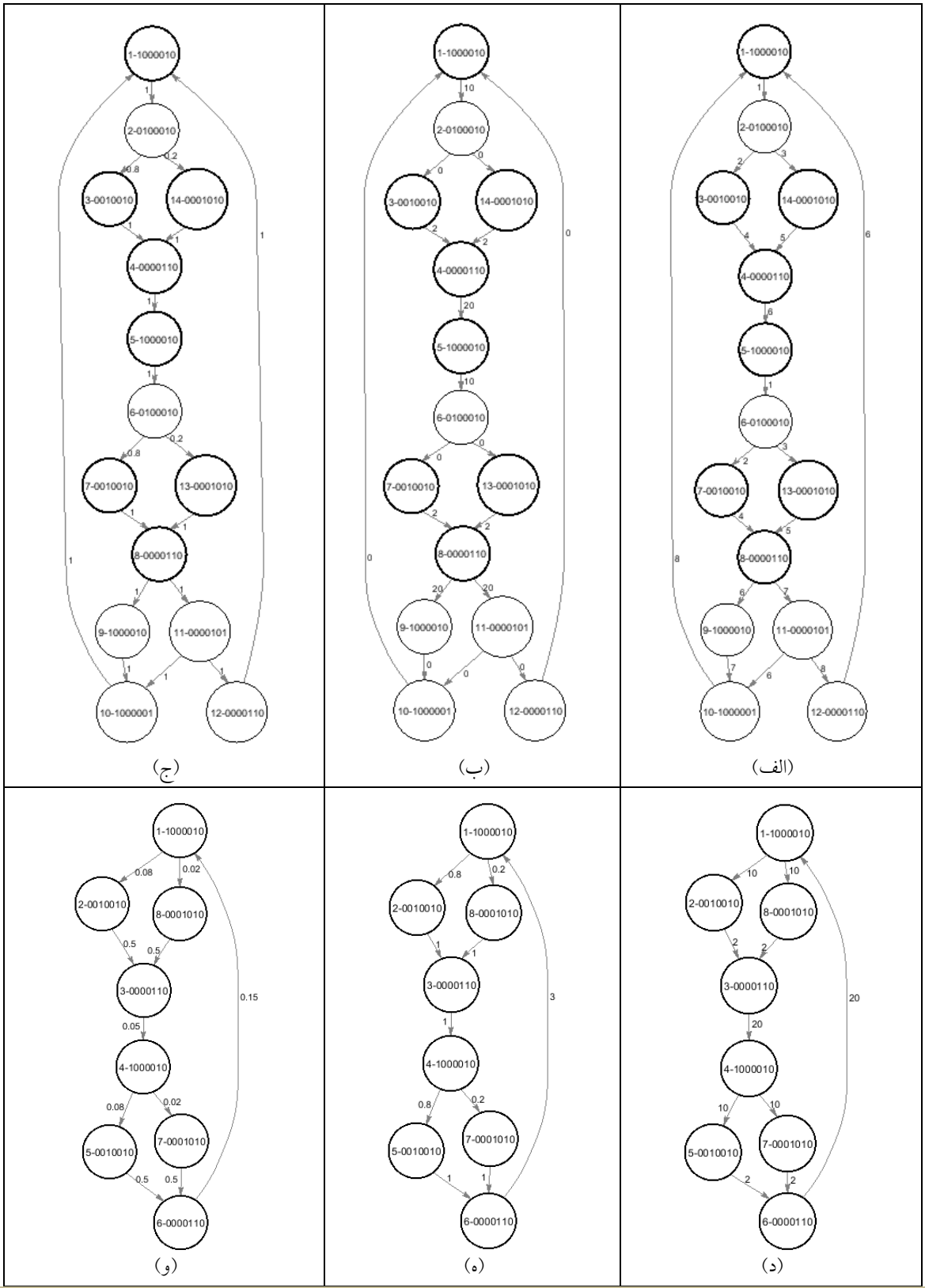
Idx	Marking	Probability
1	10000	0.0590
2	00300	0.1511
3	00201	0.1889
4	00102	0.2172
5	00003	0.0295
6	00210	0.0756
7	00111	0.0567
8	00012	0.1558
9	00120	0.0189
10	00021	0.0425
11	00030	0.0047

(د)

شکل (۳): مثال ۱ (الف) گراف دسترسی همراه با شمارهٔ گذر (ب) گراف دسترسی همراه با احتمال گذر (ج) گراف CTMC (د) حل حالت پایدار



شکل (۴): مثال ۲ (الف) گراف دسترسی همراه با شمارهٔ گذر (ب) گراف دسترسی همراه با زمان گذر (ج) گراف دسترسی همراه با احتمال گذر (د) گراف دسترسی کاهش یافته همراه با زمان گذر (ه) گراف دسترسی همراه با احتمال گذر (و) گراف CTMC



شکل (۵): مثال ۳ (الف) گراف دسترسی همراه با شمارهٔ گذر (ب) گراف دسترسی همراه با زمان گذر (ج) گراف دسترسی همراه با احتمال گذر (د) گراف دسترسی کاهش یافته همراه با زمان گذر (ه) گراف دسترسی همراه با احتمال گذر (و) گراف CTMC

علت تقریبی بودن آن هم این است که در حل حالت پایدار CTMC زمان بین دو شلیک متوالی هر گذر حالت باید تصادفی و دارای توزیع نمایی باشد، درحالی که ETRMG فاصله‌های زمانی معین و ثابت است. در روش پیشنهادی، مقدار هریک از این زمان‌های ثابت را که در حقیقت برابر با میانگین زمان شلیک هم هستند، معکوس کرده به‌عنوان میانگین نرخ شلیک در نظر گرفته و به‌صورت تقریبی مشابه نرخ شلیک نمایی با آن برخورد می‌شود.

۶.۳. حل حالت پایدار انواع شبکهٔ پتری

حل حالت دائمی هریک از انواع شبکهٔ پتری بر اساس زنجیرهٔ مارکوف زمان گسسته (DTMC) یا زنجیرهٔ مارکوف زمان پیوسته (CTMC) انجام می‌شود که بسته به نوع شبکهٔ پتری به‌نحوی که در بخش قبل شرح داده شد، از روی گراف دسترسی به دست می‌آید. منظور از حل حالت دائمی به دست آوردن احتمال هریک از حالت‌های DTMC یا CTMC در حالت دائمی است که به‌ترتیب با حل دستگاه‌های $\pi Q=0$ و $\pi P=\pi$ به دست می‌آیند. در این دستگاه‌ها π برداری به طول n (تعداد حالات زنجیرهٔ مارکوف) شامل $\pi_i, i=1,2,\ldots,n$ها است و P و Q دو ماتریس n در n هستند و به‌ترتیب ماتریس انتقال حالت و ماتریس تولید نامیده می‌شوند. برای اطلاعات بیشتر در زمینهٔ حل حالت دائمی به [۱۷] مراجعه کنید.

