



دانشگاه آزاد اسلامی - واحد قزوین

دانشکده مهندسی برق، کامپیوتر و فناوری اطلاعات
گروه کامپیوتر

پایان نامه کارشناسی ارشد مهندسی کامپیوتر
گرایش نرم افزار

یک مدل شبکه پتری تصادفی تطبیقی مبتنی بر اتوماتاهای یادگیر و
کاربردهای آن در تخصیص منابع در شبکه‌های گرید

نگارش: محمد شجاع‌فر

استاد راهنما: دکتر محمدرضا میبدي

تابستان 1389

با تشکر از مادر و پدر مهربانم که در طول تحصیل من همواره مشوق و همراه من بوده اند .

با تشکر از استاد گرانقدرم، جناب آقای دکتر محمدرضا میبیدی، که در طول تحصیل دوره‌ی کارشناسی ارشد به خصوص در انجام این پایان‌نامه، از راهنمایی‌های بی‌دریغ ایشان بهره‌مند بودم.

همچنین از دوستم سیامک برزگر که در ابتدای راه، مرا یاری نمود، سپاسگزاری می‌کنم.

چکیده

شبکه‌های پتری تصادفی وسیله‌ای برای مطالعه سیستم‌ها می‌باشند. تئوری شبکه پتری تصادفی اجازه می‌دهد که یک سیستم بتواند بوسیله آن بصورت یک مدل ریاضی مدل شود. از رفتار پویا و ساختار سیستم مدل شده توسط آنالیز شبکه پتری تصادفی، اطلاعات بسیار مفیدی اتخاذ می‌گردد که این اطلاعات می‌تواند جهت ارزشیابی، حدس‌های برای ایجاد، بهبود یا تغییرات در سیستم استفاده شود. شبکه‌های پتری تصادفی برای آنالیز سیستم‌هایی گسترده کاربرد بسزایی دارند. یکی از مشکلات شبکه پتری تصادفی عدم تطبیق پذیری آن‌ها می‌باشد و بهمین دلیل در شبکه‌های پتری تصادفی امکان دسترسی به اطلاعات قبلی وجود ندارد. اگر در هر زمان بیش از یک گذار فعال باشد، هر کدام می‌توانند به‌عنوان شلیک بعدی محسوب شوند. این ویژگی شبکه پتری حقیقتی را تداعی می‌کند که چنانچه چندین واقعه همزمان اتفاق افتد و وقوع رویدادها یکسان نباشد، هریک از رویدادها می‌تواند رخ دهد و وقوع رویدادها در طول زمان، تغییر نمی‌کند و این برخلاف دنیای واقعی و پویا می‌باشد، و شبیه سازی مشابه اجرای برنامه اصلی است، هدف آنست که از مدل شبیه سازی شده برای بررسی عملکرد سیستم استفاده شود و بدین وسیله مشکلات و نقاط ضعف مدل مشخص می‌گردد ولی ابزار شبکه پتری تصادفی به تنهایی نمی‌تواند در جهت بهبود و رفع مشکلات کاری انجام دهد و وضعیت بهینه بعدی را نمی‌توان پیشگویی نمود. در این پایان‌نامه، هدف ما ایجاد یک شبکه پتری تصادفی تطبیقی مبتنی بر اتوماتای یادگیر و کاربرد آن در تخصیص منابع در گرید های محاسباتی و اقتصادی می باشد. شبکه پتری تصادفی تطبیقی از طریق اطلاعات بدست آمده از حالات قبلی سیستم و واکنش‌های محیط پویا، حالت بهینه بعدی را پیشگویی نموده و وضعیت جاری سیستم را بروز و احتمال وقوع رویدادها را در طول زمان تغییر می‌دهد و باعث می‌شود رویدادها بر اساس احتمال وقوعشان فعال شوند. بروز شدن وضعیت‌های سیستم بر اساس واکنش محیط پویا کمک شایانی در یادگیری و آموزش شبکه‌های پتری می‌کند در اینجا، تطبیقی بودن شبکه های پتری در کاربردهای مختلف مورد بررسی قرار گرفته اند. در این پروژه از ابزار شبیه‌سازی SPNP برای شبیه‌سازی شبکه پتری تصادفی استفاده می‌گردد. در ادامه، کاربرد مدل تطبیقی پیشنهاد شده در قسمت اول، در گرید محاسباتی مورد مطالعه و بررسی قرار می‌گیرد. در این بررسی ایده ای اکتشافی با توجه به الگوریتم های زمانبندی در گریدهای محاسباتی پیشنهاد می‌شود و نتایج آن با روشهای Min.min و Max.min مقایسه شده است. در قسمت دوم تخصیص منابع در گرید اقتصادی با توجه به الگوریتم هوشمند اتوماتای یادگیر در حالت‌های مختلف مورد بررسی قرار می‌گیرد. برای این منظور در گرید اقتصادی از مدل تطبیقی ارائه شده برای تخصیص بهینه منابع در گرید با توجه به معیار زمان استفاده می‌گردد. در اینجا با در نظر گرفتن مستقل بودن کارهای تخصیص یافته به منابع و تخصیص یکباره و یک مرحله ای ایده ای

بر اساس اتوماتای یادگیر ارائه شده و با ایده های قبلی که توسط آقایان بویا و مهدوی فر ارائه شده اند بررسی شده است. الگوریتم پیشنهادی ALATO که براساس اتوماتای یادگیر ارائه شده است نسبت به الگوریتم های مشابه مدت زمان کمتری را صرف جستجو و تخصیص منابع در گرید اقتصادی می کند.

کلمات کلیدی: شبکه پتری تصادفی، تطبیقی بودن، گرید محاسباتی، زمانبندی، اتوماتای یادگیر، بهینه سازی زمان.

Abstract

Stochastic Petri Net (SPN) is a tool by which validation of discrete-event systems are studied and modeled. SPN Theory allows systems to model as mathematical models. A Petri net is a graphical model useful for modeling systems exhibiting concurrent, asynchronous or nondeterministic behavior. The nodes of a Petri net are places (drawn as circles), representing conditions, and transitions (drawn as bars), representing events. Tokens (drawn as small filled circles) are moved from place to place when the transitions, when we Trace Stochastically and randomly on PN models called it Stochastic Petri net (SPN) model. We conclude benefit information from dynamic behaviour and structured of system modeled with stochastic petri net analysis, hence, this information could use for evaluating, some guess for make, modify systems. SPN use extendly for system analysis. one of the problems in SPN are in adaptive them. therefore, SPN are unable to reach prev information .if each transaction enable each time then each of them could selected for next firing. this feathres reminding the facts that if this event accure and the eccuring is not concurrncy each firing can takeplace and event accuring not changend during the time and this is against dynamic and real world, hence, simulation is like main program. The goal is that the simulated model was used for analysis of system perfomance. thus, the problems and lack position of simulated model is appeared. But, SPN tools can not lonely resolve and modify this problems and it can not predict next optimale state. In this thesis, our main goal is making a adaptive stochastic petri net model based on learning automata and its usage in resource allocation in econimic and computing grids.. Adaptive Stochastic Petri nets based on learning automata predicts the following optimized state through getting information from former system states and dynamic environmental reactions, makes the system current states appeared based on that, changes the probability of events occurrences during the time and causes the events to be activated based on probability of occurrence. Updating system state can several major help in learning and educating petri net. Here, petri net adapting in several applications considered. We use SPNP simulation tool for simulating SPN. In another part of this thesis we used prposed method in ASPN in resource allocation in grid computing. We have proposed a new method to optimize the resource Scheduling in grid computing based on categorized requests in three layers which depend on Hierarchical Stochastic Petri net model called HSPN. These layers are Home, Local and Grid layers. We have divided our tasks in these layers. Each layer has special function to receive subtask and deliver data to up/down layer. Then, we have compare HSPN with other resource scheduling such as Min.min and Max.min algorithms. In second part of thesis we considered resource allocation in econimical grid based on learning automata in diffrent states. For this reason, we considerd time measurement for adaptive model for resource allocation in econimical grid. Here, we considered independent allocated tasks to resources as at-onec and phased, an idea based on learning automata and based on related works that Mr Bouyya and mahdavifar compared. ALATO is proposed algorithms based on Learning Automata.

Keywords: Stochastic Petri Net, Adaptivable, Grid Computing, Scheduling, Learning Automata, Time Optimization.

فهرست مطالب

فصل اول.....	2
1-1- مقدمه	2
2-1- ساختار پایان نامه	4
فصل دوم: <u>کارهای پیشین</u>	7
1-2- مقدمه	7
2-2- شبکه های پتری تصادفی.....	7
2-2-1- تکنیکهای تحلیل در شبکه پتری تصادفی	12
2-2-1-1- درخت دسترسی	12
2-2-1-1- محدودیتهای درخت دسترسی	15
2-2-1-2- معادلات ماتریس	15
2-2-2- بهبود کارایی در شبکه پتری تصادفی	16
2-2-2-1- شاخصهای کارایی	17
2-2-2-3- بهبود کارایی در شبکه های پتری تصادفی عمومی	19
2-2-2-1- ساختار شبکه پتری تصادفی عمومی	19
2-2-2-2- توزیع عمومی تعداد دفعات فعال شدن	24
2-2-2-3- روابط با مدل های دیگر	25
2-2-2-4- تجزیه ساختاری شبکه های پتری تصادفی	28
2-2-2-4- خلاصه و نتیجه گیری	29

32-3- اتوماتای یادگیر.....	32
2-3-1-.....مقدمه	32
2-3-2-.....تاریخچه ی اتوماتای یادگیر	33
2-3-3-..... اتوماتای یادگیر تصادفی (SLA)	35
2-3-3-1-..... اتوماتای تصادفی	35
2-3-3-2-.....محیط	36
2-3-4-.....الگوریتم های یادگیری	38
2-3-4-1-.....الگوریتم های یادگیری استاندارد	38
2-3-4-2-.....الگوریتم های یادگیری مدل S	40
2-3-4-1-.....الگوریتم $S-L_{RI}$	41
2-3-4-2-.....الگوریتم $S-L_{RP}$	41
2-3-4-3-.....الگوریتم $S-L_{ReP}$	42
2-3-5-.....الگوریتم های یادگیری با ساختار ثابت	42
2-3-5-1-.....اتوماتای دو حالت $L_{2,2}$	42
2-3-5-2-.....توسعه های اتوماتای دو حالت $L_{2,2}$	44
2-3-5-3-.....اتوماتای حافظه دار با دو عمل L_{2N2}	44
2-3-5-4-.....اتوماتای کرینسکی (K_1)	46

.....-5-5-3-2 اتوماتاي كرايلوف (K_2)	47
.....-6-5-3-2 اتوماتاي $G_{2N,2}$	47
.....-7-5-3-2 اتوماتاي مهاجرت اشياء	48
.....-6-3-2 بازيهاي اتوماتا	48
.....-7-3-2 خلاصه و نتيجه گيري	49
.....-4-2-4-2 گريدهاي محاسباتي	50
.....-1-4-2 مقدمه	50
.....-2-4-2 طبقه بندي سيستم هاي گريد	52
.....-3-4-2 توانمندیهاي گريد محاسباتي	54
.....-1-3-4-2 بهره برداري از منابع بدون استفاده	55
.....-2-3-4-2 موازي سازي پردازنده ها	56
.....-3-3-4-2 برنامه هاي کاربردي	57
.....-4-3-4-2 منابع مجازي و سازمانهاي مجازي براي ايجاد همكاري	58
.....-5-3-4-2 دسترسي به منابع اضافي	58
.....-4-4-2 مديريت منابع در گريد	59
.....-1-4-4-2 تعاريف و نيازمنديها	60
.....-2-4-4-2 مدل انتزاعي سيستم مديريت منبع	60
.....-3-4-4-2 سازماندهي ماشين	62

منابع.....	2-4-4-4	63
زمانبندی.....	2-4-4-5	65
خلاصه و نتیجه‌گیری.....	2-4-5	68
فصل سوم: شبکه پتری تصادفی تطابقی.....70		
1-3- مقدمه.....	70	
2-3- کارهای انجام شده در یادگیری شبکه پتری.....	71	
1-2-3- ویژگی های تطابقی.....	73	
Fusion Hybrid.....	3-2-2	74
Combination Hybrid.....	3-2-3	86
3-3- خلاصه و نتیجه گیری.....	92	
فصل چهارم: کاربرد شبکه پتری تصادفی در گزید محاسباتی.....94		
1-4- مقدمه.....	94	
2-4- مرور اجمالی بر گزید محاسباتی.....	95	
3-4- کارهای مرتبط در زمانبندی گزید محاسباتی به کمک شبکه پتری.....	96	
4-4- ایده پیشنهادی (HSPN).....	97	
5-4- شبیه سازی ایده پیشنهادی.....	102	
6-4- خلاصه و نتیجه گیری و کارهای آینده.....	109	
فصل پنجم: پیاده سازی شبکه پتری تصادفی تطابقی در گزید اقتصادی.....111		
1-5- مروری بر الگوریتم های موجود و الگوریتم پیشنهادی.....	111	
1-1-5- مراحل مشترک الگوریتم ها.....	112	
2-2-5- الگوریتم BTO و الگوریتم ABTO.....	113	
3-1-5- الگوریتم EBTO.....	118	

الگوریتم AEBTO	4-1-5
119	
الگوریتم LATO	5-1-5
122	
الگوریتم مینیمم هزینه محاسبات	6-1-5
125	
الگوریتم پیشنهادی ALATO	7-1-5
125	
2-5- شبیه سازی الگوریتم ها	131
1-2-5- مقدمه	131
2-2-5- مدل شبکه پتری تصادفی تطابق در محیط گرید	132
3-2-5- پیاده سازی الگوریتم ها	138
4-2-5- محیط گرید	138
5-2-5- مدل کاربر	139
6-2-5- مدل برنامه	139
7-2-5- مدل منابع	140
8-2-5- شبیه سازی زمان	145
1-8-2-5- بهینه سازی زمان در یک هزینه و زمان معین (حالت اول)	145
1-1-8-2-5 مدل P-LRP	146
2-1-8-2-5 مدل S-LRP	148
3-1-8-2-5 مدل S-LRI	150
4-1-8-2-5 مدل S-LReP	150
2-8-2-5- بهینه سازی زمان در زمان معین با بودجه های مختلف	152
(حالت دوم)	152
3-8-2-5- بهینه سازی زمان در ناهمگونی های مختلف (حالت سوم)	158
3-5- خلاصه و نتیجه گیری	164

فصل ششم: نتیجه‌گیری و پیشنهادات.....	167
فصل هفتم: پیوست ها.....	171
1-7- واژه نامه ها.....	171
2-7- مروری بر نرم افزار SPNP.....	175
1-2-7- راه اندازی در ویندوز XP.....	176
2-2-7- فایل‌های خروجی در SPNP.....	176
3-2-7- زبان CSPL.....	177
4-2-7- طراحی در نرم افزار SPNP.....	182
3-7- کدها... ..	185
1-3-7- کد الگوریتم اول.....	185
1-1-3-7- قسمت اصلی کد الگوریتم Min.min.....	185
2-1-3-7- قسمت اصلی کد Max.Min و الگوریتم HSPN.....	186
4-7- کد الگوریتم دوم و شبیه سازی آن در SPNP.....	188
1-4-7- کد نزولی کردن کارها و کد تخصیص کارها در میان افزار.....	188
2-4-7- کد الگوریتم های گرید اقتصادی در CSPL.....	191
3-4-7- الگوریتم زمانبندی و تخصیص کارها به منابع مختلف.....	194
مراجع.....	199

فهرست شکل‌ها

- شکل 1-2: مدل شبکه‌های پتری تصادفی برای مثال لامپ.....9
- شکل 2-2: نمایش شبکه‌های پتری تصادفی در صف $M/M/1$ 11
- شکل 3-2: یک شبکه پتری برای درخت دسترسی.....13
- شکل 4-2: مرحله اول درخت دسترسی.....13
- شکل 5-2: مرحله دوم درخت دسترسی.....13
- شکل 6-2: مرحله سوم درخت دسترسی.....14
- شکل 7-2: درخت دسترسی شبکه پتری.....14
- شکل 8-2: نمودار معادله ماتریسی.....16
- شکل 9-2: نمونه شبکه‌های پتری تصادفی عمومی از صف $M/M/1$ 20
- شکل 10-2: نمایش شبکه‌های پتری تصادفی عمومی صف $M/M/2/5/10$ 22
- شکل 11-2: سیستم دو پردازنده ای.....22
- شکل 12-2: شبکه‌های پتری تصادفی عمومی نمایش داده شده در سیستم دو پردازنده ای در شکل 11-2.....24
- شکل 13-2: نمایش مکانهای زمانبندی شده در چارچوب شبکه‌های پتری تصادفی عمومی.....26
- شکل 14-2: نمایش حالت زمانبندی شده سه مرحله ای در چارچوب شبکه‌های پتری تصادفی عمومی.....27
- شکل 15-2: رابطه‌ی بین اتوماتا و محیط احتمالی.....37
- شکل 16-2: نمودار عملکرد اتوماتای دو حالت $L_{2,2}$ 43
- شکل 17-2: گراف انتقال حالت اتوماتای $L_{2,2}$ 43
- شکل 18-2: گراف‌های انتقال حالت اتوماتای L_{2N2} 46
- شکل 19-2: گراف‌های انتقال حالت اتوماتای کرینسکی.....47
- شکل 20-2: انتقال حالات اتوماتای کرایلوف هنگام دریافت پاسخ نامطلوب از محیط.....47
- شکل 21-2: نمودار تغییر وضعیت اتوماتا.....48
- شکل 22-2: نمودار تغییر وضعیت اتوماتای مهاجرت اشیاء.....48
- شکل 23-2: طبقه‌بندی سیستم‌های گرید.....53
- شکل 24-2: بازنمایی منابع ناهمگون و پراکنده به صورت یک سازمان مجازی واحد توسط گرید.....58
- شکل 25-2: زمینه‌ی سیستم مدیریت منابع.....61
- شکل 26-2: ساختار انتزاعی سیستم مدیریت منابع.....62
- شکل 27-2: طبقه‌بندی سازماندهی ماشین‌ها در گرید.....62
- شکل 28-2: طبقه‌بندی مدل منابع.....63
- شکل 29-2: طبقه‌بندی سازماندهی فضای نام.....64
- شکل 30-2: طبقه‌بندی یافتن منبع.....65
- شکل 31-2: طبقه‌بندی انتشار منبع.....65
- شکل 32-2: طبقه‌بندی سازماندهی زمانبند.....66
- شکل 33-2: طبقه‌بندی برآورد وضعیت.....66

شکل 2-34:	طبقه‌بندی زمانبندی مجدد	67
شکل 2-35:	طبقه‌بندی سیاست‌های زمانبندی	68
شکل 3-1:	فرم استاندارد انتقال توکن بین مکانها و انتقالها با قوانین فازی تعریف شده	75
شکل 3-2:	موتور استنتاج فازی لحاظ شده در شبکه پتری توسعه یافته	76
شکل 3-3:	شبکه پتری عصبی-قسمت اول	77
شکل 3-4:	شبکه پتری عصبی-قسمت دوم	78
شکل 3-5:	شبکه پتری فازی عصبی برای نمایش سه قانون AND	80
شکل 3-6:	شبکه پتری فازی عصبی برای نمایش سه قانون OR	80
شکل 3-7:	مدل سه لایه‌های شبکه پتری فازی عصبی برای استنتاج فازی	81
شکل 3-8:	یادگیری در شبکه پتری در چهار برش ورودی و خروجی شبکه عصبی	82
شکل 3-9:	چگونگی فعالشدن و روشنشدن یک گذار در شبکه پتری یادگیر در هر برش زمانی	83
شکل 3-10:	شبکه پتری عصبی دو لایه	83
شکل 3-11:	مدل شبکه پتری خبره فازی	84
شکل 3-12:	یک مدل شبکه پتری فازی عصبی برای کنترل سطوح ناهمواری در فرزکاری دستگاه CNC	87
شکل 3-13:	نمودار شبکه پتری تصادفی مدل FMS	88
شکل 3-14:	ساختار کلی ایده پیشنهادی آقای فاکودا	90
شکل 3-15:	مدل وظیفه ایده پیشنهادی آقای فاکودا	90
شکل 3-16:	مدل وظیفه برای یک وعده غذایی	91
شکل 4-1:	مدل سه لایه ای زمانبندی کارها یا درخواست ها	98
شکل 4-2:	مدل ایده پیشنهادی HSPN	101
شکل 4-3:	زمان متوسط پاسخگویی برای هر درخواست بازای 100 درخواست	106
شکل 4-4:	متوسط زمان تخصیص داده شده برای هر کار	107
شکل 4-5:	هزینه متوسط پاسخگویی برای هر درخواست بازای 100 درخواست	108
شکل 4-6:	متوسط هزینه تخصیص داده شده برای هر کار	108
شکل 5-1:	الگوریتم BTO	115
شکل 5-2:	کد برنامه الگوریتم BTO	116
شکل 5-3:	کد قسمت دوم در ABTO	118
شکل 5-4:	الگوریتم EBTO	119
شکل 5-5:	الگوریتم AEBTO	121
شکل 5-6:	الگوریتم LATO	124
شکل 5-7:	الگوریتم ALATO	130
شکل 5-8:	سه لایه شبکه گرید در شبکه پتری تصادفی	133
شکل 5-9:	تخصیص کارها در لایه میان افزار	134
شکل 5-10:	نزولی کردن کارها	135
شکل 5-11:	انتساب کارها به منابع و اعمال پاداش و جریمه و اجرا به تعداد 10000 بار	137
شکل 5-12:	الگوریتم زمانبندی	137
شکل 5-13:	کد تعریف کارها	140
شکل 5-14:	کد و منبع GR1 کد منبع GR2	143
شکل 5-15:	توانمندی منابع در پیکربندی GR1	144

144.....	شکل 5-16: قیمت مفید منابع در پیکربندی GR1
147.....	شکل 5-17: روند یادگیری الگوریتم‌های LATO و ALATO
148.....	شکل 5-18: مقایسه زمان اجرا الگوریتم‌ها در اتوماتای استاندارد با بودجه ثابت 150000
149.....	شکل 5-19: مقایسه زمان اجرا الگوریتم‌ها در اتوماتای S-LRP با بودجه ثابت 150000
152.....	شکل 5-20: مقایسه زمان اجرا اتوماتاها با بودجه ثابت 150000 در بهترین حالت هر اتوماتا
155.....	شکل 5-21: نتایج الگوریتم‌های بهینه‌سازی زمان با بودجه‌های مختلف
156.....	شکل 5-22: مقایسه‌ی الگوریتم‌های ALATO و LATO و Minimum با بودجه‌های مختلف
158.....	شکل 5-23: روند تغییرات زمان اجرای کارها در الگوریتم ALATO با بودجه‌های مختلف
161.....	شکل 5-24: نتایج الگوریتم‌های بهینه‌سازی زمان در ناهمگونی‌های مختلف
162.....	شکل 5-25: مقایسه‌ی الگوریتم‌های LATO و ALATO و Minimum در ناهمگونی‌های مختلف
163.....	شکل 5-26: اختلاف زمان اجرای کارها در الگوریتم‌های LATO و ALATO با الگوریتم Minimum
164.....	شکل 5-27: نتایج الگوریتم‌های بهینه‌سازی زمان برای کارهای همگون و ناهمگون
182.....	شکل 7-1: قاب اصلی
183.....	شکل 7-2: پروژه‌های جدید
184.....	شکل 7-3
184.....	شکل 7-4

فهرست جدول‌ها

جدول 1-3: جدول تغییرات فازی درمقادیر نشانه‌گذاری گذرهای شبکه پتری توسعه یافته	75
جدول 2-3: وضعیت مکان‌ها و گذر‌ها	91
جدول 1-4: نشانه‌های و توضیحات آن‌ها در HSPN	99
جدول 2-4: دسته‌بندی 100 کار به 4 گروه 25 تایی و شرایط لحاظ شده برای هر کار	103
جدول 1-5: علائم اختصاری در نظر گرفته شده برای ایده پیشنهادی	128
جدول 2-5: پیکربندی GR1 برای منابع	141
جدول 3-5: پیکربندی GR2 برای منابع	142
جدول 4-5: شرایط آزمایش بهینه‌سازی زمان با بودجه ثابت	146
جدول 5-5: تغییرات میزان پاداش و جریمه در مدل P-LRP و در الگوریتم‌های بهینه‌سازی زمان با بودجه ثابت 150000	146
جدول 6-5: تغییرات میزان پاداش و جریمه در مدل S-LRP و الگوریتم‌های بهینه‌سازی زمان با بودجه ثابت 150000	149
جدول 7-5: تغییرات میزان پاداش و جریمه در مدل S-LRI و الگوریتم‌های بهینه‌سازی زمان با بودجه ثابت 150000	150
جدول 8-5: تغییرات میزان پاداش و جریمه در مدل S-LReP و الگوریتم‌های بهینه‌سازی زمان با بودجه ثابت 150000	150
جدول 9-5: شرایط آزمایش بهینه‌سازی زمان با بودجه متغیر	152
جدول 10-5: زمان اجرای برنامه در الگوریتم‌های بهینه‌سازی زمان با بودجه‌های مختلف	153
جدول 11-5: شرایط آزمایش بهینه‌سازی زمان در ناهمگونی‌های مختلف و بودجه 150000	159
جدول 12-5: زمان اجرای برنامه در الگوریتم‌های بهینه‌سازی زمان با ناهمگونی‌های مختلف	160
جدول 1-7: گزینه‌های موجود برای فایل‌های میانی	178
جدول 2-7: راه حل تحلیلی و عددی option های موجود	178
جدول 3-7: Option های شبیه‌ساز موجود	178
جدول 4-7: Option های متفرقه	179

فصل اول

مقدمه

1 فصل اول

مقدمه

شبکه های پتري تصادفی وسیله اي براي مطالعه سیستم ها می باشند. تئوري شبکه پتري تصادفی اجازه می دهد که يك سیستم بتواند بوسیله آن بصورت یک مدل ریاضی مدل شود. از رفتار پویا و ساختار سیستم مدل شده توسط آنالیز شبکه پتري تصادفی، اطلاعات بسیار مفیدی اتخاذ می گردد که این اطلاعات می تواند جهت ارزشیابی، حدسهای برای ایجاد، بهبود یا تغییرات در سیستم استفاده شود. شبکه های پتری تصادفی برای آنالیز سیستم هایی گسترده کاربرد بسزایی دارند. از جمله این سیستم ها می توان به نرم افزارهای توزیع شده، پروتکل ارتباطی، سیستم های تولید، سیستم های پایگاهی، سیستم های فهرست و آمارهای منطقی، سیستم های شی گرا، سیستم های زمان واقعی و شبکه های محاسباتی گرید اشاره کرد. یکی از مشکلات شبکه پتري تصادفی عدم تطبیق پذیری آنها می باشد و بهمین دلیل در شبکه های پتري تصادفی امکان دسترسی به اطلاعات قبلی وجود ندارد. در شبکه پتری تصادفی اگر در هر زمان بیش از يك گذار فعال باشد، هر کدام می توانند به عنوان شلیک بعدی محسوب شوند. این ویژگی شبکه پتري حقیقتی را تداعی می کند که چنانچه چندین واقعه همزمان اتفاق افتد و وقوع رویدادها یکسان نباشد، هر يك از رویدادها می تواند رخ دهد و وقوع رویدادها در طول زمان، تغییر نمی کند و این برخلاف دنیای واقعی و پویا می باشد، و شبیه سازی مشابه اجرای برنامه اصلی است، هدف آنست که از مدل شبیه سازی شده برای بررسی عملکرد سیستم استفاده شود و بدین- وسیله مشکلات و نقاط ضعف مدل مشخص می گردد ولی ابزار شبکه پتري تصادفی به تنهایی نمی تواند در جهت بهبود و رفع مشکلات کاری انجام دهد و وضعیت بهینه بعدی را نمی توان پیشگویی نمود.

گرید محاسباتی یک زیربنای سخت افزاری و نرم افزاری می باشد که دسترسی قابل اعتماد، پایدار، فراگیر و ارزان را به توانایی های محاسباتی دیگران فراهم می کند. یک گرید محاسباتی با مجموعه ای از منابع در مقیاس وسیع در ارتباط است. چنین مجموعه ای به یک زیربنای سخت افزاری برای دستیابی به ارتباطات لازم و یک زیربنای نرم افزاری برای بررسی و کنترل نتایج بدست آمده، نیاز دارد. منابع در گرید، متعلق به سازمان های مختلف می باشد که دارای

سیاست‌های مدیریتی و مدل‌های استفاده و هزینه‌ای مخصوص به خود برای کاربران مختلف در زمان‌های گوناگون می‌باشند. همچنین فراهم بودن منابع و میزان بار بر روی آنها در طول زمان تغییر می‌کند.

صاحبان و کاربران منابع دارای اهداف، استراتژی‌ها و الگوهای عرضه و تقاضای متفاوتی می‌باشند. در مدیریت چنین محیط پیچیده‌ای، نمی‌توان از رویکردهای رایج برای مدیریت منابع که سعی می‌کنند میزان کارایی را در سطح سیستم بهینه کنند، استفاده کرد. رویکردهای رایج از سیاست‌های متمرکز استفاده می‌کنند که نیازمند اطلاعات کاملی از وضعیت‌ها و یک سیاست مدیریت زیربنایی مشترک یا یک سیاست غیرمتمرکز توافقی می‌باشند. به دلیل پیچیدگی‌های موجود در ایجاد محیط‌های گرید موفق، نمی‌توان یک ماتریس کارایی قابل قبول در سطح سیستم و همچنین یک سیاست مدیریتی مشترک را تعریف کرد.

رویکرد اقتصادی، یک مبنای عادلانه برای مدیریت موفق ناهمگونی و غیرمتمرکز بودن که در مسائل اقتصادی بشری وجود دارد، فراهم کرده است. مدل‌های اقتصادی رقابتی، الگوریتم‌ها، سیاست‌ها و ابزارهایی را برای اشتراک یا تخصیص منابع در سیستم‌های گرید فراهم می‌کنند. این مدل‌ها می‌توانند بر مبنای مبادله‌ی پایا پای و یا قیمت گذاری باشند. در مدل مبتنی بر مبادله‌ی پایا پای، همه‌ی شرکت‌کننده‌ها باید دارای منابعی باشند و عمل دادوستد منابع را با تبادل آنها انجام دهند (مثلاً مبادله‌ی زمان پردازنده با فضای ذخیره‌سازی). در مدل مبتنی بر قیمت، منابع دارای قیمتی هستند که بر اساس عرضه، تقاضا و ارزش در سیستم اقتصادی تعیین می‌گردد.

واسطه‌گر منبع گرید که وظیفه‌ی زمانبندی برنامه‌ی کاربردی متعلق به کاربر را بر عهده دارد، باید دارای یک الگوریتم زمانبندی کارا باشد. این الگوریتم می‌تواند با یک استراتژی عمده را برای زمانبندی اتخاذ کند. این استراتژی که توسط بویا پیشنهاد شده است، عبارتست از:

- مینیمم کردن زمان با رعایت محدودیت بودجه (بهینه‌سازی زمان)

در این پایان‌نامه، الگوریتم جدیدی برای استراتژی فوق در مدل اقتصادی بازار کالا، پیشنهاد می‌گردد. الگوریتم پیشنهادی ALATO بر اساس اتوماتاهای یادگیر برای بهینه‌سازی

زمانبندی می‌باشد که نسبت به الگوریتم‌های گزارش شده، از کارایی بهتری برخوردارست و معایب آنها را برطرف می‌کنند. این الگوریتم در زمانی که ناهمگونی کارها زیاد است، از نظر کارایی نسبت به الگوریتم‌های مکاشفه‌ای بهبود قابل ملاحظه‌ای دارند. الگوریتم پیشنهادی و همچنین الگوریتم‌های گزارش شده، با استفاده از جعبه ابزار SPNP شبیه‌سازی شده و کارایی‌های آنها در شرایط گوناگون مورد بررسی قرار می‌گیرد.

ساختار پایان نامه

در فصل دوم کارهای پیشین که به سه دسته شبکه‌های پتری تصادفی، اتوماتای یادگیر و گرید محاسباتی تقسیم می‌شوند ارائه شده است. در بخش اول تعریف جامع و کاملی از شبکه‌های پتری تصادفی و شبکه‌های پتری تصادفی عمومی با ذکر چند مثال بیان شده است. در بخش دوم مفهوم اتوماتای یادگیر و انواع آن معرفی می‌گردد. از اتوماتای یادگیر در طراحی الگوریتم‌های کارا برای زمانبندی اقتصادی در این پایان‌نامه استفاده می‌شود. در بخش سوم و انتهایی این فصل انواع طبقه‌بندی مفاهیم مرتبط با سیستم‌های گرید و همچنین تعاریف و مفاهیم مورد نیاز، آورده شده است. همچنین مدیریت منابع و زمانبندی کارها در گرید مورد بررسی قرار می‌گیرد.

در فصل سوم یک شبکه پتری تصادفی تطابقی با ارائه دو مفهوم Fusion Hybrid و Combination Hybrid و طبقه‌بندی کارهای اجرا شده در این دو گروه ارائه شده است.

در فصل چهارم یک شبکه پتری تصادفی به صورت سلسله مراتبی برای زمانبندی در تخصیص منابع در گرید های محاسباتی ارائه شده است و نتایج آن با دو الگوریتم پرکاربرد Min.min و Max.min در پارامترهای زمان و هزینه مورد بررسی قرار گرفته است.

در فصل پنجم شبکه پتری تصادفی تطابقی که بر اساس اتوماتای یادگیر ارائه شده در رویکرد اقتصادی گرید جهت بهینه کردن زمان در تخصیص منابع مورد بررسی قرار گرفته است. نتایج بر اساس انواع اتوماتاها و ناهمگونی درخواست‌ها بررسی شده است و نتایج آنها در قالب نمودارهای مختلف آورده شده و توضیحات لازم ذکر می‌گردد.

فصل ششم شامل نتیجه‌گیری و پیشنهاد برای کارهای آینده می‌باشد.

فصل هفتم، پیوست پایان‌نامه را در بر می‌گیرد. در این پیوست، واژه‌نامه‌ی فارسی به انگلیسی برای معادله‌های به کار رفته در این پایان‌نامه، راهنمای اجمالی از نرم افزار SPNP استفاده شده برای شبیه سازی الگوریتم ها و همچنین کدهای استفاده شده در روش‌های پیشنهادی آورده شده است.

هر فصل شامل یک بخش مقدمه در ابتدا و یک بخش خلاصه و نتیجه‌گیری در انتهای فصل می‌باشد.

فصل دوم

کارهای پیشین

2 کارهای پیشین

مقدمه

در این فصل، کارهای پیشین که به سه دسته شبکه های پتری تصادفی، اتوماتای یادگیر و گرید محاسباتی تقسیم می شوند ارائه شده است. در بخش اول تعریف جامع و کاملی از شبکه های پتری تصادفی و شبکه های پتری تصادفی عمومی با ذکر چند مثال بیان شده است. در بخش دوم مفهوم اتوماتای یادگیر و انواع آن معرفی می گردد. از اتوماتای یادگیر در طراحی الگوریتم های کارا برای زمانبندی اقتصادی در این پایان نامه استفاده می شود. در بخش سوم و انتهای این فصل انواع طبقه بندی مفاهیم مرتبط با سیستم های گرید و همچنین تعاریف و مفاهیم مورد نیاز، آورده شده است. همچنین مدیریت منابع و زمانبندی کارها در گرید مورد بررسی قرار می گیرد.

شبکه های پتری تصادفی

شبکه های پتری تصادفی توسط مولی¹ و ناتکین² [1-3] به عنوان یک مدل توسعه یافته از شبکه های پتری معرفی شده است. شبکه های پتری دارای تأخیرات در فعال شدن تصادفی با همکاری حالتها یا گذارها³ می باشد که عمل فعال شدن⁴ هر یک از این حالات یک عمل اتمیک و یکتا می باشند که «شبکه های پتری تصادفی»⁵ نامیده می شوند [4].

تعریف: شبکه های پتری تصادفی از شش بخش زیر تشکیل یافته است که بصورت زیر نشان داده می شود:

(P, T, I, O, M₀, Λ)

رابطه (1-2)

که عبارت است از شبکه های پتری بدون زمان دارای علامت که زیر بنای شبکه های پتری تصادفی می باشد. با توجه به مدل بالا شبکه های پتری تصادفی طبق معمول تشکیل شده از:

- مجموعه از مکانها $P=(p_1, p_2, \dots, p_m)$
- یک مجموعه از انتقالها $T=(t_1, t_2, \dots, t_n)$
- مجموعه ای از ورودیها $I \subset P \times T$
- مجموعه ای از خروجیها $O \subset T \times P$
- یک نشان گذار⁶ اولیه $M_0=(m_{01}, m_{02}, \dots, m_{0m})$
- $\Lambda=(\lambda_1, \lambda_2, \dots, \lambda_n)$ یک رشته از (احتمالاً وابسته به نشان گذاری ها)

¹ Molloy

² Natkin

³ Transitions

⁴ Fire

⁵ Stochastic Petri Net (SPN)

⁶ Marking

سرعت‌های فعال شدن می‌باشد که با انتقال‌ها مرتبط است. فعال شدن یک انتقال، یک عملیات اتمیک می‌باشد که علائم از مکان‌های ورودی خود جدا شده و در مکان‌های خروجی خود توسط عملیات یگانه و واحد، قرار می‌گیرند (که این امر با مدل‌های شبکه‌های پتری را زما ندارد که بعدها به آن اشاره می‌کنیم و در آن عملیات فعال شدن که به سه مرحله جداگانه تقسیم می‌شود، در تضاد می‌باشد).

یک تأخیر در فعال شدن با هر یک از انتقال‌ها همراه است که مقدار زمانی که باید قبل از فعال شدن انتقال گذرانده شود را تعیین می‌کند. این تأخیر فعال شدن یک متغیر تصادفی و دارای تابع شدت احتمال نمایی منفی است. پارامتر تابع شدت احتمال⁷ که به انتقال t_i مرتبط است سرعت فعال شدن می‌باشد که با λ_i و t_i ارتباط دارد. این سرعت فعال شدن ممکن است علامت‌دار باشد در حقیقت باید به صورت $\lambda_i(M_j)$ نوشته شود. تأخیر فعال شدن متوسط انتقال t_i ، در علامت‌گذاری M_j عبارت است از $[\lambda_i(M_j)]^{-1}$.

دو نوع توصیف از یک مدل شبکه‌های پتری تصادفی امکان پذیر می‌باشد. اولین توصیف فرض می‌کند که هرگاه یک علامت‌گذاری جدید وارد شود هر کدام انتقال را قادر می‌سازند تا یک نمونه از تأخیر فعال شدن تصادفی را از تابع شدت احتمال مربوطه، نشان دهند. انتقالی که حداقل تأخیر فعال شدن را نشان دهد انتقالی خواهد بود که فعال شدن آن تغییر در علامت‌گذاری را مشخص می‌کند، بنابراین زمان توقف در علامت‌گذاری با حداقل ارزش نمونه نشانده برابر می‌باشد. علامت‌گذاری جدید توسط قوانین زیربنایی شبکه‌های پتری غیر زمانی به دست می‌آید و آنگاه فرآیند دوباره آغاز می‌شود.

به این علت حداقل دو متغیر تصادفی با تابع شدت احتمال نمایی منفی و دارای پارامترهای μ_1 و μ_2 بعنوان متغیرهای تصادفی می‌باشند که به صورت نمایی توزیع شده و پارامترهای $(\mu_1 + \mu_2)$ را دارد، زمان توقف در علامت‌گذاری M_j یک متغیر تصادفی با تابع شدت احتمال نمایی منفی است، در واقع این امر بدین معنی می‌باشد:

$$\left[\sum_{it_i \in E(M_j)} \lambda_i(M_j) \right]^{-1} \quad \text{رابطه (2-2)}$$

که در آن $E(M_j)$ ، مجموعه‌ای از تمام انتقال‌های فعال شده در M_j می‌باشد.

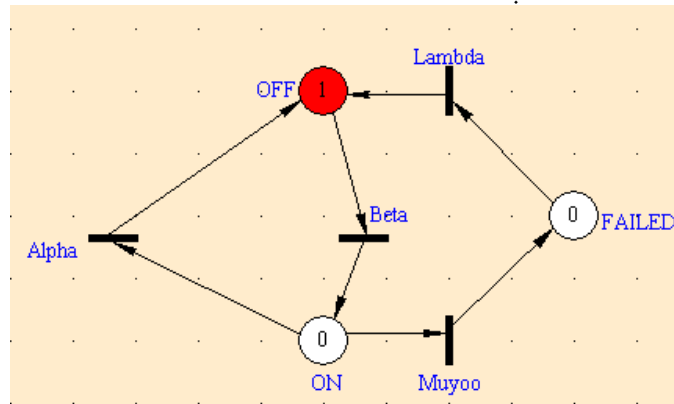
این واقعیت که تمامی تأخیرهای فعال شدن دارای تابع شدت احتمال نمایی هستند به ما امکان نوشتن عبارتی ساده برای تعیین تغییر علامت‌گذاری توسط فعال شدن را می‌دهد.

⁷ probability density function (pdf)

$$P\{t_k | M_j\} = \frac{\lambda_k(M_j)}{\sum_{i: t_i \in E(M_j)} \lambda_i(M_j)} \quad , \quad t_k \in E(M_j) \quad \text{رابطه (2-3)}$$

ویژگی بدون حافظه تابع شدت احتمال نمایی منفی، امکان توصیف مفید و متفاوتتری از مدل‌های شبکه‌های پتری تصادفی را می‌دهد. هرگاه یک تغییر در علامتگذاری، انتقال‌هایی را فعال کند که قبلاً از آخرین فعال شدن خود فعال نبوده، این نمونه‌های انتقال، یک مدل از تأخیر فعال شدن از تابع شدت احتمال نمایی منفی مربوطه را نشان می‌دهد و یک زمان سنج را برای ارزش نمونه مدل‌های تأخیری تنظیم می‌کند. در زمان فعالسازی انتقال، زمان سنج⁸ با سرعتی ثابت کاهش می‌یابد. اگر فعال شدن یک انتقال مخالف، آن را غیر فعال کند زمان سنج متوقف شده و در حین فعالسازی مجدد انتقال میزان کاهش احتمالاً با سرعتی متفاوت از سر گرفته می‌شود. هنگامی که زمان سنج به صفر نزدیک می‌شود، انتقال فعال می‌گردد.