شکل (۳ د) پاسخ حالت پایدار شبکهٔ پتری مثال ۱ را نشان می‌دهد. در هر سطر از این جواب احتمال یکی از حالات گراف CTMC کاهش‌یافتهٔ شکل را نشان می‌دهد. شکل (۶) پاسخ حالت پایدار شبکهٔ پتری مثال ۲ را نشان می‌دهد. هر سطر از این پاسخ مربوط به یکی از حالات گراف CTMC تقریبی شکل (۴ و) است. شکل‌های (۷) و (۸) نیز مربوط به مثال‌های ۳ و ۴ است که در ادامه شرح داده خواهد شد.

Index	Marking	Trans delay	Token-delay	Steady State Probability
1	10000	00021(20)	{[0],[][][][]}	0.2833
2	00300	(10)0001(20)	{[][0,0,0],[][][]}	0.0290
3	00201	(10)0001(20)	{[][0,0],[0],[][]}	0.0508
4	00102	(10)0001(20)	{[][0],[0,0],[][]}	0.0535
5	00003	(10)00210	{[][][0,0,0],[][]}	0.5666
6	00210	(10)0010(20)	{[][][0,0],[][0],}	0.0109
7	00120	(10)0010(20)	{[][][0],[][0,0],}	0.0027
8	00111	(10)0000(20)	{[][][][][0,0,0],}	0.0027
9	00030	(10)0020(20)	{[][][0,0],[0],[]}	0.0002
10	00021	(10)0020(20)	{[][0],[0],[0],[]}	0.0002

شکل (۶): نتیجهٔ حل حالت پایدار مثال ۲

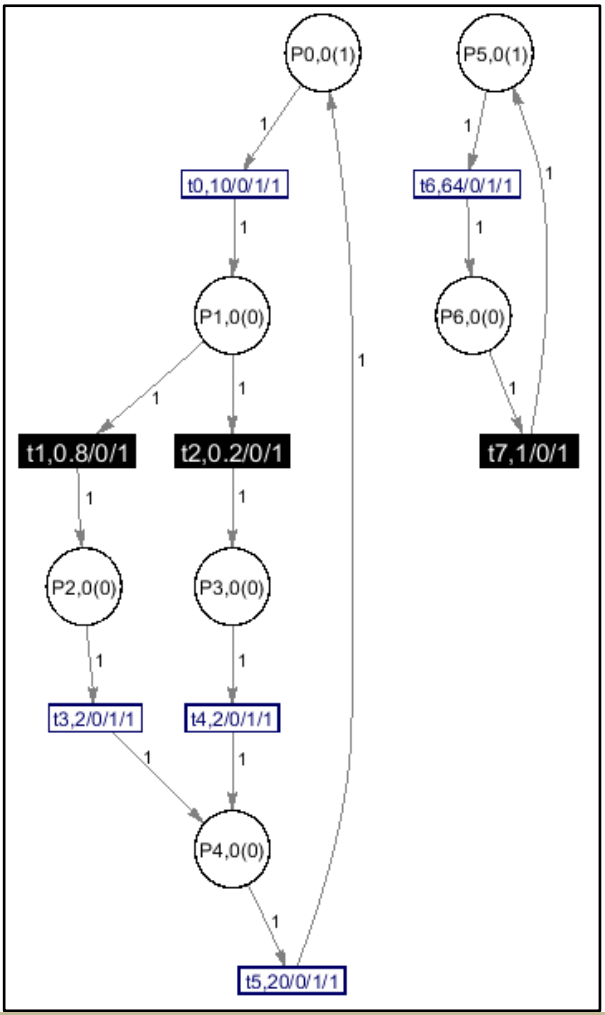
Index	Marking	Trans delay	Token-delay	Steady State Probability
1	1000010	00022(20)(54)0	{[0],[],[],[],[],[0],[]}	0.1562
2	0010010	(10)0002(20)(52)0	{[],[],[0],[],[],[0],[]}	0.0250
3	0000110	(10)00220(32)0	{[],[],[],[],[0],[0],[]}	0.3125
4	1000010	00022(20)(22)0	{[0],[],[],[],[],[0],[]}	0.1562
5	0010010	(10)0002(20)(20)0	{[],[],[0],[],[],[0],[]}	0.0250
6	0000110	(10)0022000	{[],[],[],[],[0],[0],[]}	0.3125
7	0001010	(10)0020(20)(20)0	{[],[],[],[0],[],[0],[]}	0.0062
8	0001010	(10)0020(20)(52)0	{[],[],[],[0],[],[0],[]}	0.0062

شکل (۷): نتیجهٔ حل حالت پایدار مثال ۳

Index	Marking	Trans delay	Token-delay	Steady State Probability
1	1000010	00022(20)00	{[0],[],[],[],[],[54],[]}	0.1562
2	0010010	(10)0002(20)00	{[],[],[0],[],[],[52],[]}	0.0250
3	0000110	(10)0022000	{[],[],[],[],[0],[32],[]}	0.3125
4	1000010	00022(20)00	{[0],[],[],[],[],[22],[]}	0.1562
5	0010010	(10)0002(20)00	{[],[],[0],[],[],[20],[]}	0.0250
6	0000110	(10)0022000	{[],[],[],[],[0],[0],[]}	0.3125
7	0001010	(10)0020(20)00	{[],[],[],[0],[],[20],[]}	0.0062
8	0001010	(10)0020(20)00	{[],[],[],[0],[],[52],[]}	0.0062

شکل (۸): نتیجهٔ حل حالت پایدار مثال ۴

توجه شود که در گراف دسترسی ETRMG هر حالت با سه ویژگی مشخص می‌شود: ۱. نشانه‌گذاری؛ ۲. مقدار تأخیر باقی‌مانده تا شلیک گذر که در شمارنده‌های گذرها مشخص می‌شود؛ ۳. زمان باقی‌ماندهٔ تأخیر نشانه‌ها در صورت وجود. در پیاده سازی این موضوع هر دو تأخیر (مورد دوم و سوم) در مقدار تأخیر باقی‌مانده در لحظهٔ قبل از شلیک در نظر گرفته شده است. در مثال ۲ با توجه به اینکه به‌طور کلی از تأخیر نشانه استفاده نشده، فقط مقدار شمارنده‌های تأخیر گذر و نشانه‌گذاری اهمیت پیدا می‌کند. در صورتی که دو نشانه‌گذاری یکسان دارای تأخیر گذر متفاوت و/یا تأخیر نشانهٔ متفاوت باشند، دو حالت متفاوت در نظر گرفته می‌شوند. در مثال فوق چنین حالتی اتفاق نیفتاده است. برای درک بهتر موضوع دو مثال ۳ و ۴ را ملاحظه کنید.



شکل (۹): گراف پتری مثال ۳

مثال ۳: اهمیت زمان باقی‌ماندهٔ تأخیر گذر

شکل (۹) یک شبکهٔ پتری GTPN را که دو بخش مجزا دارد نشان می‌دهد. بخش سمت چپ مشابه مثال ۲ است که وزن کمان t0 به p1 و p4 به t5 به یک تغییر کرده است. انواع گراف دسترسی این شبکه را در شکل (۵) ملاحظه می‌کنید. پاسخ حل حالت پایدار آن در شکل (۷) آمده است. در این شکل حالت‌هایی وجود دارد که از نظر نشانه‌گذاری یکسان‌اند ولی از نظر زمان تأخیر گذر متفاوت‌اند. حالت‌های ۱، ۲، ۳ و ۸ از نظر نشانه‌گذاری با حالت‌های ۴، ۵، ۶ و ۷ برابرند، درحالی‌که در رسم گراف دسترسی به‌عنوان حالت‌های متفاوت در نظر گرفته شده‌اند. به عبارت دیگر، در تعیین حالت جدید علاوه نشانه‌گذاری، زمان تأخیر گذر نیز در نظر گرفته شده است. البته در این مثال نیز مانند مثال ۲ تأخیر نشانه وجود ندارد. در مثال ۴ تأثیر تأخیر نشانه نشان داده می‌شود.

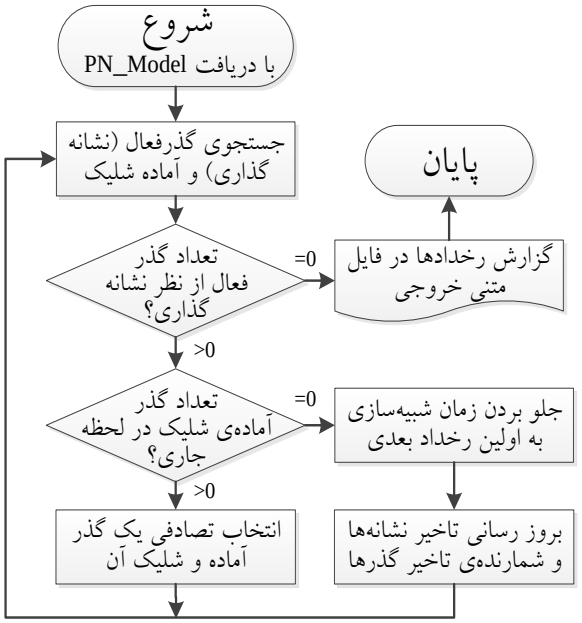
مثال ۴: اهمیت زمان تأخیر نشانه

همان مدل پتری مثال ۳ (یعنی شکل ۹) را در نظر بگیرید. با این تفاوت که زمان تأخیر گذر t6 صفر و به‌جای آن تأخیر نشانه در گذر t7 برابر ۶۴ باشد. با این تغییر، مشابه همان رفتار را با استفاده ار تأخیر نشانه ایجاد کرده‌ایم. این مدل از نظر عملکردی مشابه مثال ۳ است با این تفاوت که زمان تأخیر در گذر t6 توسط گذرهای t7 روی نشانهٔ ورودی t6 قرار داده می‌شود. انواع گراف دسترسی این مدل نیز مشابه مثال ۳ است که در شکل (۵) ملاحظه می‌کنید. پاسخ حل حالت پایدار آن در شکل (۸) آمده است. در این شکل حالت‌هایی وجود دارد که از نظر نشانه‌گذاری و باقی‌مانده تأخیر گذر یکسان‌اند ولی از نظر زمان نشانه باهم متفاوت‌اند. به عبارت دیگر، علاوه بر نشانه‌گذاری و باقی‌ماندهٔ تأخیر گذر، زمان تأخیر نشانه نیز در تعیین حالت در نظر گرفته شده است.

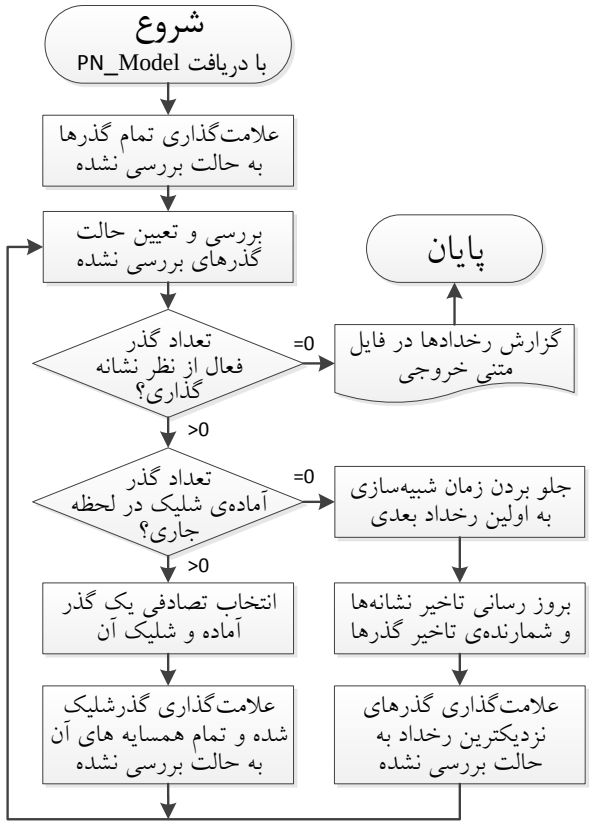
۷.۳. شبیه‌سازی انواع شبکهٔ پتری

برای شبیه‌سازی انواع شبکهٔ پتری با انواع گذر مختلف نیازمند یک الگوریتم جامع شبیه‌سازی هستیم، به‌طوری که بتواند از میان انواع گذرها به‌طور صحیح گذرهای آماده شلیک را شناسایی و

از میان آن‌ها بر اساس قوانین اولیه، یک گذر را انتخاب و شلیک کند.



شکل (۱۰): فلوچارت الگوریتم شبیه‌سازی اولیه



شکل (۱۱): فلوچارت الگوریتم شبیه‌سازی بهبودیافته

در ابزار KU_PN_Tool ابتدا یک الگوریتم شبیه‌سازی اولیه که عملکرد آن به‌صورت خلاصه در نمودار گردشی در شکل (۱۰) نمایش داده شده است در تابعی به نام Simulate پیاده‌سازی شد، سپس برای افزایش سرعت شبیه‌سازی نسخهٔ بهبودیافته‌ای به نام Simulate_Fast_version پیاده‌سازی شد که نمودار گردشی آن در شکل (۱۱) آمده است. جزئیات این دو الگوریتم در ادامه شرح داده می‌شود.