مفید بودن توصیف دوم از این واقعیت حاصل می‌شود که ممکن است فعالیتها را به انتقالها مرتبط سازیم. این فعالیتها می‌توانند در طی پویایی‌های مدل آغاز شده و در نهایت متوقف گردند و وقتی تکمیل شدند، تغییر حالتی را باعث می‌شوند. این ارتباط در توصیف قبلی وجود نداشت چون نمونه برداری مجدد بدین معنا خواهد بود که فعالیتها در هر علامتگذاری مجدداً آغاز گشته و در هر آغازشدنی، مدت زمان متفاوت داشته باشد.



شکل 2-1: مدل شبکه‌های پتری تصادفی برای مثال لامپ

به علت ویژگی بدون حافظه تابع شدت احتمال نمایی، توصیفات دیگر را نیز می‌توان معادل دومی دانست. هنگامی که شبکه‌های پتری تصادفی را با شبکه‌های توزیع شده عمومی نام می‌بریم، این معادل دیگر درست نخواهد بود. باید خاطرنشان سازیم که همراه سازی یک تأخیر در فعال شدن تصادفی با تابع شدت احتمال پیوسته در هر

⁸ Timer

انتقال منجر به احتمالی با مقدار صفر⁹ برای دو انتقال در حال برنامه ریزی به منظور فعال شدن در همان لحظه می گردد. بنابراین این، یک مدل شبکه های پتری تصادفی توسط انتقال های فعال شده به ترتیب ظاهر می شود.

مجموعه فایل دسترس یک شبکه های پتری تصادفی، به علت حمایت غیر زمانی تابع شدت احتمال در تأخیر فعال شدن، مشابه یکی از شبکه های پتری غیر زمانی زیر می باشد. در واقع این واقعیت که نمونه های تأخیر فعال شدن از مجموعه اعداد واقعی غیر منفی نشان داده می شود این امر را ضمانت می کند که در هر علامتگذاری قابل دسترس، تمامی انتقال های فعال، احتمالی غیر صفر از فعال شدن را خواهند داشت. هویت مجموعه قابل دسترس این امر را می رساند که ویژگی های ساختاری بدست آمده برای شبکه های پتری برای شبکه های پتری تصادفی نیز معتبر هستند.

نمودار سرعت انتقال حالت زنجیره مارکو پیوسته زمانی¹⁰ که مطابق با شبکه های پتری تصادفی می باشد با ساخت گراف قابل دسترس و همچنین با نامگذاری کمان ها با سرعت فعال سازی انتقال هایی که فعال شدن آنها تغییرات علامتگذار را به وجود می آورد، بدست می آید. بنابراین، به دست آوردن ژنراتور بسیار کوچک زنجیره مارکو پیوسته زمانی، دارای پیچیدگی یکسانی هم چون ایجاد مجموعه قابل دسترسی شبکه های پتری را دارا می باشد. راه حل حالت ثابت این مدل، با حل سیستم معادلات خطی بدست می آید.

رابطه (2-4)

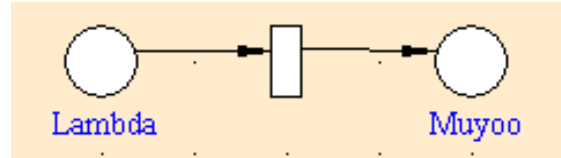
$$\pi Q = 0 \quad \sum_i \pi_i = 1$$

دقت کنید که π_i ، بر احتمال حالت ثابت علامتگذاری M_i دلالت دارد (و نیز در حالت i ، یک تطابق یک به یک بین حالات و علامتگذاری ها وجود دارد) و π ، تابع شدت احتمال تعادل در علامتگذاری های قابل دسترس می باشد.

مدل شبکه های پتری تصادفی که سیستم در نظر گرفته شده در شکل 2-1 به نمایش درآمده است. خواننده، لازم است مدل شبکه های پتری تصادفی و نمودارهای سرعت انتقال حالت زنجیره مارکو پیوسته زمانی را با هم مقایسه کند و به شباهت در مکان شناسی دقت کند. علت آن به خاطر این واقعیت است که شبکه های پتری زیربنای مدل شبکه های پتری تصادفی و بصورت یک گراف علامتدار می باشند [5]. در بسیاری از حالات این شباهت در مکان شناسی وجود ندارد. به عنوان مثال دوم، شبکه های پتری تصادفی شکل 2-2 را در نظر بگیرید. این نمایش شبکه های پتری تصادفی از صف $M/M/1$ می باشد.

⁹ Null

¹⁰ Continues-Time Markov Chain



شکل 2-2: نمایش شبکه‌های پتری تصادفی در صف M/M /1

در این مدل‌ها توصیف در زمینه‌هایی همچون بهبود کارایی¹¹ و همچنین توصیف آنکه تأخیرات تصادفی با توابع احتمالی گسترش یافته توانی منفی بکار می‌روند و نقاط ضعف و قوت آن‌ها چه هستند بحث شده است. تحقیقات زیادی روی این شبکه‌ها صورت گرفته است. همچنین امروزه توصیفات زیادی بر نتایج بدست آمده وجود دارد. در واقع، تعیین شده است که چه نتایجی به‌عنوان نتایج نامعتبر در نظر گرفته می‌شوند و چه نتایجی بیشترین تأثیر را در استفاده از این تکنیک مدلسازی در رشته‌ی کاربردی دارند. شبکه‌های پتری، در شکل‌ها و اندازه‌های مختلف به منظور مطالعه ویژگی‌های کیفیتی سیستم‌هایی به‌کار می‌روند که دارای ویژگی‌های همروندی¹² و همزمانی¹³ می‌باشند.

استفاده از تکنیک‌های مبتنی بر شبکه‌های پتری به منظور تحلیل‌های کیفی سیستم‌ها نیاز به معرفی ویژگی‌های موقتی¹⁴ مدل‌های غیر زمان و بنیادی دارند. این موقعیت از زمان‌های بسیار دور شناخته شده و پیشنهاد‌های بسیار متفاوتی برای معرفی ویژگی‌های موقتی شبکه‌های پتری در مطالعات ظاهر شده است.

پیشنهاد‌های اصلی وجود دارد که ویژگی‌های پیشنهادات مختلف را نشان می‌دهد که عبارتند از:

- عناصر شبکه‌های پتری که زمان به آن‌ها مربوط می‌شود (مکان‌ها یا تغییر حالات)
- مفهوم فعال شدن در مورد حالت زمانی¹⁵ (فعال شدن اتمیک یا فعال شدن در سه مرحله)
- ویژگی‌های موقتی معمولی (قطعی¹⁶ یا احتمالی)

در این فصل، مدل‌های شبکه‌های پتری را در نظر می‌گیریم که با یک ویژگی موقتی همراه‌اند. در این ویژگی، فعال شدن تأخیر در تغییر حالات همراه می‌شود. فعال شدن تغییر حالت اتمیک می‌باشد. علائم و نشانه‌ها¹⁷ از مکان‌های ورودی خارج می‌شود و در

¹¹ Performance Evaluation

¹² Concurrency

¹³ Synchronization

¹⁴ Temporal

¹⁵ Time Transition

¹⁶ Deterministic

¹⁷ Tokens

مکان‌های خروجی به همراه یک عملیات و محدود قرار گیرند. ویژگی تأخیر فعال شدن طبیعتی احتمالی دارد به‌طوری که تابع شدت احتمال یا تابع توزیع احتمال¹⁸ که تابع تأخیری می‌باشند و همراه با یک تغییر حالت هستند باید مشخص شوند.

در ساده‌ترین حالت فرض می‌کنیم که تمام تأخیرات دارای تابع شدت احتمال نمایی منفی هستند اما به‌صورت مختصر حالت تابع شدت احتمال عمومی را در نظر می‌گیریم. کلاس مدلهایی که در نظر می‌گیریم اغلب با نام شبکه‌های پتری تصادفی نامیده می‌شوند. هدف این قسمت، به بحث گذاشتن چندین نکته‌ی مربوط به شبکه‌های پتری تصادفی می‌باشند که عبارتند از:

- علت معرفی شبکه‌های پتری تصادفی
- علت استفاده تابع شدت احتمال نمایی برای مشخص کردن زمان
- نکات قوت و ضعف شبکه‌های پتری تصادفی
- تلاش‌های تحقیقاتی موجود
- نتایج مهم شکست در زمینه شبکه‌های پتری تصادفی

این بحث متخصصین شبکه‌های پتری را مورد خطاب می‌دهد که با زمینه‌ی مدل‌سازی عملکرد تصادفی آشنایی ندارند به این دلیل یک مرور مختصر روش کلاسیک ارزیابی عملکرد سیستم‌ها در یک چارچوب احتمالی در قسمت بعدی آورده شده است که در آن برخی از ایده‌های ابتلایی در مورد فرایندهای تصادفی و تئوری صف¹⁹ خلاصه شده‌اند.

2-2-1- تکنیک‌های تحلیل در شبکه پتری تصادفی

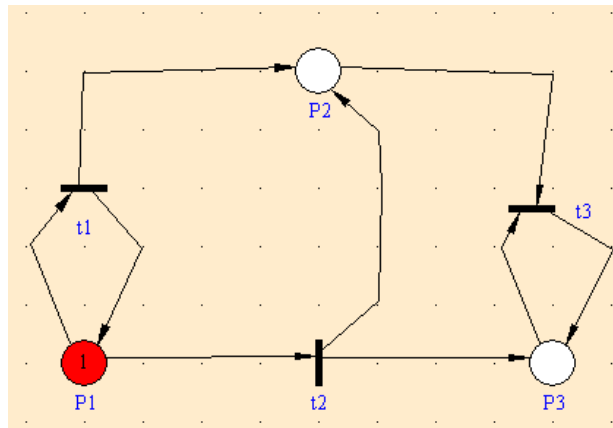
بهترین تکنیک تحلیل شبکه پتری درخت دسترسی و تکنیک دیگر معاملات ماتریسی است.

2-2-1-1- درخت دسترسی

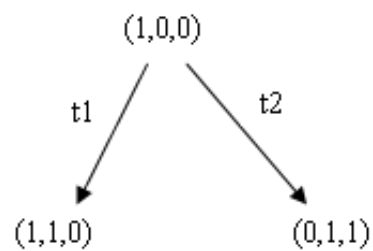
درخت دسترسی، سری دسترسی یک شبکه پتری ارائه می‌کند که جهت مطالعه شبکه بسیار مهم است.

¹⁸ Probability Distribution Function(PDF)

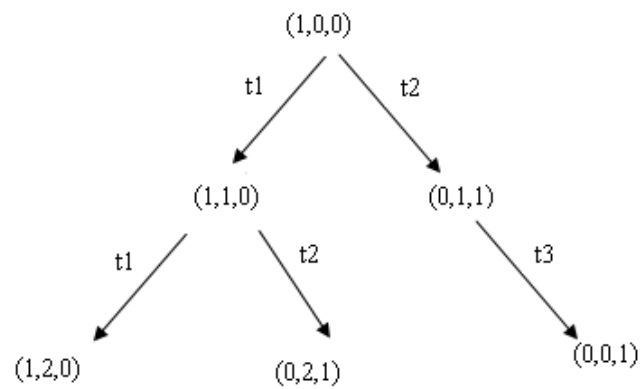
¹⁹ Queuing Theory



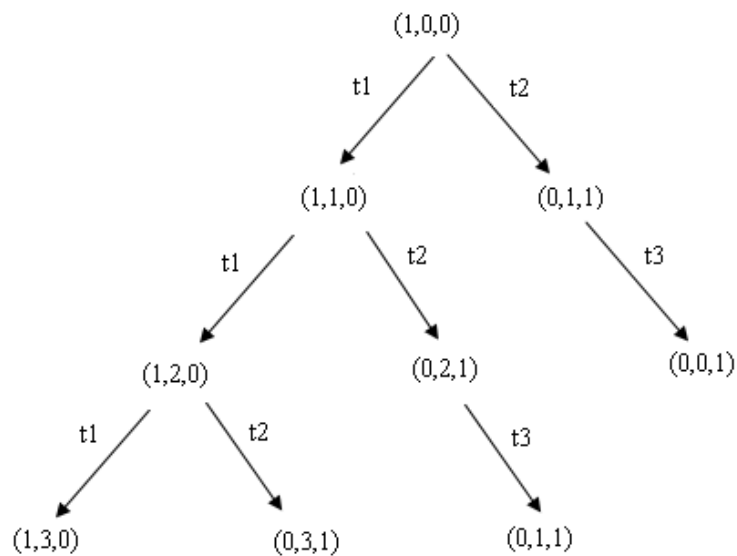
شکل 2-3: یک شبکه پتری برای درخت دسترسی



شکل 2-4: مرحله اول درخت دسترسی

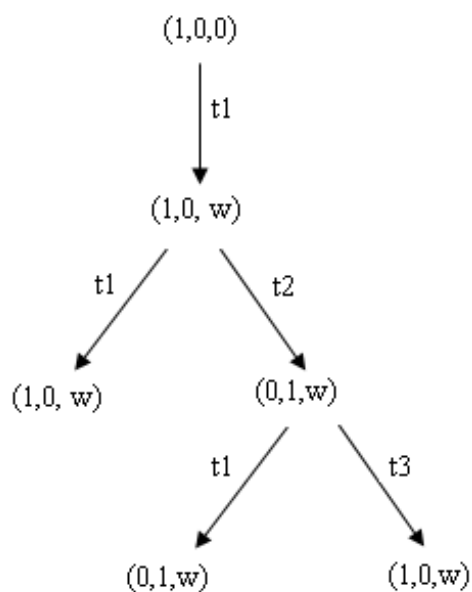


شکل 2-5: مرحله دوم درخت دسترسی



شکل 2-6: مرحله سوم درخت دسترسي

آن‌هایی که تغییری ندارند با w نشان می‌دهیم.



شکل 2-7: درخت دسترسي شبکه پتری

1-1-1-2-2 محدودیت‌های درخت دسترسی

درخت فوق برای حل مسائلی از قبیل بی‌خطری²⁰، حفاظت و امثال آن مورد استفاده قرار می‌گیرد ولی امکان حل توالی شلیک‌ها را ندارد. این امر به خاطر محدودیت وجود w است.

Submarking: بار تمام حالت‌های P_i ، $\mu''(P_i) = \mu'(P_i)$ ، با حالت اولیه μ (فقط یک μ'' است)

Zero reachability: برای تمام دخالت‌های $\mu'(P_i) = 0$ است و هیچگاه تغییر نمی‌کند (همیشه صفر است).

Single space zero reachability: برای یک حالت خاص p_i ، μ' است که $\mu'(P_i) = 0$ می‌باشد.

Reachability: یعنی چنانچه برای μ' داشته باشیم $\mu' \in R(C\mu)$ همان مقدار اولیه باشد (فقط μ' تغییر نکند).

2-1-2-2 معادلات ماتریس

راه دیگر تحلیل ارائه ماتریسی وقایع است.

$$\mu' = \mu + x.D$$

$$\mu' = \delta(\mu, \delta) = \mu + f(\delta).D \quad \mu.w = \mu'.w \quad \text{رابطه‌ی (5-2)}$$

$$\mu.w = \mu'.w = (\mu + f(\delta).D).w = \mu.w + f(\delta).D.w$$

$$f(\delta).D.w = 0 \Rightarrow D.w = 0$$

دو ماتریس D^+ و D^- برای ورودی‌ها و خروجی‌ها تعریف می‌شوند و توابع ورودی و خروجی تعریف می‌شوند و توابع ورودی و خروجی را ارائه می‌کنند. هر ماتریس M ردیف (برای هر گذار یک ردیف) و N ستون (برای هر مکان یک ستون) دارد.

$$D^-[j,i] = \#(P_i, I(t_j))$$

$$D^+[j,i] = \#(P_i, O(t_j))$$

$$D^- = \begin{bmatrix} 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} \quad D^+ = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 2 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \text{رابطه‌ی (6-2)}$$

$$D = D^+ - D^-$$

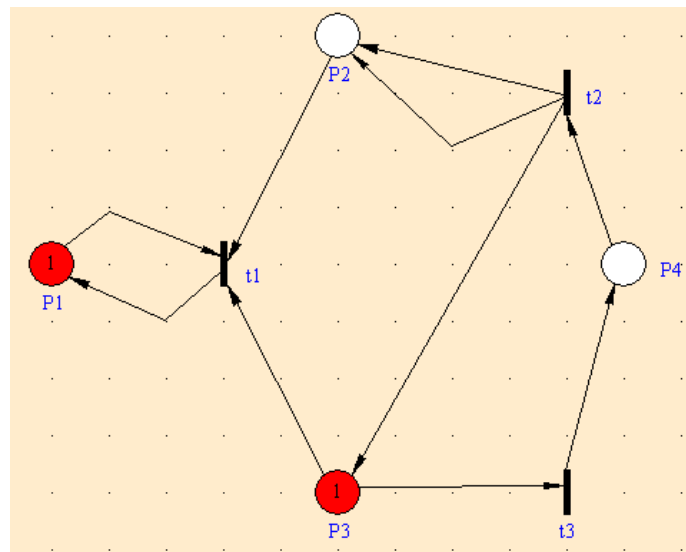
²⁰ Safeness

$$D = \begin{vmatrix} 0 & -1 & -1 & 0 \\ +1 & +2 & -1 & -1 \\ +1 & 0 & -1 & -1 \end{vmatrix}$$

مقدرا اولیه $\mu = (1,0,1,0)$ فعال است و μ' بدست می آید.

رابطه ی
(7-2)

$$\mu' = (1,0,1,0) + (0,0,1) \cdot \begin{vmatrix} 0 & -1 & -1 & 0 \\ 0 & +2 & -1 & -1 \\ 0 & 0 & -1 & +1 \end{vmatrix} = (1,0,1,0) + (0,0,-1,+1) = (1,0,0,1)$$



شکل 8-2: نمودار معادله ماتریسی

2-2-2- بهبود کارایی در شبکه پتری تصادفی

در ابتدا بهبود کارایی می تواند به دو زمینه کاری قسمت بندی شود. اولین زمینه به اندازه گیری²¹ برمی گردد که اندازه گیری بطور کلی به سه زمینه متمایز بخش بندی می شود که عبارتند از:

- اندازه گیری کردن
- الگوها و شاخص های اندازه گیری²²
- پیش نوع ها²³

²¹ Measurement

²² Benchmarks

اندازه‌گیری کردن بر روی سیستم واقعی تحت شرایط اجرایی واقعی امکان‌پذیر است. نتایج اندازه‌گیری‌ها به جزئیات ویژگی‌های سیستم‌های اندازه‌گیری وابستگی زیادی دارند و در خلال اندازه‌گیری بر روی سیستم به زمینه‌کاری خاصی اشاره می‌کنند و عمومیت خیلی کمی دارند.

هنگامی که کارایی دو سیستم مقایسه شده‌اند، اندازه‌گیری‌ها به اندازه کفایت نخواهد داشت و این از موقعی است که هیچ تضمینی بر شرایط اجرا بر اندازه‌گیری‌هایی که در نظر گرفته و معادلات آن‌ها وجود ندارد، پس، با این شرایط مقایسه کردن عادلانه نخواهد بود.

برای غلبه بر این مشکل شاخص‌های اندازه‌گیری بوجود آمدند. آن‌ها که یک بار کردن کاری²⁴ بطور مصنوعی را برای سیستم‌ها مهیا می‌کنند. مثلاً، مشاهدات می‌تواند در یک شرایط اجرایی معادل اجرا شود و مقایسات معنی‌داری بوجود آید. اندازه‌گیری و شاخص‌های اندازه‌گیری موجودیتهای یک سیستم می‌باشند که باید مورد مطالعه قرار بگیرند و بنابراین آن‌ها می‌توانند در (بسیاری) از حالت که مطالعه کارایی مربوط به یک سیستم مهیا نباشند (ممکن است هنوز عملی و اجرایی نشده باشد) مشاهده شوند. لازم است که یک چارچوب نمایشی از سیستم هم بصورت سخت‌افزاری و هم بصورت نرم‌افزاری نرم‌افزاری طراحی شود. چنین ساختاری مطمئناً نیازمند جزئیاتی می‌باشد که به‌طور معمول به آن‌ها پیش‌نمونه گوئیم. در واقع، این یک ساختار اولیه است که چارچوب کلی کار را نشان می‌دهد. مشاهدات بر روی چنین پیش‌نمونه‌هایی اجرا می‌شوند و استفاده از شاخص‌های کارایی را به‌عنوان بار کردن کارهای مصنوعی امکان‌پذیر می‌کنند.

2-2-2-1- شاخص‌های کارایی

عموماً، هدف تجزیه مدل شبکه‌های پتری تصادفی، محاسبه نمایه‌های عملکردی متراکم‌تر و منسجم‌تر می‌باشد تا محاسبه احتمالات علامت‌گذاری فردی. چندین نوع از نتایج به هم پیوسته را می‌توان به راحتی از توزیع حالت ثابت در علامت‌گذاری‌های قابل دسترس بدست آورد. در این بخش، برخی از پارامترهای عملکردی حالات ثابت بهم پیوسته را که به سادگی قابل محاسبه می‌باشند را معرفی می‌کنیم [6].

²³ Prototype

²⁴ Workload

- احتمال يك پيشامد كه از طريق علامتگذاري هاي مكاني تعريف شده (مثلاً، بدون علامت در مجموعه مكانها، و حداقل يك علامت در يك مكان، در حالي كه مكان ديگر خالي مي باشد) را مي توان با اضافه كردن احتمال هاي تمام علامتگذاري ها مشخص كرد و در آنها شرايط مشابه تعريف پيشامد، صحيح مي باشد. بنابراین، مثلاً، احتمال حالت ثابت پيشامد A ، كه از طريق حالي تعريف شده كه براي علامتگذاري هاي $M_i \in M$ درست مي باشد، از طريق $P\{A\} = \sum_{i: M_i \in M} \pi_i$ بدست مي آيد.

- pmf عدد علامتها در يك مكان، مثل p_i ، را مي توان با محاسبه احتمالات تكي در pmf مانند احتمالات پيشامد «مكان p_i كه شامل علائم k مي شود» بدست آورد.
- تعداد متوسط علائم يك مكان را مي توان با محاسبه pmf علائم در آن مكان به دست آورد.
- بسامد فعال شدن يك انتقال يعني متوسط دفعاتي كه انتقال در واحد زمان فعال مي شود را مي توان توسط مجموع سرعت فعال سازي انتقال محاسبه كرد.

$$f_j = \sum_{i: t_j \in E(M_i)} \lambda_j(M_i) \pi_i \quad \text{رابطه (2-8)}$$

كه در آن f_j ، بسامد فعال سازي t_j مي باشد، $E(M_i)$ مجموعه انتقال هاي فعال شده در M_i و $\lambda_j(M_i)$ ، سرعت فعال سازي t_j در M_i مي باشد.

- تأخير متوسط علامت در عبور از يك شبكه فرعي در شرايط حالات ثابت را مي توان با استفاده از فرمول Little محاسبه كرد [7,8].

$$E[T] = \frac{E[N]}{E[\gamma]} \quad \text{رابطه (2-9)}$$

كه در آن $E[T]$ تأخير متوسط، $E[N]$ تعداد متوسط علائم در فرآيند عبور از شبكه فرعي و $E[\gamma]$ سرعت ورودي متوسط علائم در شبكه فرعي مي باشد. اين روند را هرگاه كه امكان شناسايي علائم جالب در داخل شبكه فرعي وجود داشته باشد (كه همچنين مي تواند شامل نشانه هاي ديگر باشد كه شرايط داخلي آن را تعريف مي كند اما، اين علائم بايد از علائمي كه تأخير آنها مطالعه شده قابل تشخيص باشند) بطوري كه عدد متوسط آنها را بتوان محاسبه كرد و يك رابطه را ميان ورودي و خروجي ايجاد كرد. (مثل يك نشانه خروجي براي هر نشانه ورودي).

2-2-3- بهبود کارایی در شبکه های پتری تصادفی عمومی²⁵

عامل اساسی که کاربرد مدل های شبکه های پتری تصادفی را محدود می سازد، پیچیدگی تجزیه آنهاست. این امر به علت عوامل زیادی می باشد. احتمال تعداد بسیار زیاد نشانه گذاری ها اصلی ترین آنهاست. جنبه های دیگر، ممکن است به پیچیدگی راه حل مدل اضافه کند. یکی از آنها به علت حضور در مدل فعالیتها که نسبت به آنهایی که مربوط به پیشامدهایی هستند و نقش حیاتی در عملکرد کلی بازی می کنند، در مقیاس زمانی بسیار سریع تری رخ می دهند. این منجر به پیدایش سیستم های محاسبه ای طولی که تجزیه و تحلیل آن دشوار می باشند می شود، یعنی، حل آنها با مقدار قابل قبولی از دقت توسط تکنیکهای عددی معمول سخت می باشد. از طرفی دیگر، نادیده پنداشتن فعالیت های آنی²⁶ ممکن است منجر به مدل هایی شود که از لحاظ منطقی غیر صحیح هستند. حتی ممکن است در ساخت مکان شناسی یک مدل شبکه های پتری تصادفی، تجزیه گر انتقال هایی را وارد کند که با جنبه های کاملاً منطقی رفتار سیستم مطابق است. [9,10]. بطوری که، هیچ زمان گذاری را نمی توان به آنها مرتبط دانست. مدل های شبکه های پتری تصادفی عمومی از دو نوع انتقال تشکیل شده اند که عبارتند از:

1. انتقال زمان بندی شده که، مرتبط هستند با تأخیرهای فعال شدن توزیع شده نمایی مانند شبکه های پتری تصادفی
 2. انتقال های بلافاصل که در زمان صفر و با تقدم بر انتقال های زمان بندی شده فعال شده می باشند.
- علاوه بر آن، کمان ها مجاز می باشند، سطوح تقدم متعددی از انتقال های بلافاصل را می توان استفاده کرد و وزن ها با انتقال های بلافاصل مرتبط هستند.

2-2-3-1- ساختار شبکه پتری تصادفی عمومی

یک شبکه های پتری تصادفی عمومی، یک هشت تایی بصورت زیر می باشد:

را بطة (2-10) $(P, T, \rho, I, O, H, M_0, W)$

که در آن $(P, T, \rho, I, M_0, O, H)$ مدل غیر زمانی اساسی است که شامل:

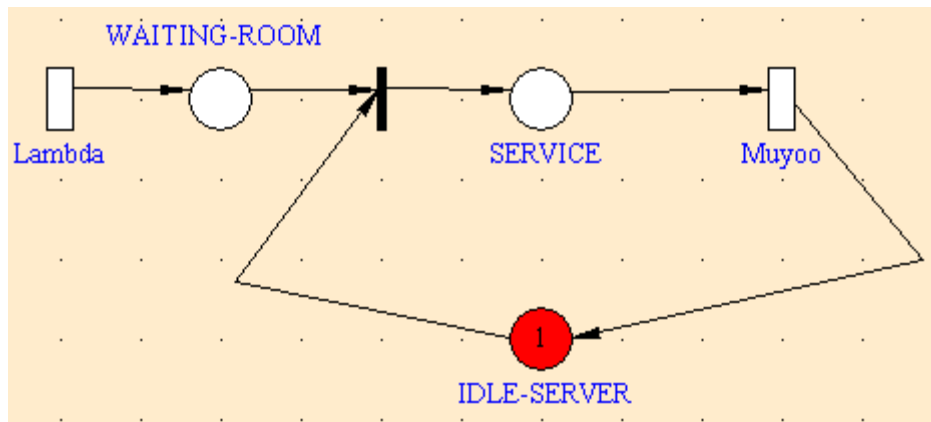
- شبکه های پتری اساسی (P, T, I, O, M_0)
- یک مجموعه از کمان های مانع بصورت $H \subset P \times T$
- تقدم های انتقال به انتقال ها، ρ که با تقدم پایین تر (0)

²⁵ General Stochastic Petri Net (GSPN)

²⁶ Real Time

دارای انتقال‌های زمانی و تقدم‌های بالاتر (≥ 1) با انتقال‌های بلافاصل ارتباط دارد و $W = (w_1, w_2, \dots, w_n)$ ، یک سری است با مدخل w_i

- یک پارامتر تابع شدت احتمال نمایی منفی از تأخیر فعال سازی انتقال است اگر t_i یک انتقال زمانی باشد.
- W یک وزن است که برای محاسبه احتمالات فعال سازی انتقال‌های بلافاصل استفاده می‌شود اگر t_i یک انتقال بلافاصل باشد. در نمونه گرافیکی شبکه های پتری تصادفی عمومی، انتقال‌های بلافاصل به عنوان بخش‌ها و انتقال‌های زمانی به عنوان مربع‌های چهارگوش طراحی شده‌اند. تفسیر یک مدل شبکه های پتری تصادفی عمومی مشابه تفسیر مدل شبکه های پتری تصادفی است که تغییراتی برای توضیح انتقال‌های بلافاصله لازم می‌باشد. هنگامی که علامتی وارد می‌شود، ابتدا لازم است که اطمینان حاصل شود که فقط انتقال‌های زمانی یا حداقل یک انتقال بلافاصل را فعال می‌سازد. علامت‌گذاری‌های نوع پیشین را محسوس²⁷ و علامت‌گذاری‌های نوع دوم را در نامحسوس می‌نامیم.



شکل 9-2: نمونه شبکه های پتری تصادفی عمومی از صف M/M/1

در حالت علامت‌گذاری محسوس، زمان سنج‌های انتقال‌های زمانی فعال شده کاهش خود را از سرگرفته یا دوباره تنظیم شده و سپس کاهش می‌یابند تا زمانی که یک انتقال زمانی درست مثل حالت شبکه های پتری تصادفی فعال شود.

در حالت علامت‌گذاری نامحسوس، انتخاب اینکه کدام انتقال فعال شده، نمی‌تواند براساس توصیف موقتی باشد زیرا تمام انتقال‌های بلافاصل دقیقاً در زمان صفر فعال می‌شوند. پس این انتخاب

²⁷ Sensitive

براساس تقدمات و وزنهای می باشد. مجموعه انتقالهای دارای بیشترین سطح تقدم در ابتدا شناسایی میشود و اگر شامل بیش از يك انتقال شود، انتخاب بعدی، از نوع احتمالی و براساس وزنهای انتقال طبق عبارت زیر خواهد بود:

$$P\{t_k\} = \frac{w_k}{\sum_{it_i \in E(M)} w_i} \quad \text{رابطه (2-11)}$$

که در آن $E(M)$ عبارت است از مجموعه انتقالهای بلافاصل فعال شده در علامتگذاری M یعنی انتقالهای دارای تقدم با سطح بالا. دقت کنید که معناشناسی مدل شبکه های پتری تصادفی عمومی همیشه بر این فرض استوار نیست که انتقالها، حتی در علامتگذاری غیر محسوس که شامل انتقالهای بلافاصل فعال شده یکسان است، يك به يك فعال شوند. معادله این رفتار با معادله ای که از فعال شدن همزمان برخی از انتقالهای بلافاصل در این مدل ناشی میشود را میتوان برای کاهش پیچیدگی الگوریتمهای راه حل بکار برد [11]. وجود کمان های مانع و تقدمات اولیه تعداد علامتگذاریهای قابل دسترس را با توجه به شبکه های پتری بنیادی کاهش میدهد. مجموعه قابل دسترس شبکه های پتری تصادفی عمومی، برابر است با یکی از شبکه های پتری بنیادی که شامل کمان های مانع و تقدمات میباشد. تجزیه يك مدل از شبکه های پتری تصادفی عمومی باز به حل يك سیستم معادلات خطی دارد که شامل تعداد زیادی معادلات مانند علامتگذاریهای محسوس²⁸ قابل دسترس میشود. ژنراتور بسیار ریز²⁹ زنجیره مارکو پیوسته زمانی که به مدل شبکه های پتری تصادفی عمومی مرتبط است با کاهش گراف قابل دسترس که تحت عنوان سرعتها یا وزنهای انتقالهای باعث تغییر علامت قرار گرفته است، بوجود می آید.

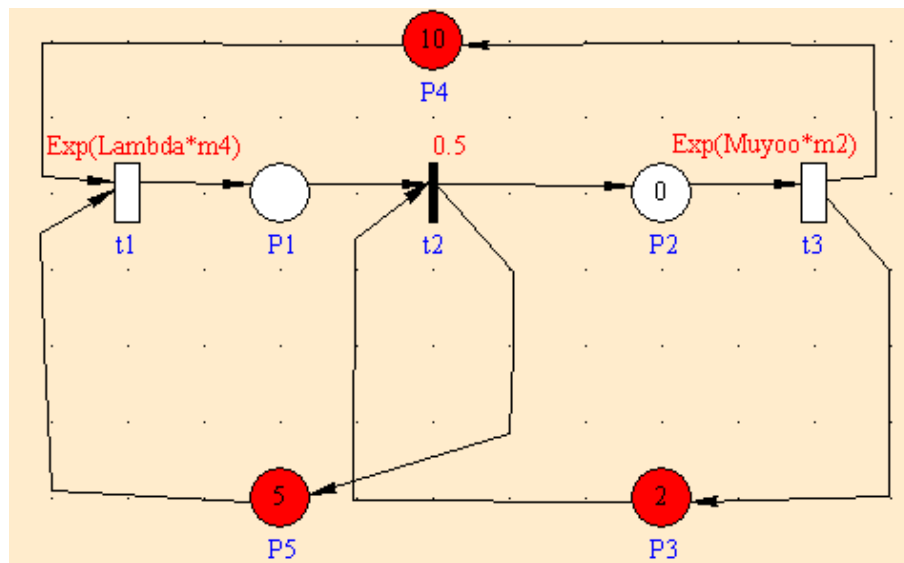
در شکل 2-9 به عنوان اولین نمونه از مدل شبکه های پتری تصادفی عمومی، نمایشی از صف $M/M/1$ را نشان داده ایم. احتمال استفاده از يك انتقال بلافاصله نمایش آشکار³⁰ اتاق انتظار، از ایستگاه سرویس و از سرور بلااستفاده را ممکن میسازد. يك مدل نسبتاً پیچیده تر، نمونه شبکه های پتری تصادفی عمومی از صف $M/M/2$ است که دارای اتاق انتظار محدود (5 جایگاه) و جمعیت کاربری محدود (10 مشتری) میباشد (مانند شکل 2-10).

²⁸ Tangible Marking

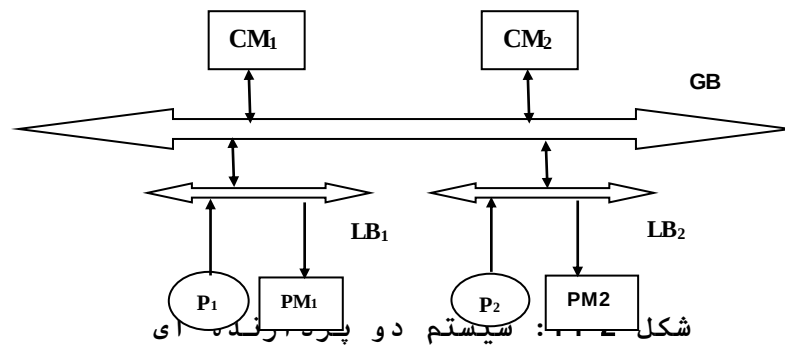
²⁹ Infinitesimal Generator

³⁰ Explicit Representation

این سیستم صفی عموماً $M/M/2/5/10$ نام دارد. اتاق انتظار توسط مکان p_1 نشان داده می‌شود، سرورهای بلا استفاده توسط نشانه‌های p_3 در آن مکان به نمایش درمی‌آیند؛ سرورهای مشغول نشانه p_2 در محل را دارند. مکان p_4 ، دارای مشتریانی است که ممکن است به صف برسند. مکان p_5 دارای نشانه‌هایی است که مکان‌های خالی را در اتاق انتظار نشان می‌دهد و باعث ایجاد اعتبار برای پذیرش در صف می‌گردد. علامتگذاری p_4 ، سرعت فعال سازی ورود مشتریان مدل انتقال و علامتگذاری p_2 ، سرعت فعال سازی خروج مشتریان مدل انتقال را تعیین می‌کند.



شکل 2-10: نمایش شبکه های پتری تصادفی عمومی صف $M/M/2/5/10$



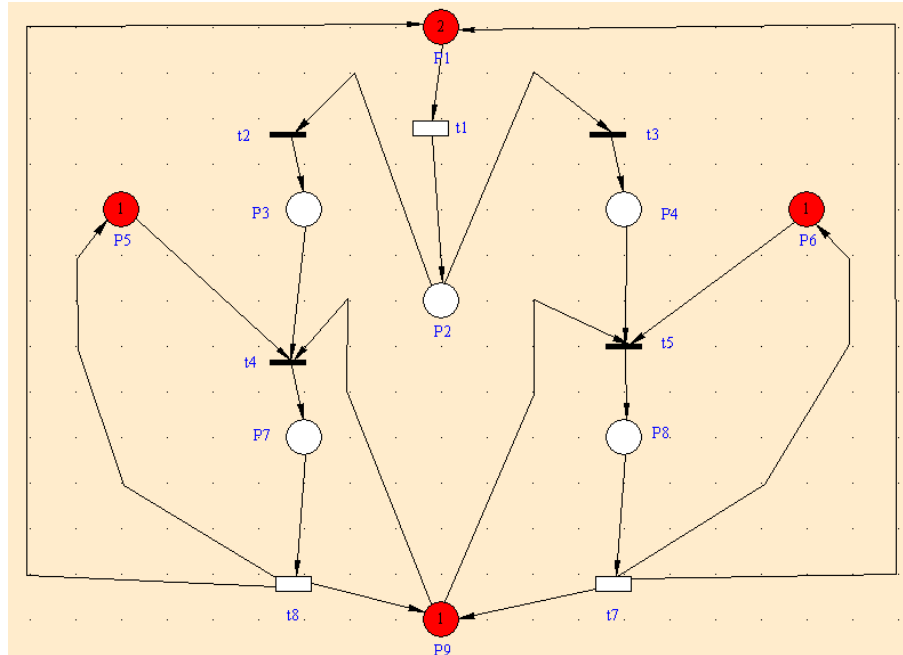
در واقع، سیستم‌های صفی جمعیت محدود عموماً با استفاده از یک فرآیند ورودی به همراه زمانهای ورودی میانی توزیع شده به صورت

نهایی که سرعت آن با توجه به تعداد مشتریان خارج از صف نسبی است، مدل‌بندی می‌شوند.

به عنوان سومین نمونه از یک مدل شبکه های پتری تصادفی عمومی، عملکرد سیستم دارای 2 پردازشگر شکل 2-10 را مورد تجزیه قرار می‌دهیم [12].

واحدهای دارای دو پردازشگر، از یک پردازشگر (P_1, P_2) و یک حافظه اختصاصی (PM_1, PM_2) تشکیل شده که توسط دو باس محلی (LB_1, LB_2) به هم مرتبط می‌شوند و از دو ماژول حافظه معمولی (CM_1, CM_2) توسط یک باس کلی و سراسری (GB) به هم مرتبط می‌شوند. پردازشگرها یا می‌توانند فعال باشند یعنی در حافظه اختصاصی عمل کنند یا سعی کنند به یک واحد حافظه مشترک دست یابند. تنها زمانی تقاضاها پاسخ داده می‌شوند که باس سراسری و حافظه عمومی تقاضا شده در دسترس باشند (توجه کنید که در این حالت، در دسترس بودن باسهای عمومی بدین معناست که حافظه‌های کلی هر دو در دسترس هستند، طوری که مدل را می‌توان ساده نمود) زمانهای فعال پردازشگر و زمانهای دستیابی به حافظه را متغیرهای تصادفی با تابع شدت احتمال نمایی منفی در نظر می‌گیریم.

مدل شبکه های پتری تصادفی عمومی بدست آمده از این سیستم دارای دو پردازشگر می‌باشد که در شکل 2-12 نشان داده شده است. نشانه‌ها در مکان p_1 ، پردازشگرهای فعال را به نمایش درمی‌آورند. انتقال t_1 ، مورد یک تقاضا دسترسی به حافظه کلی را نمایش می‌دهد. انتقال‌های t_1 و t_3 ، انتخاب یکی از دو واحد حافظه کلی را مدل‌گذاری می‌کنند. نشانه‌ها در مکان‌های p_5 و p_6 ، در دسترس بودن دو واحد حافظه کلی را نیز نشان می‌دهد. یک نشانه در مکان p_9 ، حاکی از در دسترس بودن باس عمومی می‌باشد. انتقال‌های t_4 و t_5 آغاز دسترسی به حافظه کلی را مدل‌گذاری می‌کنند که مدت آن با تأخیر در انتقال‌های t_6 و t_7 نشان داده می‌شود، سرعت‌های فعال شدن t_1, t_6 و t_7 ، عبارتند از λ ، μ و μ .



شکل 2-12: شبکه های پتری تصادفی عمومی نمایش داده شده در سیستم دو پردازنده ای در شکل 2-11

کاربرد تکنیک‌های مدلسازی شبکه‌های پتری تصادفی و شبکه های پتری تصادفی عمومی در زمینه‌های مختلف فاصله زیادی با هم دارند. عاملی که پذیرش آن‌ها را به‌عنوان ابزارهای مدلسازی محدود کرده است در پیچیدگی واقع گرایانه³¹ مدل وجود دارد یعنی این عاملی که در این سیستم‌ها باعث محدود بودن شده است. مسأله پیچیدگی را به‌طور دقیق‌تر در بخش بعدی مورد بحث و بررسی قرار می‌دهیم. همچنین باید در این مسأله تأکید نمود که کاربرد شبکه‌های پتری تصادفی و شبکه های پتری تصادفی عمومی وابستگی شدیدی به ابزارهای کافی نرم‌افزاری موجود برمی‌گردد که ساخت مدل و راه‌حل‌ها فقط به مثال‌های بسیار ریز امکان‌پذیر خواهند بود.