۱.۷.۳. الگوریتم شبیه‌سازی اولیه

خلاصهٔ عملکرد الگوریتم اولیهٔ شبیه‌ساز به شرح زیر است:

۱. در تیک زمانی جاری فهرست گذرهای توانا از نظر نشانه‌گذاری، بر اساس شرایط زیر تهیه شوند.
   - ✓ تمام مکان‌های ورودی دارای حداقل تعداد توکن برابر وزن کمان ورودی باشند.
   - ✓ همهٔ مکان‌های بازدارنده دارای تعداد توکنی کمتر از وزن کمان بازدارندهٔ مربوط باشند.
   - ✓ تعداد توکن در مکان‌های خروجی پس از شلیک از ظرفیت آن‌ها تجاوز نکند.
۲. اگر فهرست استخراج‌شده از مرحلهٔ اول خالی است، شبیه‌سازی متوقف و نتایج گزارش شود، وگرنه مرحلهٔ ۳ انجام شود.
۳. از میان گذرهای مرحلهٔ اول فهرست گذرهایی را که بر اساس شرایط زیر در تیک زمانی جاری آمادهٔ شلیک هستند استخراج کنید.
   - ✓ اگر گذر زمانی یا زمان تصادفی است، تأخیر توکن‌های ورودی به صفر رسیده باشد (این شرط برای گذرهای فوری چک نمی‌شود یعنی گذر فوری می‌تواند با وجود تأخیر غیر صفر در توکن ورودی شلیک شود).
   - ✓ اگر گذر از نوع زمان ثابت یا زمان تصادفی است، باید شمارندهٔ مربوط به گذر (یعنی CountT) مقدارش صفر شده باشد.
۴. اگر فهرست استخراج‌شده از مرحلهٔ ۳ دارای عضوی هست مراحل زیر انجام شود:
   - ✓ از فهرست گذرهای فعال، فهرستی شامل گذرهایی که

به‌طور مشترک در بالاترین سطح اولویت قرار دارند استخراج شود.

- ✓ از زیرفهرست انتخاب‌شده یکی از گذرها به‌طور تصادفی و بر اساس وزن احتمال آن‌ها انتخاب و شلیک شود.
- ✓ به مرحلهٔ ۱ برود.

۵. اگر فهرست مرحلهٔ ۳ خالی است فاصلهٔ زمانی تا رخداد بعدی به‌عنوان مقدار پرش زمانی (num_of_skip_tick) محاسبه و به زمان (ساعت) شبیه‌سازی اضافه شود. همراه با جلو بردن زمان عدد شمارنده‌های گذرهای زمانی نیز به همان میزان کاهش یافته و میزان تأخیر توکن‌های زمان‌دار نیز کاهش می‌یابد. مقدار پرش زمانی حداقل زمان لازم برای صفر شدن توکن‌های یک گذر یا صفر شدن زمان شمارندهٔ آن است. پس از این مرحله به مرحلهٔ ۱ برود.

با توجه به الگوریتم فوق موتور شبیه‌ساز KU_PN_Tool بین انواع گذر فعال (یعنی انواع فوری، زمان ثابت و زمان تصادفی) به گذرهای فوری اولویت می‌دهد. سپس از میان گذرهای فعال فوری گذرهایی را که دارای اولویت بالاتر (عدد Priority کمتر) است انتخاب می‌کند و در مرحلهٔ آخر و در شرایط مساوی سراغ وزن احتمال (ProbWeight) رفته و به‌صورت تصادفی یکی را انتخاب می‌کند. اگر عدد ProbWeight برای همهٔ گذرها برابر (یا مساوی صفر) باشد وزن احتمال را برابر در نظر می‌گیرد.

۲.۷.۳. الگوریتم شبیه‌سازی بهبودیافته

در الگوریتم بهبودیافته برای هر گذر یک ویژگی به نام Tr_State تعریف شده است. این ویژگی ترکیبی از ۴ ویژگی فرعی دیگر است که هرکدام با یک بیت نشان داده می‌شود. شرح بیت‌ها و معرفی هریک از ویژگی‌های فرعی در زیر آمده است:

bit0: برقرار بودن شرط نشانه‌گذاری (marking) گذر فعال؛

bit1: صفر بودن زمان تأخیر تمام نشانه‌های ورودی گذر؛

bit2: اتمام تأخیر گذر؛

bit3: گذر چک نشده و باید چک شود.

بنابراین حالت‌های مختلف یک گذر به‌صورت ترکیبی از این ۴ بیت به شکل جدول (۷) خواهد بود.

جدول (۷): معرفی حالات گذر در هنگام شبیه‌سازی

شرح حالت	مقدار حالت	مقدار بیت‌های Tr_State			
		bit0	bit1	bit2	bit3
شرط نشانه‌گذاری برقرار نیست. تأخیر نشانه صفر نیست. تأخیر گذر هم صفر نیست.	0	0	0	0	0
شرط نشانه‌گذاری برقرار است. تأخیر نشانه صفر نیست. تأخیر گذر هم صفر نیست.	1	1	0	0	0
شرط نشانه‌گذاری برقرار است. تأخیر نشانه صفر است ولی تأخیر گذر تمام نشده	3	1	1	0	0
شرط نشانه‌گذاری برقرار است، تأخیر نشانه صفر نیست ولی تأخیر گذر تمام شده	5	1	0	1	0
شرط نشانه‌گذاری برقرار است، تأخیر نشانه صفر است و تأخیر گذر تمام شده آمادهٔ شلیک است	7	1	1	1	0
تاکنون هیچ‌یک از شرط‌های فعال بودن برقرار نبوده ولی باید بررسی شود.	8	0	0	0	1
شرط نشانه‌گذاری برقرار است ولی باید از نظر صفر بودن تأخیر نشانه و تأخیر گذر بررسی شود.	9	1	0	0	1

در الگوریتم بهبودیافتهٔ شبیه‌سازی، برای یافتن گذر فعال، به‌جای بررسی تک‌تک گذرها، تنها گذرهایی که احتمال فعال شدن آن ها وجود دارد بررسی می‌شود. هر گذر در هر لحظه می‌تواند در هر یکی از حالات جدول (۷) باشد و در طی اجرای الگوریتم وضعیت‌های تغییر می‌کند. با توجه به فلوچارت شکل (۱۱) مراحل اجرا الگوریتم به‌صورت زیر است.

۱. ابتدا تمام گذرها به حالت بررسی‌نشده (۸) تنظیم شوند.

۲. فقط وضعیت گذرهای بررسی‌نشده (۸ و ۹) به شکل زیر بررسی و به‌روزرسانی شوند.

- ✓ گذرهای حالت ۸ از نظر نشانه‌گذاری و تأخیرها بررسی

شوند و به یکی از حالت‌های ۰، ۱، ۳، ۵ یا ۷ برده شوند.

✓ گذرهای حالت ۹ فقط از نظر تأخیر (نشانه‌گذاری و گذر) بررسی و به یکی از حالات ۱، ۳، ۵ یا ۷ برده شوند.

۳. در صورتی که تمام گذرها در وضعیت ۰ باشند (بن بست) یا تعداد گذرهای شلیک‌شده از ماکزیمم تعیین‌شده توسط کاربر بیشتر شود، شبیه‌سازی به پایان برسد وگرنه کار از مرحلهٔ بعد ادامه پیدا کند.

۴. در صورت موجود بودن گذرهایی در حالت (۷) به مرحلهٔ ۵ وگرنه به ۶ برود.

۵. از میان گذرهای حالت (۷) یکی به تصادف انتخاب و شلیک شده و گذر شلیک شده و گذرهای همسایهٔ آن به حالت ۸ برده شوند و کار از مرحلهٔ ۲ به بعد ادامه پیدا کند.

۶. زمان شبیه‌سازی به زمان رخداد بعدی جلو برده شود و تمام گذرهایی که در حالت ۱، ۳ و ۵ هستند به حالت ۹ برده شوند و کار از مرحلهٔ ۲ به بعد ادامه پیدا کند.

نتیجهٔ شبیه‌سازی در فایلی متنی که توسط کاربر در منوی اصلی برنامه مشخص شده نوشته می‌شود. این فایل شامل فهرستی از رخدادهای شلیک گذر و نشانه‌گذاری‌هاست. اگر شبکهٔ پتری بدون زمان باشد، رخدادها به ترتیبی که اتفاق می‌افتند ثبت می‌شوند. اگر شبکه شامل گذرهای زمان ثابت یا زمان تصادفی باشد، زمان رخداد آن‌ها نیز مشخص می‌شود. در پایان فایل متنی، زمان شبیه‌سازی و زمان اجرای مدل ذکر می‌شود. زمان شبیه‌سازی و زمان اجرا علاوه بر ثبت در پایان فایل متنی، به‌عنوان دو پارامتر خروجی نیز توسط تابع شبیه‌ساز برگردانده می‌شوند. زمان اجرا وابسته به تأخیرهای درون مدل بوده و زمان اجرای واقعی سیستمی را که مدل شده نشان می‌دهد. درحالی‌که زمان شبیه‌سازی زمانی است که برنامهٔ شبیه‌ساز در حال اجرا بوده تا به پایان برسد. زمان شبیه‌سازی علاوه بر الگوریتم شبیه‌سازی به مشخصات سخت‌افزاری و سیستم‌عامل ماشینی که برنامه شبیه‌ساز روی آن اجرا می‌شود نیز بستگی خواهد داشت.

جدول (۸) زمان شبیه‌سازی را برای دو الگوریتم شبیه‌سازی ارائه شده، شامل الگوریتم اولیه و الگوریتم بهبودیافته، با هم مقایسه کرده است. شبیه‌سازی برای ۱۰۰۰۰ رخداد روی تعدادی از مثال‌های بدون بن‌بست که در پوشهٔ examples از ابزار ساخته‌شده قرار دارد انجام شده است.

جدول (۸): مقایسهٔ نتایج شبیه‌سازی الگوریتم اولیه و بهبودیافته

	Old Simulation Algorithm (s)		New Simulation Algorithm (s)	
Model Name	Run_Time	Sim_Time	Run_Time	Sim_Time
CloseLoop_1in1	102520.00	7.6150	102520.00	4.4862
CloseLoop_2in1	0533.5042	8.6850	0532.6928	4.3923
Counter6	0000.0000	3.7613	0000.0000	1.7295
Exclusion1	0807.4433	9.3130	0790.1657	4.6799
Exclusion2	0000.0000	5.2204	0000.0000	2.8689
GSPN1	0610.2585	6.9163	0619.9797	4.9283
InhArcExample	0000.0000	3.7936	0000.0000	1.2342
Paper_example1	2051.9575	6.8726	2052.2560	3.9572
Paper_example2	43512.000	7.8264	43525.000	3.7894
Paper_example3	64000.000	7.1043	64000.000	3.9298
Paper_example4	39999.000	6.8459	39998.000	3.4916
Paper_example5	2014.0239	7.4396	2016.7607	4.5861
philosopher_dining	0000.0000	6.4995	0000.0000	3.6301
Resource_Sharing	700040.00	6.5730	700040.00	4.5498
	average	6.7476	average	3.7324

۴. معرفی قابلیت‌های انحصاری ابزار ساخته‌شده

هدف این فصل بیان قابلیت‌های منحصربه‌فرد ابزار ساخته‌شده است. برخی از این قابلیت‌ها با ارائهٔ مثال‌های ساده و برخی دیگر با ارجاع به کارهای انجام‌شده نشان داده خواهد شد.

۱.۴. تولید خودکار مدل با استفاده از حلقهٔ تکرار

از جمله قابلیت‌های این ابزار تولید خودکار مدل با استفاده از حلقهٔ تکرار در تابع سازنده است. در صورتی که مدل پتری دارای ساختار منظم و شامل تعداد زیادی گذر، مکان و کمان باشد، می‌توان با فراخوانی توابع ایجاد گذر و مکان و کمان در داخل حلقهٔ تکرار با یک طول کد کوتاه‌تر تابع سازنده را نوشت. شکل (۱۲) کد تابع سازندهٔ مدل مسابقات فوتبال بین تیم‌های یک گروه چهارتیمه را نشان می‌دهد. همان طور که کد این تابع نشان می‌دهد، برای ساخت مدل از حلقه‌های تکرار استفاده شده است. با هر بار شبیه‌سازی این تابع نتایج تک‌تک

بازی‌ها به‌صورت تصادفی و بر اساس وزن‌های احتمال تعیین‌شده مشخص و امتیاز نهایی هر تیم مشخص می‌شود. این مثال کوچک یکی از قابلیت‌های مهم ابزار ساخته‌شده را نشان می‌دهد. این قابلیت ما را قادر می‌سازد با استفاده از تکنیک‌های برنامه‌نویسی، شبکه‌های پتری بزرگ با ساختارهای متنوع ایجاد کرد. شبکه‌هایی که به‌علت پیچیدگی و بزرگ بودن نه با استفاده از ترسیم گرافیکی و نه کدنویسی ساده بدون تکرار، ایجاد آن‌ها بسیار دشوار، زمانبر و همراه با خطاهای انسانی زیاد خواهد بود.

```
function PN_model=Football_Play_4team()
  [PN_model] = Init_PN('Football_Play_4team');
  for i=1:4
  [PN_model,Team(i)]=New_Place(PN_model,['Team',num2str(i)],0,1,{[1,1,0,0],[1,1,0,0],[1,1,0,0]});
   [PN_model,Score(i)]=New_Place(PN_model,['Score',num2str(i)],0,1,{});
end
t=1;
for i=1:3
  for k=i+1:4
    [PN_model,Playing(t)]=New_Place(PN_model,['Played',num2str(t)],0,1,{});
    [PN_model,Permission(t)]=New_Place(PN_model,['Permit',num2str(t)],0,1,{[1,1,0,0]});
  [PN_model,Do_Play(t)]=New_Transition(PN_model,['Do_Play',num2str(t)],'General_func',0,0,0,t,1);
    [PN_model,Win(t)]=New_Transition(PN_model,['Win',num2str(t)], 'General_func',0,0,0,t,0.3);
    [PN_model,Lose(t)]=New_Transition(PN_model,['Lose',num2str(t)], 'General_func',0,0,0,t,0.3);
    [PN_model,Equal(t)]=New_Transition(PN_model,['Equal',num2str(t)],'General_func',0,0,0,t,0.4);
    PN_model=Weighted_Arc_P2T(PN_model,Team(i),Do_Play(t),1);
    PN_model=Weighted_Arc_P2T(PN_model,Team(k),Do_Play(t),1);
    PN_model=Weighted_Arc_P2T(PN_model,Permission(t),Do_Play(t),1);
    PN_model=Weighted_Arc_P2T(PN_model,Playing(t),Win(t),1);
    PN_model=Weighted_Arc_P2T(PN_model,Playing(t),Lose(t),1);
    PN_model=Weighted_Arc_P2T(PN_model,Playing(t),Equal(t),1);

    PN_model=Weighted_Arc_T2P(PN_model,Do_Play(t),Playing(t),1);
    PN_model=Weighted_Arc_T2P(PN_model,Win(t),Score(i),3);
    PN_model=Weighted_Arc_T2P(PN_model,Lose(t),Score(k),3);
    PN_model=Weighted_Arc_T2P(PN_model,Equal(t),Score(i),1);
    PN_model=Weighted_Arc_T2P(PN_model,Equal(t),Score(k),1);
    t=t+1;
   end
end
```