2-3-2-2- توزیع عمومی تعداد دفعات فعال شدن

برخی از تحقیقات تلاش بر این است تا تلاش را به این اختصاص دهد تا قدرت مدلسازی شبکه‌های پتری تصادفی و شبکه های پتری تصادفی عمومی را با اجازه دادن تأخیرات فعال شدن تا متغیرهای تصادفی تابع شدت احتمال مطلق مجاز بشمرد. این نتایج در تعاریف شبکه‌های پتری تصادفی توزیع شده³² یا شبکه‌های پتری تصادفی گسترش یافته³³ [13]، شبکه‌های پتری تصادفی معین [14,15]³⁴ یا تاخیر انتشار در

³¹ Realistic

³² Distributed SPN(DSPN)

³³ Extended SPN(ESPN)

فاز گسترش آن ها بیان شده است. ذکر این نکته ضروری است که موجود بودن در این مدل ها دارای تأخیر در فعال سازی تصادفی با تابع شدت احتمال با تأخیرهای نمایی منفی در روند تعریف اجرای شبکه های پتری تصادفی می باشد. در واقع تابع شدت احتمال نمایی آن ها منفی است یا به تعبیر دیگر، هرگاه تغییر علامتگذاری اتفاق بیافتد حالتی را فعال می کند که از آخرین فعال سازی خود فعال نشده بودند. این حالت نمونه ای از تأخیر فعال سازی را از تابع شدت احتمال عمومی مرتبط نشان می دهد و یک زمان سنج ارزش یک نمونه ی تأخیر در نظر گرفته شده را تنظیم می کند [16,17].

وقتی که انتقال فعال می شود زمان سنج با سرعت یکسانی کاهش می یابد و اگر حالت انتقال با فعال شدن یک حالت متضاد غیر فعال شود آنگاه زمان سنج متوقف می شود و زمانی که انتقال دوباره فعال می شود زمان سنج دوباره فعال می شود و کارش را ادامه می دهد. هنگامی که زمان سنج به مقدار 0 می رسد انتقال فعال می شود. روندهای فعال سازی دیگر نیز مانند رقابت با حافظه فعال سازی که در آن موقع که یک انتقال فعال می شود نمونه گیری مجدد صورت می گیرد و پیش انتخاب که در آن یک انتقال که باید فعال شود و مستقل از تأخیرهای فعال سازی انتخاب می شود.

تفاوت بین حالات رقابت میان طول عمر و حافظه فعال کننده از این واقعیت ناشی می شود که در حالت توزیع های عمومی ویژگی بدون حافظه وجود ندارد بنابراین، زمان باقی مانده قبل از فعال سازی هنگامی که انتقالی که غیر فعال شده بود دوباره فعال می شود توزیعی متفاوت از زمان فعال سازی نمونه گیری شده از توزیع مربوط به انتقالی غیر فعال فعال شده بود ایجاد می کند. متأسفانه قدرت مدل سازی کاهش یافته ای این مدل های شبکه های پتری تصادفی توسعه یافته، هزینه اش با یک پیچیده افزایش یافته راه حل شان پرداخت می شود. طوری که، این نوع مدل ها اغلب به منظور شبیه سازی بافت ها استفاده می شود برخی از نتایج جدید نشان داده اند که در حالت شبکه های پتری تصادفی گسترش یافته امکان آن وجود دارد که در ساخت از مدل شبکه های پتری بنیادی استفاده کرد تا کاهش های قابل توجهی را در کاهش پیچیدگی راه حل به دست آورد [18]. علاوه بر آن، در مقاله [19] هندرسون و لوسیک در بحثی شرایطی که تحت آن ها توزیع حالت پایدار³⁵ بر نشانه ها تنها بستگی به تأخیر فعال سازی متوسط یک انتقال دارد و وابسته به نوع توزیع نیست تعریف کردند. در بخش 2-3-1-2 در شبکه پتری تصادفی به این نتایج برمی گردیم.

2-3-2-3 روابط با مدل های دیگر

پیشنهادهای مختلفی متفاوت از آنچه در این بحث در نظر گرفته

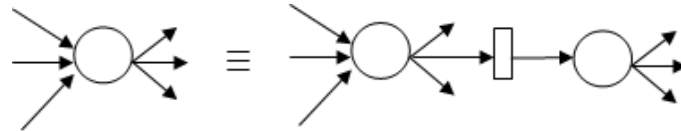
³⁴ Deterministic Stochastic Petri Net (DSPN)

³⁵ Steady-State

شده اند به منظور معرفی ویژگی‌های موقتی احتمالی در درون يك مدل معرفی شده اند و در محافل علمی منتشر شده است. دو جریان اصلی را می‌توان شناسایی نمود :

❖ مدل‌های شبکه‌های پتری که در آن‌ها زمان‌گذاری با مکان‌ها ارتباط دارد [20,21].

❖ مدل‌های شبکه‌های پتری که در آن‌ها انتقال‌های زمان‌گذاری شده در سه زمان فعال می‌شوند [22-25].



شکل 2-13: نمایش مکانهای زمانبندی شده در چارچوب شبکه‌های پتری تصادفی عمومی

ما بدون در نظر گرفتن فرضیه‌ی برتری هر يك از مدل‌های شبکه‌های پتری زمان‌گذاری شده با توجه به بقیه توضیح می‌دهیم که چگونه معنای حالت بالا (شکل 2-13) را می‌توان با نشانه‌ی شبکه‌های پتری تصادفی عمومی ارائه داد. هنگامی که زمان‌گذاری با مکان‌ها مرتبط می‌شود در حالت فعال شدن نشانه‌های انتقال فعال شده از مکان‌های ورودی خارج می‌شوند و در خروجی با يك عملیات مستقل قرار می‌گیرند. اگر چه، نشانه‌ها در مکان‌های خروجی انتقال قرار می‌گیرند بلافاصله برای فعال کردن سایر انتقال‌ها آماده و مهیا نمی‌باشند. فقط آن‌ها بعد تأخیری که در ارتباط با مکان‌های گذشته دارند فعال می‌شوند. پس امکان آن وجود دارد با جایگزینی هر مکان زمان‌گذاری شده با يك مکان استاندارد و يك انتقال زمان‌گذاری شده و يك انتقال استاندارد دیگر، شبکه‌های پتری زماندار مرتبط با مکان‌ها را تبدیل به يك فرمول شبکه‌های پتری تصادفی عمومی بکنیم. پیکان‌های ورودی مکان زمان‌گذاری شده بر اولین مکان غیر زمانی مرتبط هستند و پیکان‌های خروجی با مکان زمان‌گذاری شده از مکان غیر زمان‌گذاری شده دومی جدا می‌شوند. (طبق شکل 2-13)

توجه کنید که طول عمر نشانه‌هایی که در مکان زمان‌گذاری قرار دارند به صورت موازی پیش می‌روند به طوری که انتقال نشان داده شده در شکل 2-13 مستلزم آنست که سرعت فعالسازی انتقال زماندار به تعداد نشانه‌ها در مکان ورودی نسبی باشند (انتقال که عبارت صفبندی را قرض گرفته گفته می‌شود از نوع سرور نامعین می‌باشد).

زمان‌گذاری به انتقال‌هایی مربوط است که در سه مرحله فعال می‌شوند، يك انتقال فعال شده بلافاصله شروع به فعال شدن می‌کند و نشانه‌ها را از مکان‌های ورودی خود حذف می‌کند. نشانه‌ها، به هر حال، در مکان‌های خروجی انتقال قرار نمی‌گیرند تا زمانی که تأخیر انتقال سپری شود. آنها برای کل تأخیر انتقال در انتقال نگهداری می‌شوند. به عبارت دیگر، فعالسازی يك انتقال از سه

مرحله تشکیل شده است که عبارتست از:

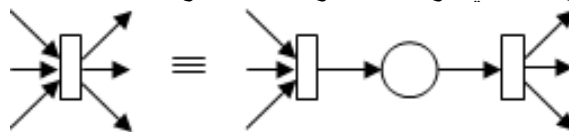
1. يك مرحله شروع فعالسازي که در آن نشانه‌ها از مکانهای ورودی خارج می‌شوند.

2. يك فعالسازي در مرحله پیشرفت، که تأخیر زمانی به آن مرتبط است.

3. يك مرحله فعالسازي نهایی، که در آن نشانه‌ها در درون مکانهای خروجی مرتبط هستند.

انتقال يك شبکه‌های پتری زماندار با انتقال‌هایی که در سه مرحله به فرمول شبکه‌های پتری تصادفی عمومی تبدیل می‌شوند، به راحتی با جایگزینی هر انتقال سه مرحله‌ای با انتقال بلافاصله، يك مکان و يك انتقال زماندار صورت می‌گیرد. پیکانهای ورودی انتقال سه مرحله‌ای، به انتقال بلافاصله مرتبطند و پیکانهای خروجی انتقال سه مرحله‌ای از انتقال زماندار جدا می‌شوند (مانند شکل 2-14).

عموماً، انتقال‌های زماندار سه مرحله‌ای می‌توانند يك فعالسازي را از ابتدا آغاز کنند و این در حالی است که فعالسازي دیگر در حال اجرا شدن می‌باشد. انتقال زماندار در شکل 2-14، از نوع سرور نامعین می‌باشد. هنگامی که انتقال زماندار سه مرحله‌ای می‌تواند تنها زمانی که يك فعالسازي قبلی تکمیل نشده، فعالسازي جدید را آغاز کند (سرعت فعالسازي انتقال زماندار شکل 2-14 ثابت است گفته می‌شود انتقال از نوع سرور و حد³⁶ می‌باشد). يك مانع می‌تواند از آن به مکان انتقال بلافاصله اضافه شود و این در صورتی است که در طی تأخیر انتقال نشانه‌ها باید در مکانهای ورودی انتقال سه مرحله‌ای زماندار حفظ شوند.



شکل 2-14: نمایش حالت زمانبندی شده سه مرحله‌ای در چارچوب شبکه‌های پتری تصادفی عمومی

باید خاطر نشان ساخت که در حالت انتقال‌های زماندار که در سه مرحله فعال می‌شوند، تضادهای، به انتقال‌هایی محدود می‌شوند که همزمان فعال شده‌اند. در واقع، ممکن نیست که انتقالی که فعال شده در زمان دیرتری با فعالسازي يك انتقال فعال شده قبلی، غیر فعال گردد. به‌منظور شامل کردن این ویژگی که ممکن است در برخی از حالات برای مدلسازي سیستم‌ها مفید باشد، مدل‌های شبکه‌های پتری زمانی که در آنها انتقال در سه مرحله فعال می‌شود، پیکانهای خاصی که انتقال خاصی را غیر فعال³⁷ می‌کنند پیشنهاد شده‌اند [26].

³⁶ Single Server

³⁷ Unfired

2-2-3-4- تجزیه ساختاری شبکه های پتری تصادفی

همانطور که در 2-2-3-1 گفته شد، شبکه های پتری تصادفی از ابتدا، توسط محققینی پیشنهاد شد که در زمینه مدل سازی تصادفی کاربردی فعال بودند و آن را به عنوان یک نشان گرافیکی مناسب برای تعریف انتزاعی مدل های مارکوفین معرفی کردند. بنابراین، احتمال تعریف سرعت های فعال سازی مستقل از علامت گذاری در شبکه های پتری تصادفی بر این فرضیه بود که تجزیه گر از تمام علائم قابل دسترس آگاهی داشته باشد تا بدرستی وابستگی علامت گذاری را مشخص کند.

حتی، تعریف ابتدایی شبکه های پتری تصادفی عمومی نیز موجود بوده که در آن انتخاب انتقال بلافاصله که باید در یک علامت گذاری ناپدید شود فعال می شود، بر اساس یک توزیع راه گزینی کاربردی قرار دارد. که، آن را می توان با دانستن مجموعه قابل دسترسی بکه های پتری تصادفی عمومی تعریف نمود. با تعریف اصلاح شده شبکه های پتری تصادفی عمومی، تلاشی به منظور تلفیق ویژگی های ساختاری و اتفاقی مدل صورت گرفت طوری که تعریف مدل در سطح شبکه و بدون نیاز به ایجاد مجموعه قابل دسترسی، ممکن شد.

یکی از مزایای اصلی مدل شبکه های پتری تصادفی بنیادی، علاوه بر سادگی و سهولت استفاده، هویت قابل دسترسی گراف آن به همراه یکی از شبکه های پتری بنیادی می باشد. این بدان معناست که تمام نتایج موجود برای تجزیه ساختاری شبکه های انتقال را بتوان بر سادگی در محیط شبکه های پتری تصادفی بکار برد.

شبکه های پتری تصادفی عمومی، در عوض، شبکه قابل دسترسی، مشابه یکی از شبکه های انتقال بنیادی به همراه پیکانهای مانع و اولویت ها می باشند، یک مدل غیر زمانی که هر گونه توجه کوچک را از تئوری پردازان شبکه دریافت می دارد. خوشبختانه، وجود پیکانهای مانع و اولویت ها، تنها مجموعه قابل دسترسی را با توجه به یکی از شبکه های بنیادی اساسی که از طریق حذف اولویت ها، پیکانهای مانع و زمان گذاری بدست می آید محدود می کند. این بدان معناست که برخی از ویژگی های ساختاری معتبر برای شبکه های پتری بنیادی اساسی توسط شبکه های پتری تصادفی عمومی بدست می آید. در واقع، محدود سازی شبکه قابل دسترسی، ضمانت می کند که تمام متغیرهای مکانی که در مطالعه شبکه های پتری بنیادی اساسی بدست آمده، همچنان برای شبکه های پتری تصادفی عمومی وجود داشته باشند. در اصل، متغیرهای مکانی دیگری موجود است که در مورد شبکه های پتری تصادفی عمومی صدق می کند اما، برای شبکه های پتری بنیادی اساسی معتبر نمی باشد. تجربه نشان داده است که این مسئله عموماً رخ نمی دهد زیرا، آنچه مربوط به متغیرهای انتقال است یعنی وجود پیکانهای مانع و اولویت ها، ممکن است متغیری که قابل فعال سازی در شبکه های پتری بنیادی اساسی است را غیر محتمل کند. در حالتی که در تابع شدت احتمال تأخیر عمومی ممکن است

مشکل تجزیه ساختاری مشکل‌تر باشد، زیرا شاید این اتفاق بیفتد که ویژگی‌های احتمالی این مدل تأثیری بر رفتار کیفی داشته باشد تأثیری به این صورت که اگر چه دو انتقال همزمان فعال شوند، یکی از آنها نمی‌تواند به علت محدودیت‌های زمانی فعال گردد، یعنی ویژگی‌های ساختاری شبکه را نمی‌توان کاملاً بررسی نمود حتی با در نظر گرفتن پیکانهای مانع و اولویت‌های ممکن. بنابراین، زمانگذاری باید به عنوان یک جز ساختاری محسوب گردد. یک شرط کافی به منظور اجتناب از این تأثیر نامطلوب ویژگی موقتی بر رفتار کیفی مدل این است. تمام تابع شدت احتمال تأخیر فعال سازی میزانی نامعین داشته باشند. به هر حال، این محدودیت، استفاده از تأخیرهای فعال سازی سریع را که از دیدگاه مدلسازی اهمیت خیلی دارد، ممکن نمی‌سازد. به هر حال، باید تأکید نمود که امروزه ویژگی‌های ساختاری مدل‌های شبکه‌های پتری تصادفی در هر حالت برای تعریف مدل یا محاسبه نتایج نسبی استفاده می‌شوند [27,28]. چیزی در مورد نقش ویژگی‌های ساختاری مدل بر کاهش پیچیدگی محاسبه pmf بر علامتگذاری‌های قابل دسترس وجود ندارد.

2-2-4- خلاصه و نتیجه گیری

اکثر تحقیقات کنونی در زمینه شبکه‌های پتری تصادفی، به کاربرد شبکه‌های پتری تصادفی و مدل‌های مربوط به عمل کرد و مطالعات معتبر محدوده وسیعی از سیستم‌ها می‌شود که شامل زمینه های زیر می‌باشد که عبارتست از: معماری‌های سیستم‌های محاسبه توزیع شده، نرم افزار توزیع شده، پروتکل ارتباطی، سیستم‌های تولید، VLSI، پایگاه داده، فهرست و آمارهای منطقی، سیستم‌های شی گرا³⁸ و سیستم‌های زمان واقعی³⁹.

در برخی از این مطالعات کاربردی، جنبه تجزیه عملکرد با اثبات رسمی درست بودن سیستم تحت بررسی تلفیق شده که این کار را با بهره‌گیری از توصیف سیستم رسمی بدست آمده با فرمول شبکه‌های پتری صورت داده اند [29]. همانطور که گفتیم، مشکل اساسی در استفاده از تکنیک‌های شبکه‌های پتری تصادفی برای تجزیه سیستم‌های واقعی از پیچیدگی راه حل سیستم نشأت می‌گیرد. معمولاً این مدل‌ها شامل فضاهای عظیمی هستند که تولید مجموعه قابل دسترس را برای اجرا بسیار هزینه‌بر می‌سازد (هم از لحاظ زمانی و هم فضایی). چندین تحقیق به تلاش‌هایی به منظور کاهش هزینه راه حل اختصاص یافته است.

یک راه حل، استفاده از تکنیک‌های شبیه سازی می‌باشد. با انجام این کار، مشکلاتی که از پیچیدگی فضایی ناشی می‌شوند از بین می‌روند. زیرا، ایجاد مجموعه قابل دسترس دیگر لازم نیست، اما، مشکلات مربوط به پیچیدگی زمان همچنان به قوت خود باقی است.

³⁸ Object Oriented Systems

³⁹ Real-Time Systems

(مثل مورد شبیه سازی). از سوی دیگر، هاس⁴⁰ و شیلدر⁴¹[12,30,31] نشان داده اند که قدرت مدلسازی شبکه های پتری تصادفی در چهارچوب شبیه سازی بسیار برجسته است زیرا آنها معادل فرآیندهای نیمه - مارکو عمومی هستند.

روش های دیگر را که می توان برای کاهش پیچیدگی راه حل مدل استفاده نمود که مربوط به استفاده از ویژگی های ساختاری مدلهای شبکه های پتری برای ایجاد الگوریتم های راه حل مؤثر می باشد. متغیرهای مکانی برای بهبود اثربخشی فضایی ایجاد مجموعه قبا بل دسترسی مفید می باشد. ارتباط اتفاقی و حذف دو جانبه به همراه متغیرها را می توان برای شناسایی زیر مدلهایی بکار برد که در حالات خاص قابل مطالعه به صورت مجزا می باشند و نتایجی ارائه می دهد که ترکیب بعدی آنها، راه حل کلی مطلوبی را ارائه می دهند. تلاش به منظور استفاده از ویژگی های ساختاری مدلهای شبکه های پتری بنیادی در فازهای مختلف یک مطالعه عمل کردی با استفاده از متغیرهای مکان و انتقال می باشد. مدل و فرآیند اشکال زدائی⁴² مدل توصیف شده است[32]. استفاده از متغیرهای مکانی، ارتباط اتفاقی⁴³، اجرای دوجانبه⁴⁴ برای بررسی اینست که آیا مدل را می توان در کلاسی خاص طبقه بندی نمود؟ استفاده از متغیرهای انتقال برای شناسایی راه های باریک، استفاده از تضادها به منظور سهولت تعریف مدل شبکه های پتری تصادفی عمومی می باشد. پیچیدگی گرافیکی مدلهای شبکه های پتری تصادفی را می توان در صورت استفاده از توصیف سطح بالا کاهش داد اما، تحقیقات به منظور دستیابی به تعادل کافی حداکثر میان قدرت مدل سازی و سهولت استفاده ادامه دارد.

در نهایت، برخی از محققان در حال بررسی احتمال ترکیب شبکه های پتری تصادفی با صف ها هستند تا روش ارزیابی عملکرد تلفیقی را بدست آورند که از ویژگی های مطلوب هر دو روش استفاده کند .

هدف این بحث، معرفی شبکه های پتری تصادفی برای متخصصین شبکه پتری می باشد که با رشته مدلسازی عملکرد تصادفی آشنایی ندارند. بدین علت، یک مرور مختصر از رشته تجزیه عملکرد، از فرآیندهای مارکوفین و از مدلهای صف بندی ارائه شد. ویژگی های مدلهای شبکه های پتری تصادفی و انگیزه ای که به تعریف آنها منجر می شود توصیف شد. برخی از نمونه ها، همچنین برای توضیح کاربرد احتمالی بیان شد و روش شبکه های پتری تصادفی برای تجزیه عملکرد سیستم های استفاده شد. خلاصه مختصری از فعالیتهای تحقیقاتی موجود و بارزترین مشکلات ارائه شده است. با این امید که برخی

⁴⁰ Hass

⁴¹ Shilder

⁴² Debugging

⁴³ Casual Connection

⁴⁴ Mutual Exclusion

از خوانندگان به رشته شبکه های پتری تصادفی علاقه مند شوند و نتایج جدیدی که کاربرد روش شبکه های پتری تصادفی را گسترش می دهد، بدست آورند [33, 34].

اتوماتای یادگیر

2-3-1- مقدمه

فرآیندی که طی آن، موجودات زنده مطالب مختلف را یاد می‌گیرند، مدت‌های مدیدی مورد علاقه متخصصان بوده است. تحقیقاتی که در این زمینه انجام شده است، در دو شاخه‌ی کلی متمرکز می‌باشد:

1. درک فرآیندی که موجودات زنده در طی آن، عمل به یادگیری نمایند.
2. بدست آوردن روش‌هایی که با استفاده از آنها بتوان این قابلیت یادگیری را در ماشین‌ها ایجاد نمود.

یادگیری به روش‌های مختلفی تعریف شده است. بنابراین تعریف یادگیری، اعمال یک سری تغییرات در کارایی سیستم بر اساس تجارب قبلی می‌باشد. مهم‌ترین مشخصه‌ی این سیستم یادگیر، قابلیت آن در افزایش کارایی در طول زمان می‌باشد. اگر بخواهیم کاملاً به صورت ریاضی این تعریف را بیان نماییم، می‌گوییم: هدف یک سیستم یادگیر، بهینه‌سازی یک عملکرد می‌باشد که امکان شناسایی آن به طور کامل وجود ندارد. با این تعریف می‌توان وظیفه‌ی سیستم یادگیر را به یک مسئله‌ی بهینه‌سازی که بر روی یک مجموعه از پارامترها تعریف شده است کاهش داد و با بدست آوردن مجموعه‌ی پارامترهای بهینه، از طریق روش‌های موجود آن را حل نمود. در این بخش روش‌های یادگیر که براساس اتوماتاهای یادگیر تصادفی⁴⁵ (SLA) می‌باشند، بررسی می‌شود.

مفهوم اتوماتای تصادفی اولین بار توسط گروه تحقیقاتی تستلین⁴⁶ در اوایل سال 1960 در اتحاد جماهیر شوروی مطرح گردید. پس از آن در تحقیقات بعدی، نمونه‌های مختلفی از کاربرد روش‌های یادگیری در سیستم‌های مهندسی بوجود آمد که از جمله‌ی آنها می‌توان کاربرد در مسیریابی تلفن، شناسایی الگو، تقسیم‌بندی اشیاء⁴⁷ و کنترل قابل تطبیق⁴⁸ را نام برد.

⁴⁵ Stochastic Learning Automata (SLA)

⁴⁶ Tsetlin

⁴⁷ Object Partitioning

⁴⁸ Adaptive Control

اتوماتای یادگیر⁴⁹ [35, 36] را می‌توان یک شیء مجرد در نظر گرفت که دارای تعداد محدودی عمل⁵⁰ می‌باشد. عملکرد این شیء به این صورت است که در هر زمان، یک عمل از بین مجموعه‌ی اَعمال انتخاب می‌گردد و سپس در یک محیط تصادفی مورد ارزیابی قرار می‌گیرد. جوابی که از محیط دریافت می‌شود، توسط اتوماتا برای انتخاب عمل بعدی به کار گرفته می‌شود و به این ترتیب اتوماتا به تدریج، عمل بهینه را شناسایی می‌نماید. روشی که اتوماتا با استفاده از آن پاسخ محیط را برای انتخاب عمل بعدی به کار می‌برد، توسط الگوریتم یادگیری که استفاده شده است، تعیین می‌گردد. در بخش‌های بعدی جزئیات بیشتری در مورد اجزای SLA مطرح خواهیم نمود.

2-3-2- تاریخچه‌ی اتوماتای یادگیر

در دهه‌ی 1960 میلادی، سیپکین⁵¹ روشی برای ساده کردن مسائل فوق به مسأله‌ای برای تشخیص پارامترهای بهینه و به کار بردن روش‌های تپه نوردی برای حل آن معرفی کرد [37]. ستلین و همکارانش [38] در همان زمان، کار بر روی اتوماتای یادگیر را آغاز کردند. مفهوم اتوماتای یادگیر، نخستین بار توسط وی مطرح شد. وی به مدل‌سازی رفتارهای سیستم‌های بیولوژیکی علاقمند بود و یک اتوماتای قطعی را که در محیطی تصادفی فعالیت می‌کرد، به عنوان مدلی برای یادگیری، معرفی نمود. در تحقیقات انجام شده‌ی بعدی، استفاده از یادگیری در سیستم‌های مهندسی نیز در نظر گرفته شد. رویکرد دیگری که توسط نارندرا⁵² و ویسواناتان⁵³ ارائه شد [39]، در نظر گرفتن مسأله به صورت پیدا کردن عمل بهینه از میان مجموعه‌ای از عمل‌های مجاز برای یک اتوماتای تصادفی بود. تفاوت دو روش اخیر در این است که در روش اول در هر تکرار، فضای پارامترها بروز می‌شود، اما در روش دوم، فضای احتمال بروز می‌گردد. پس از آن، اکثر کارهای انجام شده در تئوری اتوماتای یادگیر، در ادامه‌ی مسیر معرفی شده، توسط ستلین انجام شدند. وارشاوسکی⁵⁴ و وورونتسوا⁵⁵ [40] اتوماتای یادگیر با ساختار متغیر

⁴⁹ Learning Automata (LA)

⁵⁰ Action

⁵¹ Y.Z. Tsyppkin

⁵² Narendra

⁵³ Viswanatan

⁵⁴ Varshavski

⁵⁵ Vorontsova

را مطرح کردند که احتمالات عمل‌های⁵⁶ خود را بروز می‌کرد و در نتیجه باعث کاهش تعداد حالات⁵⁷، نسبت به اتوماتای قطعی می‌شد.

نخستین تلاش‌ها برای استفاده از اتوماتای یادگیر در کاربردهای کنترلی، توسط فو⁵⁸ و همکارانش انجام شد [41]. از جمله این کارها می‌توان به کاربردهایی از اتوماتای یادگیر در تخمین پارامترها⁵⁹، تشخیص الگو⁶⁰ و تئوری بازی⁶¹ اشاره کرد [42-44]. مک لارن⁶² روش‌های بیروزرسانی خطی و ویژگی‌های آنها را بررسی نمود [45] و پس از آن چاندراسیکار⁶³ و شن⁶⁴ روش‌های بیروزرسانی غیرخطی را بررسی کردند [46]. کتاب نارندرا و تاتاچار⁶⁵ تحت عنوان اتوماتای یادگیر، مقدمه‌ای به تئوری اتوماتا است که کلیه تحقیقات انجام شده تا اواخر دهه 1980 را در بر می‌گیرد [47] مثال‌ها و کاربردهای متعدد دیگری از اتوماتای یادگیر نیز توسط نجیم⁶⁶ و پوژنیاک⁶⁷ در کتابی با عنوان "اتوماتای یادگیر: تئوری و کاربردها" ⁶⁸ ارائه شده است [48].

نتایج این تحقیقات، به استفاده از اتوماتا به عنوان مدلی از یادگیری با کاربردهایی در مسیریابی شبکه‌های تلفن، تشخیص الگو و کنترل تطبیق‌پذیر منجر شد [49-52]. علاوه بر این، از کاربردهای اخیر اتوماتای یادگیر در مسائل واقعی می‌توان به تشخیص الگو [53] جداسازی گراف [54]⁶⁹ و برنامه‌ریزی مسیر [55]⁷⁰ اشاره کرد. دیگر کاربردهای اتوماتای یادگیر در [56-60] ارائه شده اند.

یک اتوماتای یادگیر را می‌توان به صورت یک شیء مجرد که دارای تعداد متناهی عمل است، در نظر گرفت. نحوه کار یک اتوماتای یادگیر بدین صورت است که یک عمل از مجموعه عمل‌های

⁵⁶ Action Probability

⁵⁷ States

⁵⁸ Fu

⁵⁹ Parameter Estimation

⁶⁰ Pattern Recognition

⁶¹ Game Theory

⁶² McLaren

⁶³ Chandrasekar

⁶⁴ Shen

⁶⁵ Thathachar

⁶⁶ Najim

⁶⁷ Pznyak

⁶⁸ Learning Automata: Theory and Application

⁶⁹ Graph Partitioning

⁷⁰ Path Planning

خود را انتخاب کرده و آن را به محیط اعمال می‌کند. سپس عمل مذکور توسط یک محیط تصادفی ارزیابی می‌شود و اتوماتا از پاسخ محیط برای انتخاب عمل بعدی خود استفاده می‌کند. در طی این فرآیند، اتوماتا یاد می‌گیرد که چگونه عمل بهینه را انتخاب نماید. نحوه استفاده از پاسخ محیط به عمل انتخابی اتوماتا که در جهت انتخاب عمل بعدی اتوماتا استفاده می‌شود، توسط الگوریتم یادگیری اتوماتا مشخص می‌گردد. در بخش بعد جزئیات قسمت‌های یک اتوماتای با ساختار متغیر⁷¹ معرفی می‌شود.

3-3-2 اتوماتای یادگیر تصادفی (SLA)

هر SLA از دو جزء عمده تشکیل شده است.

1. یک اتوماتای تصادفی که دارای تعداد محدودی عمل می‌باشد و با یک محیط تصادفی در حال تعامل می‌باشد.
2. الگوریتم یادگیری که با استفاده از آن، اتوماتا عمل بهینه را یاد می‌گیرد.

2-3-3-1 اتوماتای تصادفی

هر اتوماتا را می‌توان یک ماشین حالت متناهی⁷² در نظر گرفت که به وسیله پنج‌تایی زیر قابل نمایش است:

$$SA = \{\alpha, \beta, F, G, \phi\}$$

رابطه (2-12)

- پارامترهای موجود در این پنج‌تایی عبارتند از:
- مجموعه عمل‌های اتوماتا $\alpha \equiv \{\alpha_1, \alpha_2, \dots, \alpha_r\}$
- مجموعه ورودی‌های اتوماتا $\beta \equiv \{\beta_1, \beta_2, \dots, \beta_r\}$
- تابعی که ورودی و حالت جاری را به حالت بعدی نگاشت می‌کند $F \equiv \phi \times \beta \rightarrow \phi$
- تابع خروجی که حالت جاری را به خروجی بعدی نگاشت می‌کند $G \equiv \phi \rightarrow \alpha$
- مجموعه حالت‌های داخلی اتوماتا در لحظه n -ام $\phi(n) \equiv \{\phi_1, \phi_2, \dots, \phi_r\}$

مجموعه α شامل اعمال اتوماتا می‌باشد که اتوماتا در هر بار تکرار، یکی از آنها را انتخاب می‌کند. مجموعه ورودی β ،

⁷¹ Variable Structure Learning Automata

⁷² Finite State Machine (FSM)

ورودی‌های اتوماتا را تعریف می‌کند که جزئیات بیشتر مربوط به آن را در بخش بعدی بیان خواهیم کرد. نگاشته‌های F و G حالت جاری و ورودی فعلی را به خروجی (عمل) بعدی که توسط اتوماتا انتخاب شده است تبدیل می‌کنند. حال اگر نگاشته‌های F و G قطعی باشند، اتوماتا نیز اتوماتای قطعی⁷³ گفته می‌شود. در چنین حالتی با در دست داشتن حالت اولیه و ورودی، خروجی و حالت بعدی به صورت منحصر به فردی بدست می‌آید. حال اگر نگاشته‌های F و G تصادفی باشند، اتوماتا نیز اتوماتای تصادفی گفته می‌شود. در چنین حالتی فقط می‌توان احتمالات مربوط به حالت بعدی و نیز خروجی‌های مربوطه را مشخص نمود. اتوماتای تصادفی خود به دو دسته تقسیم می‌شود:

1. اتوماتای با ساختار ثابت⁷⁴

2. اتوماتای با ساختار متغیر⁷⁵

در نوع اول، احتمالات مربوط به عمل‌های مختلف ثابت هستند؛ اما در نوع دوم احتمالات در هر بار تکرار بهنگام می‌شوند. تعریف و انواع مختلف اتوماتای احتمالی با ساختار ثابت و متغیر در ادامه آمده است.

2-3-3-2- محیط

محیط احتمالی را می‌توان به صورت ریاضی با سه‌تایی E بیان نمود:

$$E \equiv \{\alpha, \beta, c\}$$

$$\alpha \equiv \{\alpha_1, \alpha_2, \dots, \alpha_r\}$$

$$\beta \equiv \{\beta_1, \beta_2, \dots, \beta_r\}$$

$$c \equiv \{c_1, c_2, \dots, c_r\}$$

رابطه‌ی (2-13)

α مجموعه‌ی ورودی‌های محیط و β مجموعه‌ی خروجی‌های آن می‌باشد. مدل‌های مختلفی برای محیط‌های احتمالی تعریف شده است. در P-Model، محیط مقادیر صفر و یک را به عنوان خروجی اختیار می‌کند. صفر به معنی پاداش⁷⁶ و یک به معنی جریمه⁷⁷ می‌باشد. در Q-Model، مقادیر خروجی به صورت اعداد گسسته‌ی بین صفر و یک هستند و

⁷³ Deterministic Automata

⁷⁴ Fixed Structure

⁷⁵ Variable Structure

⁷⁶ Reward

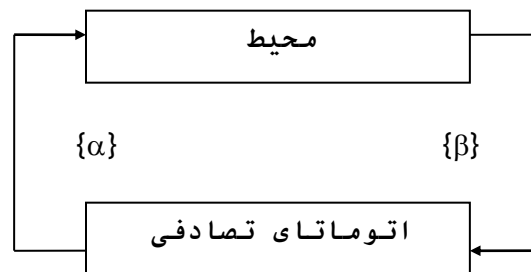
⁷⁷ Penalty

در S-Model، خروجی محیط، مقدار پیوسته‌ای بین صفر و یک می‌باشد. C نیز مجموعه‌ی احتمال جریمه‌شدن عمل‌های اتوماتای احتمالی است که به صورت زیر تعریف می‌شود:

$$c_i = \text{Prob}[\beta(i)=1 / \alpha(i)=\alpha_i], \quad i=1,2,\dots,r \quad \text{رابطه‌ی (2-14)}$$

c_i یعنی احتمال آنکه عمل α_i نتیجه‌ی نامطلوب داشته باشد. در محیط‌های پایدار⁷⁸ مقادیر c_i به صورت ثابت باقی می‌ماند؛ در حالی که در محیط‌های ناپایدار⁷⁹ این مقادیر در طی زمان تغییر می‌کنند. c_i ها معمولاً مشخص نیستند و دانستن آنها به معنی شناخت کامل محیط احتمالی است که در اکثر کاربردها امکان ندارد و هدف اتوماتای یادگیر، مشخص کردن این مقادیر است.

رابطه‌ی بین محیط احتمالی و یک اتوماتای احتمالی در شکل 2-15 نشان داده شده است.



شکل 2-15: رابطه‌ی بین اتوماتا و محیط احتمالی

بنابراین در نهایت SLA را می‌توان توسط پنج‌تایی زیر نمایش داد.

$$SLA = \{\alpha, \beta, P, T, C\} \quad \text{رابطه‌ی (2-15)}$$

که در آن داریم:

- مجموعه‌ی خروجی‌های اتوماتا / مجموعه‌ی ورودی‌های محیط
- $$\alpha \equiv \{\alpha_1, \alpha_2, \dots, \alpha_r\} \equiv$$

⁷⁸ Stationary

⁷⁹ Non-Stationary

- مجموعه‌ی ورودی‌های اتوماتا / مجموعه‌ی خروجی‌های محیط

$$\beta \equiv \{\beta_1, \beta_2, \dots, \beta_r\} \equiv$$
- بردار احتمالات

$$P \equiv \{p_1, p_2, K, p_r\} \equiv$$
- الگوریتم یادگیر $T \equiv P(n-1) = T[\alpha(n), \beta(n), p(n)] \equiv$
- مجموعه‌ی احتمالات جریمه‌ی تعریف شده برای محیط

$$C \equiv \{c_1, c_2, \dots, c_r\} \equiv$$

همان طور که قبلاً بیان شد، جهت سادگی ریاضی، هر یک از حالت‌های داخلی اتوماتا را بر یک عمل منحصر به فرد از اتوماتا منطبق می‌نمایند.

2-3-4- الگوریتم‌های یادگیری

2-3-4-1 الگوریتم‌های یادگیری استاندارد

یک الگوریتم یادگیری را می‌توان به صورت زیر بیان نمود:
رابطه‌ی (2-16)

$$P(n+1) = T[P(n), \alpha(n), \beta(n)]$$

اگر عملگر T خطی باشد، الگوریتم یادگیری خطی و در غیر این صورت غیرخطی خوانده می‌شود. می‌توان گفت که ایده‌ی اصلی تمام الگوریتم‌های یادگیری به صورت زیر است: اگر SLA در تکرار n -ام عمل α_i را انتخاب نماید و پاسخ مطلوبی برای آن از محیط دریافت نماید، در آن صورت احتمال $P_i(n)$ مربوط به عمل افزایش یافته و احتمالات مربوط به سایر اعمال کاهش می‌یابد. در صورت دریافت جواب نامطلوب، $P_i(n)$ کاهش یافته و احتمالات مربوط به سایر اعمال افزایش می‌یابد. بنابراین داریم:

برای پاسخ مطلوب⁸⁰:

$$P_i(n+1) = P_i(n) + \sum_{j=1, j \neq i}^r f_j[P_j(n)]$$

رابطه‌ی (2-17)

$$P_j(n+1) = P_j(n) - f_j[P_j(n)]; \forall j; j \neq i$$

برای پاسخ نامطلوب⁸¹:

⁸⁰ Favorable

⁸¹ Unfavorable

$$P_j(n+1) = P_j(n) - g_j[P_j(n)]; \forall j; j \neq i$$

$$P_i(n+1) = P_i(n) + \sum_{j=i}^r g_j[P_j(n)] \quad \text{رابطه ی (2-18)}$$

توابع f_i و g_j توابعی غیر منفی می‌باشند که به ترتیب توابع پاداش و جریمه نامیده می‌شود. هم اکنون مطالعات بسیاری بر روی الگوریتم‌های یادگیر خطی به علت قابلیت انعطاف بالای آنها انجام می‌شود. برای این نوع از الگوریتم‌ها، توابع پاداش و جریمه به صورت زیر تعریف می‌شوند [61]:

$$f_j[p_j(n)] = ap_j(n) \quad 0 < a < 1 \quad \text{رابطه ی (2-19)}$$

$$g_j[p_j(n)] = \frac{b}{r-1} - bp_j(n) \quad 0 \leq b < 1 \quad \text{رابطه ی (2-20)}$$

در روابط فوق داریم:
 $r \equiv$ تعداد اعمال یک اتوماتا
 $a \equiv$ پارامتر پاداش
 $b \equiv$ پارامتر جریمه

بر اساس مقادیر مختلفی که برای پارامترهای یادگیری a و b در نظر گرفته می‌شود، الگوریتم‌های مختلفی بدست می‌آید که در زیر مواردی از آنها را نام می‌بریم:

روش L_{RP} : اگر a و b مساوی باشند، روش جریمه‌ی پاداش خطی⁸² حاصل می‌شود.

روش L_{ReP} : اگر b چندین برابر از a کوچکتر باشد، روش یادگیری حاصل، روش جریمه‌ی جزئی پاداش خطی⁸³ خواهد بود.

روش L_{RI} : اگر پارامتر جریمه (b) صفر باشد، روش حاصل، روش عدم پاداش خطی⁸⁴ نامیده می‌شود.

⁸² Linear Reward Penalty (LRP) Scheme

⁸³ Linear Reward Penalty Epsilon Penalty (L_{ReP}) Scheme

⁸⁴ Linear Reward Inaction (L_{RI}) Scheme

اگر رابطه‌ی (19-2) و رابطه‌ی (20-2) برای توابع f_i و g_j به کار برده شوند، شکل عمومی الگوریتم یادگیری به صورت زیر خواهد بود:

اگر در تکرار n عمل α انتخاب شود، در این صورت در تکرار $(n+1)$ داریم:

دریافت پاسخ مطلوب از محیط:

$$p_i(n+1) = p_i(n) + a[1 - p_i(n)] \quad \text{رابطه‌ی (21-2)}$$

$$p_j(n+1) = p(1-a)p_j(n); \forall j; j \neq i \quad \text{رابطه‌ی (22-2)}$$

دریافت پاسخ نامطلوب از محیط:

$$p_i(n+1) = (1-b)p_i(n) \quad \text{رابطه‌ی (23-2)}$$

$$p_j(n+1) = \frac{b}{1-r} = (1-b)p_j(n); \forall j; j \neq i \quad \text{رابطه‌ی (24-2)}$$

اگر در معادلات فوق a و b برابر باشند، روش L_{RP} بدست می‌آید و اگر $b = 0$ باشد، روش L_{RI} حاصل می‌گردد و اگر $b \ll a$ اتوماتای یادگیر $L_{R\&P}$ حاصل می‌شود. لازم است در اینجا یادآور شویم که روش‌های بهنگام‌سازی غیرخطی نیز توسط محققین دنبال می‌شود؛ اما استفاده از این روش‌ها منجر به افزایش قابل ملاحظه‌ای در کارایی نسبت به روش‌های خطی نشده است. مهم‌ترین عاملی که منجر به محدودیت کاربرد SLA می‌شود، نرخ کند همگرایی آن می‌باشد. این امر به خصوص زمانی که تعداد عمل‌ها زیاد باشد و SLA در هر بار تکرار باید تعداد اعمال زیادی را بهنگام نماید، بیشتر احساس می‌شود. با توجه به این مشکل به بیان الگوریتم‌های غیرخطی نخواهیم پرداخت.

2-4-3-2 الگوریتم‌های یادگیری مدل S

در محیط‌های S پاسخ محیط به عمل‌های اتوماتای یادگیر، یک متغیر تصادفی در بازه $[1,0]$ می‌باشد. بنابراین خروجی محیط (ورودی اتوماتا) بصورت زیر تغییر می‌کند:

رابطه ی (25-2)

$$\beta(n) \equiv \{\beta_1, \beta_2, \dots, \beta_r\} \equiv \beta_i \in [0,1]; \quad \forall i$$

از آنجاییکه پاسخ محیط از نوع S یک متغیر تصادفی در بازه [1,0] است، استفاده از مدل S برای سیستمهای یادگیر نیاز به اطلاعات اولیه ای از کران بالا و پایین شاخصهای کارایی سیستم دارد تا بتواند پاسخهای محیط را در مقیاس [1,0] تنظیم نماید. کارایی مناسب مدل S در [62] نشان داده شده است. در [63] یک الگوریتم غیر خطی اما بهینه برای یک اتوماتا با دو عمل در محیط مدل S معرفی شده است. یک روش زیربهینه برای یک اتوماتا با چند عمل در محیط مدل S نیز در [64] معرفی شده است.