شکل (۱۲): کد سازندهٔ مدل پتری بازی فوتبال در یک گروه چهارتیمه

۲.۴. تغییر ساختار مدل پتری در حین اجرا

منظور از تغییر ساختار، حذف یا اضافه کردن گذر و مکان و کمان‌های ارتباطی آن‌ها در حین اجراست. برای نشان دادن قابلیت یادشده به دو نمونه‌کار انجام‌شده که با استفاده از نسخه‌های اولیه از ابزار KU_PN_Tool پیاده‌سازی شده است اشاره می‌شود. در [۸] روشی مبتنی بر مدل پتری برای ارزیابی نگاشت یک کاربرد روی یک شبکهٔ درون‌تراشه‌ای ارائه شد. در این روش، ابتدا گراف وظیفه به‌صورت یک شبکهٔ پتری مدل شده، سپس اطلاعات نگاشت به آن اضافه و شبیه‌سازی می‌شد. اطلاعات نگاشت شامل ارتباط (تبادل پیغام) بین هسته‌های پردازشی نیز می‌شد. به عبارت دیگر، برای مدل‌سازی کاربرد نگاشت‌شده باید تبادل پیغام‌ها نیز به‌عنوان یک سری وظایف ارتباطی به مدل افزوده شود تا تأخیر ارتباطات را در نظر گرفته و تخمین دقیق‌تری از زمان اجرای کاربرد ارائه کند. ازاین‌رو این وظایف ارتباطی به‌صورت گذرهایی بین گذرهای پردازشی اضافه شد تا از مدل به‌دست‌آمده برای ارزیابی نگاشت مورد نظر استفاده شود.

در ادامه در [۱۸] انحصار متقابل در استفاده از مشترک از منابع ارتباطی (مسیرها و لینک‌ها) نیز به مدل اضافه شد. این موضوع که دو یا چند گذر ارتباطی که از یک مسیر مشترک (یا مسیرهایی که لینک مشترک دارند) استفاده کنند نمی‌توانند همزمان اجرا شوند،

باید به نحوی به مدل پتری اضافه می‌شد؛ به همین دلیل گذرهایی به‌منظور ایجاد انحصار متقابل بین آن‌ها به مدل اضافه شد تا مدل به‌دست‌آمده مدل دقیق‌تری برای ارزیابی نگاشت مورد نظر باشد.

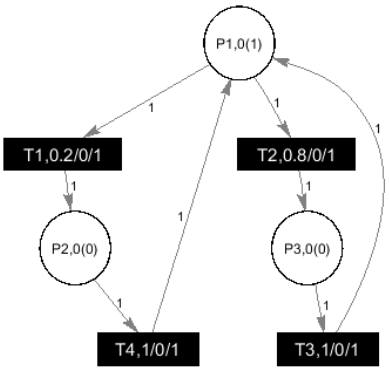
در هر دو روش یادشده ([۸ و ۱۸]) که با هدف ارزیابی انواع نگاشت ارائه شده‌اند، از قابلیت این ابزار برای دستکاری ساختار شبکهٔ پتری و افزودن تعدادی گذر (که به‌ترتیب گذرهای ارتباطی و گذرهای انحصار متقابل‌اند) استفاده کرده‌اند.

۳.۴. تطبیق‌پذیری مدل پتری

یکی از قابلیت‌های ابزار ارائه‌شده تغییر در مدل به‌وسیلهٔ تابع شلیک سفارشی شده است که می‌توان آن را پشتیبانی از شبکه‌های پتری یادگیرنده و یا شبکه‌های پتری تطبیقی محسوب کرد. برای نشان دادن این قابلیت به یک مثال ساده بسنده می‌شود.

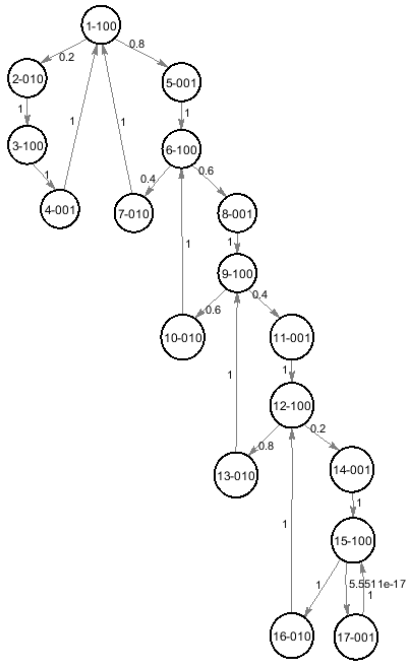
مثال ۵: مدل پتری تطبیق‌پذیر

مدل پتری شکل (۱۳) را در نظر بگیرید. احتمال شلیک T1 و T2 به‌ترتیب 0.2 و 0.8 است. فرض کنید دو تابع شلیک سفارشی‌شده برای گذرهای T1 و T2 نوشته شده به‌طوری که تابع شلیک T3 مقدار 0.2 از وزن احتمال T2 کم کرده و در مقابل 0.2 به وزن احتمال T1 اضافه کند. از طرف دیگر، تابع شلیک T4 مقدار 0.2 از وزن احتمال T1 کم کرده و به T2 اضافه کند. در ابتدا چون احتمال T2 بیشتر است، نرخ شلیک گذرهای شاخهٔ T2 و T3 بیشتر از شاخهٔ T1 و T4 است ولی به‌مرور با ادامهٔ کار با کاهش احتمال T2 توسط T3 نرخ شلیک دو شاخه به تعادل می‌رسد.