1-2-4-3-2 الگوریتم $S-L_{RI}$

در محیط مدل P محیط توسط احتمال جریمه ها تعریف می شود. برای هر عمل مانند α_i ، محیط پاسخی با یک مقدار تصادفی $[\beta_i(n) | \alpha_i]$ به اتوماتا می دهد که ورودی اتوماتا را تشکیل می دهد. در مدل P، پاسخ $\beta_i(n)$ با احتمال c_i برابر 1 (پاسخ نامطلوب) و با احتمال c_i -1 صفر (پاسخ مطلوب) در نظر گرفته می شود. اما در محیط مدل S، محیط بصورت زیر تعریف می شود:

$$E \equiv \{\alpha, \beta, s\}$$

رابطه ی (26-2)

$$s(n) \equiv \{s_1, s_2, \dots, s_r\}; \quad s_i = E\{\beta_i(n) | \alpha_i\}; \quad \forall i$$

s_i مقدار میانگین پاسخ β_i برای عمل α_i می باشد و در واقع s_i ها بعنوان شدت جریمه ها محسوب می شوند. فرض کنید در گام n ام عمل α_i انتخاب شده باشد و پاسخ محیط به آن $\beta_i(n)$ بوده باشد. در چنین حالتی احتمال عملهای اتوماتای $S-L_{RI}$ بصورت زیر بروز می شوند [47]:

$$p_i(n+1) = p_i(n) + a(1 - \beta_i(n))(1 - p_i(n))$$

رابطه ی (27-2)

$$p_j(n+1) = p_j(n) - a(1 - \beta_i(n))p_j(n) \quad \forall j, \quad j \neq i$$

که a پارامتر یادگیری و $0 < a < 1$ می باشد.

2-2-4-3-2 الگوریتم $S-L_{RP}$

الگوریتم یادگیری تقویتی اتوماتای یادگیر در مدل $S-L_{RP}$ بصورت زیر بردار احتمالات عملهای اتوماتا را بروز می کند [47]. در اتوماتایی با r عمل، اگر در تکرار n ام عمل α_i انتخاب شده باشد

و پاسخ محیط به آن $\beta_i(n)$ باشد، بردار احتمالات اتوماتا طبق رابطه زیر بروز می‌شود.

$$\begin{aligned} p_i(n+1) &= p_i(n) + a(1-\beta_i(n))(1-p_i(n)) - a\beta_i(n)p_i(n) \\ p_j(n+1) &= p_j(n) - a(1-\beta_i(n))p_j(n) + a\beta_i(n)\left[\frac{1}{r-1} - p_j(n)\right] \quad \forall j \neq i \end{aligned}$$

رابطه -
ی (28-2)

3-2-4-3-2 الگوریتم $S-L_{ReP}$

اتوماتای یادگیری $S-L_{ReP}$ با r عمل و پارامتر پاداش a و پارامتر جریمه b ، بصورت زیر بردار عملهای خود را بروز می‌کند. اگر در تکرار n عمل α_i انتخاب شده باشد و پاسخ محیط به آن $\beta_i(n)$ باشد، بردار احتمالات اتوماتا طبق رابطه زیر بروز می‌شود.

$$\begin{aligned} p_i(n+1) &= p_i(n) + a(1-\beta_i(n))(1-p_i(n)) - b\beta_i(n)p_i(n) \\ p_j(n+1) &= p_j(n) - a(1-\beta_i(n))p_j(n) + b\beta_i(n)\left[\frac{1}{r-1} - p_j(n)\right] \quad \forall j \neq i \end{aligned}$$

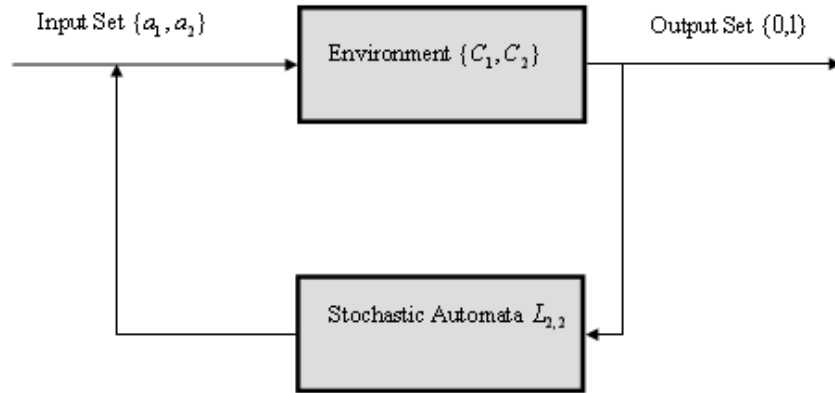
رابطه -
ی (29-2)

3-2-5- الگوریتم‌های یادگیری با ساختار ثابت

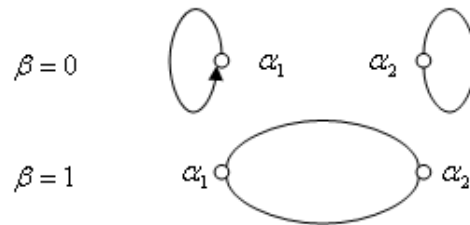
الگوریتم‌های یادگیری را می‌توان برحسب نوع بردار احتمالات نیز تقسیم‌بندی نمود. بر اساس این نوع از تقسیم‌بندی، اگر بردار احتمالات در طول مدت زمان اجرا ثابت باشد، اتوماتا از نوع ساختار ثابت خواهد بود و اگر این بردار در طول زمان اجرا تغییر نماید، اتوماتای با ساختار متغیر حاصل خواهد شد. در هر دو نوع اتوماتای ذکر شده، احتمالات در بازه‌ی $[0,1]$ قرار دارند. الگوریتم‌های یادگیری که پیش از این در مورد آن بحث گردید، مربوط به اتوماتای از نوع ساختار متغیر می‌باشند. در ادامه به بررسی انواع دیگری از اتوماتای یادگیر با ساختار ثابت می‌پردازیم و تعدادی از بهترین ایده‌های معرفی شده در سال 1960 را معرفی می‌نماییم.

3-2-5-1 اتوماتای دو حالت $L_{2,2}$

این اتوماتا دارای دو حالت ϕ_1 و ϕ_2 و دو خروجی α_1 و α_2 می‌باشد (شکل 2-16). ورودی‌های پذیرفته شده توسط اتوماتا مجموعه‌ی $\{0,1\}$ می‌باشد. اتوماتا هرگاه ورودی 1 را دریافت نماید (پاسخ نامطلوب) حالت خود را عوض می‌کند و در صورت دریافت ورودی 0 (پاسخ مطلوب) در حالت قبلی باقی می‌ماند.



شکل 2-16: نمودار عملکرد اتوماتای دو حالت $L_{2,2}$



شکل 2-17: گراف انتقال حالت اتوماتای $L_{2,2}$

در نتیجه، ماتریس انتقال⁸⁵ این اتوماتا به صورت زیر می‌باشد که گراف مرتبط با آن نیز در شکل 2-17 نمایش داده شده است. هر اتوماتایی که منطبق بر استراتژی فوق باشد $L_{2,2}$ نامیده می‌شود. در اینجا اندیس اول، تعداد حالت‌ها و اندیس دوم نیز تعداد اعمال را مشخص می‌نماید. خصوصیات محیط، توسط مجموعه‌ی احتمالات جری‌مه $\{C_1, C_2\}$ بیان می‌گردد. در اینجا هر $C_i (i=1,2)$ بیانگر احتمال دریافت پاسخ $\beta=1$ برای ورودی α_i است.

$$F(0) = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \quad F(1) = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \quad \text{رابطه‌ی (2-30)}$$

استراتژی اتوماتا به شکلی طراحی شده است که تا زمان دریافت پاسخ مطلوب از محیط، در یک عمل باقی می‌ماند و به محض آن‌که پاسخ نامطلوبی دریافت شود به سوی عمل دیگر حرکت می‌نماید.

⁸⁵ Transition matrix

2-5-3-2- توسعه‌های اتوماتای دو حالتی $L_{2,2}$

از آنجایی که تمامی سطرهای $F(i) (i=1,2)$ فقط دارای مقدار صفر و یا یک هستند و در هر سطر نیز یک عنصر منفرد وجود دارد ماتریس انتقال موجود در رابطه‌ی (2-30) قطعی می‌باشد. حال این سوال مطرح می‌شود که آیا همگرای اتوماتا را می‌توان با تصادفی نمودن ماتریس‌های انتقال به صورت زیر بهبود بخشید؟

$$F(0) = \begin{bmatrix} 1-\gamma_1 & \gamma_1 \\ \gamma_1 & 1-\gamma_1 \end{bmatrix} \quad F(1) = \begin{bmatrix} \gamma_2 & 1-\gamma_2 \\ 1-\gamma_2 & \gamma_2 \end{bmatrix}$$

رابطه‌ی (2-31)

$$0 < \gamma_i < 1 \quad (i=1,2)$$

در این مورد هنگامی که یک عمل منجر به پاسخ مطلوب گردد، اتوماتا از همان عمل با احتمال $(1-\gamma_1)$ استفاده نموده و به عمل دیگر با احتمال γ_1 سوئیچ می‌نماید. همچنین هنگامی که خروجی محیط نامطلوب باشد، این عمل را با احتمالات γ_2 و $(1-\gamma_2)$ انجام می‌دهد و می‌توان اثبات نمود که این نوع از اتوماتا، قطعی می‌باشد.

2-5-3-3- اتوماتای حافظه‌دار با دو عمل L_{2N2}

اتوماتای $L_{2,2}$ عموماً نمی‌تواند بهینه باشد. به همین دلیل، تستلین نوع تغییر یافته‌ای از اتوماتای $L_{2,2}$ را پیشنهاد نموده است که اگر احتمالات جریمه‌ی محیط دارای شرایط خاص باشند، می‌تواند بهینه گردد. چنین اتوماتایی L_{2N2} نامیده می‌شود و دارای $2N$ حالت و دو عمل می‌باشد. این نوع از اتوماتا سعی می‌کند که رفتار گذشته‌ی سیستم را جهت تصمیم‌گیری در مورد چگونگی انتقال از اعمال، مورد استفاده قرار دهد. اتوماتای $L_{2,2}$ هنگامی که یک جواب نامطلوب از محیط دریافت نماید، از عملی به عمل دیگر سوئیچ می‌نماید؛ در حالی که اتوماتای L_{2N2} اقدام به نگهداری تعداد موفقیت‌ها⁸⁶ و شکست‌های⁸⁷ دریافت شده به ازای هر عمل می‌نماید و فقط زمانی که تعداد شکست‌ها بیش از تعداد موفقیت‌ها گردد (یا از یک مقدار حداکثر N تجاوز نماید) به عمل دیگر سوئیچ می‌نماید.

⁸⁶ Successes

⁸⁷ Failures

این کار با توسعه دادن فضای حالت به $2N$ و تعیین قواعد انتقال از یک حالت به حالت دیگر صورت می‌پذیرد.

اتوماتای L_{2N2} دارای $2N$ حالت $\varphi_1, \varphi_2, \dots, \varphi_{2N}$ و دو عمل α_1 و α_2 می‌باشد.

حالت‌های $\varphi_1, \varphi_2, \dots, \varphi_N$ مربوط به عمل α_1 و حالات $\varphi_{N+1}, \dots, \varphi_{2N}$ نیز متعلق به عمل α_2 می‌باشند.

بنابراین:

$$G[\varphi_i] = \begin{cases} \alpha_1 & i = 1, 2, \dots, N \\ \alpha_2 & i = N+1, \dots, 2N \end{cases} \quad \text{رابطه ی (2-32)}$$

که در آن G تابع نگاشت خروجی می‌باشد.

اگر اتوماتا در یک حالت φ_i ($1 \leq i \leq N$) باشد، عمل α_1 را انجام می‌دهد. حال اگر این عمل منجر به پاسخی نامطلوب شود، حالت به صورت زیر تغییر می‌کند:

$$\varphi_i \rightarrow \varphi_{i-1} \quad (i = 1, 2, \dots, N-1) \quad \text{رابطه ی (2-33)}$$

و

$$\varphi_N \rightarrow \varphi_{2N} \quad \text{رابطه ی (2-34)}$$

به صورت مشابه اگر عمل انتخابی منجر به پاسخ مطلوب شود، انتقال حالت‌ها به صورت زیر صورت می‌گیرد:

$$\varphi_i \rightarrow \varphi_{i-1} \quad (i = 2, 3, \dots, N) \quad \text{رابطه ی (2-35)}$$

و

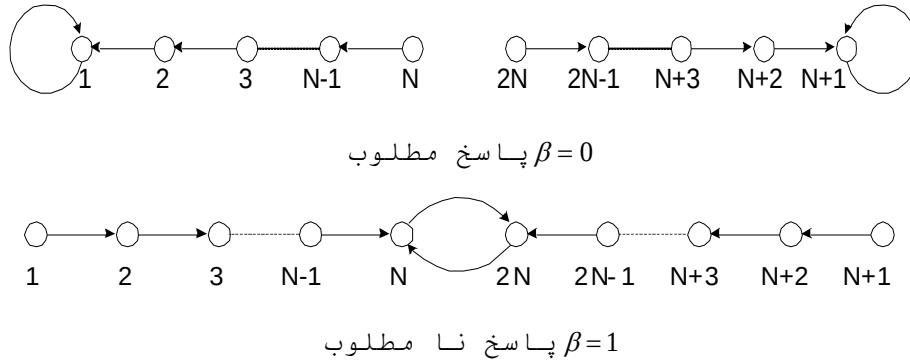
$$\varphi_1 \rightarrow \varphi_1 \quad \text{رابطه ی (2-36)}$$

به شکلی مشابه می‌توان انتقال حالت را برای φ_1 هنگامی که $N+1 \leq i \leq 2N$ و عمل α_2 در حال انجام است به شکل زیر بیان نمود:

$$\left. \begin{aligned} \varphi_i &\rightarrow \varphi_{i+1} & (i = N+1, \dots, 2N-1) \\ \varphi_{2N} &\rightarrow \varphi_N \end{aligned} \right\} \alpha_2 \text{ منجر به یک پاسخ نامطلوب شده است.}$$

$$\left. \begin{aligned} \varphi_i &\rightarrow \varphi_{i-1} & (i = N+2, \dots, 2N) \\ \varphi_{N+1} &\rightarrow \varphi_{N+1} \end{aligned} \right\} \alpha_2 \text{ منجر به یک پاسخ مطلوب شده است.}$$

گراف‌های انتقال حالت در شکل 18-2 نمایش داده شده‌اند و ماتریس‌های انتقال آنها نیز توسط $F(0)$ و $F(1)$ بیان شده است.

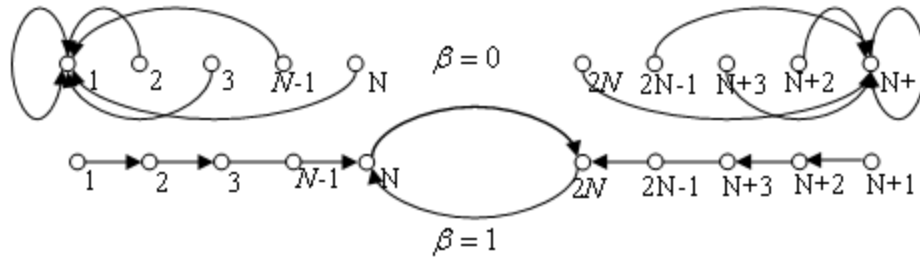


شکل 18-2: گراف‌های انتقال حالت اتوماتای L_{2N2}

هر عمل با N حافظه مرتبط می‌باشد و در مجموع، اتوماتا دارای $2N$ حافظه است. هنگام دریافت هر پاسخ مطلوب، حالت اتوماتا به حالت عمیق‌تری در حافظه تغییر کرده و اگر پاسخ نامطلوبی دریافت گردد، حالت آن به عمق کمتری در حافظه‌ی اتوماتا تغییر می‌یابد. عمق یک حالت φ_a در ارتباط با عمل α را می‌توان به صورت قراردادی به شکل زیر تعریف نمود: حداقل طولی از سیگنال‌های ورودی که φ_a را از مجموعه‌ی حالت‌های متعلق به عمل α ($\varphi_a = \{\varphi_\alpha\}$) بیرون می‌آورد. همچنین عمق یک اتوماتا عبارت است از بزرگترین عمق حالت‌های آن که در این مثال N حالت می‌باشد.

4-5-3-2- اتوماتای کرینسکی (K_1)

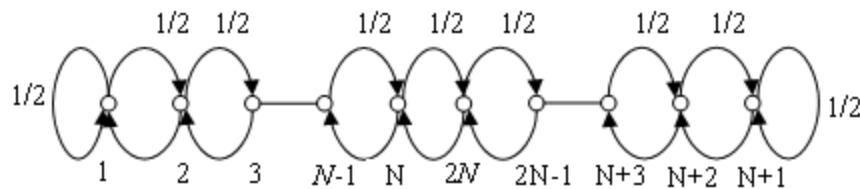
نمودار انتقال حالت نمایش داده شده در شکل 19-2 نشان می‌دهد که رفتار اتوماتای کرینسکی برای پاسخ نامطلوب از محیط همانند L_{2N2} می‌باشد. به هنگام دریافت پاسخ مطلوب، هر حالت φ_i ($i=1,2,\dots,N$) به حالت φ_1 رفته و حالت φ_i ($i=N+1,N+2,\dots,2N$) نیز به حالت φ_{N+1} می‌رود. در مجموع، برای رفتن از یک عمل به عمل دیگر نیاز به دریافت N پاسخ نامطلوب متوالی دارد.



شکل 19-2: گرافهای انتقال حالت اتوماتای کرینسکی

2-3-5-5- اتوماتای کرایلوف (K_2)

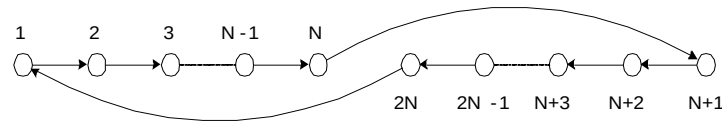
این اتوماتا برای پاسخ مطلوب محیط، نمودار تغییر حالتی مانند اتوماتای $L_{2N,2}$ دارد. زمانی که پاسخ محیط نامطلوب است، اتوماتا با احتمال 0.5 از حالت $\forall \varphi_i$ ($i=1, \dots, N-1, N, \dots, 2N$) به حالت‌های φ_{i+1} یا φ_{i-1} می‌رود (شکل 20-2). همچنین زمانی که $i=1$ یا $i=N-1$ باشد، حالت بعدی اتوماتا، با احتمال 0.5، همان حالت یا حالت φ_{i+1} خواهد بود. وقتی که $i=N$ ، با احتمال 0.5 حالت بعدی، φ_{N-1} یا φ_{2N} می‌باشد و هنگامی که $i=2N$ ، با احتمال 0.5 به φ_N یا φ_{2N+1} تغییر حالت می‌دهد.



شکل 20-2: انتقال حالات اتوماتای کرایلوف هنگام دریافت پاسخ نامطلوب از محیط

2-3-5-6- اتوماتای $G_{2N,2}$

در این اتوماتا بر خلاف $L_{2N,2}$ ، عمل حداقل N بار انجام می‌گردد (پس از گرفتن N جریمه) تا اینکه در نهایت عمل α_1 دوباره انتخاب شود. گراف تغییر وضعیت این اتوماتا برای پاسخ مطلوب مانند اتوماتای $L_{2N,2}$ بوده و برای پاسخ نامطلوب مطابق شکل 21-2 می‌باشد.



$G_{2N,2}$ پاسخ نا $\beta=1$
مطلوب

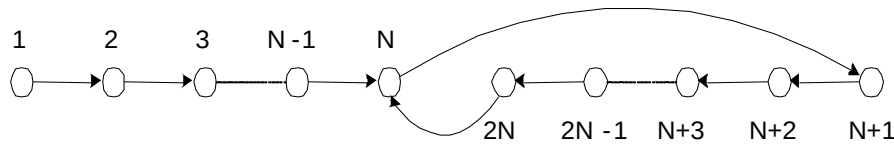
تغییر وضعیت

شکل 21-2: نمودار

اتوماتا

7-5-3-2- اتوماتای مهاجرت اشیاء⁸⁸

نمودار تغییر وضعیت در این اتوماتا، برای پاسخ مطلوب مانند اتوماتای $I_{2N,2}$ بوده و برای پاسخ نامطلوب مطابق شکل 22-2 می‌باشد.



$\beta=1$ پاسخ نا مطلوب

شکل 22-2: نمودار تغییر وضعیت اتوماتای مهاجرت اشیاء

2-3-6- بازی‌های اتوماتا

در مسائل مختلف اجتماعی و اقتصادی که تداخل علایق در بین عوامل تصمیم‌گیری نقش مهمی را در تصمیم‌گیری نهایی بازی می‌کند، تئوری بازی‌ها نقشی بسیار عمده ایفا می‌نماید. در زمانی که هر دو طرف، عملی را انتخاب می‌کنند و به ازای آن جوابی از محیط دریافت می‌دارند، می‌گوییم که یک بازی در حال انجام است. هر یک از بازیگران می‌تواند اطلاعات کاملی از طرف مقابل و گزینه‌های او داشته یا نداشته باشد و هر بازیگر، حرکت بعدی خود را بر پایه‌ی جواب بدست آمده از محیط استوار می‌سازد.

در یک بازی اتوماتا، تعدادی اتوماتا در یک محیط، بدون آنکه دانش کاملی از دیگران در اختیار داشته باشند عمل می‌کنند. هر یک از آنها می‌تواند دارای تعداد محدودی اعمال و قواعد متفاوت با دیگران باشد. برای بازی‌های اتوماتا یک قاعده‌ی عمومی به صورت

⁸⁸ Object Migration Automata (OMA)

زیر می‌توان بیان نمود: اگر N اتوماتا $\{A^1, A^2, \dots, A^N\}$ در بازی داشته باشیم و هر اتوماتای دلخواه A_j به و سیله‌ی پنج‌تایی $\{\alpha^j, \beta^j, \phi^j, F^j, G^j\}$ توصیف گردد که در آنها تعاریف زیر را داشته باشیم:

- مجموعه‌ی اعمال $\alpha^j = \{\alpha_1^j, \alpha_2^j, \dots, \alpha_r^j\}$
- مجموعه‌ی ورودی‌ها $\beta^j = \{\beta_1^j, \beta_2^j, \dots, \beta_r^j\}$
- مجموعه‌ی حالت‌های داخلی $\phi^j = \{\phi_1^j, \phi_2^j, \dots, \phi_r^j\}$
- قواعد بهنگام‌سازی اتوماتا F^j, G^j

در آن صورت یک بازی اتوماتا $\alpha(n)$ به عنوان مجموعه‌ای از استراتژی‌های انتخاب شده توسط اتوماتا در تکرار n -ام تعریف شده و به صورت زیر بیان می‌گردد:

$$\alpha(n) = \{\alpha^1(n), \alpha^2(n), \dots, \alpha^N(n)\} \quad \text{رابطه‌ی (2-37)}$$

نتیجه‌ی بازی $\alpha(n)$ نیز با نام $\beta(n)$ به صورت زیر تعریف می‌شود:

$$\beta(n) = \{\beta^1(n), \beta^2(n), \dots, \beta^N(n)\} \quad \text{رابطه‌ی (2-38)}$$

حال اگر احتمال نتیجه‌ی $\beta(n)$ وابسته به بازی $\alpha(n)$ باشد گفته می‌شود که N اتوماتا در بازی، شریک هستند.

2-3-7- خلاصه و نتیجه‌گیری

الگوریتم‌های یادگیری همواره سعی دارند با شنا سایی مشخصات محیطی که با آن در ارتباط هستند، عملکرد و خصوصیات خود را در جهت رسیدن به یک هدف خاص بهینه نمایند. اتوماتای یادگیر نیز نوعی از این الگوریتم‌ها است که سعی دارد بر حسب پاسخی که از محیط دریافت می‌کند احتمال مربوط به حالات خود را تغییر دهد و در هر حالت نیز عملکرد خاص خود را اجرا نماید. اتوماتاهای یادگیر بر حسب ساختارشان به دو دسته‌ی عمده تقسیم می‌شوند. در دسته‌ی اول، ساختار اتوماتا ثابت می‌باشد و بر حسب یک توالی خاص از پیش تعریف شده، انتقال حالات صورت می‌گیرد. این نوع از اتوماتاها را اتوماتای ساختار ثابت می‌نامند. در نوعی دیگر از اتوماتاها، هنگامی که اتوماتا در حال فعالیت است، احتمالات مربوط به هر یک از حالات دائماً در حال تغییر می‌باشد؛ لذا این نوع از اتوماتا را اتوماتای ساختار متغیر می‌نامند.

اتوماتاهایی که در این دسته قرار می‌گیرند، برحسب چگونگی تعیین احتمال جدید بر حسب احتمالات گذشته، به دو نوع خطی و غیرخطی طبقه‌بندی می‌شوند. اتوماتاهای نوع غیرخطی به علت پیچیدگی‌هایی که در عملکرد و تحلیل رفتار خود دارند، کمتر مورد استفاده‌ی عملی قرار می‌گیرند.

گرایدهای محاسباتی

2-4-1- مقدمه

گرایدهای محاسباتی⁸⁹ به عنوان یک رویکرد جدید برای حل مسائل در مقیاس بزرگ در زمینه‌های علمی، مهندسی و تجاری بوجود آمده‌اند [65-68]. آنها این قابلیت را بوجود می‌آورند که بتوان بنگاه‌های مجازی⁹⁰ (VE) را به منظور اشتراک و یکپارچه‌سازی میلیون‌ها منبع که از نظر جغرافیایی در سطح سازمان‌ها و حوزه‌های مدیریتی پراکنده‌اند، ایجاد کرد (مانند [69] SETI@Home). آنها از یک سری منابع ناهمگون (کامپیوترهای شخصی، ایستگاه‌های کاری⁹¹، کلاسترها و ابرکامپیوترها)، سیستم‌های مدیریت زیربنایی (سیستم عامل واحد، سیستم‌های صف و غیره)، سیاست‌ها و برنامه‌های کاربردی (علمی، مهندسی و تجاری) با نیازمندی‌های مختلف (پردازنده، ورودی و خروجی، حافظه و شبکه) تشکیل شده‌اند. منابع، متعلق به سازمان‌های مختلف می‌باشد که دارای سیاست‌های مدیریتی و مدل‌های استفاده و هزینه‌ای مخصوص به خود برای کاربران مختلف در زمان‌های گوناگون می‌باشند. همچنین فراهم بودن منابع و میزان بار بر روی آنها در طول زمان تغییر می‌کند.

⁸⁹ Computational Grids

⁹⁰ Virtual Enterprises (VE)

⁹¹ Workstations

محاسبات گریدی⁹² در اوایل دهه‌ی 90 در انجمن ابرمحاسبات⁹³ مطرح شد و هدف آن به کار گرفتن منابع محاسباتی موجود به شکلی آسان برای محاسبات پیچیده توسط سایت‌هایی بود که به صورت جغرافیایی توزیع شده می‌باشند. ایده‌ی اصلی‌ی گرید این است که منابع ماشین‌های شرکت‌کننده، از طریق یک لایه‌ی نرم‌افزاری به صورت شفاف⁹⁴ و مطمئن مورد استفاده قرار بگیرد. این لایه‌ی نرم‌افزاری، وظایفی همچون مجازی‌سازی⁹⁵ منابع، یافتن و جستجوی منابع، و مدیریت برنامه‌های در حال اجرا را بر عهده دارد. محاسبات گریدی، در ساده‌ترین بیان، محاسبات توزیع شده‌ای است که به سطح تکامل بالاتری رسیده است. هدف این است که تصویری از یک کامپیوتر مجازی ساده و در عین حال بزرگ و پرقدرت که توانایی مدیریت خودش را دارا می‌باشد از یک مجموعه‌ی وسیعی از کامپیوترها بوجود بیاید. این مجموعه از یک سری سیستم‌های ناهمگون متصل به هم که ترکیبات مختلفی از منابع را به اشتراک گذاشته‌اند، تشکیل شده است.

دانشمندان در اواسط دهه‌ی 90 تحقیقاتی را بر روی طراحی و پیاده‌سازی گریدهای محاسباتی آغاز کردند تا امکان انجام محاسبات موازی و توزیع شده در یک ناحیه‌ی گسترده فراهم شود. انگیزه‌ی رویکرد گریدهای محاسباتی در ابتدا وجود برنامه‌های کاربردی علمی در مقیاس بزرگ بود که منابع (محاسباتی و داده‌ای) زیادی را بیشتر از آنچه که می‌توانست در یک کامپیوتر (کامپیوتر شخصی، ایستگاه کاری، ابرکامپیوتر یا کلاستر) قرار بگیرد، نیاز داشتند. یک گرید محاسباتی می‌تواند منابع مورد نیاز برای این برنامه‌های کاربردی را در یک حوزه‌ی مدیریتی واحد فراهم کند. محاسبات گریدی به عنوان یک روش کارآمد برای نسل جدید محاسبات مطرح شده است. این روش امکان به اشتراک گذاشتن، انتخاب و مجتمع کردن منابع ناهمگونی را می‌دهد که از نظر جغرافیایی توزیع شده هستند. به این ترتیب، مسائل علمی، مهندسی و تجاری در مقیاس بزرگ، اکنون با استفاده از گریدهای محاسباتی قابل حل هستند. تلاش‌های زیادی در قالب پروژه‌های گوناگون در زمینه‌ی گرید (سیستم‌های گرید یکپارچه، میان‌افزارهای درونی و در سطح کاربر، و برنامه‌های کاربردی) تاکنون انجام شده است که در شکل 2-23

⁹² Grid Computing

⁹³ Supercomputing

⁹⁴ Transparent

⁹⁵ Virtualization

طبقه‌بندی شده و اطلاعات مربوط به کشور و سازمان انجام دهنده و کاربردهای آنها آورده شده است.

معمولاً اصطلاحاتی مانند محاسبات توزیع شده، محاسبات سراسری⁹⁶، فرامحاسبات⁹⁷ و محاسبات کلاستری⁹⁸ با محاسبات گریدی اشتباه می‌شوند. اگر چه همه‌ی اینها شباهتهایی با هم دارند، ولی متفاوت هستند. محاسبات توزیع شده یک اصطلاح عام است که همه‌ی موارد ذکر شده را شامل می‌شود؛ در حالی که محاسبات گریدی یا کلاستری و سایرین، زیرمجموعه‌ای از آن هستند.

در محاسبات کلاستری تأکید بر بسترهایی متشکل از گره‌های همگون متصل به هم است و معمولاً در یک حوزه‌ی مدیریتی⁹⁹ قرار دارند. کلاسترها عموماً از کامپیوترهای شخصی و ایستگاه‌های کاری¹⁰⁰ و شبکه‌های پرسرعت و مطمئن تشکیل شده‌اند. تأکید برنامه‌ها روی محاسبات حلقه ربایی¹⁰¹، محاسبات با توان عملیاتی¹⁰² بالا و محاسبات توزیع شده می‌باشد. محاسبات گریدی بر مجموعه‌ای از منابع ناهمگون توزیع شده تأکید دارد که از آنها به منظور ایجاد یک بستر برای محاسبات با کارایی بالا استفاده می‌شود. بستر ارتباطی و محاسباتی به صورت نامطمئن و پویا می‌باشد و منابع به شدت ناهمگون می‌باشند.

2-4-2- طبقه‌بندی سیستم‌های گرید

اهداف مورد نظر در طراحی سیستم گرید و همچنین کاربردهایی که برای آن در نظر گرفته شده است، از عوامل تعیین‌کننده در نوع کاربری گرید می‌باشند. اهداف طراحی را می‌توان به سه گروه عمده تقسیم‌بندی کرد که عبارتند از: افزایش کارایی برنامه‌ی کاربردی، دسترسی به داده‌ها، و افزایش و بهبود سرویس‌ها. با توجه به این اهداف می‌توان طبقه‌بندی شکل 2-23 را برای سیستم‌های گرید ارائه کرد.

⁹⁶ Global Computing

⁹⁷ Meta Computing

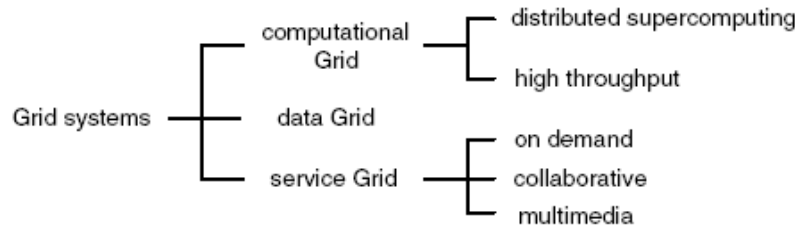
⁹⁸ cluster Computing

⁹⁹ Administrative Domain

¹⁰⁰ Workstations

¹⁰¹ Cycle-Stealing

¹⁰² Throughput



شکل 23-2: طبقه‌بندی سیستم‌های گرید

گریدهای محاسباتی به دو دسته‌ی عمده تقسیم می‌شوند: 1- ابرمحاسبات توزیع شده و 2- گرید دارای توان عملیاتی بالا. گرید ابرمحاسبات توزیع شده، برنامه را به صورت موازی روی چندین ماشین اجرا می‌کند و هدفش کاهش زمان اتمام¹⁰³ یک کار است. نمونه‌ی برنامه‌هایی که نیاز به ابرمحاسبات توزیع شده دارند، مسائل بزرگ و پرچالشی مانند شبیه‌سازی هسته‌ای یا مدل‌سازی آب و هوا می‌باشد. گرید با توان عملیاتی بالا، نرخ اتمام یک جریان از کارها را افزایش می‌دهد و برای کاربردهای پارامترروب¹⁰⁴ مانند شبیه‌سازی مونت کارلو مناسب است. برنامه‌های کاربردی پارامترروب از تعداد زیادی کار مستقل از هم تشکیل شده‌اند که برای آنها پارامترهای یکسان و مشخصی وجود دارد. هر کدام از کارها تنها در مورد مقادیر این پارامترها با کارهای دیگر متفاوت است. در دنیای واقعی، از برنامه‌های کاربردی پارامترروب که با استفاده از گریدهای محاسباتی به نتیجه رسیده‌اند، می‌توان به مدل‌سازی مولکول‌ها برای طراحی دارو [71] و تشخیص بیماری‌ها از روی پرتونگاری اشعه‌ی X [72] اشاره کرد.

گریدهای داده‌ای¹⁰⁵ [73]، یک بستر برای ساختن اطلاعات جدید از منابع داده‌ای توزیع شده همچون کتابخانه‌های دیجیتال یا انبارهای داده¹⁰⁶ را فراهم می‌کنند. البته گریدهای محاسباتی نیز به فراهم نمودن خدمات داده‌ای نیاز دارند، ولی عمده تفاوت یک گرید داده‌ای و گرید محاسباتی، بستر ویژه‌ای است که برای کاربردهای مدیریت ذخیره‌سازی و دستیابی داده، فراهم شده است. در گرید محاسباتی، برنامه‌ی کاربردی، خود، شمای مدیریت ذخیره‌سازی را پیاده‌سازی می‌کند. نمونه‌ای از کاربردهای این سیستم‌ها (گرید

¹⁰³ Completion Time

¹⁰⁴ Parameter Sweep

¹⁰⁵ Data Grids

¹⁰⁶ Data Warehouse

داده‌ای (داده‌کاوی¹⁰⁷ برای اهداف خاص است که اطلاعات را از منابع داده‌ای متفاوت به هم مربوط می‌سازد. پروژه‌های گرید داده‌ای اروپایی¹⁰⁸ و گلوباس¹⁰⁹ نمونه‌هایی هستند که در حال کار بر روی توسعه دادن سازمان‌های داده‌ای با مقیاس بزرگ هستند.

سیستم‌هایی که در یک گرید خدماتی¹¹⁰ [74] حضور دارند، خدماتی را فراهم می‌کنند که توسط یک ماشین و محدود قابل ارائه نیست. گریدهای خدماتی به سه دسته تقسیم می‌شوند:

1. **سیستم‌های گرید on-demand**: منابع مختلف را به صورت پویا

با یکدیگر ترکیب می‌کنند تا یک سرویس جدید ارائه دهند.

2. **سیستم‌های گرید هم‌کاری‌کننده¹¹¹**: کاربران و برنامه‌های

کاربردی را به گروه‌های کاری همکاری‌کننده متصل می‌کنند.

این سیستم‌ها امکان تعامل بلادرنگ بین کاربران و

برنامه‌های کاربردی را از طریق یک فضای کاری مجازی

فراهم می‌کنند.

3. **سیستم‌های گرید چندرسانه‌ای¹¹²**: یک بستر برای برنامه‌های

بلادرنگ چندرسانه‌ای فراهم می‌کنند.

اکثر پژوهش‌های فعال در توسعه‌ی سیستم‌های گرید، در یکی از

دسته‌بندی‌های فوق قرار می‌گیرد. برای توضیحات بیشتر می‌توان به

[75] مراجعه کرد.

2-4-3- توانمندی‌های گرید محاسباتی

هنگامی که یک گرید محاسباتی سازماندهی می‌شود، باید یک سری

از نیازمندی‌های کاربران را برطرف کند. برای اینکه نیازهای

موجود به طور بهتری با توانمندی‌های یک گرید منطبق گردد، باید

دلایل استفاده از محاسبات گریدی را در نظر داشته باشیم. در این

بخش به بررسی توانمندی‌های مهم محاسبات گریدی می‌پردازیم.

¹⁰⁷ Data Mining

¹⁰⁸ European DataGrid

¹⁰⁹ Globus

¹¹⁰ Service grid

¹¹¹ Collaborative

¹¹² Multimedia

2-4-3-1 بهره‌برداری از منابع بدون استفاده

ساده‌ترین راه استفاده از محاسبات گزیدی این است که یک برنامه‌ی کاربردی موجود در یک ماشین را بر روی ماشین دیگری اجرا کنیم. ماشینی که برنامه‌ی کاربردی در حالت عادی بر روی آن در حال اجرا می‌باشد، ممکن است به علت مواجه شدن با تعداد زیادی فعالیت، به طور غیرعادی درگیر محاسبات شود. این برنامه‌ی کاربردی می‌تواند بر روی یک ماشین بدون استفاده که در جای دیگری در گرید موجود است، اجرا شود.

حداقل دو پیش‌نیاز برای این مسأله وجود دارد: اول اینکه برنامه‌ی کاربردی باید بتواند بدون ایجاد کردن سربار¹¹³ غیرضروری، در یک ماشین غیر از ماشینی که در آن قرار دارد، اجرا شود. ثانیاً، ماشین مقصد باید سخت‌افزار و نرم‌افزار و منابع مورد نیاز برنامه را دارا باشد.

برای مثال یک برنامه‌ی دسته‌ای¹¹⁴ که زمان قابل ملاحظه‌ای را صرف پردازش یک مجموعه از داده‌های ورودی می‌کند تا یک سری خروجی را تولید کند، ممکن است ایده‌آل‌ترین و ساده‌ترین کاربرد برای یک گرید محاسباتی باشد. اگر تعداد ورودی‌ها و خروجی‌ها زیاد باشد، طراحی و تفکر بیشتری برای استفاده‌ی بهینه از گرید برای این کار مورد نیاز است. معمولاً اجرای یک برنامه‌ی پردازش کلمه¹¹⁵ بر روی یک ماشین دیگر در گرید بی‌معنی است؛ زیرا باعث بوجود آمدن تأخیرها و خطاهای بالقوه‌ی زیادی می‌شود. در بیشتر سازمانها، تعداد زیادی منابع پردازشی بدون استفاده وجود دارد. بیشتر کامپیوترهای رومیزی، کمتر از پنج درصد زمان، فعالیت می‌کنند. در بعضی از سازمانها حتی ماشین سرور نیز نسبتاً بیکار است. محاسبات گزیدی چارچوبی را برای بهره‌برداری از این منابع بیکار فراهم می‌کند و بنابراین این امکان را بوجود می‌آورد که به مقدار زیادی، استفاده از منابع افزایش پیدا کند.

منابع پردازشی تنها منابعی نیستند که ممکن است بدون استفاده باشند. بعضی اوقات، ماشینها دارای فضای ذخیره‌سازی زیادی هستند که استفاده نمی‌شود. محاسبات گزیدی و به طور خاص یک گرید داده‌ای می‌تواند امکان جمع نمودن این فضاها و تبدیل آن به یک انبار داده‌ی مجازی بسیار بزرگتر را فراهم کند. به این

¹¹³ Overhead¹¹⁴ Batch Application¹¹⁵ Word Processing

ترتیب، موجب می‌شود که کارایی و قابلیت اطمینان نسبت به یک ماشین واحد بهبود یابد. اگر یک کار دسته‌ای بخواهد مقدار زیادی داده را از ورودی دریافت کند، می‌توان نسخه‌های یکسانی از این داده‌ها را در نقاط مهم گرید قرار داد. بنابراین اگر یک برنامه بخواهد بر روی یک ماشین دور¹¹⁶ اجرا شود، لازم نیست داده‌های مورد نیاز آن نیز جابجا شوند، بلکه داده‌ها در نزدیکی و یا در خود آن ماشین موجود خواهند بود.

وظیفه‌ی دیگر گرید محاسباتی بهبود توازن در استفاده از منابع می‌باشد. در یک سازمان ممکن است در زمانهایی مقدار فعالیتها به اوج خود برسد که باعث افزایش تقاضای منابع می‌شود. اگر برنامه‌های کاربردی، قابلیت اجرا در گرید را داشته باشند، در چنین مواقعی می‌توان آنها را برای اجرا به ماشین‌های بدون استفاده منتقل نمود. در واقع، بعضی از پیاده‌سازی‌های گرید محاسباتی می‌توانند کارهای نیمه‌تمام را بین ماشینها جابجا کنند. در حالت کلی، یک گرید می‌تواند توازن بار محاسباتی را در مجموعه‌های گسترده‌تری از منابع انجام دهد. این منابع می‌تواند شامل پردازنده، حافظه‌ی جانبی و انواع دیگری از منابع موجود در گرید باشد.

2-3-4-2 موازی‌سازی پردازنده‌ها

ظرفیت انبوه پردازنده‌های موازی یکی از ویژگی‌های جذاب گریدهای محاسباتی می‌باشد. علاوه بر نیازهای کاملاً علمی، این قدرت محاسباتی می‌تواند باعث بوجود آمدن تحول بزرگی در صنایعی مانند حوزه‌ی پزشکی و زیست‌شناسی، مدل‌سازی مالی، اکتشاف نفت، متحرک‌سازی تصاویر و بسیاری دیگر شود.

ویژگی مشترک چنین استفاده‌هایی، استفاده از الگوریتم‌هایی برای تقسیم برنامه‌های کاربردی به بخش‌های اجرایی مستقل می‌باشد. یک برنامه‌ی کاربردی قابل اجرا در گرید که کار محاسباتی دارد را می‌توان به صورت تعداد زیادی کارهای کوچک در نظر گرفت که هر کدام بر روی ماشین متفاوتی در گرید اجرا می‌شوند. هر اندازه که این کارها به ارتباط با یکدیگر نیاز نداشته باشند، برنامه‌ی کاربردی قابلیت مقیاس‌پذیری¹¹⁷ بیشتری خواهد داشت. یک برنامه‌ی

¹¹⁶ Remote

¹¹⁷ Scalability

کاربردی کاملاً مقیاس‌پذیر، اگر از ده پردازنده برای کارهای خود استفاده کند، اجرای آن ده برابر زودتر تمام می‌شود.

موانعی در برابر مقیاس‌پذیری کامل وجود دارد. اولین مانع، الگوریتم‌هایی هستند که برای تقسیم برنامه‌های کاربردی بین تعداد زیادی پردازنده استفاده می‌شوند. اگر الگوریتم فقط به تعداد محدودی بخش‌های مستقل اجرایی قابل تقسیم باشد، مانعی در برابر مقیاس‌پذیری خواهد بود. دومین مانع هنگامی بوجود می‌آید که بخش‌های اجرایی کاملاً مستقل از هم نباشند. این باعث درگیری کارها با هم می‌شود و مقیاس‌پذیری را محدود می‌کند. برای مثال، اگر لازم باشد همه‌ی کارها عمل نوشتن و خواندن از یک فایل یا پایگاه داده‌ی مشترک را انجام دهند، محدودیت‌های دسترسی به آن فایل یا پایگاه داده، یک عامل محدودکننده برای مقیاس‌پذیری برنامه‌ی کاربردی خواهد بود. دلایل دیگر ایجاد درگیری بین کارها عبارتند از: تأخیر در تبادل پیام بین کارها، ظرفیت ارتباطات در شبکه، پروتکل‌های همزمان‌سازی، پهنای باند ارتباط با حافظه-های جانبی و وسایل دیگر، و تأخیرهایی که با نیازهای بلادرنگ تداخل پیدا می‌کند.

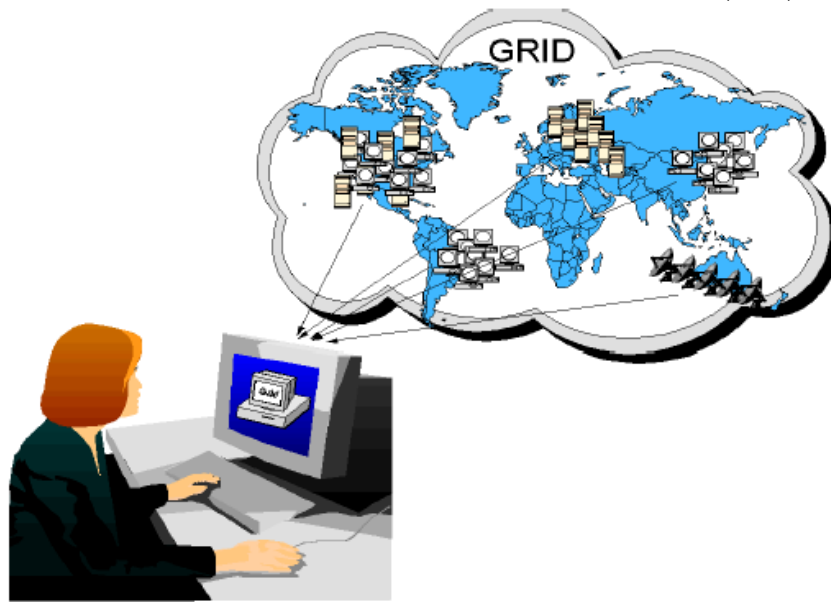
3-3-4-2 برنامه‌های کاربردی

برای اینکه یک برنامه‌ی کاربردی بتواند بر روی گرید اجرا شود، عوامل زیادی را باید در نظر گرفت. یکی از آنها این است که باید قبول کرد که همه‌ی برنامه‌های کاربردی این قابلیت را ندارند که به صورت موازی بر روی گرید محاسباتی اجرا شوند و در عین حال، مقیاس‌پذیری قابل توجهی نیز بدست بیایند. علاوه بر آن، در عمل، هیچ ابزار خودکاری وجود ندارد که برنامه‌های کاربردی را طوری تغییر دهد تا بتوانند از قابلیت‌های موازی گریدهای محاسباتی استفاده کنند. ابزارهایی وجود دارند که این مهارت را به طراحان برنامه‌های کاربردی می‌دهند که بتوانند برنامه‌های موازی را به مرحله‌ی پیاده‌سازی برسانند. البته دانش تبدیل خودکار برنامه‌های کاربردی در مراحل ابتدایی قرار دارد. این کار واقعاً مشکل است و حتی در شرایطی که امکان‌پذیر باشد به دانش ریاضیات و مهارت‌های بالای برنامه‌نویسی نیاز دارد. اغلب برنامه‌های کاربردی دارای محاسبات زیاد که امروزه پیاده‌سازی می‌شوند، دارای ساختار موازی هستند و اگر استناداردها و

پروتکل‌های جدید گرید را در خود نداشته باشند، به راحتی می‌توان قابلیت اجرا بر روی گریدهای محاسباتی را در آنها بوجود آورد.

2-4-3-4- منابع مجازی و سازمان‌های مجازی برای ایجاد همکاری

کمک مهم دیگری که یک گرید محاسباتی به ما می‌کند این است که همکاری بین تعداد زیادی سیستم‌های متصل به هم در یک محدوده وسیع را امکان‌پذیر و ساده می‌سازد. در گذشته، محاسبات توزیع شده در حال رسیدن به این مرحله از همکاری‌ها بود و تا حدی نیز به آن دست یافته بود. اکنون محاسبات گریدی این توانایی‌ها را حتی در مقیاس وسیع‌تری نسبت به آنچه تصور می‌شد در اختیار قرار می‌دهد. همچنین همراه با آن، استانداردهای مهمی را ارائه می‌کند که سیستم‌های بسیار ناهمگون را قادر می‌سازد تا با یکدیگر همکاری کنند و به این ترتیب، تصویری از یک سیستم محاسباتی مجازی بزرگ که منابع مجازی بسیاری را فراهم می‌کند ایجاد شود. این دید یکپارچه در شکل 2-24 نشان داده شده است.



شکل 2-24: بازنمایی منابع ناهمگون و پراکنده به صورت یک سازمان مجازی واحد توسط گرید

2-4-3-5- دسترسی به منابع اضافی

علاوه بر منابع پردازشی و ذخیره‌سازی اطلاعات، یک گرید می‌تواند دسترسی به تعداد زیادی از منابع دیگر و همچنین به

تجهیزات، نرم افزارها، جوازهای¹¹⁸ استفاده و خدمات خاص دیگر را فراهم کند. منابع اضافی می‌تواند در تعداد و یا در ظرفیتهای اضافی فراهم شود.

برای مثال، اگر یک کاربر نیاز داشته باشد که به منظور پیاده‌سازی یک سیستم جستجوی داده‌کاوی، پهنای باند ارتباط خود با شبکه‌ی جهانی اینترنت را افزایش دهد، کار او می‌تواند بین چندین ماشین موجود در گرید که هرکدام ارتباط مستقلى با شبکه‌ی جهانی برقرار کرده است تقسیم شود. در این صورت، مجموع توانایی‌های جستجو در اینترنت چند برابر می‌شود؛ زیرا هر ماشین ارتباط جداگانه‌ای با اینترنت دارد. اگر این ماشین‌ها مجراهای ارتباطی خود را در گذشته به اشتراک گذاشته بودند، افزایش چندانى برای پهنای باند دسترسی به اینترنت بوجود نمی‌آمد. بعضی از ماشین‌ها ممکن است جواز استفاده از یک نرم افزار گران‌قیمت را که بر روی آنها نصب شده است در اختیار داشته باشند. کارهای مورد نظر یک کاربر که به چنین نرم‌افزاری نیاز دارد می‌تواند به ماشین مربوطه فرستاده شود تا از جواز استفاده از نرم‌افزار موجود بر روی آن ماشین بهره ببرد.

بعضی از ماشین‌ها در گرید ممکن است دارای وسایل جانبی خاص باشند. بسیاری از ما از چاپگرهای دور استفاده کرده‌ایم تا از توانمندی‌های چاپ تصادفی پیشرفته و یا سرعت بالای آنها استفاده کنیم. به طور مشابه، می‌توان از یک گرید برای استفاده از تجهیزات ویژه‌ی دیگر استفاده کرد. برای نمونه، یک ماشین ممکن است دارای یک دستگاه ذخیره‌سازی داده بر روی لوح دی‌وی دی¹¹⁹ باشد و دیگران می‌توانند از طریق گرید، برای انتشار سریع اطلاعات از آن استفاده کنند. همچنین بعضی از ماشین‌ها در گرید می‌توانند به یک میکروسکوپ الکترونی متصل باشند که قابلیت کار از راه دور را داراست و کاربران گرید در سایر نقاط می‌توانند از آن استفاده کنند.

4-4-2 مدیریت منابع در گرید

در ادامه، مدیریت منابع در سیستم‌های گرید توضیح داده خواهد شد.

¹¹⁸ License

¹¹⁹ DVD

2-1-4-4-2 تعاریف و نیازمندی‌ها

در گریدها، منبع¹²⁰ عبارت است از یک نهاد¹²¹ با قابلیت استفاده‌ی مجدد¹²² که برای برآورده کردن یک کار یا تقاضا به خدمت گرفته می‌شود. منبع می‌تواند یک ماشین، شبکه یا هر سرویس دیگر (ترکیبی از ماشین‌ها، شبکه‌ها و نرم افزار) باشد.

فراهم‌کننده‌ی منبع¹²³، به عنوان یک عامل¹²⁴ که منبع را کنترل می‌کند، تعریف می‌گردد. برای مثال، یک واسطه‌گر¹²⁵ منبع می‌تواند به عنوان یک فراهم‌کننده‌ی منبع عمل کند. به همین صورت، مصرف‌کننده‌ی منبع¹²⁶، عاملی است که مصرف‌کننده را کنترل می‌کند.

سیستم مدیریت منبع¹²⁷ (RMS)، سرویسی است که توسط یک شبکه‌ی محاسباتی¹²⁸ فراهم می‌گردد تا مجموعه‌ی وسیعی از منابع دارای نام که برای شبکه‌ی محاسباتی موجود می‌باشند را مدیریت کند، به طوری که شاخص کارایی (سیستم محور¹²⁹ یا کار محور¹³⁰) بهینه گردد [77].

به علت مواردی چون توسعه‌پذیری¹³¹، سازگارپذیری¹³²، خودمختاری سایت¹³³ و کیفیت سرویس، مدیریت منبع در سیستم‌های گریدها بسیار پرچالش‌تر از محیط‌های محاسباتی توزیع شده‌ی¹³⁴ قدیمی است.

2-2-4-4-2 مدل انتزاعی سیستم مدیریت منبع

تجربه‌ی کار با سیستم‌های محاسباتی شبکه‌ای نشان داده است که برنامه‌ی کارا و کارایی سیستم، الزاماً یکی نیستند. به عبارت دیگر، یک زمانبند ممکن است نتواند هم برنامه را بهینه کند و هم کارایی سیستم را. یک راه‌حل استفاده از سیستم چندلایه است. علاوه بر این، از آنجایی که مقیاس مورد انتظار در سیستم‌های

¹²⁰ Resource

¹²¹ Entity

¹²² Reusability

¹²³ Resource Provider

¹²⁴ Agent

¹²⁵ Broker

¹²⁶ Resource Consumer

¹²⁷ Resource Management System (RMS)

¹²⁸ Computational Network

¹²⁹ System Centric

¹³⁰ Job Centric

¹³¹ Extensibility

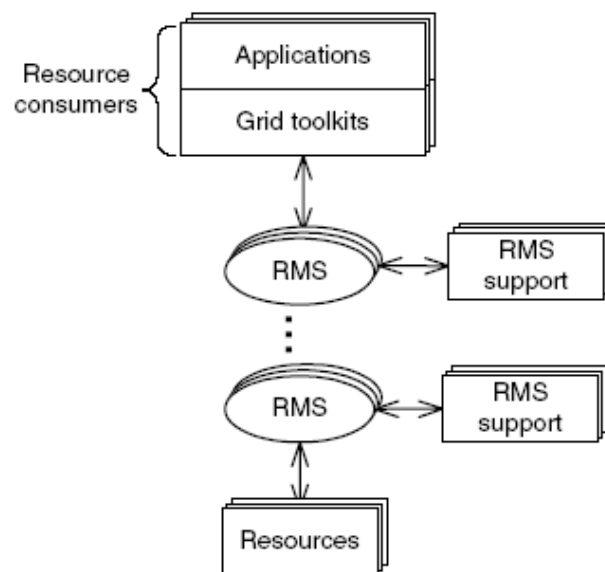
¹³² Adaptability

¹³³ Site Autonomy

¹³⁴ Distributed Computing Environments (DCE)

گريد، بزرگ است لذا يك سيستم مديريت منابع معمولاً از اتصال سيستم مديريت منابعي متعددي تشكيل شده است كه با يكديگر همكاري مي كنند. شكل 2-25 يك نمودار بلوكي براي يك سيستم همراه با سيستم مديريت منابعي متعدد به هم متصل را نشان مي دهد كه هر سيستم مديريت منابع داراي چندين سطح مي باشد. برنامه هاي کاربردي از سرويس هاي جعبه ابزارهاي¹³⁵ گريد استفاده مي كنند.

شكل 2-26 يك مدل انتزاعي از توابع اصلي كه توسط يك سيستم مديريت گريد پشتيباني مي شود را نشان مي دهد. اين مدل داراي واحدهاي عملياتي متعدد و چهار رابط¹³⁶ مي باشد: رابط مصرف كننده منبع، رابط فراهم كننده منبع، رابط پشتيباني مدير منبع¹³⁷ و رابط هم تاي مدير منبع¹³⁸.



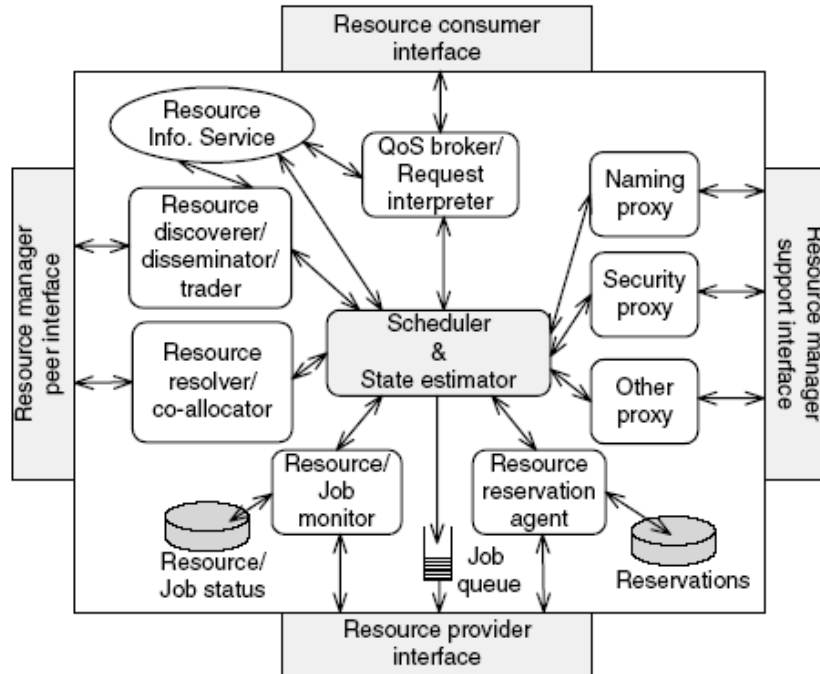
شكل 2-25: زمینه ی سیستم سیستم مدیریت منابع

¹³⁵ Toolkits

¹³⁶ Interface

¹³⁷ Resource Manager Support Interface (RMSI)

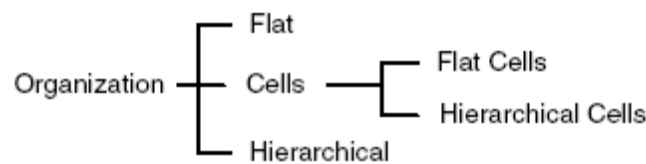
¹³⁸ Resource Manager Peer Interface (RMPI)



شکل 26-2: ساختار انتزاعی سیستم مدیریت منابع

3-4-4-2- سازماندهی ماشین

سازماندهی ماشین‌ها در گرید، الگوی ارتباطی سیستم مدیریت منابع را تحت تأثیر قرار می‌دهد و لذا نقش تعیین‌کننده‌ای در توسعه‌پذیری معماری حاصل دارد. شکل 27-2 طبقه‌بندی سازماندهی ماشین‌ها را نشان می‌دهد.



شکل 27-2: طبقه‌بندی سازماندهی ماشین‌ها در گرید

سازماندهی برای ماشین‌های درگیر در مدیریت منابع، در واقع، چگونگی اقدام به زمانبندی، ساختار ارتباطی بین ماشین‌ها و نقش‌های متفاوتی که ماشین‌ها در جهت زمانبندی ایفا می‌کنند را روشن می‌سازد.

در سازماندهی مسطح¹³⁹، همه‌ی ماشین‌ها می‌توانند بدون نیاز به یک رابط، مستقیماً با یکدیگر ارتباط داشته باشند. در سازماندهی سلسله‌مراتبی¹⁴⁰، ماشین‌های واقع در یک سطح می‌توانند با ماشین‌هایی که مستقیماً در بالا یا پایین آنها قرار دارند و همچنین با هم‌تاهای خود در همان سطح، ارتباط مستقیم داشته باشند.

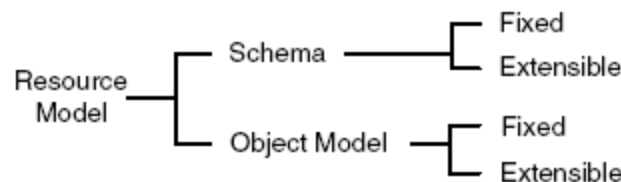
در ساختار سلولی، ماشین‌های واقع در یک سلول می‌توانند با یکدیگر از طریق سازماندهی مسطح، ارتباط داشته باشند. در هر سلول، ماشین‌هایی تعیین می‌شوند که وظیفه‌ی ارتباط با ماشین‌های خارج از سلول را بر عهده دارند. ساختار درونی یک سلول از سایرین پنهان است. سلول‌ها خود می‌توانند بصورت سلسله‌مراتبی یا مسطح سازماندهی شوند.

4-4-4-2- منابع

در این بخش، جنبه‌های مختلف سیستم‌های مدیریت منابع را بررسی خواهیم کرد که رابط بین منابعی که توسط سیستم مدیریت منابع اداره می‌شود را فراهم می‌سازند. برای توضیحات بیشتر می‌توان به [78,79] مراجعه کرد.

مدل منبع

مدل منبع، چگونگی توصیف منابع گرید توسط برنامه‌ها و سیستم مدیریت منابع را تعیین می‌کند. شکل 2-28 طبقه‌بندی مدل منابع را نشان می‌دهد.



شکل 2-28: طبقه‌بندی مدل منابع

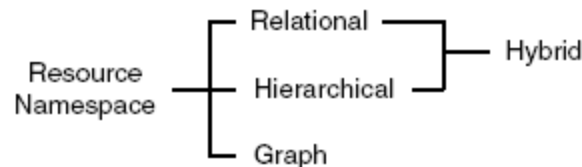
¹³⁹ Flat

¹⁴⁰ Hierarchical

در روش مبتنی بر شما¹⁴¹، داده ای که یک منبع را شامل می‌شود، توسط یک زبان، همراه با تعدادی محدودیت‌های جامعیت¹⁴² توصیف می‌شود. در شما توسعه‌پذیر، برخلاف نوع ثابت، انواع جدید شما برای توصیف منابع را می‌توان اضافه نمود. در یک مدل شیء¹⁴³، عملیات روی منابع به عنوان بخشی از مدل منبع تعریف گردیده است. همانطور که در مورد شماها گفته شد، مدل شیء نیز می‌تواند دو شکل داشته باشد: از پیش تعیین شده و تثبیت شده به عنوان بخشی از تعریف (مدل ثابت¹⁴⁴) و یا این‌که مکانیزم‌هایی برای توسعه در آن پیش‌بینی شده باشد (مدل توسعه‌پذیر).

سازماندهی فضای نام منابع

یک سیستم مدیریت منبع گریده، بر روی انبوهی از منابع نامگذاری شده‌ی سراسری عمل می‌کند؛ لذا سازماندهی فضای نام¹⁴⁵ تأثیر عمده‌ای بر پروتکل‌ها و متدهای یافتن منابع می‌گذارد. شکل 2-29 طبقه‌بندی سازماندهی فضای نام را نشان می‌دهد. فضای نام رابطه‌ای، منابع را به شکل رابطه تقسیم می‌کند و از مفاهیم موجود در پایگاه داده‌ی رابطه‌ای برای این کار استفاده می‌کند. فضای نام سلسله‌مراتبی، منابع گریده را در سلسله‌مراتب‌هایی پخش می‌کند؛ و در نهایت فضای نام مبتنی بر گراف از گره‌ها و اشاره‌گرها برای تقسیم منابع استفاده می‌کند.



شکل 2-29: طبقه‌بندی سازماندهی فضای نام

یافتن و انتشار منبع

یافتن و انتشار منبع¹⁴⁶ را می‌توان به عنوان توابع تکمیلی لحاظ کرد. یافتن، توسط یک برنامه‌ی کاربردی شبکه‌ای¹⁴⁷، برای

¹⁴¹ Schema

¹⁴² Integrity Constraints

¹⁴³ Object Model

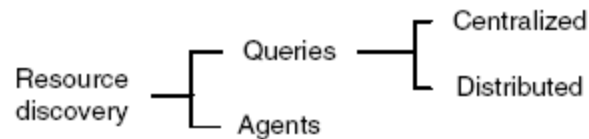
¹⁴⁴ Fixed

¹⁴⁵ NameSpace

¹⁴⁶ Resource Discovery and Dissemination

¹⁴⁷ Network Application

پیدا کردن منابع مناسب در گرید صورت می‌گیرد. انتشار، توسط یک منبع به منظور یافتن برنامه‌های مناسب که می‌توانند از آن منابع استفاده کنند، انجام می‌شود. یافتن و انتشار منابع باید بر اساس معیارهای مشخصی انجام گیرد (مانند کیفیت سرویس [92]). شکل 30-2 طبقه‌بندی یافتن منابع و شکل 31-2 طبقه‌بندی انتشار آنها را نشان می‌دهد.



شکل 30-2: طبقه‌بندی یافتن منبع



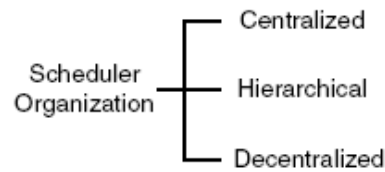
شکل 31-2: طبقه‌بندی انتشار منبع

5-4-4-2 زمانبندی

در این بخش طبقه‌بندی‌هایی را که وظایف زمانبندی و تخصیص منابع را در سیستم مدیریت منابع توصیف می‌کنند، مورد بررسی قرار می‌دهیم.

سازماندهی زمانبندی

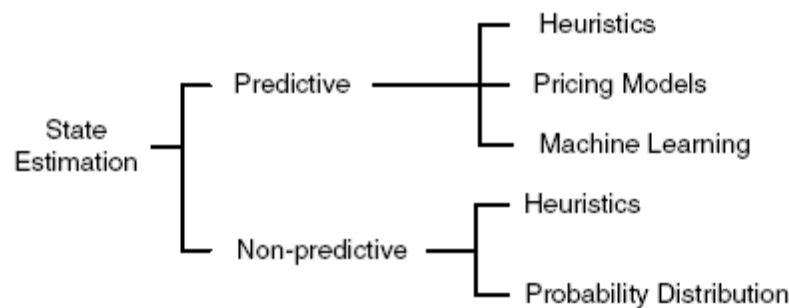
مؤلفه‌های زمانبندی در سیستم مدیریت منابع را می‌توان به سه شکل، سازماندهی کرد (شکل 32-2). در سازماندهی متمرکز فقط یک کنترل‌کننده‌ی زمانبندی وجود دارد که در سطح سیستم تصمیم‌گیری می‌کند. در سازماندهی سلسله‌مراتبی، کنترل‌کننده‌های زمانبندی در یک سلسله‌مراتب قرار دارند؛ به طوری که هر چه به سطوح بالاتر برویم، کنترل‌کننده‌ها می‌توانند بر مجموعه‌ی بزرگتری از منابع نظارت کنند. سازماندهی غیرمتمرکز، حالتی غیر از دو حالت فوق‌الذکر است که در آن کنترل‌کننده‌ها در سطح گرید به صورت توزیع شده قرار می‌گیرند. هر کدام از این سه حالت دارای مزایا و معایبی هستند. (برای اطلاعات بیشتر به [76,77] مراجعه کنید)



شکل 2-32: طبقه‌بندی سازماندهی زمانبند

برآورد وضعیت¹⁴⁸

به علت تأخیر انتشار اطلاعات در سیستم‌های توزیعی مانند سیستم‌های گرید، برآورد وضعیت همیشه برحسب اطلاعات جزئی و کهنه صورت می‌گیرد. تأکید طبقه‌بندی ارائه شده در این بخش روی مکانیزم‌های به کار گرفته شده برای برآورد وضعیت‌هایی است که پیاده‌سازی مدل انتزاعی را تحت تأثیر خود قرار می‌دهند. شکل 2-33 طبقه‌بندی برآورد وضعیت را نشان می‌دهد.



شکل 2-33: طبقه‌بندی برآورد وضعیت

برآورد وضعیت غیرپیش‌گویانه¹⁴⁹ فقط از اطلاعات وضعیت کار و منبع موجود استفاده می‌کند؛ زیرا نیازی به در نظر گرفتن اطلاعات گذشته ندارد. روش‌های غیرپیش‌گویانه یا از مکاشفه¹⁵⁰ بر اساس مشخصات کار و منبع استفاده می‌کنند و یا یک مدل توزیع احتمالی بر اساس تحلیل آماری غیربرخط¹⁵¹ مشخصات کار مورد انتظار، را به کار می‌گیرند. یک روش پیش‌گویانه، اطلاعات کنونی و سوابق را برای برآورد وضعیت لحاظ می‌کند. مدل‌های پیش‌گویانه از یکی از روش‌های

¹⁴⁸ State Estimation

¹⁴⁹ Non-predictive

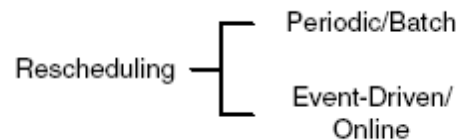
¹⁵⁰ Heuristic

¹⁵¹ Off-line

مکاشفه ای، مدل های قیمت گذاری و یا یادگیری ماشین استفاده می کنند. در روش مکاشفه ای، از قوانین از پیش تعیین شده برای هدایت برآورد وضعیت استفاده می شود. این قوانین، بر اساس رفتار مورد انتظار گرید برای برنامه ها استخراج می شوند. در روش مدل قیمت گذاری، با استفاده از پویایی های بازار، منابع خرید و فروش می شوند. در یادگیری ماشین، با استفاده از توزیع های نا شناخته ی بالقوه، از شمای برخط¹⁵² یا غیربرخط برای برآورد وضعیت استفاده می شود.

زمانبندی مجدد¹⁵³

ویژگی های زمانبندی مجدد در یک سیستم مدیریت منابع مشخص می کند که چه هنگام، زمانبندی موجود، مورد بررسی مجدد قرار بگیرد و چه هنگام، اجرای کارها دوباره منظم شود. نظم بخشی مجدد به کارها برای این منظور انجام می گیرد که بهره وری منابع، توان عملیاتی، یا سایر معیارها بسته به سیاست زمانبندی، ماکزیمم شود. روش به کار گرفته شده برای زمانبندی مجدد، مناسب بودن یک سیستم مدیریت منبع را برای یک کاربرد خاص مشخص می کند. شکل 2-34 طبقه بندی زمانبندی مجدد را نشان می دهد.



شکل 2-34: طبقه بندی زمانبندی مجدد

سیاست زمانبندی

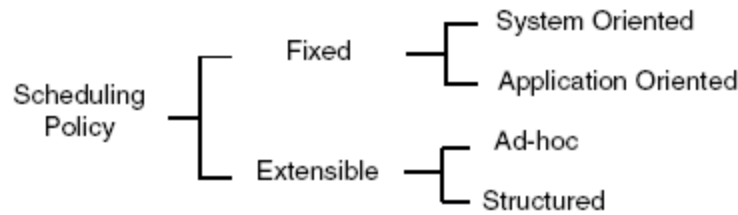
سیستم مدیریت منبع از یک سیاست زمانبندی¹⁵⁴ استفاده می کند تا ترتیب جزئی تقاضاها و کارها را هنگام زمانبندی تعیین کند. شکل 2-35 طبقه بندی سیاست های زمانبندی را نشان می دهد. در این طبقه بندی، تأکید بر میزان متأثر بودن سیاست زمانبندی از نهادهای خارجی می باشد. سیستم های گرید بزرگ با حوزه های مدیریتی مختلف، معمولاً دارای سیاست های به کارگیری منابع متعددی می باشند. در روش ثابت، سیاست پیاده سازی شده توسط مدیریت منبع،

¹⁵² On-line

¹⁵³ Rescheduling

¹⁵⁴ Scheduling Policy

از پیش تعیین شده است؛ در حالی که سیاست زمانبندی تو سعه‌پذیر به نهادهای خارجی این امکان را می‌دهد که سیاست زمانبندی را عوض کنند.



شکل 2-35: طبقه‌بندی سیاست‌های زمانبندی

2-4-5- خلاصه و نتیجه‌گیری

محیط گرید یک زیربنای سخت‌افزاری و نرم‌افزاری را فراهم می‌کند تا در یک شبکه از ماشین‌های توزیع‌شده و ناهمگون، منابع محلی هر ماشین برای دیگران قابل استفاده باشد. در واقع، یک گرید با مجموعه‌ای از منابع در مقیاس وسیع در ارتباط است که به صورت مشترک مورد استفاده قرار می‌گیرند. در طبقه‌بندی سیستم‌های گرید می‌توان سه نوع گرید را در نظر گرفت: محاسباتی، داده‌ای و خدماتی. گریدهای محاسباتی توانمندی‌های مختلفی دارند؛ از جمله استفاده از منابع بیکار، به کار گرفتن منابع اضافه و موازی‌سازی پردازنده‌ها برای افزایش کارایی. سیستم مدیریت منابع به عنوان مهم‌ترین بخش یک گرید محاسباتی نقش مهمی در کنترل و نظارت بر استفاده از منابع دارد. مهم‌ترین وظیفه‌ی این سیستم، زمانبندی کارها بر روی منابع می‌باشد. این عمل می‌تواند به صورت متمرکز، سلسله‌مراتبی و غیرمتمرکز انجام شود. با توجه به شرایط حاکم بر گریدهای محاسباتی، زمانبندی غیرمتمرکز و توزیع شده از بقیه موفق‌تر خواهد بود.

فصل سوم

شبکه پتری تصادفی تطابقی
(ASPN)

3 شبکه پتری تصادفی تطابقی

مقدمه

شبکه های پتری تصادفی [81] ابزاری مناسب برای مدل سازی ریاضی و گرافیکی به حساب می آیند. از این ابزار می توان برای مدل سازی، توصیف و تحلیل سیستم هایی که دارای ماهیتی همزمان، آ سنکرون، توزیع شده، موازی، نامعین و یا اتفاقی هستند استفاده نمود [82]. در واقع، شبکه های پتری جزء مدل هایی هستند که قادرند به صورت همزمان حالت و عملکرد یک سیستم را نشان دهند [83]. شبکه پتری تصادفی و تصادفی عمومی یک الگوی گسسته و گسترده در مدل سازی ها بشمار می آیند. این ابزارها توانایی مدل کردن احتمالی یک سیستم را با توجه به شرایط مختلف دارا می باشند. شبکه های پتری تصادفی، همزمانی، همروندی و عملیات غیر همزمانی پدیده ها را مورد پردازش قرار می دهند. از رفتار پویا و ساختار سیستم مدل شده بوسیله آنالیز شبکه پتری تصادفی، اطلاعات بسیار مفیدی اتخاذ می گردد که این اطلاعات می تواند جهت ارزشیابی و حدس هایی برای ایجاد، بهبود و تغییرات در سیستم ها بکار برده شود. خاصیت گرافیکی شبکه های پتری تصادفی [84] اجازه تجسم پیچیدگی سیستم را به مدل شبیه سازی شده می دهد. شبکه های پتری تصادفی می توانند به راحتی رابطه های اولویت دار و اثر متقابل ساختاری را در رویدادهای همزمان و غیر همزمان به نمایش بگذارند [4]. نرم افزارهای متعددی برای شبیه سازی و ارزیابی کارایی سیستم ها توسط شبکه های پتری و شبکه های پتری تصادفی از سوی افراد متخصص و موسسات تحقیقاتی در دانشگاه های مختلف ارائه شده است. از جمله این نرم افزارها می توان به PN Tool، Petri net Simulator، SPNP اشاره کرد [11,32,85]. شبکه های پتری تصادفی برای آنالیز سیستم هایی گسترده کاربرد بسزایی دارند. از جمله این سیستم ها می توان به نرم افزارهای توزیع شده، پروتکل ارتباطی، سیستم های تولید، VLSI، سیستم های پایگاهی، سیستم های فهرست و آمارهای منطقی، سیستم های شی گرا، سیستم های زمان واقعی و شبکه های محاسباتی گرید اشاره کرد [86,87].

یکی از مشکلات شبکه پتری تصادفی عدم تطبیق پذیری آنها می باشد و به همین دلیل در شبکه های پتری تصادفی امکان دسترسی به اطلاعات قبلی وجود ندارد [81]، مسئله کنترل زمان در شبکه های پتری تصادفی مورد بررسی قرار نمی گیرد [88] و در سیستم هایی که پیچیدگی (زیاد بودن حجم درخواستها و زیاد بودن عملیات) زیادی دارند شبکه های

پتری تصادفی با سرعت بسیار کمی کار تحلیل و آنالیزها را انجام می دهند [89].

هدف اصلی از این پروژه ارائه يك شبکه پتری تصادفی تطبیقی مبتنی بر اتوماتاهای یادگیر می باشد. شبکه پتری تصادفی تطبیقی مبتنی بر اتوماتاهای یادگیر از طریق اطلاعات بدست آمده از حالات قبلی سیستم و واکنش های محیط پویا، حالت بهینه بعدی را پیشگویی نموده و وضعیت جاری سیستم را بر اساس آن بروز کرده و احتمال وقوع پیش آمدها را در طول زمان تغییر می دهد و باعث می شود پیش آمدها بر اساس احتمال وقوعشان فعال شوند (رخ دهند). این بروز شدن وضعیت های سیستم بر اساس محیط پویا کمک شایانی در یادگیری و آموزش شبکه های پتری می کند. در این پروژه از ابزار شبیه سازی SPNP برای شبیه سازی شبکه پتری تصادفی استفاده می گردد.

در قسمت دیگری از این پروژه کاربرد مدل تطبیقی پیشنهاد شده در قسمت اول، در شبکه های گرید مورد مطالعه و بررسی قرار می گیرد. برای این منظور از مدل تطبیقی ارائه شده برای تخصیص بهینه منابع در گرید با توجه به معیار هزینه استفاده می گردد. این مساله با توجه به فرضیات زیر حل می گردد: 1. کارهای تخصیص یافته به منابع مستقل از هم هستند. 2. تخصیص منابع به صورت یکباره و در یک مرحله صورت می گیرد [68,90,91]. الگوریتمهای طراحی شده با الگوریتم های مکاشفه ای جدید (ABTO, EBTO, AEBTO) و الگوریتم یادگیری تقویتی (LATO) مقایسه می گردد [92-97].

کارهای انجام شده در یادگیری شبکه پتری

شبکه های پتری به طور گسترده ای به منظور ساخت، تجزیه تحلیل، طراحی و کنترل سیستم های همروند پیچیده مورد استفاده قرار گرفته اند. اگرچه، هنوز تحقیق در توضیح چگونگی اعمال این شبکه ها به سیستم های مختلف در حال انجام است، علاقه روز افزونی در سال های اخیر به توسعه یک شیوه و متد جدید که زمینه را در طرح خصوصیات تطابقی در مدل های عظیم شبکه پتری آماده کرده، به وجود آمده است [98-99].

در این بررسی، هدف ما خلاصه کردن نتایج تحقیق به عنوان حرکتی رو به جلو در خصوص شبکه های پتری تطابقی می باشد که، امروزه، علاقه زیادی را در جامعه پژوهشی ایجاد کرده اند. چنین رویکردهایی از گسترش کلی در عرصه تکنیک های هوشمند نظیر: شبکه

های عصبی مصنوعی¹⁵⁵، منطق فازی¹⁵⁶ و سیستم های مبتنی بر دانش¹⁵⁷ بهره می برند. تمام تلاش ها با هدف بهبود بخشیدن و توسعه نمونه هایی که بتوانند خصوصیت قابل سازش را به شبکه های پتری انتقال دهند انجام گرفته است.

وضعیت این رشته نوپا از طریق مشاهده زمینه های موضوعی مرتبط که در گرد همایی ها و کنفرانس های مختلف در این زمینه در سال های اخیر مورد بررسی قرار گرفته امکان پذیر است. بدلیل کارهای تحقیقی محدودی که در این حوزه انجام گرفته، همچنین، به منظور همگام بودن با مقاله های نو و مبتکرانه که نشان دهنده علاقه پر شور محققین در این زمینه است، بر آن شدیم تنها گزارشاتی که در ژورنال ها و کنفرانسها مختلف آمده بود را بگنجانیم.

در این زمینه، متوجه شدیم که پژوهش در چند سال اخیر از جنبه های مختلفی توسعه یافته است که این امر گسترش ایده های جدید را که اغلب از نتایج کاری رشته های مشابه سیستم های هوشمند موجب می شوند را می طلبد. رویکردی که روشهای مبتنی بر دانش، منطق فازی و شبکه عصبی را در هم می آمیزد و این آغازی در تغییر دور نمای پژوهش در این زمینه می باشد.

با تاکید بر موضوعات متودولوژی ها¹⁵⁸ در چارچوب تطابقی به نقل گزارشاتی خواهیم پرداخت که به نحوی تکنیک های هوشمند و شبکه های پتری را در بر می گیرد. این گزارشات به دو دسته تقسیم شده اند: 1- Fusion Hybrid و 2- Combination Hybrid.

در دسته اول، تحقیقات انجام گرفته در سال های اخیر با هدف تعریف شرایط اساسی می باشد که تحت این شرایط شبکه پتری قادر است برای توصیف اعمال تطابقی مشابه به شبکه عصبی زیستی ساخته شود. این دیدگاه نظریات خود را منحصرًا از نحوه عملکرد مدل زیستی مغز گرفته است و ظرفیت های شبکه پتری را به عنوان نماینده مناسبی برای تقلید از رفتارهای بیولوژیکی مغز از نقطه نظر ویژگی های توابع توزیع آن مورد آزمایش قرار داده است. با این همه، بیشتر کارهای تحقیقی مستلزم هم کوشی شبکه های پتری و تکنیک های هوشمند می باشد که ایده های آن از مفاهیم منطق فازی و بازشناسی الگوها در شبکه های عصبی مصنوعی با توجه به

¹⁵⁵ Artificial Neural Network(ANN)

¹⁵⁶ Fuzzy Logic(FL)

¹⁵⁷ Knowledge Base System(KBS)

¹⁵⁸ Methodology

وزن ها و خصوصیات یادگیری در آن ها ناشی شده است. در این دسته یک سیستم واقعی پیاده سازی شده وجود دارد که می خواهیم آن را بصورت مصنوعی مدل کنیم. این مدل مصنوعی که شمایی از جهان واقعی می باشد را براحتی می توان مورد تحلیل، آنالیز و بررسی قرار داد و نتایج را در دنیای واقعی اعمال کرد. علت تطبیقی بودن این دسته به تغییرات بنیادی پیکربندی شبکه پتری و همسان شدن آن با چارچوب سیستم واقعی مربوط می شود. تغییراتی در ساختار اصلی شبکه پتری در این دسته مقالات اعمال شده است و باعث شده که با استفاده از تکنیک های هوشمندی به شبکه پتری قدرت و توانایی یادگیری و آموزش را بدهد. معایبی هم این دسته دارند. اغلب اوقات یادگیری در شبکه های پتری از این دسته باعث افزایش پیچیدگی عملیات و افزایش سربار محاسباتی می شود. چون، زمان زیادی برای مدل کردن تمام خصوصیات سیستم واقعی در کنار استفاده از تکنیک های هوشمند در شبکه پتری طول می کشد.