شکل (۱۳): مدل پتری مثال ۵

شکل (۱۴) گراف DTMC ی این مدل را نشان می‌دهد. در این DTMC یک حالت تنها شامل نشانه گذاری نیست بلکه نرخ احتمال گذرهای خروجی آن نیز در تعیین حالت دخالت دارند. به عبارت دیگر، حالت‌هایی که نشانه‌گذاری یکسان ولی وزن احتمال گذرهای خارج‌شوندهٔ متفاوت دارند حالات متفاوت در نظر گرفته شده و همهٔ آن‌ها در DTMC رسم می‌شوند. پاسخ حل حالت پایدار این DTMC در شکل (۱۵) آورده شده است.



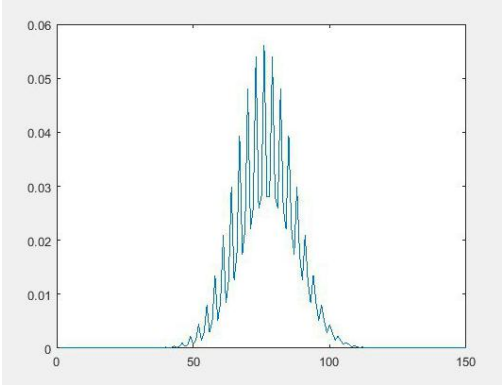
شکل (۱۴): گراف DTMC مدل پتری مثال ۵

Index	Pl Marking	Steady State Prob.
1	100	0.0781
2	010	0.0156
3	100	0.0156
4	001	0.0156
5	001	0.0625
6	100	0.1563
7	010	0.0625
8	001	0.0938
9	100	0.1562
10	010	0.0937
11	001	0.0625
12	100	0.0781
13	010	0.0625
14	001	0.0156
15	100	0.0156
16	010	0.0156
17	001	0.0000

شکل (۱۵): حد حالت پایدار مدل پتری مثال ۵

در این مدل، اگر گام‌های افزایش و کاهش وزن احتمال را به‌جای 0.2 برابر 0.02 در نظر بگیریم، تعداد حالت‌های DTMC افزایش می‌یابد. شکل (۱۶) پاسخ حالت دائمی مدل را در این شرایط به‌صورت نمودار نشان می‌دهد. همان طور که در شکل مشاهده می‌شود، از حدود ۱۵۰ حالت ممکن احتمال حدود ۲۰ حالت که در وسط نمودار قرار گرفته‌اند، سطح احتمال بالاتری

نسبت به بقیه دارند. این حالت‌ها همان حالت‌هایی هستند که در آن‌ها وزن احتمال گذرهای T1 و T2 در آن‌ها نزدیک به 0.5 است.



شکل (۱۶): حل حالت پایدار مدل پتری مثال ۵ با گام‌های ۰.۰۲

۴.۴. جست‌وجوی خودکار راه حل

همان طور که در مقدمهٔ این مقاله شرح داده شد، به‌علت یکپارچه بودن کد مدل با اجزای دیگر ابزار KU_PN_Tool امکان تغییر در ساختار و/یا ویژگی‌های مدل پتری در حین اجرا وجود دارد. این تغییرات می‌تواند به‌طور هدفمند و برای جست‌وجوی مدل بهینه و پیدا کردن راه حل انجام شود. برای نشان دادن این قابلیت مثال۶ را در نظر بگیرید. در مثال ۶ همان مدل پتری مثال۱ با تغییر وزن هر دو کمان وزن‌دار از ۳ به ۵ استفاده شده است. این شبکهٔ پتری را می‌توان مدل توزیع وظایف بین دو پردازنده در نظر گرفت که یک برنامهٔ کاربردی شامل ۵ وظیفه را برای اجرا بین دو پردازندهٔ t3 و t4 توزیع می‌کند. توزیع به‌صورت تصادفی توسط گذرهای t1 و t2 انجام می‌شود. فرض کنید وزن احتمال گذرهای t1 و t2 به‌ترتیب برابر PW و 1-PW باشد. می‌خواهیم PW را طوری تعیین کنیم که توان عملیاتی اجرای برنامه کاربردی ماکزیمم یا به عبارتی زمان اجرای مدل حداقل شود. برای جست‌وجو و پیدا کردن جواب بهینه از الگوریتم زیر استفاده می‌شود. این الگوریتم پس از اجرا PW=0.8 را به‌عنوان بهترین جواب اعلام می‌کند.

۱. مقداردهی اولیه صفر به توان عملیاتی (Trput=0).

۲. مقداردهی اولیه صفر به وزن احتمال (PW=0).

۳. به دست آوردن گراف دسترسی و از روی آن CTMC و سپس پاسخ حالت پایدار مدل پتری جاری.

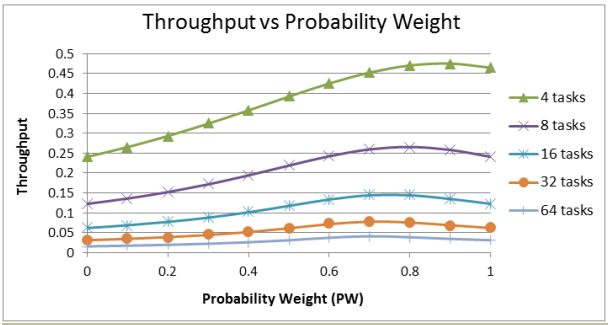
۴. به دست آوردن توان عملیاتی سیستم با ضرب احتمال حالت پایدار وضعیتی که تمام نشانه‌ها در P4 قرار دارند (یا به عبارت دیگر حالت اتمام اجرای کاربرد) در نرخ خروج از آن.

۵. اگر توان عملیاتی نسبت به مرحلهٔ قبل بهبود داشته مقداری کوچکی (مثلاً 0.1) به PW اضافه کن و مراحل ۳ و ۴ را تکرار کن. وگرنه مقدار قبلی PW و توان عملیاتی متناظر آن را به‌عنوان جواب اعلام کن.

یک نمونهٔ عملی از جست‌وجوی خودکار راه حل بهینه که از ابزار پیشنهادی این مقاله استفاده کرده، در [۱۹] آمده است.

۵.۴. بررسی اثر تغییرات در مدل بر روی کارایی

در شکل (۱۷) مقدار توان عملیاتی سیستم برای مقادیر مختلف PW در بازهٔ [0,1] نموداروار رسم شده است. برای رسم این نمودار دو پارامتر از شبکهٔ پتری مثال ۶ تغییر داده شده است. یکی از این دو پارامتر PW و دیگری وزن کمان‌های t0 به p1 و p4 به t5 را مشخص می‌کند. وزن تعیین‌شده برای هر دو کمان در هر آزمایش با هم برابر تعیین شده و اعداد ۲، ۴، ۸، ۱۶، ۳۲ و ۶۴ در نظر گرفته می‌شود. وزن کمان در مثال ۶ به تعداد وظایف کاربرد مورد نظر تعبیر می‌شود. تغییرات با دو حلقهٔ تکرار متداخل (یکی برای وزن کمان‌ها و دیگری برای وزن احتمال توزیع) اعمال و هر بار با تحلیل حالت پایدار توان عملیاتی محاسبه رسم شده است.