در دسته دوم، گروه نسبتاً کوچکی از محققان در زمینه به کارگیری و اجرای تکنیک های هوشمند تطابقی به همراه متودولوژی های شبکه پتری در مسائل دنیای واقعی فعالیت می کنند. مدل های این دسته ساده تر از مدل های دسته اول می باشند. در حالی که، در برخی اوقات زمان اجرای آن ها بیشتر است. همچنین، میزان پیچیدگی در طراحی کمتر از روش اول می باشد. در این روش تکنولوژی های مختلف در کنار هم قرار می گیرند و با تغییرات نسبی در هر کدام، مدل ارائه شده برای اجرا در محیط با بهره وری و دقت بیشتری نسبت به حالت عادی اعمال می شود. این روش از خصوصیات اولیه شبکه پتری در کنار تکنیک های هوشمندانه استفاده می کند که به کارها و درخواست های داخلی هر بخشی از سیستم واقعی اعمال می شود. در اینجا، یک ترکیب واقعی و روشن از ویژگی های هوشمند با شرایط فعلی سیستم در جهت بهبود وضعیت سیستم جاری پیشنهاد شده است. در این بررسی، بر آن خواهیم بود که مروری بر آخرین تلاشها و مقالاتی که در زمینه شبکه پتری توسط محققان مختلفی از دیدگاه تطابقی آن انجام گرفته است داشته باشیم و به برجسته سازی تلاش هایشان از دیدگاه متودولوژی تحلیلی بپردازیم.

3-2-1- ویژگی های تطابقی

هنوز هم شبکه های پتری بعنوان معرف کوچکی در حل مسائل دنیای واقعی که در ماهیتی پویا و غیر خطی دارند در نظر گرفته می شود. این شبکه ها، قابلیت یادگیری برای تطبیق خود در محیط

های متغیر را ندارند. از سوی دیگر، تکنیک های هوشمند ویژگی هایی را در اختیار می گذارند که اغلب دارای قابلیت تطبیق می باشند و قادر به ساخت استنتاج سطح بالا¹⁵⁹، رفتارهای فازی و توانایی های یادگیری هستند. مسئله ای که در میان این تکنیک های هوشمند مشترک و یکسان می باشد این است که همه آنها به یک یا چندین جنبه از شناخت انسان می پردازند که این جوانب عبارتند از: پردازش اطلاعات، بازنمایی دانش¹⁶⁰ و یادگیری. لذا، معقول به نظر می رسد که هدفی برای دستیابی به مدلی که بتواند با نقاط قوت تکنیک های هوشمند به همراه شبکه های پتری به شکل یک مدل پیوندی و تلفیقی باشد را در نظر بگیریم.

در واقع، مدلی طراحی شود تا بتوان تکنیک های هوشمند و شبکه های پتری را بصورت یک مدل پیوندی و ترکیبی در آن بیان کرد.

گرایش های اخیر، راه حل های ترکیبی بوجود آمده از تکنیک های هوشمند استفاده می کند که ابعاد دیگر از مسائلی می باشد که مسائل آنی را در بر می گیرد [100-102]. راه های مختلفی وجود دارد که تکنولوژی های پیوندی یا ترکیبی را به منظور ارائه راه حل هایی برای کاربرد های متنوع مورد استفاده قرار داده است [103]. این بخش به مرور تحقیقاتی می پردازد که بر شیوه های متنوعی تاکید می ورزند که در تلاش اند تا خلاصه متفاوتی از سطوح آگاهی را به مدل های شبکه ادغام کنند. ویژگی های مختلفی نظیر معیارهای عضویت، وزن ها، فاکتورهای قطعیت و قانون یادگیری که همه و همه اقتباسی هستند از تکنیک های هوشمند که به روشهای مختلفی جهت ساخت مدلی قابل تطابقی از شبکه های پتری معرفی شده اند که در این بخش تحت عنوان Fusion Hybrid و Combination Hybrid طبقه بندی شده اند.

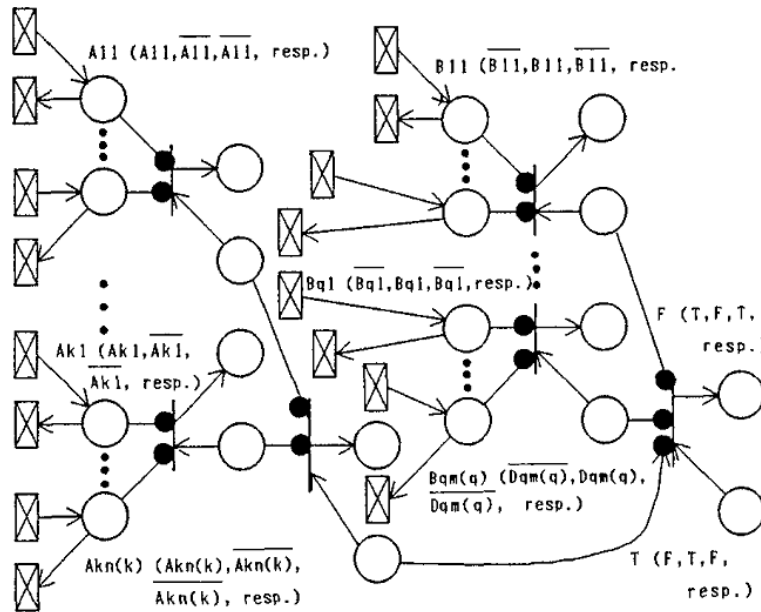
2-2-3 Fusion Hybrid

این رویکرد مستلزم مکاتب کردن ارائه یک متودولوژی به متودولوژی دیگر دلالت بر اینکه چنین گذری به طور طبیعی اتفاق نمی افتد می باشد. در اینجا، ایده های محققین مختلف در مسائل مختلف مورد بررسی قرار گرفته است. سه تحقیق اول بر رویکردی تاکید می کنند که به منظور بیان خصوصیات یادگیری تطابقی نوعی قیاس تمثیلی بین شبکه های پتری و شبکه های عصبی ارائه می کند.

¹⁵⁹ High level Reasoning

¹⁶⁰ Knowledge Representation

تسوجی [104] در سال 1990 مدل شبکه پتری توسعه یافته ای¹⁶¹ را ارائه داد که خصوصیات استنتاج فازی و شبکه عصبی را با هم ادغام نمود. در این مدل، به هر توکن یک مقدار حقیقی مثبت (که بیشتر از یک هم نیست) داده می شود. این مقدار میزان آستانه فعال شدن یک انتقال (گذر) می باشد. ویژگی های تحریک پذیری¹⁶²، قابلیت دسترسی¹⁶³، حد و مرز پذیری¹⁶⁴ و روش الگوسازی موتور استنتاج فازی¹⁶⁵ مورد تجزیه و تحلیل قرار گرفت. همانطور که در شکل 1-3 نشان داده شده، چهار قانون فازی تعریف شده در این مقاله به کمک شبکه پتری توسعه یافته مدل شده است. در جدول 1-3 میزان تغییرات نشانه گذاری ها در مکان ها و میزان تغییرات آن در هر گذر آورده شده است.



شکل 1-3: فرم استاندارد انتقال توکن بین مکان ها و انتقال ها با قوانین فازی تعریف شده

جدول 1-3: جدول تغییرات فازی در مقادیر نشانه گذاری گذرهای شبکه پتری توسعه یافته

¹⁶¹ Extended Petri Net (EPN)

¹⁶² Liveness

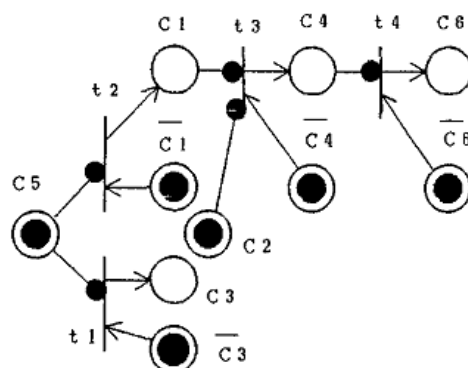
¹⁶³ Reachability

¹⁶⁴ Boundness

¹⁶⁵ Fuzzy inference engine

فعال شدن گذرها	C_1	C_2	C_3	C_4	C_5	C_6	$\bar{C}_1$	$\bar{C}_3$	$\bar{C}_4$	$\bar{C}_6$
نشانه گذاری اولیه	0	0.8	0	0	0.5	0	1	1	1	1
بعد فعال شدن t_1	0	0.8	0.5	0	0.5	0	1	0.5	1	1
بعد فعال شدن t_2	0.5	0.8	0.5	0	0.5	0	0.5	0.5	1	1
بعد فعال شدن t_3	0.5	0.8	0.5	0.5	0.5	0	0.5	0.5	0.5	1
بعد فعال شدن t_4	0.5	0.8	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5
نشانه گذاری انتهایی	0.5	0.8	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5

همانطور که در شکل 2-3 نشان داده شده است موتور استنتاج فازی لحاظ شده در شبکه پتری توسعه یافته نشان داده شده است.

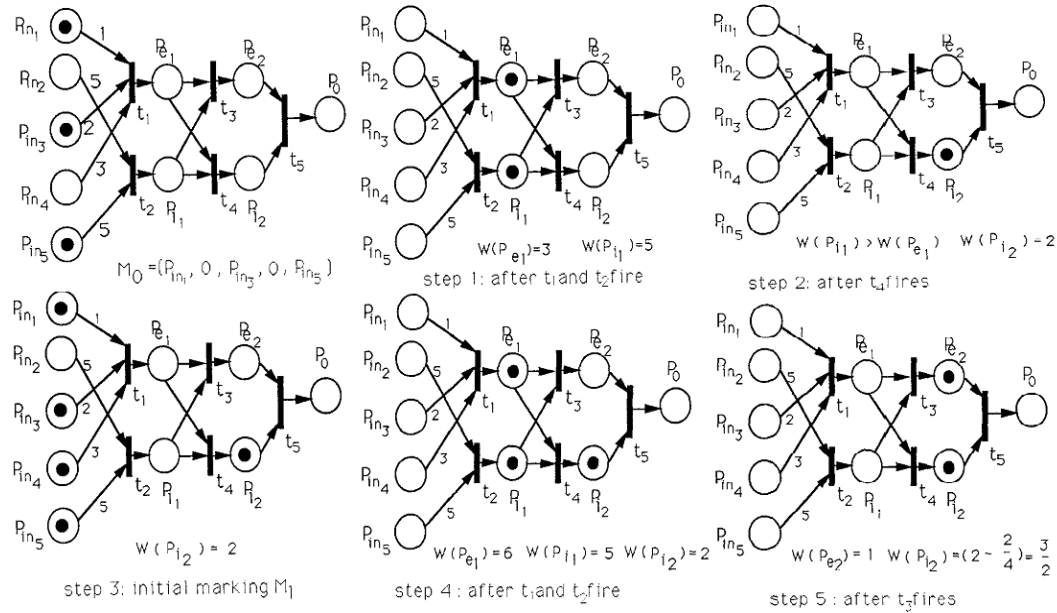


شکل 2-3: موتور استنتاج فازی لحاظ شده در شبکه پتری توسعه یافته

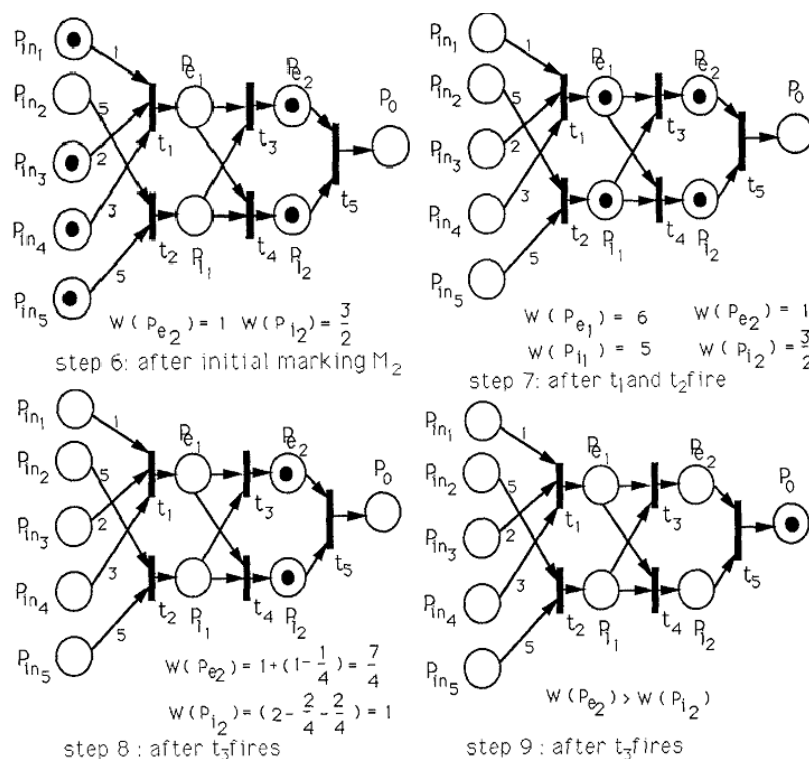
هدجیئیکولاو [105] در سال 1990 یک مدل شبکه پتری عصبی پیشنهاد کرد که از شبکه پتری برای تحلیل رفتار یک شبکه عصبی بهره می‌گرفت. مدل شبکه پتری عصبی ارائه شده در این مقاله از فاکتور تضعیف کننده¹⁶⁶ (که یک عدد حقیقی است) وزن در واحد زمان استفاده می‌کند که تعداد توکن‌ها را در برخی از واحدهای زمانی تغییر می‌دهد که، این فاکتور از طریق یک بیان‌کننده کاهش مکرر توان قابلیت پیش سیناسپی است هدایت می‌شود. این مدل از قابلیت برای ذخیره مرحله قبل از یک فعالیت تقلید می‌کند که شرایط مدل هوپ فیلد Hopfield را قبول می‌کند. خروجی شبکه پتری عصبی به مولفه آغازین شباهت دارد و بدنه اصلی نرون‌ها مشابه چهار مکان و پنج گذر هستند. مدل شبکه پتری عصبی با سه معیار و نشان آغازی جهت ارائه روشی که رفتار نرون‌ها را شبیه سازی کند آزمایش شد. تحقیقات زیادی در تلاش برای ترکیب کردن شبکه پتری با یک

¹⁶⁶ Decay Factor

تعداد بیشتری از تکنیک های هوشمند مثل شبکه های عصبی، منطق فازی و سیستم مبتنی بر دانش در مسیرهای موازی انجام گرفته است. شبکه پتری عصبی در شکل 3-3 و شکل 4-3 نشان داده شده است. در این شکل مراحل حرکت 3 توکن در شبکه پتری نشان داده شده است.



شکل 3-3: شبکه پتری عصبی-قسمت اول



شکل 4-3: شبکه پتری عصبی-قسمت دوم

ونکاتش و همکارانش [106] در سال 1993 یک رویکرد اساسی و بنیادی را برای مدلسازی رفتاری نرون ها به کمک شبکه پتری سطح بالا¹⁶⁷ ارائه داد. در این مقاله، یک المنت¹⁶⁸ جدید به HRN اضافه شده تا تمام خصوصیات شبکه عصبی را در مدل اعمال کند. این المنت در مکان ها، انتقال ها، کمان ها و وزن ها اعمال شده اند. اگرچه، چنین رویکردی توسط ونکاتش جزئیات عملیاتی از الگوسازی فعال آن ارائه نمی دهد. اما، کمکی به مقایسه شبکه عصبی و نمونه های شبکه پتری از نقطه نظر ویژگی ها و شباهت ها می کند. این ایده، به شرح و توصیف بخش های مغز و بخش قابل قیاس آن در شبکه های پتری می پردازد. به عنوان مثال، مکان ها می توانند شامل بدنه سلول های مغز باشند و هم شامل شکاف های پس سیناسپی¹⁶⁹ (سیناپس پیوندگاه شکاف سلول می باشد) باشند. اگرچه که هر دو عملکرد و وظایف مختلفی دارند. این شکاف ها به عنوان نقاط ورودی و خروجی عمل می کنند. این، در حالی است که، بدنه سلول ها نقاط سیگنالی را جمع

¹⁶⁷ Hierarchical Petri Net (HPN)

¹⁶⁸ Element

¹⁶⁹ Pre-Synaptic Clefts

آوری کرده و آستانه ای برای آغاز تحرک می باشند. کمان ها بطور جداگانه نماینده دندریته ها¹⁷⁰ (شاخه های متعدد سلول عصبی) می باشند و آکسون ها¹⁷¹ بعنوان توابع تحریک¹⁷² و بازداشت¹⁷³ می باشند. دو نوع مختلف از وزن مورد بررسی قرار گرفته است که یکی با اعداد واقعی (حقیقی) است و دیگری با اعداد صحیح. فرمول مدل شبکه پتری در نظر گرفته شده در شبکه عصبی زیستی از طریق ساخت پیاز بویایی یک خرگوش توضیح داده شده است.

آهسون [107] در سال 1995 مدلی از شبکه پتری عصبی فاز¹⁷⁴ پیشنهاد کرد که در این مدل خصوصیات شبکه عصبی و استنتاج فاز¹⁷⁵ را در هم آمیخت. در این مدل، شبکه پتری عصبی¹⁷⁶ مکان هایی را مورد استفاده قرار می دهد که شامل سه زیر مجموعه می باشد که عبارتند از: ورودی ها، انتقال ها و خروجی ها. گذرها (انتقال ها) شامل دو نوع معمولی و آغازی هستند. وزن های گذر موجود در ورودی ثابت و برابر یک است در حالی که وزن های لایه های انتقال و خروجی قابل آموزش و تمرین بوده و تنها مقادیر بین 0 و 1- را می پذیرند. پس، مدل شبکه پتری عصبی با استفاده از خصوصیات منطبق فاز¹⁷⁷ اصلاح شده است.

در ادامه در این مقاله، مدل شبکه پتری عصبی با استفاده از خصوصیات منطبق فاز¹⁷⁸ تغییر داده شد و سپس شبکه پتری عصبی فاز¹⁷⁹ ایجاد شد. این مدل شامل هفت تاپل¹⁷⁷ بود که به ویژگی های شبکه پتری عصبی اضافه می شد. این تاپل ها شامل مجموعه محدودی از گزاره ها¹⁷⁸ و مجموعه محدودی از مقادیر درستی فاز¹⁷⁹ بر روی گزاره ها می باشد. همچنین، شبکه پتری عصبی فاز¹⁷⁹ از فاکتورهای اضافه شده ای استفاده می کند که از مکان ها به گذرها نگاشت می شوند و نیز بعنوان توابع اتصال و پیوند که گذرها را به مقادیر واقعی و درست نگاشت می کنند. این متدولوژی پیشنهاد شده در دسته بندی الگوهای دودویی¹⁷⁹ تست شد. در شکل 3-5 و 3-6 شبکه پتری فاز¹⁷⁹

¹⁷⁰ Dendrite

¹⁷¹ Axons

¹⁷² Excitatory

¹⁷³ Inhibitory

¹⁷⁴ Fuzzy Neural Petri Net(FNPN)

¹⁷⁵ Fuzzy inference

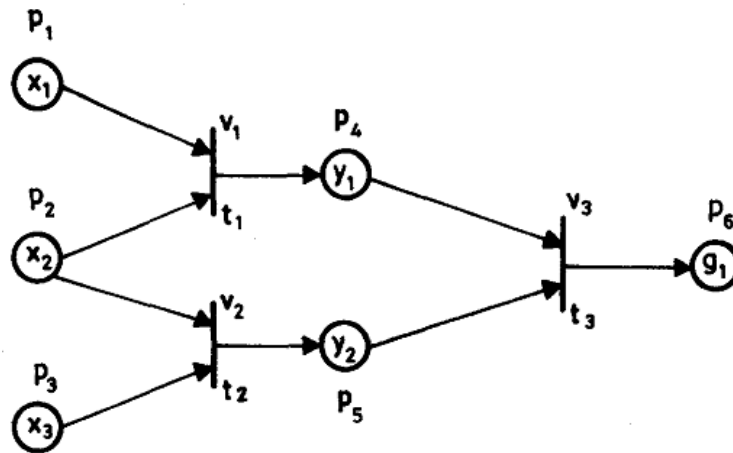
¹⁷⁶ Neural Petri Net(NPN)

¹⁷⁷ Tuple (ستون جدول)

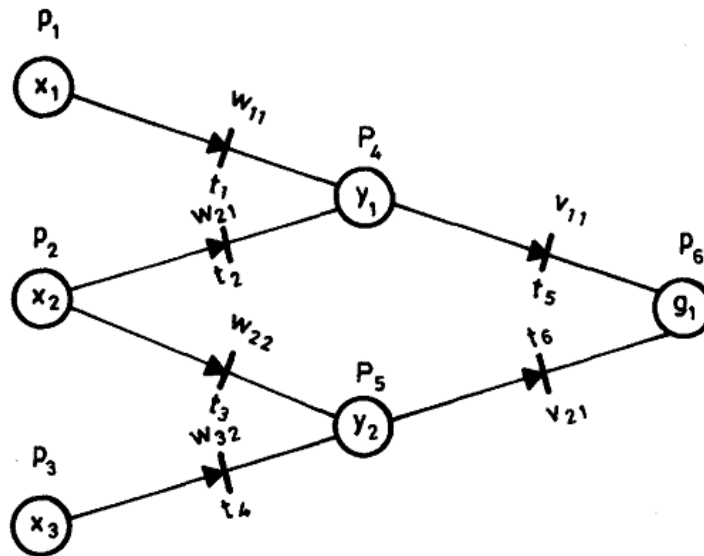
¹⁷⁸ Propositions

¹⁷⁹ Binary Pattern Classification

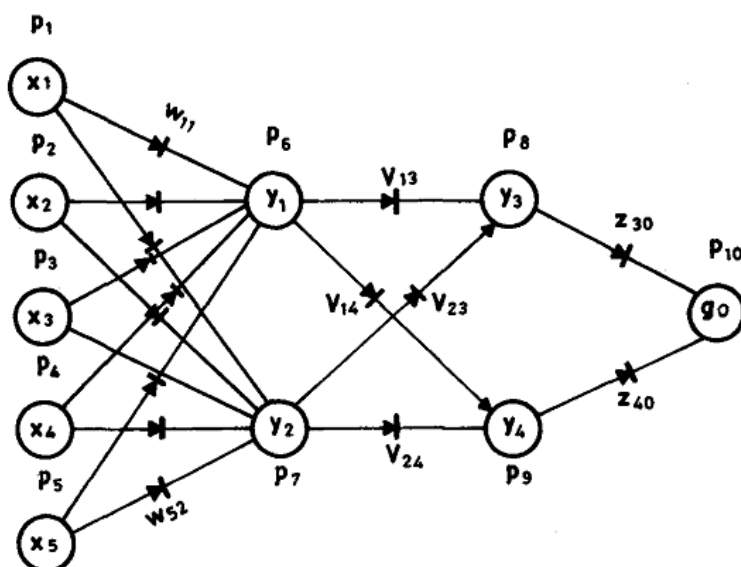
عصبی برای قوانین OR و AND نشان داده شده است. در شکل 7-3 مدل سه لایه ای شبکه پتری فازی عصبی برای استنتاج فازی نشان داده شده است.



شکل 3-5: شبکه پتری فازی عصبی برای نمایش سه قانون AND



شکل 3-6: شبکه پتری فازی عصبی برای نمایش سه قانون OR



شکل 3-7: مدل سه لایه ای شبکه پتری فازی عصبی برای استنتاج فازی

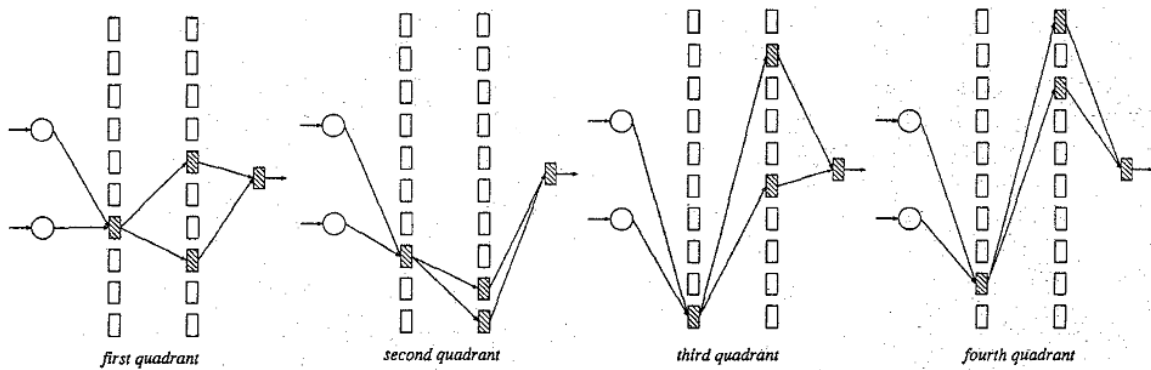
هیراساوا و همکارانش [108,109] در سال های 1995 و 1998 قابلیت ها و امکانات شبکه پتری را از نقطه نظر توزیع کاربردی که در شبکه عصبی زیستی مغز انسان وجود دارد آزمایش کرده و مدل بازبینی شده ای از شبکه پتری پیشنهاد کرد که قابلیت یادگیری را همانند شبکه عصبی دارا می باشد. در این مقاله مدلی بر اساس شبکه پتری بنام شبکه پتری یادگیر¹⁸⁰ ارائه شده است که توانایی یادگیری L.P.N از طریق تنظیم و تعدیل کردن وزن هایی که در مسیر اجرایی باشند کسب می گردد. ساختار شبکه پتری یادگیر وزن های در حال اجرا قرارگرفتن را در کمان های ورودی گذرها معرفی شد و علامت های در حال اجرا قرارگرفتن را که با تغییر مسیر توکن یا نشانه ها حرکت می کند مورد استفاده قرار می دهد. مطابق فرآیند یادگیری، توکن یا نشانه های منتقل شده به شبکه پتری یادگیر - بسته به مقدار علامت های در حال اجرا قرار گرفتن توکن یا نشانه - ها در مکان های ورودی و مقدار وزن های در حال اجرا قرار گرفتن بسته به کمان های شبکه پتری یادگیر می باشد. الگوریتم یادگیری که در شبکه پتری یادگیر استفاده شده است آمیخته ای است از قانون یادگیری هب¹⁸¹ و انتشار به عقب¹⁸² که در هر دو وزن در حال اجرا قرار گرفتن و مقادیر آغازین طوری تنظیم شده اند که علامت در حال

¹⁸⁰ Learning Petri Network(L.P.N)

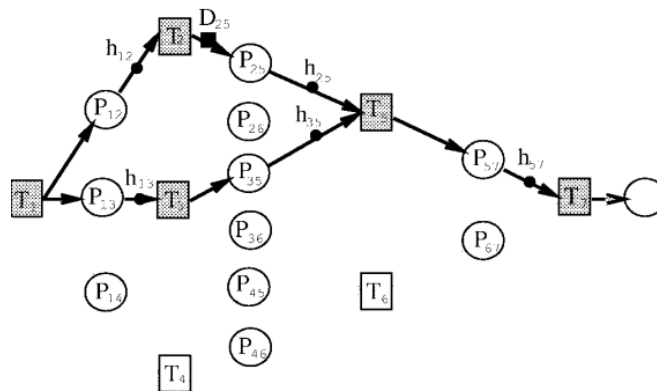
¹⁸¹ HEBB

¹⁸² Back propagation

اجرا قرار گرفتن یک گذر به مقدار مورد هدف خود نزدیک می شود. تنظیم و تعدیل وزن نیز به گونه ای تغییر داده شده است که عبور یک توکن یا نشانه زمانی که انتخاب می شود تقویت شده و بنابراین تثبیت می شود. شبیه سازی به منظور تثبیت کنترل مسیر شبکه پتری یادگیر و همچنین مقایسه عملکرد شبکه پتری یادگیر با مدل آن شبکه عصبی اجرا می شود. شبکه پتری یادگیر پیشنهادی به مسئله شناسایی سیستم های غیرخطی ربط داده شد و نتایج شبکه پتری یادگیر ثابت کرد این مسئله برای سیستمی که توسط شبکه عصبی ساده تری بررسی می شوند بهتر باشد. شکل 3-8 یادگیری در شبکه پتری در چهار برش ورودی و خروجی شبکه عصبی نشان داده شده است [108]. شکل 3-9 نمونه ای از چگونگی فعال شدن و روشن شدن یک گذر در شبکه پتری یادگیر در هر برش زمانی را نشان می دهد. این شبکه پتری یادگیر که بصورت لایه ای فرض شده است از یک گذر ورودی، یک گذر خروجی و پنج گذر معمولی در دو لایه پنهان تشکیل شده است [109].

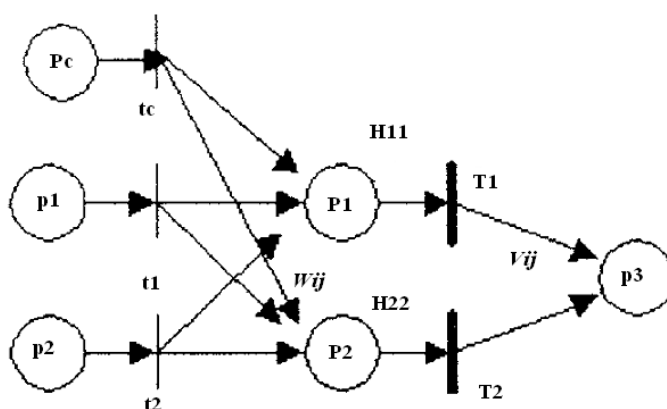


شکل 3-8: یادگیری در شبکه پتری در چهار برش ورودی و خروجی شبکه عصبی



شکل 9-3: چگونگی فعال شدن و روشن شدن یک گذار در شبکه پتری یادگیر در هر برش زمانی

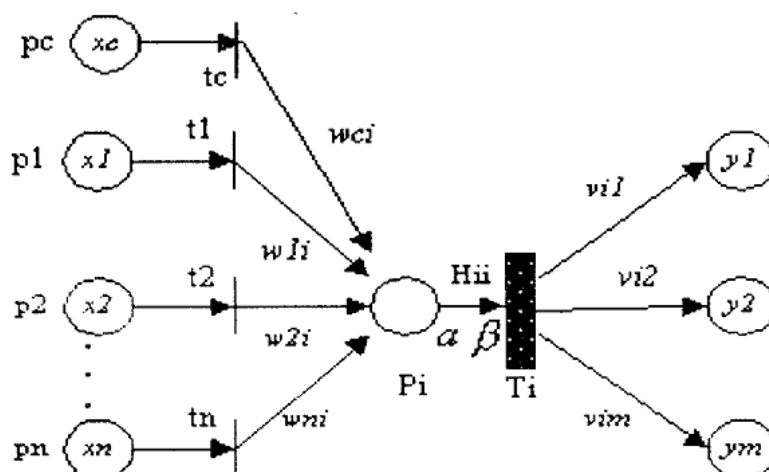
ژا و همکارانش [110] در سال 1998 مدل شبکه پتری خبره عصبی فازی¹⁸³ را که ساختار آن همانند [107] می باشد را ارائه داد و این مدل را در دو کاربرد مورد ارزیابی قرار داد. یکی از کاربردها ارزیابی سیستم های غیر خطی و طراحی مجدد آن ها می باشد و کاربرد دیگر در کنترل پردازش های روبات های هوشمند می باشد. شکل 10-3 یک شبکه پتری عصبی دو لایه را نشان می دهد. در اینجا، p_c بایاس کنترل جریان، p_1 و p_2 لایه ورودی، P_1 و P_2 لایه پنهان، p_3 لایه خروجی، t_1 و t_2 گذرهای نرمال، T_1 و T_2 گذرهای انتقال دهنده، H_{11} و H_{22} مقادیر آستانه فعال شدن و α و β توابع انتقال کلی حلقوی یا مارپیچی¹⁸⁴ می باشند. در شکل 11-3 مدل شبکه پتری خبره فازی نشان داده شده است.



شکل 10-3: شبکه پتری عصبی دو لایه

¹⁸³ Neural Fuzzy Expert Petri Net (NFEPN)

¹⁸⁴ Generic Sigmoid Transfer Function



شکل 3-11: مدل شبکه پتری خبره فازی

تسنگ و همکارانش [111] در سال 1999 ظرفیت یادگیری شبکه های پتری فازی را از طریق مدلسازی تولید قوانین فازی¹⁸⁵ و الگوریتم یادگیری شبکه عصبی مورد بررسی قرار دادند. الگوریتم یادگیری بکار رفته در مدل پیشنهادی در این مقاله شامل مقداردهی اولیه، ارائه نمونه های آزمایشی، روش های محاسبات Feed Forward و Feed Backward برای تنظیم کمان ها می باشد. این مقاله، یک مدل 14-Tuple از شبکه فازی پتری تعریف شده است که شامل مکان ها، گذرها، میزان آستانه آغازها در دستورالعمل های ورودی خروجی، مجموعه های فازی، مجموعه وزن های محلی یا میانی از قوانین تولید فازی، فاکتور قطعیت قوانین و مجموعه کمان ها می باشند. مجموعه کمان ها ارائه کننده فاکتورهای قطعیت و انواع مختلف دستورالعمل های ارتباط که به هر مکان یک مجموعه فازی و یک فاکتور قطعیت، به هر کمان یک وزن مکانی و یک نقشه مطرح دقیق از مکان ها به مقدارهای آغازی اختصاص می دهند هستند. این مدل پیشنهادی در مجموعه قواعد تولید فازی که معرف پایگاه دانش سیستم خبره فازی جهت انتخاب کامپیوتر متخصص و کارآمدتر در یک سازمان می باشد بکار رفته است.

لی و همکارانش [112,113] در سال 1999 و 2000 مدلی به نام مدل شبکه پتری فازی تطابقی پیشنهاد کرد که از فواید توصیفی شبکه پتری فازی سود می برد اما در همان حال نیز همانند شبکه عصبی دارای قابلیت یادگیری بود. این بررسی برای طراحی یک نظام

¹⁸⁵ Fuzzy production Rules

استنتاجی دانش پویا تاکید داشت که بر اساس تنوع دانش مثلا شناخت انسان¹⁸⁶ و درک و تفکر انسان قابل توصیف بود. این مدل به معرفی مفهوم تولید قانون تنظیم شده فاز¹⁸⁷ می پردازد که این مفهوم به هریک از طرح ها با توجه به سابقه کاربردی آنها و بر اساس یک قانون یک درجه اهمیت و ارزش در نظر می گیرد. در این مقاله، الگوریتم های یادگیری مجزا توسعه پیدا کردند و الگوریتم های استنتاج فاز¹⁸⁸ نامیده شدند و الگوریتم های انتشار به عقب برای تأیید وزن های قابل تعدیل و یادگیری دانش تحت شرایط کلی استفاده شده است.

چن و همکارانش [114] در سال 2002 مدلی از شبکه پتری فاز برای طبقه بندی الگوها¹⁸⁹ مبتنی بر قاعده (قانون) ارائه دادند. در این مدل پیشنهادی قوانین فاز در حوزه دانش¹⁹⁰ اعمال می گردد بطوری که به کمک آن وزن های محلی و فاکتور قطعیت¹⁹¹ توسط یک روال یادگیری بصورت آنلاین تنظیم می گردد. در این جا، هر مکان ورودی به عنوان یک ویژگی از الگو در نظر گرفته می شود و در هر انتقال (گذر) یک تولید قوانین فاز می باشد. نشانه گذاری در هر مکان به عنوان مقدار عضویت کلاس¹⁹² می باشد. شبکه از هفت لایه استفاده می کند. این مدل برای دسته بندی مجموعه داده های آیریس¹⁹³ که به منظور آزمایش توانایی استنتاج شبکه های پتری فاز در شناسایی نمونه های مختلف مورد استفاده قرار می گیرد.

جونز و همکارانش [115] در سال 2003 مدلی از یک تست آزمایشگاهی از یک خط آهن و عبور قطار بر روی آن بر اساس زمان واقعی توسط چند پردازنده را ارائه کردند. در این مدل، خصوصیات شبکه عصبی، زمان بندی و توکن های تصادفی در شبکه پتری با یک دیگر ادغام شده است. ساخت و شکل مسیر قطار به طور ساختاری با بهره گیری از مکان ها، گذرها و کمان های راست شده ساخته شده است. این در حالی است که، خود قطار توسط رنگ خاص و منحصر به فرد توکن یا نشانه ها ارائه می شود. سوزن های ریل با استفاده از شبکه پتری عصبی که مجموعه ای از گذرها را که از آرایش و نحوه فعال-

¹⁸⁶ Human Cognition

¹⁸⁷ Weighted Fuzzy Production Rule

¹⁸⁸ Fuzzy Reasoning Algorithm

¹⁸⁹ Pattern Classification

¹⁹⁰ Domain Knowledge

¹⁹¹ Certainty Factor

¹⁹² Class Membership Value

¹⁹³ Iris Data set

سازی یک اثر در ساختار چندگانه و لایه بندی شده ورودی و خروجی ساخته شده اند مورد استفاده قرار می دهد. ترکیبات مکان های زمانی-، گذرها و کمان های شبکه های پتری ابزاری را در اختیار می گذارد که میزان در حال اجرا قرار گرفتن سیناسپی را نمونه سازی کرده و سرعت اجرا را پردازش می کند. یادگیری از طریق شبیه سازی سوییچ-های شبکه پتری عصبی با بردارهای نشانه گذاری شبکه و تنظیم وزن سیناپس ها تا زمانی که رفتار مورد نظر کسب شود انجام می شود.

گووا و همکارانش [116] در سال 2003 یک شبکه پتری استدلال فازی¹⁹⁴ را برای سیستم تولید قوانین فازی ارائه کردند. در این مدل یک الگوریتم استدلالی استنتاج می شود که فرمت ماتریکس تساوی را برمیگزیند. این ماتریکس تصاویر بسیار شبیه به نوع خود در شبکه پتری معمولی می باشد و توانایی استدلال موازی را به خوبی درجه صحت تمامی طرح ها و مسائل از نخسیتن آنها حتی بدون استفاده از روش تحلیل قابلیت دسترس نشان می دهد. آنها به چگونگی این امر از طریق نشان دادن، تامل کردن و استنتاج قوانینی که شامل اشتباهات چاپی منفی در الگوی شبکه پتری پیشنهادی خود هستند می-پردازند چنین مدلی از طریق یک سیستم تخصصی خطا یاب مورد اعتبار قرار می گیرد.

3-2-3- Combination Hybrid

از نقطه نظر بهره گیری از توانایی هر روش در ترکیب و ادغام به گونه ای که هویت و ویژگی های جداگانه آنها حفظ شود، این رویکرد شامل پیوند صریح و آشکار می باشد.

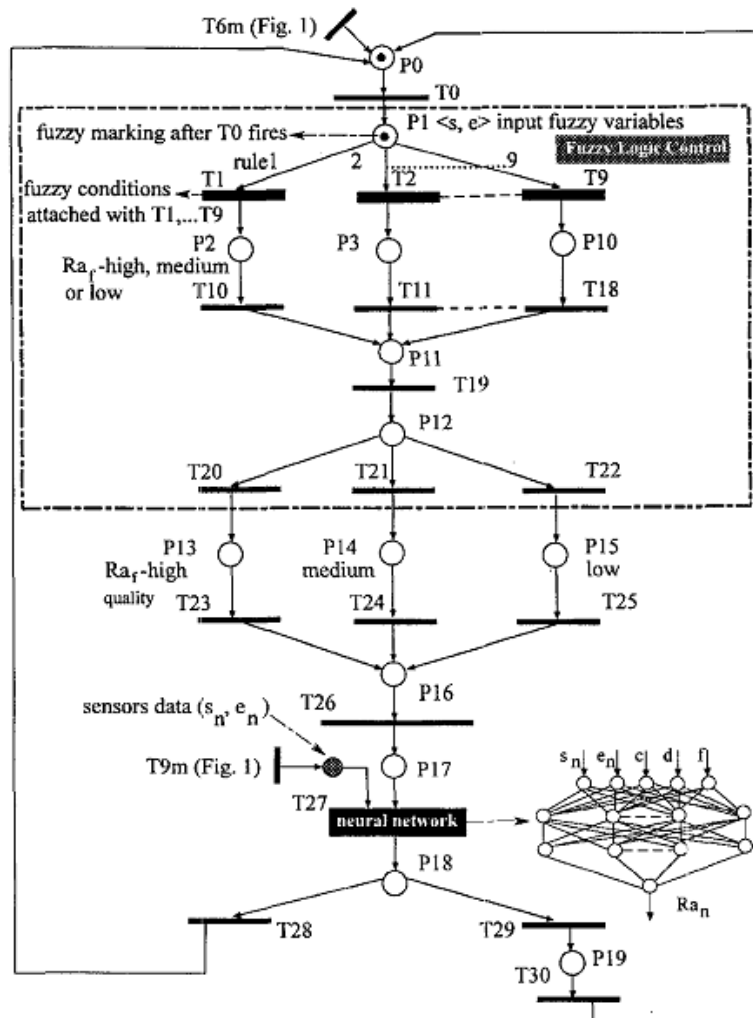
هانا و همکارانش [117] به منظور حفظ و نگهداری کیفیت ویژگی های محصول در یک مرکز فرز¹⁹⁵ دستگاه های CNC، مدلی مبتنی بر ترکیب و بکاربردن دو روش ANN و شبکه پتری فازی را ارائه دادند. در اینجا، از شبکه پتری فازی به همراه شبکه عصبی مصنوعی برای مدل سازی و کنترل ناهموار سطوح فلزات¹⁹⁶ در فرآیند فلزکاری CNC استفاده می شود. این مقاله مثالی است از کار تحقیقاتی که در قالب مدل ترکیبی با استفاده از شبکه پتری و ANN انجام گرفته است. در این مدل ترکیبی علاوه بر گذرهای مرتبط با فرآیند فلزکاری دستگاه CNC، یک گذر هنگام فعال شدن ANN را راه اندازی

¹⁹⁴ Fuzzy Reasoning Petri Net (FRPN)

¹⁹⁵ Milling

¹⁹⁶ Metal Surface Roughness

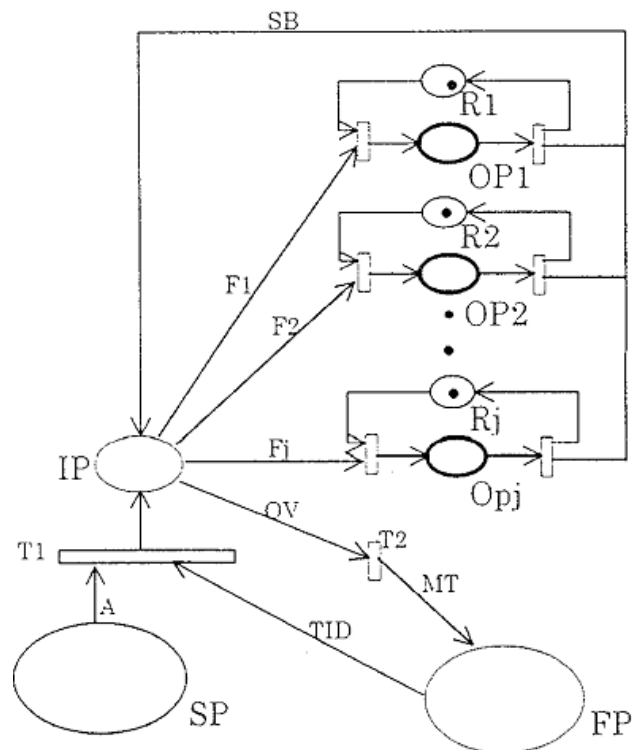
میکنند¹⁹⁷ تا شبکه شروع بکار کند و گذر دی گر باعث فعال شدن خروجی در شبکه عصبی مصنوعی می شود. اگر خروجی ANN مشابه خروجی شبکه پتری باشد آنگاه گذر دیگری برای تکرار این چرخه فعال می شود، در غیر این صورت، گذر ترمیم خطا به منظور فعال ساختن بخش های مختلف دستگاه جهت تنظیم مجدد فعال می شود. در شکل 3-12 یک مدل شبکه پتری فازی عصبی برای کنترل سطوح ناهمواری در فرز کاری دستگاه CNC نشان داده شده است.



شکل 3-12: یک مدل شبکه پتری فازی عصبی برای کنترل سطوح ناهمواری در فرزکاری دستگاه CNC

¹⁹⁷ Triggers

سانگ و همکارانش [118] یک تکنیک بهینه سازی برای سیستم های تولید قابل انعطاف صنعتی (FMS)¹⁹⁸ مطرح کردند، با کمک این تکنیک، مسئله زمانبندی تخصیص منابع بر اساس مدل شبکه عصبی هاپ فیلد¹⁹⁹ و شبکه پتری تصادفی مدل سازی می شود. در اینجا، تاخیر زمانی با مکان ها و انتشار زمانی با توکن ها همراه است، همچنین این مدل محدودیت ارتباط بین وظایف، دستگاه ها و ساختار عینی را به یک ساختار انرژی نگاشت می کند. با استفاده از این ساختار انرژی با مدل شبکه عصبی هاپ فیلد، الگوریتم طراحی بهینه سازی برای FMS به دست می آید. نمودار مدل سیستم های تولید قابل انعطاف صنعتی در شکل 3-13 نشان داده شده است.



شکل 3-13: نمودار شبکه پتری تصادفی مدل FMS

سرانجام، آقای فاکودا و همکارانش [119] در سال 2002 برای بهبود صحت نمایش الگوی الکترومایگرام²⁰⁰ روش کنترل مایوالکتریک را با استفاده از شبکه عصبی سازمان دهی شده و رخدادی که الگوی

¹⁹⁸ Flexible Manufacturing System(FMS)

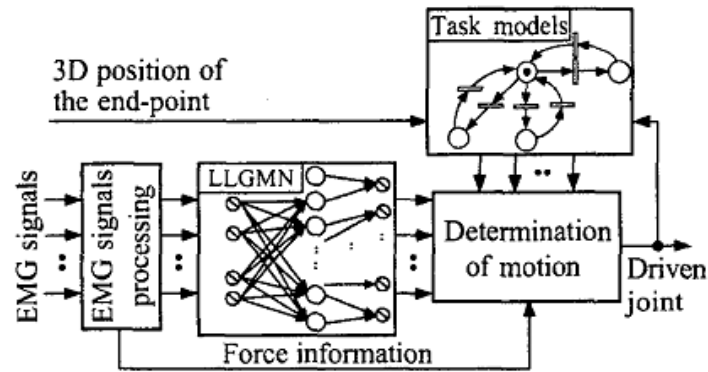
¹⁹⁹ Hopfield

²⁰⁰ Electromyogram

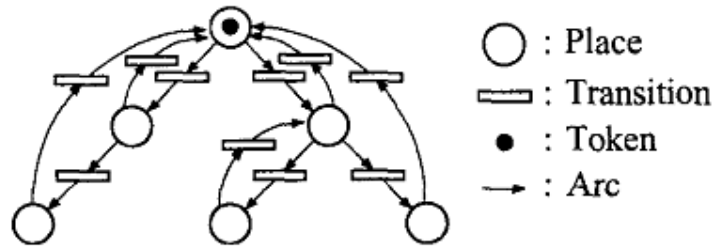
وظیفه را استخراج می‌کند مطرح کردند. این مدل، باعث بهبود در کنترل دقت دستگاههای الکتروم‌یوگرام با استفاده از هوشمندی شبکه های عصبی و مدل وظیفه توسط شبکه های پتری می‌باشد. این ایده در آزمایشگاه ربات دانشگاه هیروشیما ژاپن بر روی یک دست مصنوعی مورد بررسی قرار گرفت. ساختار ایده پیشنهادی در شکل 3-14 نشان داده شده است.

هدف در این ایده ایجاد یک روش بهینه برای کاهش حرکت های اضافی دست مصنوعی است. این یک نمونه از ادغام دو تکنولوژی مختلف بصورت یک سیستم واحد است. در این ایده، کنترل میو الکتریک برای بهبود و دقت در دستگاه های الکترو کاردیوگراف برای ایجاد طرح های آماری متفاوت رسم شده با استفاده از شبکه عصبی، سازماندهی شده و بصورت یک مدل وظیفه رویداد پیشنهاد شده است. هر الگو و طرح از الکتروکاردیوگراف²⁰¹ بعنوان یک ورودی از شبکه عصبی می‌باشد که می‌خواهد روال آموزش در آن اجرا شود. در اینجا سنسورهای سیگنالهای الکتروکاردیوگراف بر روی دست و فواصل قرار گرفته اند و در بازه زمانی خاص یک الگو از حرکت دست و وضعیت محیط را به الگوی شبکه عصبی ارسال می‌کند. شبکه عصبی با توجه به اطلاعات اولیه یک مدل وظیفه از کارها و وظایف یک الگو می‌سازد و آن را به شبکه پتری می‌دهد و این شبکه با توجه به حالت بهینه و ایده آل مقایسه می‌کند و این روال، تا ساخت الگوی بهینه اعمال می‌شود و در انتها آن را در محیط، پیاده سازی می‌کند. هر درخواست آمده به شبکه پتری بعنوان یک وظیفه محسوب می‌شود. پس، شبکه پتری بکار رفته در این پروژه، مسئول تحلیل درخواست داده شده به دست مصنوعی را بصورت یک مدل وظیفه، برعهده دارد. مدل وظیفه با استفاده از شبکه پتری جهت دانش وابسته به وظیفه که برای تغییر خروجی شبکه عصبی مورد استفاده قرار می‌گیرد نشان داده می‌شود. مدل وظیفه در شکل 3-15 نشان داده شده است.

²⁰¹ Electrocardiograph



شکل 3-14: ساختار کلی ایده پیشنهادی آقای فاکودا



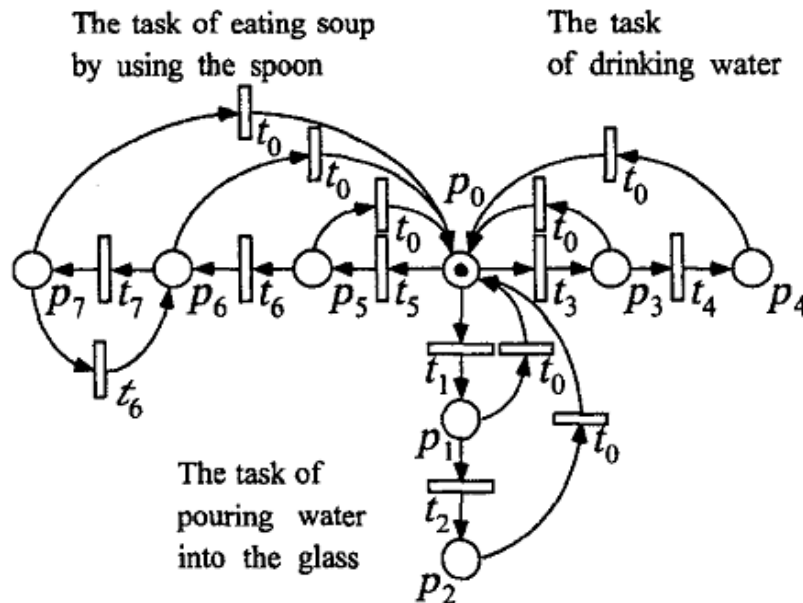
شکل 3-15: مدل وظیفه ایده پیشنهادی آقای فاکودا

شبکه عصبی یک شبکه سه لایه ای Feed Forward است که براساس مدل ثبت وقایع بصورت خطی²⁰² و یک شبکه مرکب گوسین²⁰³ می باشد و می تواند از طریق یادگیری برای تابع شدت احتمال هر کلاس متفاوت در الگوهای ورودی باز شناسی را مدل کند. مدل وظیفه با بهره گیری از تاریخچه نتایج تمایز شبکه عصبی، موقعیت عملی اپراتور را برآورد می کند و مطابق موقعیت برآورد شده بردار تعدیل کننده را بیرون می دهد. مدل وظیفه از شبکه پتری برای توصیف اطلاعات مستقلى می باشد که در توصیف خروجی شبکه عصبی استفاده می شود. مدل وظیفه با استفاده از سابقه ای که نتایج شبکه عصبی دارد حالت عملگر وظیفه را تخمین می زند. شبکه عصبی به الگوهای دستگاه الکتروکاردیوگراف متصل است و الگو بصورت ورودی وارد شبکه عصبی می شود. خروجی شبکه عصبی یک مدل وظیفه است که توسط شبکه پتری مورد آنالیز و طراحی قرار گرفته است. این رویکرد

²⁰² Log Linear Model

²⁰³ Gaussian Mixed Network

پیشنهادهای با استفاده از بهره گیری کنترل کننده انسانی (بازوی مصنوعی) که توسط نویسنده خود گسترش یافت مورد آزمایش قرار گرفت. این ایده برای یک ربات در روی میزی که یک بطری آب، لیوان، بشقاب غذا و قاشق سوپ بود مورد آزمایش واقع شد. شکل 3-16 مدل شبکه پتری که وظیفه خوردن در یک وعده غذایی را نشان می دهد. جدول 2-3 مکان ها و گذر های یک وعده غذا خوردن ربات را نشان می دهد.



شکل 3-16: مدل وظیفه برای یک وعده غذایی
جدول 2-3: وضعیت مکان ها و گذر ها

مکان ها		گذر ها	
p0	Standby	t0	Opening
p1	برداشتن بطری	t1	عمل بلند کردن بطری
p2	ریختن	t2	عمل بد داخل چرخاندن قاشق
p3	بلند کردن لیوان	t3	عمل بلند کردن لیوان
p4	خوردن آب	t4	عمل به داخل بردن غذا به دهن
p5	بلند کردن قاشق	t5	عمل بلند کردن بازو دست
p6	پرکردن قاشق با غذا	t6	عمل به داخل چرخاندن مچ دست
p7	خوردن سوپ	t7	عمل به داخل بردن غذا به دهن

چنین رویکردی سبب بهبود صحت باز شناسی کنترل مصنوعی نیز شد. همانطور که در این مثال دیدیم در اینجا شبکه پتری بصورت تطابقی با توجه به ورودی و خروجی شبکه عصبی وضعیت بهتر را برای محیط در نظر می گیرد. در فصل های بعدی به توضیح ایده های پیشنهادی خواهیم پرداخت.

خلاصه و نتیجه گیری

در این بخش ایده های انجام شده در شبکه پتری مورد بررسی قرار گرفت. در اینجا ایده ها به دو دسته تقسیم شدند. هر دسته روش هایی صورت گرفته بود که بطور کامل مورد بررسی قرار گرفت. در واقع شبکه پتری تصادفی تطابقی یک شبکه پتری تصادفی است که با توجه به اطلاعات محیط بتواند در گام های دیگر طراحی به کمک وضعیت حالت فعلی حالت بعدی سیستم را پیش بینی کند و به مرور زمان خود را با شرایط محیط بروز کند. با توجه به تعریفی که در این فصل در مورد شبکه پتری تصادفی تطابقی²⁰⁴ صورت گرفت به این نتیجه می رسیم که یک شبکه پتری تصادفی را می توان در سیستم های هوشمند و سیستم هایی که پویایی و تغییراتی زیادی دارند بکاربرد.

در فصل های بعدی به کاربرد ASPN در گرید های محاسباتی و اقتصادی می پردازیم.

²⁰⁴ Adaptive Stochastic Petri Net(ASPN)

فصل چهارم

کاربرد شبکه پتری تصادفی تطابقی
در گزید محاسباتی

4 کاربرد شبکه پتری تصادفی در گرید محاسباتی

مقدمه

در گرید محاسباتی مسئله زمانبندی و تخصیص کارها به منابع یکی از مسائل مهم می باشد. از آنجایی که منابع پیچیده²⁰⁵، توزیع شده، ناهمگون²⁰⁶، پویا و خودمختار²⁰⁷ می باشند انجام تخصیص و زمانبندی منابع با دشواری هایی همراه می باشد. بنابر این، ما بر آن شدیم تا روشی را پیشنهاد کنیم تا روند زمانبندی و تخصیص منابع در گرید های محاسباتی را بهبود بخشیم. در این بخش، یک الگوی سلسله مراتبی سه لایه ای برای گرید های محاسباتی پیشنهاد شده است که و مساله زمانبندی در گرید را تحت الشعاع قرار داده است. این ایده پیشنهادی به همراه شبکه پتری تصادفی و بصورت تطابقی شبیه سازی شده و نتایج بدست آمده از این شبیه سازی در GridSim لحاظ شده است.

در اینجا، زمانبندی کارها در شبکه پتری تصادفی که بصورت سلسله مراتبی²⁰⁸ لحاظ شده است. در اینجا این مدل را HSPN می نامیم که نام اقتباس شده از مدل آن می باشد. در اینجا درخواست ها به سه دسته تقسیم می شوند. لایه ها به نام های لایه خانگی²⁰⁹، لایه محلی²¹⁰ و لایه گرید²¹¹ تقسیم بندی شده اند. کار های ما در این سه لایه قرار می گیرند. هر لایه دارای یک سری توابع خاص برای دریافت زیر کارها²¹² و تحویل داده با لایه های پایین یا بالا دارند که در ادامه توضیح داده خواهد شد. در ادامه، ما نتایج بدست آمده از مدل HSPN را با سایر الگوریتم های مطرح موجود همچون Min.Min و Max.Min مقایسه خواهیم کرد. این نتایج نشان خواهند داد که HSPN نسبت به الگوریتم Max.Min نتیجه بهتری را ارائه می کند ولی از نتایج الگوریتم Min.Min در شرایط یکسان ضعیفتر است.

²⁰⁵ Complex

²⁰⁶ Heterogeneous

²⁰⁷ Autonomous

²⁰⁸ Hierarchical Stochastic Petri Net(HSPN)

²⁰⁹ Home Layer

²¹⁰ Local Layer

²¹¹ Grid Layer

²¹² Subtasks

مرور اجمالی بر گزید محاسباتی

در سال های اخیر کاربرد گزید محاسباتی افزایش قابل توجهی داشته است. در واقع، گزیده های محاسباتی شبکه ای از کامپیوتر ها می باشند که بطور مجتمع²¹³ قرار گرفته اند تا یک کامپیوتر مجازی با سرعت بسیار بالا را کارایی بالاتر از کارایی هر کامپیوتر را مهیا کنند. گزیده های محاسباتی از منابع بسیاری از کامپیوتر ها در شبکه استفاده می کند تا یک مساله علمی یا تکنیکی را در زمان یکسان هنگامی که نیاز به حجم وسیعی از اطلاعات یا انجام محاسبات بسیار زیاد می باشد حل کند. گروه های مختلفی در گزید محاسباتی در کنار هم مشارکت می کنند که به آن ها سازمان های مجازی²¹⁴ گفته می شود. VO ها برای حل یک مساله در کنار هم قرار می گیرند و با حل آن مساله همکاری شان به پایان می رسد و ناپدید می شوند (منظور آنست که بصورت مجازی کنار هم برای همکاری قرار می گیرند و پس از انجام کار مربوطه از حالت همکاری با هم خارج می شوند و به کار خود شان می پردازند). یکی از انتظاراتی که از گزید محاسباتی می رود امکان اجرای برنامه های مختلف در سایت های مختلف می باشد. هر سایت شامل چندین کامپیوتر و در واقع شامل چندین منبع می باشد. منابع در گزید بصورت نا همگون می باشند. بعضی از این برنامه ها نیازمند دسترسی به منابع مدیریت شده توسط واحدهای خودمختار می باشند. واحد مدیریت منابع همان زمانبندی منابع می باشند. تکنولوژی گزیده های محاسباتی این امکان را فراهم می کند که برنامه های مختلف که اغلب از سوی کاربران مختلف درخواست می شود که نسبت به هم ناهمگونی و توزیع شدگی دارند امکان دریافت منابع یکسان یا مختلف را بنا بر این سیاست مدیریت منابع یا زمانبندی منابع را داشته باشند. ابزارهای مختلفی برای مدل کردن و کنترل تخصیص منابع در گزید ارائه شده اند. از آنجایی که پویایی در خواست ها در گزیده های محاسباتی زیاد می باشند بنابر این ابزارهای شبیه سازی که توانایی همزمانی درخواست ها را کنترل کند نیاز می باشد. شبکه پتری تصادفی یکی از ابزارهای کارآمد برای این منظور می باشد. در ادامه به توضیح کلی روشهای تخصیص و زمانبندی منبع در شبکه پتری می پردازیم. در قسمت 4-4 توضیح کاملتری از ایده پیشنهادی خواهیم پرداخت. در قسمت 4-5 به

²¹³ Integrated

²¹⁴ Virtual Organizations(VOs)

ارزیابی ایده پیشنهادی با Min.Min و Max.Min می پردازیم و در انتها نتیجه گیری و کارهای آینده ذکر خواهد شد.

کارهای مرتبط در زمانبندی گزید محاسباتی به کمک شبکه پتری

یکی از مهمترین بخش ها در سیستم های گزید، چگونگی تخصیص منابع و زمانبندی آن ها در سیستم های محاسباتی می باشد. از گذشته تا به امروز، دانشمندان و محققین در علم شبکه و سیستم های گزید تلاش های زیادی را در این زمینه انجام داده اند. کارهای زیاد و گوناگونی در زمینه محاسبات گزید انجام شده است. آقایان یو²¹⁵ و بویا²¹⁶ در سال های 2005 و 2006 تحقیقات زیادی را در زمینه گردش کار و وظایف در گزید انجام داده اند [120][121]. آن ها یک چارچوب و ساختار کلی را در دسته های مختلف ارائه دادند که این چارچوب و الگوی پیشنهادی آن ها باعث می شد تا روند گردش کار در سیستم های گزید را تا حد قابل توجهی منظم تر، سریع تر و کاملتر کنترل کند. این مدل بر اساس الگوریتم های زمانبندی گراف جهت دار غیر حلقوی²¹⁷ می باشد که توانایی مدل کردن چندین درخواست را در زمان دارا می باشد. گراف جهت دار غیر حلقوی برای مدل کردن سیستم های پویا با خصوصیات و ویژگی های متنوع مناسب می باشد. اما، در مدل های پیچیده محیط های گزیدی مناسب نبوده و بدون هیچ تغییری در ساختار آن پاسخگوی نیازهای محیط نمی باشد. بنابر این، محققین به فکر استفاده از ابزارهای مدلسازی افتادند که این توانمندی را داشته باشد که بتواند همزمانی و همروندی برنامه ها و درخواست ها را با دقت و سرعت بیشتری پاسخ گو باشد. از این رو، خانواده شبکه های پتری همچون CPN^{218} [122] و SPN [34] بعنوان ابزاری کارآمد و کارا در زمینه زمانبندی و تخصیص منابع در گزید معرفی شدند. در [123] یک مدل سه لایه ای بر اساس شبکه پتری زمانی سلسله مراتبی²¹⁹ در سال 2005 ارائه شدند. این مدل بر اساس مدل های لایه ای در شبکه پتری برای زمانبندی در گزید های محاسباتی ارائه شده است. در این مقاله، سه لایه زمانبند محلی، خانگی و گزید معرفی شدند. این ایده براساس کارهای مستقل تعریف شده است و کارهای وابسته را

²¹⁵ Yu

²¹⁶ Buyya

²¹⁷ Directed Acyclic Graph(DAG)

²¹⁸ Colored Petri Net(CPN)

²¹⁹ Hierarchical Timed Petri Net(HTPN)

مورد بررسی قرار نمی دهد. مدل کردن کارهای وابسته در [124] توسط یاوجین²²⁰ و همکارانش بوسیله شبکه پتری گسترش یافته در سال 2006 ارائه شد. مدل زمانبندی برنامه ای در گریدهای بر اساس شبکه پتری بود در سال 2007 توسط ژاو²²¹ و همکارانش در [125] ارائه شد. در این بررسی، تمامی کارهای انجام شده بر روی زمانبندی گریدهای بر اساس خانواده شبکه پتری آورده شده است. وی در این مقاله، الگوریتم های زمانبندی را بصورت زمانبندی چهارسطحی ارائه داده است. در اینجا کارهای مستقل بطور کامل و واضح زمانبندی شده اند.

در ادامه، یک الگوریتم بر اساس شبکه پتری تصادفی سلسله مراتبی ارائه می دهیم. ما یک روشی برای زمانبندی و تخصیص منابع به کمک خصوصیات شبکه پتری معرفی کرده ایم که در ادامه با جزئیات بیشتر توضیح داده خواهد شد. در قسمت بعدی ایده پیشنهادی شرح داده خواهد شد. در بخش 4-5 ایده را با چند الگوریتم زمانبندی مقایسه می کنیم و نتایج را با هم در حالت های مختلف مقایسه می کنیم. در انتها، نتیجه گیری نهایی از ایده پیشنهادی و کارها آینده ذکر خواهد شد.

ایده پیشنهادی (HSPN)

همانطور که می دانیم، زمانبندی و تخصیص منابع در گریدهای محاسباتی مسئله مهمی است. دو روش برای تخصیص منابع به درخواست ها در سایت های مختلف وجود دارد. [126,127]

1. Distributed Scheme (طرح توزیع شده)
2. Hierarchical Scheme (طرح سلسله مراتبی)

روش اول طرحی است که برای تخصیص منابع به درخواست ها در سایت های مختلف بصورت توزیع شده می باشد یعنی در واقع بروکر ها بازای یک درخواست منابع را بصورت توزیع شده در نظر می گیرند و درخواست را بصورت توزیع شده به ایستگاه های کاری که حاوی منابع می باشند ارسال می کند و نتایج را از هماهنگ کننده منابع به آن صورت می گیرند.

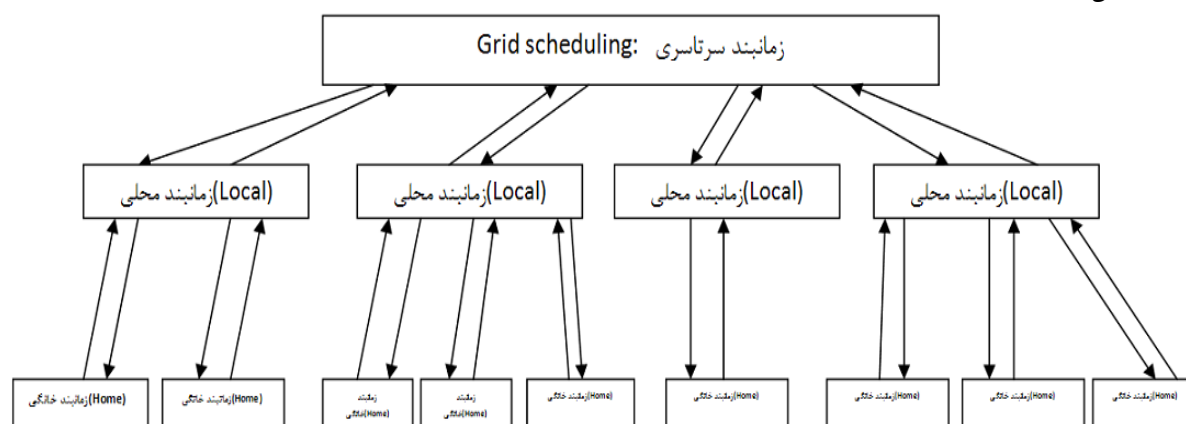
در روش دوم تخصیص منابع به درخواست ها با توجه به نوع درخواست ها در سایت ها که بصورت سلسله مراتبی مرتب شده اند صورت می گیرند. در نتیجه منابع را با توجه به شرایط محیطی و

²²⁰ Yaojin

²²¹ Zhao

هماهنگ کننده در سایت ها در سطوح مختلف دسته بندی می کنند و درخواست ها را بصورت سلسله مراتبی بروکر ها به ایستگاههای - حاوی منابع ارسال می کنند. طرح توزیع شده زمانی از طرح سلسله مراتبی بهتر خواهد بود که تمام درخواست ها روی سایتها بصورت محلی (locally) تقاضا داده شود. یعنی درخواستی به بروکر بیاید که منبع مورد نظر آن در همان سایت موجود باشد. در غیر اینصورت، در تخصیص منبع و پاسخ دهی به درخواست ها طرح دوم از اول بهتر می باشد. در اینجا زمانبندی در سه لایه تخصیص داده شده است. هر سایت می تواند دارای یک بروکر و یک زمانبند برای چگونگی تخصیص منبع داشته باشد.

در کل هم یک زمانبند سراسری در نظر گرفته می شود. با توجه به طرح سلسله مراتبی و توزیع شده ای که در [128] ذکر شده است یک الگوی زمانبندی را همانطور که در شکل 1-4 آورده شده است ارائه کرده ایم. در طرح زمانبندی سه-لایه ای منابع به یک شبکه سلسله مراتبی سه لایه ای متصل شده اند.



شکل 1-4: مدل سه لایه ای زمانبندی کارها یا درخواست ها

هر درخواست در زمانبند محلی مورد تحلیل و بررسی قرار می گیرد. در واقع محل ذخیره سازی منابع در لایه سوم یا لایه خانگی قرار گرفته اند. هر درخواست می تواند به زیر درخواست های زیادی بخش بندی شود. نحوه تقسیم بندی هر درخواست می تواند بصورت واحدهای مجزا (مستقل) یا واحدهای وابسته باشد. تخصیص واحدهای درخواستی بصورت وابسته به نتایج میانی پاسخ بخش های دیگر وابستگی دارد. در اینجا نیاز به داشتن روال وابستگی و نحوه تحولی اطلاعات بدست آمده در هر واحد درخواستی به سایر واحد مربوط می شود. در این ایده روال واحدهای مستقل مورد

بررسی قرار گرفته است. در ادامه، روال توابع در هر زمانبند را با جزئیات آورده ایم.

تابع های زمانبند خانگی:

- دریافت درخواست کاربر و تخصیص آن به منابع
- ارسال هر درخواست به لایه دوم یا زمانبند محلی
- تقسیم بندی زیر درخواست های آمده توسط زمانبند محلی و تخصیص آن ها به منابع
- مهیا کردن اطلاعات برای کامپیوتر های موجود و مرتبط با زمانبند خانگی برای زمانبند محلی

تابع های زمانبند محلی:

- دریافت اطلاعات و کارهای آمده از کامپیوتر های موجود و مرتبط با زمانبند خانگی که از زمانبند خانگی ارسال شده است
- ارسال درخواست کار به زمانبند سرتاسری (grid schedule)
- دریافت کارهای تخصیص داده شده توسط زمانبند گرید و تخصیص زیر درخواست ها به زمانبند محلی
- مهیا کردن اطلاعات از لایه دوم (Local Area Network) به زمانبند گرید

تابع های زمانبند گرید:

- دریافت اطلاعات از لایه دوم که توسط زمانبندهای محلی انجام شده است
- تخصیص کارها یا درخواستها به زمانبند محلی

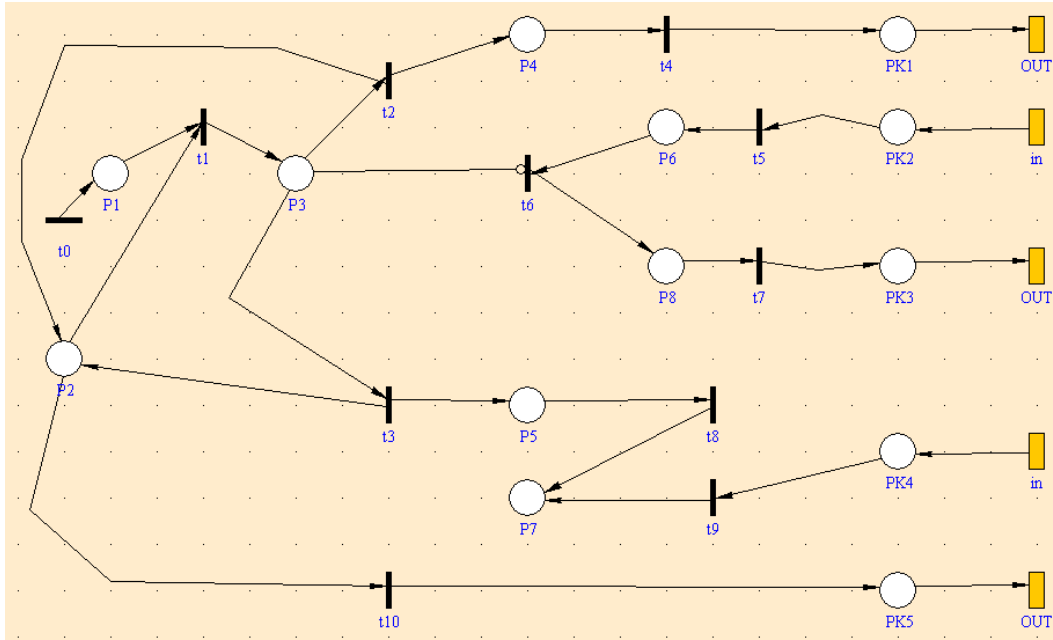
جدول 1-4 حاوی مکان ها و انتقال های در نظر گرفته شده را برای مدل شبیه سازی طرح سلسله مراتب در نظر گرفته شده می باشد.

جدول 1-4: نشانه های و توضیحات آن ها در HSPN

Place	توضیح	Transition	توضیح
P ₁	درخواست های تایید شده توسط کاربر	t ₀	تایید درخواست به کامپیوتر خانگی (ماشین خانگی)
P ₂	منابع	t ₁	دریافت درخواست و اطلاعات کامپیوتر خانگی (ماشین خانگی)

P ₃	آماده برای زمانبندی درخواست	t ₂	کار یا درخواست در زمان مقرر (Deadline) نتوانست به پایان برسد
P ₄	آماده برای تایید درخواست	t ₃	کار یا درخواست در زمان مقرر (Deadline) نتوانست به پایان برسد
P ₅	آماده برای اجرای درخواست	t ₄	ارسال کردن کار یا درخواست به زمانبند محلی
P ₆	آماده برای اجرا در خواست از راه دور (Remote)	t ₅	ارسال کردن کار یا درخواست به زمانبند خانگی
P ₇	درخواست به اتمام رسیده (Complete) یا کار تکمیل شده	t ₆ یا t ₈	اجرا کردن کار یا درخواست بصورت خانگی/راه دور (Remote)
P ₈	زیردرخواست هایی که بصورت remote تکمیل شده است	t ₇	تایید نهایی زیردرخواست های اجرا شده بصورت راه دور (Remote)
Pk ₁ ,Pk ₃ ,Pk ₅	واحد ها و بخش های خروجی تایید درخواست، تکمیل شدن کارهای کوچک یا زیردرخواست ها، حاوی اطلاعات ماشین خانگی (کامپیوتر خانگی)	t ₉	بازگرداندن درخواست پاسخ داده شده و به اتمام رسیده
Pk ₂ ,Pk ₄	واحد ها و بخش های ورودی تخصیص زیر درخواست ها یا همان کارهای کوچکتر شده که بصورت remote تخصیص پیدا کرده اند و تکمیل شدن و کامل شدن درخواست یا کار	t ₁₀	مهیا کردن اطلاعات ماشین یا کامپیوتر خانگی

در شکل 2-4 مدل HSPN نشان داده شده است. این مدل ایده پیشنهادی نشان داده شده است.



شکل 2-4: مدل ایده پیشنهادی HSPN

حال این مدل را به کمک GridSim مورد تست و آزمایش قرار می دهیم. در ابتدا توضیح اجمالی بر GridSim ارائه می کنیم. در نرم افزار GridSim در ابتدا هر درخواست به بروکر داده می شود و بروکر برای هر درخواست یک نهاد تعریف می کند و پروتکل های تعامل بین نهادهای²²² GridSim، با استفاده از رخدادهای پیاده سازی شده است. هر نهاد برای ارسال وضعیت خود به بروکر و زمانبندی ها آن را بصورت یک رخداد منتقل می کند. در GridSim نهادها هم برای درخواست اطلاعات و هم برای تحویل اطلاعات از رخدادهای استفاده می کنند. رخدادهای می تواند توسط هر نهادی تولید گردد تا بلافاصله و یا با تأخیر زمانی تعیین شده به نهادهای دیگر یا به خود او تحویل گردد. رخدادهایی که توسط همان نهاد تولید شده باشند رخدادهای داخلی²²³ و رخدادهایی که توسط نهادهای خارجی تولید شده باشند رخدادهای خارجی²²⁴ نامیده می شوند. نهادها می توانند این رخدادهای را با توجه به نشانی مبدأ رخداد که در کنار آن قرار دارد از هم تشخیص دهند.

²²² Gridlet

²²³ Internal Events

²²⁴ External Events

از پروتکل های GridSim برای تعریف خدمات نهاد استفاده می شود. با توجه به پروتکل های سرویس، رخدادهای GridSim را می توان به رخدادهای همزمان و ناهمزمان²²⁵ نیز تقسیم بندی نمود. یک رخداد را همزمان می گوئیم اگر نهاد مبدأ رخداد، صبر کند تا نهاد مقصد، تمام اعمالی که در ارتباط با رخداد است را انجام دهد (مانند تحویل یک سرویس کامل). یک رخداد را ناهمزمان می گوئیم اگر نهاد مبدأ رخداد، آن را تولید کند و سپس بدون صبر کردن برای کامل شدن آن، به فعالیت های دیگر خود ادامه دهد. هنگامی که نهاد مقصد چنین رخدادهای درخواست های سرویس را دریافت کرد، نتایج را به عنوان جواب با یک یا چندین رخداد به نهاد مبدأ برمی گرداند که خود می تواند باعث انجام اعمال مناسب بعدی گردد. باید توجه کرد که رخدادهای خارجی می توانند به صورت همزمان یا ناهمزمان تولید شوند ولی رخدادهای داخلی باید فقط به صورت ناهمزمان تولید شوند تا از به بن بست²²⁶ رسیدن نهاد جلوگیری شود. رخدادهای در زمان بندی گرید بصورت خارجی و در زمان بندی های دیگر داخلی عمل می کند. در ادامه به ارزیابی روش پیشنهادی با الگوریتم های Min.Min و Max.Min می پردازیم.

شبیه سازی ایده پیشنهادی

این بخش یکی از مهمترین بخش ها در این قسمت می باشد. در اینجا از پارامترهای یکنواخت و یکسان برای الگوریتم های Min.Min، Max.min و RHSPN استفاده شده است. در ابتدا باید هر یک از الگوریتم ها در نرم افزار Gridsim پیاده سازی شود.

در اینجا، از 4 ماشین (سایت) استفاده شده است که هر کدام از 4 پردازنده (کامپیوتر) تشکیل شده اند و توانایی اطلاعاتی هر سایت 1000 گیگابایت در نظر گرفته ایم. هر پردازنده منبع یکسانی دارد. در واقع پردازنده های هر ماشین (سایت) به طور همگن دارای منبع یکسانی می باشند ولی ماشین های هر سایت با ماشین های سایت دیگر از نظر حجم متفاوت می باشند. این بدان معنا است که هر سایت نسبت به سایت دیگر ناهمگن می باشند. یک درخواست را به 100 کار²²⁷ مجزا تقسیم می کنیم و آن ها را به هر ماشین اختصاص می دهیم. هر کار یک زمان و یک هزینه خاص خود دارد. توزیع زمان

²²⁵ Asynchronous

²²⁶ Deadlock

²²⁷ Task

و هزینه در هر کار بصورت تابع یکنواخت فرض شده است. در واقع تعداد منابع ما 16 منبع می باشند که با توجه به فضای Heap در نظر گرفته شده در جاوای نرم افزار Gridsim 100 کاربر در نظر می گیریم که هر کاربر یک کار را به بروکر GridSim درخواست می دهد. کارها با توجه به جدول 2-4 به 4 گروه تقسیم بندی شده اند. هر گروه شامل 25 کار می باشند.

جدول 2-4: دسته بندی 100 کار به 4 گروه 25 تایی و شرایط لحاظ شده برای هر کار

در نظر گرفتن هزینه	در نظر گرفتن زمان	تعداد کارهای در نظر گرفته شده
در نظر نمی گیریم	در نظر نمی گیریم	25 کار اول
در نظر نمی گیریم	در نظر می گیریم	25 کار دوم
در نظر می گیریم	در نظر نمی گیریم	25 کار سوم
در نظر می گیریم	در نظر می گیریم	25 کار چهارم

منظور از در نظر نگرفتن زمان یا هزینه برای هر کار یعنی چگونگی جستجو در الگوریتم Min.Min و Max.min می باشد. الگوریتم RHSPN این مسائل تغییری در مقدار آن ایجاد نمی شود.

هر دسته کارها از کم محتواترین²²⁸ درخواست تا پرمحتواترین²²⁹ درخواست مرتب شده اند. کم محتواترین درخواست یعنی درخواستی که پاسخگویی به آن راحت می باشد و نیاز به منابع زیادی نمی باشد. در این شبیه سازی، این درخواست هایی که تعداد منابع درخواستی آن ها کمتر یا مساوی 3 منبع می باشند منظور می باشند. پرمحتواترین درخواست یعنی درخواستی که پاسخگویی به آن دشوار-تر(زمان و هزینه بیشتر) می باشد و نیاز به مساوی یا بیش از 8 منبع می باشند.

برای پیاده سازی ایده ذکر شده در دو الگوریتم Min.min و Maxmin کافی است تغییراتی در کد بروکر Gridsim ایجاد کنیم. یک پکیج ایجاد می کنیم و آن را در Gridsim اضافه می کنیم. کلاس Main

²²⁸ Thin Request

²²⁹ Thick Request

Task ، Job و Node الگوریتم های Min.min و Max.min در پیوست آورده شده است. در ادامه توضیح اجمالی در مورد دو الگوریتم Min.min و Max.min می دهیم [128,129].

- **الگوریتم Min.min :** در این الگوریتم حداقل زمان کامل شدن هر Task در کل ماشین ها محاسبه می شود. Task ای که با حداقل زمان کلی کامل شدن می باشد بین Task ها انتخاب می شود و به ماشین مورد نظر اختصاص می یابد. وقتی کار Task مورد نظر به اتمام رسید. آن Task از لیست Task ها خارج می شود و این روال برای Task های باقی مانده ادامه می یابد تا تمام Task ها به منابع مورد نظر خودشان دسترسی پیدا کنند.

- **الگوریتم Max.min :** این الگوریتم بسیار شبیه به الگوریتم Min.min می باشد. مجموعه ای از زمان های حداقل برای هر Task محاسبه می شود. Task ای که زمان کامل شدن حداکثری در مجموعه اش داشته باشد انتخاب می شود و به ماشین مورد نظرش اختصاص داده می شود.

همانطور که در قسمت 4-4 ذکر شد در ابتدا در ماشین خانگی شروع به جستجو در پایگاه داده می کنیم. زمانبند جستجو در هر ماشین زمانبند خانگی گفته می شود. در صورتی که منبع مورد نظر را با توجه به درخواست پیدا نشود این درخواست به هماهنگ کننده سایت ارسال می شود تا در سایت محلی شروع به جستجو کند. بنابر این، زمانبند محلی فعال شده و شروع به جستجو در محدوده تعیین شده اش می کند. همچنین، در صورتی که منبع مورد نظر را با توجه به پارامترهای مورد نظر پیدا نشود یک درخواست به بروکر ارسال می شود تا در سایت های دیگر شروع به جستجو کند. در این هنگام، زمانبند گزید فعال می شود و شروع به جستجو می کند. در ارزیابی کارایی چندین پارامتر مورد بررسی قرار گرفته است که در زیر ذکر شده است.

- **زمان متوسط پاسخگویی بازای هر درخواست²³⁰(ART)**

یکی از مهمترین پارامترها برای ارزیابی می باشد. این پارامتر نشاندهنده توالی بین دریافت پاسخ و ارسال پاسخ می باشد که زمان پاسخگویی نامیده می شود. زمان پاسخگویی در صفحات وب برای ارسال/دریافت پاسخ ها مهم می باشد. زمان متوسط پاسخگویی در HSPN همان مجموع زمان ها برای

²³⁰ Average Response Time per Request

عبور یک در خواست از هر سه لایه می باشد. ممکن است Task هایی باشند که که مدت زمان مورد نظر در زمانبند خانگی و محلی پاسخگو نباشند بنابر این بروکر باید نزدیکترین منبع را بر اساس پارامترهای هر منبع انتخاب کند. هرچه مدت زمان پاسخگویی به یک در خواست کوتاهتر باشد نشاندهنده آنست که درخواست در لایه های داخلی پاسخ مطلوب را گرفته است. در اینجا، این پارامتر را برای هر کار در نظر می گیریم و آن را برای هر سه الگوریتم مورد بررسی قرار می دهیم.