شکل (۱۷): نمودار تغییرات توان عملیاتی بر اثر تغییرات وزن احتمال برای تعداد وظایف مختلف

۳. نتیجه

در کار حاضر، ابزاری مبتنی بر شبکهٔ پتری ارائه شد. مجموعهٔ توابع و امکاناتی که برای ایجاد، شبیه‌سازی و تحلیل حالت پایدار فراهم‌شده معرفی گردید. این ابزار که KU_PN_Tool نامگذاری شده، امکان رسم گراف پتری، گراف دسترسی و گراف‌های CTMC و DTMC برای انواع شبکهٔ پتری را فراهم

می‌کند. ابزار یادشده علاوه بر قابلیت‌های گفته‌شده، دارای ویژگی‌های منحصربه‌فردی است. برخی از این ویژگی‌ها با ذکر مثال‌هایی نشان داده شد و برای برخی ویژگی‌های دیگر به کارهایی که قبلاً با استفاده از این ابزار انجام شده، ارجاع داده شد. از جمله ویژگی‌های مهم این ابزار می‌توان به پشتیبانی از تولید خودکار مدل پتری با اندازه‌های بزرگ، پشتیبانی از تغییرات پویا در حین اجرا، پشتیبانی از انواع مختلف شبکهٔ پتری و همچنین متن باز بودن و قابلیت ارتقا اشاره کرد. بخشی از این ویژگی‌ها مرهون جامعیت مدل پتری پیشنهادی و برخی نتیجهٔ یکپارچگی کدهای ایجاد مدل پتری، توابع ابزار و برنامه‌ای است که کاربر برای رسیدن به اهداف تعیین‌شده (مثلاً برای جست‌وجوی راه حل بهینه) می‌نویسد.

سپاسگزاری

نگارنده از معاونت پژوهشی دانشگاه کاشان به‌دلیل پشتیبانی از طرح پژوهشی مرتبط با موضوع این مقاله (به شماره قرارداد ۷۰۷۲۱۳) که انجام آن منجر به نگارش این مقاله شد بسیار سپاسگزاری می‌کند.

مراجع

[1] David R. and Alla H., *Discrete Continuous and Hybrid Petri Nets*, New York, Springer, 2004.

[2] Jensen K., Kristensen L. M. and Wells L., "*Coloured Petri Nets and CPN Tools for modelling and validation of concurrent systems*", International Journal of Software Tools Technology Transfer, Vol. 9, pp. 213–254, 2007.

[3] Gehlot V. and Hayrapetyan A., "*Systems Modeling and Analysis Using Colored Petri Nets: a Tutorial Introduction and Practical Applications*", in Proc. of the 45th Annual Southeast Regional Conference, March 2007, pp. 514.

[4] Hrúz B. and Zhou M. C., *Modeling and Control of Discrete-event Dynamical Systems: with Petri Nets and Other Tools*, 1st Edition, London, Springer, 2007

[5] Haas P. J., "*Stochastic Petri Nets for Modelling and Simulation*", in Proc. of the 36th Conference on Winter Simulation, Washington, DC, USA, December 2004, pp. 101-112.

[6] Su Y., Qiu J., Liu G., Xu Y. and Qian Y., "*Method of Testability Index Determination Based on Generalized Stochastic Petri Net*", 2009 IEEE Circuits and Systems International Conference on Testing and Diagnosis, Chengdu, 2009, pp. 1-4.

[7] Li L., Zeng W., Hong Z., Zhou L., "*Stochastic Petri Net-based Performance Evaluation of Hybrid Traffic for Social Networks System*", Neurocomputing, Vol. 204, pp. 3-7, Sep. 2016.

[8] Sabaghian-Bidgoli H., Shahabi S. A. and Navabi Z., "*A Novel Modeling Approach for System-Level Application Mapping Targeted for Configurable Architecture*", in Canadian Journal of Electrical and Computer Engineering, Vol. 37, No. 4, pp. 192-202, December 2014.

[9] Wu N., Chu F., Chu C. and Zhou M., "*Petri Net Modeling and Cycle-Time Analysis of Dual-Arm Cluster Tools With Wafer Revisiting*", in IEEE Transactions on Systems, Man, and Cybernetics: Systems, Vol. 43, No. 1, pp. 196-207, Jan. 2013.

[10] List G. F. and Cetin M., "*Modeling traffic signal control using Petri nets*", in IEEE Transactions on Intelligent Transportation Systems, Vol. 5, No. 3, pp. 177-187, Sept. 2004.

[11] http://people.ee.duke.edu/~kst/chirel/MANUAL/manual.pdf, Accessed in April 2017.

[12] Sahner R., Trivedi K. S. and Puliafito A., "*Performance And Reliability Analysis Of Computer Systems (an Example-based Approach Using The Sharpe Software*", in IEEE Transactions on Reliability, Vol. 46, No. 3, pp. 441-441, Sept. 1997.

[13] https://sharpe.pratt.duke.edu/, Accessed in April 2017.

[14] Dingle N. J., Knottenbelt W. J. and Suto T., "*PIPE2: A Tool for the Performance Evaluation of Generalised Stochastic Petri Nets*", ACM SIGMETRICS Performance Evaluation Review, Vol. 36(4), pp. 34-39, March 2009.

[15] Thong W. J. and Ameedeen M. A., "*A Survey of Petri Net Tools*", in Advanced Computer and Communication Engineering Technology, Vol. 315, pp. 537-551, Jan. 2015.

[16] https://github.com/hsabaghianb/KU_PN_Tool/blob/master/KU_PN_Tool_v1.72.zip, Uploaded in October 2019.

[17] Bolch G., Greiner S., De-Meer H. and Trivedi K. S., Queueing Networks and Markov Chains, second edition, New Jersey, John Wiley & Sons, 2006.

[۱۸] رئیسی ورزنه، مصطفی، «**مدل‌سازی رقابت بر سر کانال‌های ارتباطی با استفاده از شبکهٔ پتری با هدف ارزیابی دقیق‌تر کارایی نگاشت‌های مختلف یک کاربرد**»، پایان‌نامه کارشناسی ارشد، دانشگاه کاشان، دی‌ماه ۱۳۹۷.

[۱۹] قدیری‌فرد، زهرا، «**ارائهٔ روشی مبتنی بر شبکهٔ پتری برای زمانبندی اجرای وظایف با هدف افزایش کارایی نگاشت کاربرد در سیستم‌های نهفتهٔ چندهسته‌ای**»، پایان‌نامهٔ کارشناسی ارشد، دانشگاه کاشان، شهریور ۱۳۹۸.