• **هزینه متوسط پاسخگویی بازای هر درخواست²³¹(ARC)**

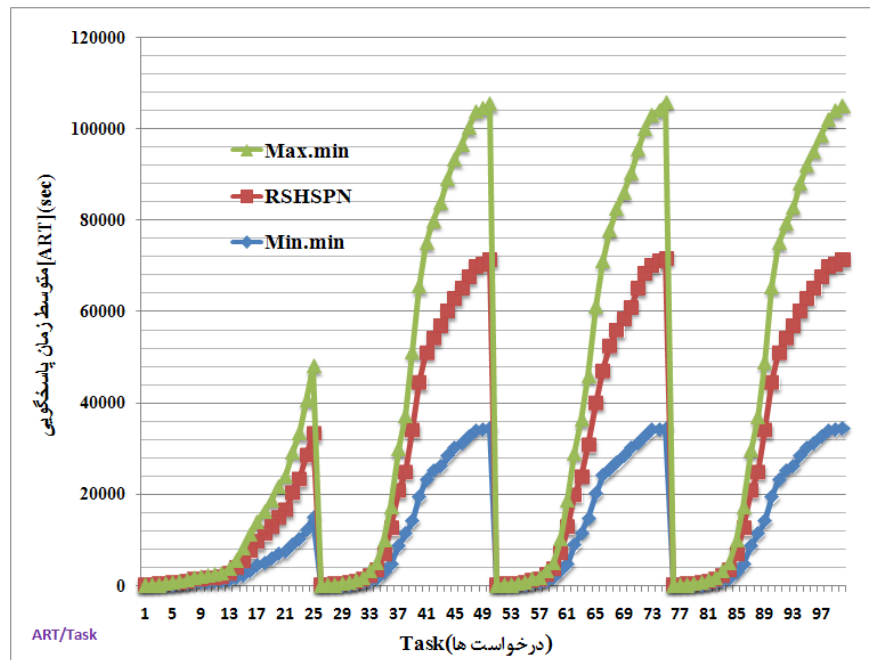
این پارامتر نیز در وب برای کارها از اهمیت ویژه ای برخوردار است. هزینه تحویل دادن داده یا منبع برای هر Task را هزینه پاسخگویی می نامیم. هزینه متوسط پاسخگویی در HSPN مجموع هزینه عبور کردن از سه لایه (در صورت لزوم) برای اختصاص منبع مناسب به Task مورد نظر می باشد. این را در نظر بگیرید که برخی از درخواست ها نیاز نیست که به لایه های خارجی همچون لایه گرید و یا لایه محلی بروند. پس در مقایسه با سایر روش ها میزان هزینه کمتری را مصرف می کنند.

تمام الگوریتم های زمانبندی در طراحی و اجرای خود این پارامتر را مورد بررسی قرار می دهند. این الگوریتم ها منبع را در نزدیکترین حالت برای سایت Task در نظر می گیرند و هزینه آن ها از هزینه اجرا هر Task کمتر می باشد. اما، ممکن است منابعی وجود داشته باشند که هزینه آن ها از هزینه مورد انتظار ما کمتر باشند ولی در اینجا در نظر گرفته می شوند. چون ممکن است زمان دسترسی به آن نسبت به منابع دیگر بخاطر نزدیک بودن یا در سایت کناری بودن کمتر باشد. بطور مثال، اگر Task اول منبع مورد نظر خود را با هزینه \$300 پیدا کند، ما میزان $300 + (100/1) * 300$ یعنی \$303 را در نظر می گیریم. این بدین معنی است که ما به دنبال منابع با هزینه کمتر از \$303 می گردیم.

شکل 3-4 زمان متوسط پاسخگویی برای هر درخواست از صد Task در نظر گرفته شده برای یک کاربر نشان داده شده است. این نمودار نشان می دهد که الگوریتم HSPN از الگوریتم Max.min در هر 4

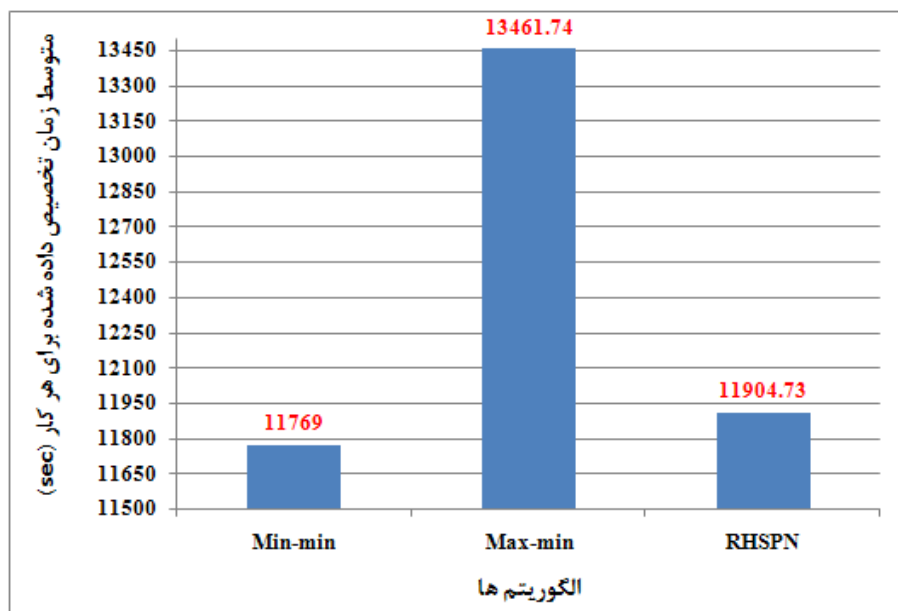
²³¹ Average Response Cost per Request

دسته بهتر عمل می کند و ولی در هر چهار حالت از الگوریتم Min.min زمان بیشتری دارد. در اینجا حداکثر زمان 36000 واحد و حداکثر هزینه \$3000 در نظر گرفته شده است. میزان زمان از Task بیست و ششم تا Task یکصدم افزایش قابل توجهی داشته است. همانطور که می بینید در درخواست های پر محتوی میزان ART در HSPN بسیار به الگوریتم Min.min نزدیکتر می باشد و گاهی اوقات از آن نیز بهتر شده است.



شکل 3-4: زمان متوسط پاسخگویی برای هر درخواست بازای 100 درخواست

همانطور که در شکل 4-4 نشان داده شده است متوسط زمان اختصاص داده شده به هر کار برای هر الگوریتم نشان داده شده است. همانطور که می بینیم الگوریتم RHSPN به میزان 150 واحد زمانی از الگوریتم Min.min زمان متوسط مصرفی هر کارش بیشتر می باشد و آن هم به هزینه تعویض بستر از یک لایه به لایه دیگر مربوط می باشد. این الگوریتم نزدیک به 1500 واحد از الگوریتم Max.min زمان کمتری مصرف می کند.



شکل 4-4: متوسط زمان تخصیص داده شده برای هر کار

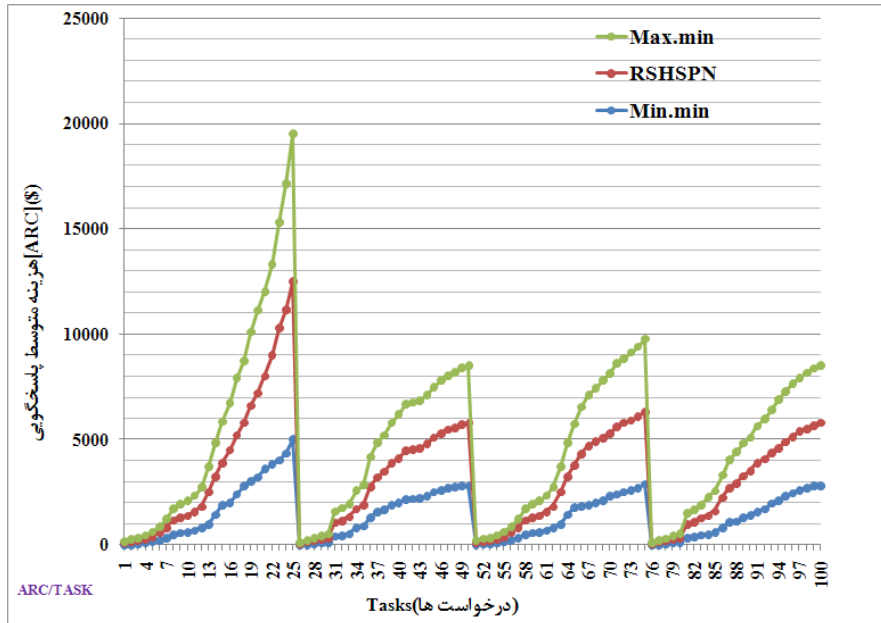
شکل 4-5 میزان هزینه متوسط پاسخگویی برای هر درخواست برای یک کاربر آورده شده است. این نمودار نشان می دهد برای Task 25 اول (دسته اول) بخاطر عدم در نظر گرفتن هزینه و زمان هر سه الگوریتم مقدار هزینه بالایی را برای هر Task بدون در نظر گرفتن پریا محتوی بودن آن در نظر می گیرند. اما در دسته دوم Task ها، ما حداکثر هزینه را 3000 دلار را برای هر کار در نظر گرفته ایم. در اینجا زمانبندها بر اساس هزینه در نظر گرفته شده باید منابع مورد نظر را برای هر درخواست در نظر بگیرند. میزان در نظر گرفتن هزینه برای جستجوی منابع بر اساس رابطه 1-2 می باشد.

$$\text{Cost(Task)} + 1\% \text{Cost(Task)}$$

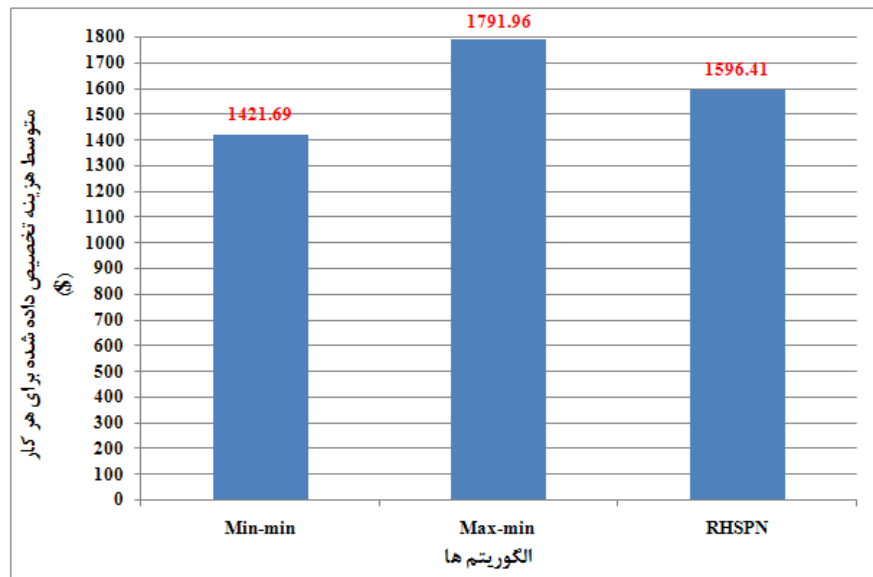
رابطه (4-1)

همانطور که نشان داده شده است HSPN هزینه را با محدود کردن دسترسی به منابع کاهش می دهد اما این میزان کاهش هزینه بهتر از الگوریتم Min.min نمی باشد، ولی، از الگوریتم Max.min بهتر می باشد. این الگوریتم در درخواست های پرمحتوی مثل الگوریتم Min.min عمل می کند و حتی گاهی از الگوریتم Min.min بهتر عمل می کند. همانطور که در شکل 4-6 نشان داده شده است متوسط هزینه اختصاص داده شده به هر کار برای هر الگوریتم نشان داده شده است. همانطور که می بینیم الگوریتم HSPN به میزان 170 واحد از الگوریتم Min.min هزینه متوسط مصرفی هر کارش بیشتر می باشد و آن

هم به هزینه تعویض بستر از یک لایه به لایه دیگر مربوط می باشد. این الگوریتم نزدیک به 200 واحد از الگوریتم Max.min هزینه کمتری مصرف می کند.



شکل 4-5: هزینه متوسط پاسخگویی برای هر درخواست بازای 100 درخواست



شکل 4-6: متوسط هزینه تخصیص داده شده برای هر کار

با توجه به تست انجام شده بر روی الگوریتم پیشنهادی بازای Task 100 می‌توان گفت که این الگوریتم زمانبندی را تا حدود 10% زمان و 11% هزینه را نسبت به روش Max.min بهبود داده است.

خلاصه و نتیجه گیری و کارهای آینده

این را مد نظر داشته باشید که هیچ راه بهینه ای برای تخصیص منابع و زمانبندی آن ها در سیستم های گرید وجود ندارد. زیرا، بخاطر ناهمگونی بودن، پویایی و گستردگی این سیستم ها در دنیا این امر امکان پذیر نمی باشد.

در این بخش، یک طرح و الگوی زمانبندی منبع با استفاده از سه لایه ای بودن محیط های گرید ارائه شد. طرح زمانبندی منابع توسط شبکه پتری تصادفی سلسله مراتبی مدل شد و مورد آنالیز و تحلیل قرار گرفت. در این جا، برای لایه های سه گانه زمانبندی های مختلف ارائه شد (زمانبند خانگی، زمانبند محلی و زمانبند گرید) و خصوصیات و توابع مورد استفاده و نحوه چگونگی ارتباط زمانبندها در مدل پیشنهادی مورد بررسی قرار گرفت. ایده پیشنهادی (RHSPN) با الگوریتم های Min.min و Max.min که از الگوریتم های پرکاربرد در سیستم های گرید می باشند مورد بررسی قرار گرفت. روش RHSPN بر اساس شبیه سازی SPN در شبکه گرید کار می کند. نتایج در نرم افزار Gridsim مورد بررسی قرار گرفتند. این نتایج که بر روی 100 کار با طول یکسان مورد بررسی قرار گرفتند نشان داد که الگوریتم پیشنهادی در زمان و هزینه به ترتیب 10% و 11% کمتر از الگوریتم Min.min زمانبندی در خواست ها را انجام می دهد. همچنین، زمانبند گرید که در ساختار خارجی و نهایی مدل پیشنهادی قرار گرفته است توانایی جستجو و تخصیص منابع را به کارها و یا زیر کارها دارا می باشد.

از کارهای که در آینده بر روی این ایده صورت خواهد گرفت پیاده سازی ایده پیشنهادی در کارهای وابسته به هم و استفاده از الگوریتم های هوشمند در بهبود ایده پیشنهادی در زمانبندی و تخصیص منابع می باشد.

فصل پنجم

کاربرد شبکه پتری تصادفی تطابقی
در گزید اقتصادی

5 پیاده سازی شبکه پتری تصادفی تطابقی در گرید اقتصادی

مروری بر الگوریتم های موجود و الگوریتم پیشنهادی

الگوریتمی که برای بهینه سازی زمان توسط بویا در [92] پیشنهاد شده و در [95] مورد ارزیابی قرار گرفته است، الگوریتمی مکاشفه ای است که برای کارهای ناهمگون جوابهای قابل قبولی را به کاربر ارائه نمی کند. بویا این الگوریتم را برای زمانبندی برنامه های کاربردی پارامترروب طراحی کرده است. از آنجا که ویژگی این برنامه ها این است که از تعدادی کارهای همگون مستقل از هم تشکیل شده اند، الگوریتم گزارش شده فرض می کند که کارها دارای ناهمگونی بسیار کمی هستند و به همین دلیل برای کارهای ناهمگون نمی تواند زمانبندی مناسبی را برای کاربر انجام دهد. بنابراین این انگیزه وجود می آید که با استفاده از الگوریتم های مکاشفه ای جدید و الگوریتم های یادگیری بتوان نتایج موجود را بهبود داد. همچنین، بویا منابع را زمان مشترک فرض می کند و باعث می شود که پیچیدگی ها و نقص هایی در الگوریتم او وجود داشته باشد.

در ادامه، آقای مهدوی فر پس از بررسی و شبیه سازی روش های مختلف برای بهینه سازی زمان، الگوریتم هایی را برای این کار پیشنهاد کرد. ما الگوریتمی تحت عنوان ALATO برای ایجاد توازن بار در تخصیص کارها به منابع گرید محاسباتی ارائه داده ایم. به این ترتیب، الگوریتم های موجود برای بهینه سازی زمان عبارتند از:

- | | |
|---|------------|
| Buyya Time Optimization (BTO) | (1) |
| Advanced Buyya Time Optimization (ABTO) | (2) |
| Extended Buyya Time Optimization (EBTO) | (3) |
| Advanced Extended Buyya Time Optimization (AEBTO) | (4) |
| Learning Automata Time Optimization (LATO) | (5) |
| Advanced Learning Automata Time Optimization (ALATO) | (6) |

در نتیجه می توان گفت که 6 الگوریتم برای بهینه سازی زمان در گریدهای محاسباتی ارائه شده است که الگوریتم BTO توسط بویا و الگوریتم های ABTO, EBTO, AEBTO, LATO توسط آقای مهدوی فر ارائه شده اند و الگوریتم ALATO توسط نگارنده پیشنهاد شده است. چهار الگوریتم اول روش های مکاشفه ای هستند و الگوریتم پنجم و ششم با

استفاده از یادگیری و توسط مجموعه‌ای از اتوماتاهای یادگیر، عمل زمانبندی را انجام می‌دهد. هدف همه‌ی این الگوریتم‌ها این است که بتوانند از بودجه‌ی تعیین شده توسط کاربر، استفاده‌ی حداکثر را کرده و زمان اجرای برنامه را تا جایی که امکان دارد کاهش دهند. الگوریتم ABTO با تغییراتی جزئی در الگوریتم BTO بدست آمده است و بدون دستکاری هسته‌ی اصلی الگوریتم، نتایج بهتری را ارائه می‌کند. الگوریتم EBTO از نظر مکانیزم زمانبندی با الگوریتم BTO متفاوت است ولی از ایده‌ی مطرح شده توسط بویا برای بهینه‌سازی زمان استفاده می‌کند. الگوریتم AEBTO تغییراتی را در الگوریتم EBTO ایجاد می‌کند که به تولید زمانبندی‌های بهتر منجر می‌شود. الگوریتم LATO با استفاده از اتوماتاهای یادگیر سعی می‌کند زمان را با صرف بودجه‌ی تعیین شده به حداقل برساند. الگوریتم ALATO با تغییراتی در نحوه پیاده‌سازی و پرداخت پاداش و جریمه در الگوریتم LATO شکل گرفته است و بهترین نتیجه را در میان الگوریتم‌های فوق بدست می‌آورد. در ادامه یک توضیح اجمالی در مورد خصوصیت مشترک الگوریتم‌ها می‌پردازیم.

5-1-1- مراحل مشترک الگوریتم‌ها

الگوریتم‌های بهینه‌سازی، در ابتدای کار و قبل از انجام زمانبندی، اقداماتی را انجام می‌دهند که در همه‌ی آنها مشترک است. این اعمال را می‌توان در سه مرحله به صورت زیر بیان کرد:

1. **یافتن منابع:** شناسایی منابعی که می‌توانند در اجرای کارها مورد استفاده قرار بگیرند و همچنین بدست آوردن توانمندی‌ها و ویژگی‌های آنها. این اعمال از طریق سرویس اطلاعاتی گرید انجام می‌گیرد.

2. **تجارت منابع:** شناسایی هریک از منابع بر حسب هزینه در واحد زمان (پول در ثانیه، $G\$/sec$). سپس تشخیص میزان توانمندی منبع در واحد زمان (نرخ اجرای میلیون دستورالعمل در ثانیه، MI/sec) و در نهایت بدست آوردن قیمت واقعی و مفید منبع که همان هزینه‌ی اجرای هر میلیون دستورالعمل می‌باشد (پول به ازای هر میلیون دستورالعمل، $G\$/MI$)

3. **مرتب‌سازی منابع:** یک الگوریتم بهینه‌سازی، حتماً به یک ترتیب خاص از منابع بر حسب قیمت آنها نیاز دارد. در این مرحله، منابع به ترتیب صعودی از نظر قیمت مفید (که در مرحله‌ی قبل بدست آمده است) مرتب‌سازی می‌شوند. در واقع اولویت‌واگذاری کار با منابع ارزان‌تر می‌باشد. اگر دو منبع دارای قیمت مفید

یکسان باشند، منبعی که توانمندی بیشتری دارد در اولویت قرار می‌گیرد تا کارایی الگوریتم تا حد ممکن بیشتر باشد. در واقع برای بدست آوردن بهترین ترتیب، منابع با دو پارامتر مرتب می‌شوند: 1. قیمت مفید 2. توانمندی

5-1-2- الگوریتم BTO و الگوریتم ABTO

این الگوریتم توسط بویا در [92,93] پیشنهاد شده و در [95] مورد ارزیابی قرار گرفته است. عمل بهینه‌سازی زمان نسبت به بهینه‌سازی هزینه دارای پیچیدگی بیشتری است و نمی‌توان با یک ایده‌ی ساده، زمانبندی را انجام داد. بویا برای زمانبندی کارها با هدف بهینه‌سازی زمان ایده‌ی زیر را مطرح می‌کند:

- با در نظر گرفتن مقداری از بودجه برای هر کار، منبعی را برای یک کار انتخاب می‌کنیم که بتواند اجرای آن کار را زودتر از منابع دیگر به اتمام برساند و هزینه‌ی انجام کار بر روی آن از بودجه‌ی اختصاص یافته به کار کمتر باشد.

ایده‌ی اختصاص دادن بودجه به هر کار تنها عملی است که می‌توان با استفاده از آن بودجه‌ی تعیین شده توسط کاربر را به طور کامل مصرف کرده و بهینه‌سازی زمان را انجام داد. عمل اختصاص بودجه به کارها عمل ساده‌ای نیست و می‌توان این عمل را با روش‌ها و معیارهای مختلفی انجام داد. در روش BTO این عمل با استفاده از معیار طول کارها انجام می‌گیرد که یک معیار معقول به نظر می‌رسد. در این روش هر کار به اندازه‌ی طول خود بودجه دریافت می‌کند. الگوریتم BTO به این صورت عمل می‌کند که در ابتدا کارها را به طور صعودی از نظر طول مرتب می‌کند.

سپس از اولین کار در صف شروع می‌کند و به اندازه‌ی طول آن نسبت به مجموع طول همه‌ی کارها به آن بودجه اختصاص می‌دهد و مقدار اختصاص داده شده را از کل بودجه کسر می‌کند. سپس منابع ممکن از بین همه‌ی منابع انتخاب می‌شود. منابع ممکن منابعی هستند که هزینه‌ی اجرای کار بر روی آنها از بودجه‌ی اختصاص یافته به کار کمتر است. سپس از بین این منابع، کار به منبعی نگاشت می‌شود که بتواند زودتر از منابع دیگر اجرای کار را تمام کند. البته این منبع لزوماً توانمندترین منبع نیست زیرا قبلاً کارهایی به این منابع واگذار شده است و بار کاری بیشتری تأثیر را در زمان اتمام اجرای یک کار خواهد داشت. البته اگر همه‌ی منابع ممکن دارای بار کاری یکسانی باشند آنگاه مطمئناً

منبع انتخابی منبعی است که از دیگران توانمندی بیشتری دارد (نرخ اجرای دستورالعمل در آن از بقیه بیشتر است) زیرا می‌تواند کار را زودتر تمام کند. پس از نگاشت کار می‌دانیم هزینه‌ی اجرای کار بر روی منبع انتخابی از بودجه‌ی اختصاص یافته کمتر است بنابراین هزینه‌ی اجرا از بودجه کسر شده و بقیه‌ی آن به بودجه‌ی کل برگردانده می‌شود. همچنین طول کار نیز از مجموع طول کارها کم می‌شود. سپس الگوریتم زمانبندی به سراغ دومین کار می‌رود و همین عملیات را تکرار می‌کند. این عمل ادامه می‌یابد تا همه‌ی کارها به منابع نگاشت شوند. با انجام عمل‌های ذکر شده مرحله‌ی زمانبندی به پایان می‌رسد و بعد از آن عمل واگذاری کارها در مرحله‌ی توزیع انجام می‌شود (شکل 5-1). در این مرحله به هر منبع حداکثر به تعداد پردازنده‌های بیکار آن، کار واگذار می‌شود. همانطور که پیش از این نیز گفته شد، بویا از زمانبندی مرحله‌ای برای تخصیص منابع استفاده می‌کند. یعنی دو مرحله‌ی زمانبندی و توزیع به عنوان یک مرحله آنقدر تکرار می‌شود تا عمل واگذاری همه‌ی کارها انجام شود. البته بین هر دو تکرار باید برای مدت زمانی، الگوریتم متوقف شود تا پردازنده‌های بیکار برای توزیع کارها بوجود بیاید. جزئیات بیشتر در توضیحات زمانبندی مرحله‌ای آمده است.

تا زمانی که کارهای پردازش نشده وجود دارد و از محدودیت‌های بودجه و مهلت تجاوز نکرده ایم تکرار کن:

1. **زمانبندی:** برای هر کار در صف کارهای پردازش نشده انجام بده:

a. با توجه به نسبت طول کار به مجموع طول کارهای باقیمانده، بودجه‌ای را به آن اختصاص بده و از بودجه‌ی کل کم کن.

b. گروهی از منابع را که هزینه‌ی اجرای کار بر روی آنها از بودجه‌ی اختصاص داده شده کمتر است از بقیه‌ی منابع جدا کن.

c. در گروه منابع انتخاب شده، منبعی را که بتواند زودتر از منابع دیگر گروه، اجرای کار را تمام کند انتخاب کرده و کار را به آن منتسب کن.

d. هزینه‌ی اجرای کار بر روی منبع انتخابی را از بودجه‌ی اختصاص داده شده کم کرده و مابقی را به بودجه‌ی کل برگردان.

e. کار را از صف کارهای پردازش نشده حذف کن.

2. **توزیع:** تعداد کارهایی را که می‌توان به یک منبع

واگذار کرد، بدون اینکه با حالت **overload** مواجه شود، مشخص کرده و آنها را واگذار کن. سپس بقیه‌ی کارها را به صف کارهای پردازش نشده برگردان. سیاست پیش‌فرض این

شکل 5-1: الگوریتم BTO

کد این الگوریتم در شکل 5-2 آورده شده است.

```
#include "stdafx.h"
#include "conio.h"
double User_Time, User_Cost, Sum, cost, Sum1, Avg_Sum;
//Time For User and Cost For User Assigned
Q1, Q2: Queue;
struct Task{
    int No[200];
    double cost[200];
    double T[200];
}T[200];
//tahrif task ha-200 task
struct Resource{
    int flage=0;
    double cost[8];
    double Time[8];
}R[200];
int k=0;
//Q1: saf karhaye pardazesh nashode
//Q2: saf karhaye pardazesh shode va ejra shode
```

(قسمت)

اول)

```

int _tmain(int argc, _TCHAR* argv[])
{
    for (int j=1;j<200;j++)
        { insert(Q1,T1);Order_ASC_Q1;}//ascending 200 tasks (قسمت دوم)
    for (int i=1;i<20;i++)
    {
        while (Q1!= Empty()) do
        {
            k=1;
            for (int j=1;j<200;j++)
            {
                while(k!=9)do
                {
                    if((R[j].flage==0)&&(R[j].Time[k]<=T[j].No[j])&&(Sum<=User_Time)
                    &&(cost<=User_Cost))
                    {
                        Sum=Sum+T[j].T;
                        cost=cost+R[j].cost[k];
                        insert(T[j].No[j],Q2);
                        //ezafe kardan task be saf Q2
                        Delete(T[j].No[j],Q1);
                        //hazf kardan task be saf Q1
                        R[k].flage=1;
                        k++;
                    }
                    //if
                    else if (R[k].flage==1) k++;
                    else if (R[k].Time[k]>T[j].No[j]) k++;
                    else if (Sum>User_Time)
                        printf("Resource Allocation Mode
error for Not Compiling Assigment Because of Time");
                    else if (cost>User_Cost)
                        printf("Fatal Error for Not
Sufficiant Deadline Cost");
                    else if ((k>=9)&&(j<200)) Wait( Untile Max-
Cost(R1.R8)); Break();

                    }//while-2
                    if (Qi==Empty()) Sum1=Sum1+Sum;
                }//while-1
                Avg_Sum=Sum1/20;
                printf("Mizan natayej baraye 20 bar azmayesh BTO baraye
200 Tasks be ezaye nahie tain shode baraye task ha");
            }
        }
    }
    return 0;
}

```

شکل 5-2: کد برنامه الگوریتم BTO

الگوریتم ABTO تغییری در روند الگوریتم BTO ایجاد نمی‌کند؛ بلکه تنها با توجه خاص به ترتیب بررسی و واگذاری کارها، کارایی الگوریتم را بهبود می‌دهد. در الگوریتم BTO توجه خاصی به ترتیب کارها نمی‌شود زیرا این الگوریتم برای زمانبندی کارهای همگون طراحی شده است. به هر حال بویا ترتیب صعودی کارها از نظر طول را برای مرتب‌سازی انتخاب می‌کند. در این الگوریتم صف کارهای پردازش نشده همواره بر حسب طول کارها به صورت صعودی مرتب می‌شود.

همانطور که در ایده‌ی اصلی بهینه‌سازی زمان ذکر شد، هر کار با توجه به بودجه‌ی خود به منبعی واگذار می‌شود که بتواند زودتر اجرای او را تمام کند. اگر در ابتدا کارهای کوچک را زمانبندی کنیم و در پایان کارهای بزرگتر را بین منابع توزیع کنیم، اختلاف زمان اجرای کارها در منابع زیاد می‌شود و در نتیجه اختلاف زمان اجرا در پرکارترین منبع با منابع دیگر زیاد خواهد بود؛ زیرا به علت محدودیت بودجه‌ی باقیمانده نمی‌توان به راحتی شرایط انتقال یک کار بزرگ را از پرکارترین منبع به یک منبع سریع‌تر فراهم کرد و این وضعیت برای بهینه‌سازی زمان مطلوب نیست. در حالی که اگر ابتدا کارهای بزرگتر را توزیع کنیم و سپس به کارهای کوچکتر بپردازیم، می‌توانیم در پایان زمانبندی، با استفاده از بودجه‌ی باقیمانده، این کارهای کوچک را طوری بین منابع توزیع کنیم که زمان لازم برای اجرای کارهای واگذار شده در پرکارترین منبع نسبت به منابع دیگر اختلاف زیادی نداشته باشد. بنابراین برای کاهش زمان اجرای کارهای برنامه‌ی کاربردی متعلق به کاربر (زمان اجرای کارهای واگذار شده به پرکارترین منبع) که در واقع هدف یک الگوریتم بهینه‌سازی زمان است، بهتر است که ترتیب واگذاری کارها از نظر طول کارها به صورت نزولی باشد.

برای رسیدن به کارایی بالاتر، الگوریتم ABTO نسبت به الگوریتم BTO دارای یک عمل اضافه با تأکید خاص خواهد بود:

- در ابتدای مرحله‌ی زمانبندی در الگوریتم BTO، صف کارهای پردازش نشده را بر حسب طول کارها به صورت نزولی مرتب کن.

با افزودن عمل فوق به الگوریتم BTO، به الگوریتم ABTO دست پیدا می‌کنیم که دارای کارایی بالاتری خواهد بود. این الگوریتم نیز از روش زمانبندی مرحله‌ای استفاده می‌کند و تنها تفاوت آن با الگوریتم پیشنهادی بویا در ترتیب بررسی کارها می‌باشد.

بنابر این قسمت دوم در کد شکل 5-2 تغییر می کند. این تغییر بصورت شکل 5-3 خواهد بود که جایگزین قسمت دوم در کد شکل 5-2 خواهد شد.

```
for (int j=1;j<200;j++)
{
    insert(Q1,T1);Order_DSC_Q1;}//Desscending 200 tasks
```

شکل 5-3: کد قسمت دوم در ABTO

3-1-5- الگوریتم EBTO

الگوریتم EBTO از تغییر الگوریتم BTO و استفاده از روش زمانبندی یکباره حاصل می شود؛ البته همچنان از ایده ی اصلی این الگوریتم استفاده می کند و روش اختصاص بودجه به کارها را نیز تغییر نمی دهد. همانطور که گفته شد، یکی از نقاط ضعف الگوریتم BTO این است که عمل زمانبندی و نگاشت کارهای پردازش نشده را در هنگام اجرای کارهای واگذار شده به منابع انجام می دهد و در واقع از روش زمانبندی مرحله ای استفاده می کند. ولی الگوریتم EBTO از روش زمانبندی یکباره استفاده می کند که در آن فرض می شود همه ی منابع به صورت فضا مشترک رفتار می کنند. همانطور که قبلاً گفته شد این روش تنها راهکار قابل استفاده در گریدهای اقتصادی برای زمانبندی کارهای متعلق به چندین کاربر مختلف به صورت همزمان می باشد. الگوریتم EBTO سعی می کند با استفاده از ایده ی الگوریتم BTO، پس از نگاشت هر کار به یک منبع، بلافاصله عمل واگذاری آن را به منبع انتخابی انجام دهد. به این ترتیب با زمانبندی یکباره، امکان سرویس دهی همزمان به چندین کاربر را فراهم می کند. همچنین از آنجا که این الگوریتم، کارها را به صورت نزولی مرتب می کند، از نظر کارایی نیز الگوریتم BTO را بهبود می دهد.

الگوریتم EBTO در ابتدا کارها را از نظر طول به صورت نزولی مرتب می کند. سپس با استفاده از روشی که بویا برای اختصاص بودجه به کارها مطرح کرده بود، اولین کار را از مجموعه ی کارها جدا کرده و بودجه ای را به آن اختصاص می دهد. سپس از بین منابعی که می توانند این کار را با هزینه ای کمتر از بودجه ی اختصاص یافته به آن اجرا کنند، منبعی را انتخاب می کند که بتواند اجرای کار را زودتر از بقیه تمام کند و کار را به این منبع واگذار می کند. این عمل برای همه ی کارها به ترتیب انجام می شود تا واگذاری همه ی آنها به اتمام برسد.

1. کارها را به صورت نزولی از نظر طول مرتب کن.
2. برای هر کار به ترتیب انجام بده:
 - a. با توجه به نسبت طول کار به مجموع طول کارهای باقیمانده، بودجه‌ای را به آن اختصاص بده و از بودجه‌ی کل کم کن.
 - b. گروهی از منابع را که هزینه‌ی اجرای کار بر روی آنها از بودجه‌ی اختصاص داده شده، کمتر است، از بقیه‌ی منابع جدا کن.
 - c. در گروه منابع انتخاب شده، منبعی را که بتواند زودتر از منابع دیگر گروه، اجرای کار را تمام کند انتخاب کرده و کار را به آن نگاشت کن.
 - d. هزینه‌ی اجرای کار بر روی منبع انتخابی را از بودجه‌ی اختصاص داده شده کم کرده و مابقی را به بودجه‌ی کل برگردان.

شکل 4-5: الگوریتم EBTO

همانطور که در شکل 4-5 مشاهده می‌شود، الگوریتم EBTO با توجه به طول کارها به آنها بودجه تخصیص می‌دهد. در واقع نسبت طول کار به مجموع طول کارهایی که در صف قرار دارند برابر است با نسبت بودجه‌ی اختصاص یافته برای کار به کل بودجه‌ی باقیمانده. این، همان روشی است که بویا در الگوریتم BTO برای تخصیص بودجه ارائه کرده بود. در این روش کارهای بزرگتر به نسبت طولشان بودجه‌ی بیشتری دریافت می‌کنند. می‌دانیم اگر کارهای بزرگتر در ابتدا و کارهای کوچکتر در انتهای صف کارهای واگذار شده در منابع قرار بگیرند می‌توان با جابجایی کارهای کوچک، تعادل کاری بین منابع برقرار کرده و زمان را در پرکارترین منبع کاهش داد.

به همین دلیل بهتر است که به کارهای بزرگ، بیشتر و به کارهای کوچک کمتر از نسبت طولشان بودجه تخصیص بدهیم. الگوریتم AEBTO که در بخش بعد خواهد آمد یک روش جدید برای تخصیص بودجه ارائه می‌کند که به افزایش کارایی الگوریتم منجر می‌شود.

4-1-5 الگوریتم AEBTO

نحوه‌ی تخصیص بودجه به کارها در کارایی یک الگوریتم زمانبندی با هدف بهینه‌سازی زمان تأثیر زیادی دارد. مجموع هزینه‌های اجرای کارها نباید از بودجه‌ی تعیین شده توسط کاربر فراتر رود زیرا در این صورت زمانبندی بدست آمده ارزشی نخواهد

داشت. از طرف دیگر نحوه ی تخصیص بودجه باید طوری باشد که مجموع هزینه های صرف شده برای کارها تا حد ممکن به بودجه ی تعیین شده توسط کاربر نزدیک باشد تا بتوان حداکثر استفاده را از آن داشت. روشی که بویا برای این کار ارائه کرده است دارای این ویژگی ها می باشد. اولاً در این روش به هر کار با توجه به نسبت طول آن به مجموع طول کارها بودجه اختصاص می یابد بنابراین از آنجا که هیچ کاری نمی تواند بیش از بودجه ای که دارد هزینه صرف کند، تضمین می شود که مجموع هزینه های صرف شده از بودجه ی تعیین شده کمتر باشد. ثانیاً می دانیم هر کار، منبعی را انتخاب می کند که هزینه ی اجرایش در آن منبع از بودجه ی اختصاص یافته کمتر باشد. بنابراین مقداری از بودجه برای اجرای کار بر روی منبع انتخابی اضافه است. در روش بویا، کارها این مقدار اضافه را به بودجه ی باقیمانده برمی گردانند تا در هنگام تخصیص بودجه به کارهای دیگر استفاده شود. به این ترتیب سعی می شود مجموع هزینه های اجرای کارها تا حد ممکن به مقدار بودجه ی تعیین شده نزدیک باشد و در پایان عملیات زمانبندی همه ی کارها، مقدار کمی از بودجه باقی بماند.

الگوریتم AEBTO با تغییر دادن روش پیشنهادی بویا برای تخصیص بودجه باعث می شود کارایی نسبت به الگوریتم EBTO که از روش تخصیص بودجه ی بویا استفاده می کند افزایش یابد. می دانیم برای اینکه بتوان زمان اجرا در پرکارترین منبع را کاهش داد باید نوعی تعادل بین زمان اجراهای منابع استفاده شده ایجاد کرد. برای این عمل باید هنگامی که در حال زمانبندی کارهای پایانی هستیم طوری آنها را بین منابع توزیع کنیم که این تعادل برقرار شود. زمانی می توان این اقدام را به خوبی انجام داد که قبلاً نوعی تعادل در کارهایی که بین منابع توزیع شده است وجود داشته باشد. یعنی کارهای بزرگ به یک منبع و کارهای کوچک به یک منبع دیگر واگذار نشده باشد. از طرف دیگر می دانیم کارهای بزرگ برای اجرا بر روی منابع گران نسبت به کارهای کوچک به بودجه ی بیشتری نیاز دارند. بنابراین اگر بتوانیم به کارهای بزرگ تر بودجه ی بیشتری نسبت به طولشان بدهیم می توان امیدوار بود که کارهای بزرگ و کوچک به صورت تقریباً یکسان بر روی منابع ارزان و گران توزیع می شوند. در حالی که در روش پیشنهادی بویا برای تخصیص بودجه، کارهای بزرگ تنها می توانستند به منابع ارزان تر واگذار شوند و صف کارها در منابع گران پر از کارهای کوچک می شد.

روش تخصیص بودجه‌ی پیشنهادی ما برای الگوریتم AEBTO سعی می‌کند از مقداری که از بودجه‌ی اختصاص یافته به هر کار زیاد می‌آید برای تخصیص بودجه به کارهای بزرگ استفاده کند. این الگوریتم در ابتدا صف کارها را از نظر طول به صورت نزولی مرتب می‌کند. سپس هنگامی که همانند الگوریتم EBTO در حال زمانبندی کارها به ترتیب می‌باشد، به جای اینکه مقدار اضافی بودجه‌ی اختصاص یافته به هر کار را به بودجه‌ی کل برگرداند، این مقدار را به بودجه‌ی کار بعدی اضافه می‌کند. از آنجا که در ابتدا کارهای بزرگتر زمانبندی می‌شوند، این عمل باعث می‌شود کارهای بزرگ بودجه‌ی بیشتری نسبت به طولشان دریافت کنند و بتوانند بر روی منابع گرانتر نیز اجرا شوند. جزئیات الگوریتم AEBTO در شکل 5-5 آمده است.

1. کارها را به صورت نزولی از نظر طول مرتب کن.
2. برای هر کار به ترتیب انجام بده
 - a. با توجه به نسبت طول کار به مجموع طول کارهای باقیمانده، بودجه‌ای را به آن اختصاص بده و از بودجه‌ی کل کم کن.
 - b. مقداری را که از بودجه‌ی کار قبل باقیمانده است، به بودجه‌ی اختصاص یافته به این کار، اضافه کن.
 - c. گروهی از منابع را که هزینه‌ی اجرای کار بر روی آنها از بودجه‌ی اختصاص داده شده، کمتر است، از بقیه‌ی منابع جدا کن.
 - d. در گروه منابع انتخاب شده، منبعی را که بتواند زودتر از منابع دیگر گروه، اجرای کار را تمام کند، انتخاب کرده و کار را به آن نگاشت کن.
 - e. هزینه‌ی اجرای کار بر روی منبع انتخابی را از بودجه‌ی اختصاص داده شده، کم کرده و مابقی را برای کار بعدی ذخیره کن.

شکل 5-5: الگوریتم AEBTO

باید توجه کرد که روش تخصیص بودجه‌ای که در اینجا پیشنهاد شده است هر دو ویژگی یک روش تخصیص خوب را داراست. اولاً با توجه به بودجه‌های اختصاص یافته به کارها هیچ گاه هزینه‌ی انجام کارها از مقدار بودجه‌ی تعیین شده توسط کاربر فراتر نمی‌رود. زیرا مقدار بودجه‌ی اختصاصی هر کار از بودجه‌ی کل کارها کم می‌شود و هیچ مقداری نیز به بودجه‌ی کل اضافه نمی‌شود. ثانیاً مقدار اضافی بودجه‌ی اختصاص یافته به هر کار، به کار بعدی

داده می‌شود و به این ترتیب سعی می‌شود از بودجه‌ی تعیین شده به طور کامل استفاده شود.

5-1-5- الگوریتم LATO

الگوریتم LATO سعی می‌کند با استفاده از مجموعه‌ای از اتوماتاهای یادگیر عمل پیچیده‌ی بهینه‌سازی زمان را با کارایی بهتری نسبت به الگوریتم‌های مکاشفه‌ای انجام دهد. این الگوریتم برای زمانبندی کارهای برنامه‌ی کاربردی متعلق به کاربر، هر کدام از کارها را به یک اتوماتای یادگیر مجهز می‌کند. این اتوماتاهای یادگیر دارای ساختار متغیر بوده و اعمال آنها منابع گرید می‌باشد. در واقع اتوماتای یادگیر متعلق به یک کار، منبعی را که کار باید به آن واگذار شود، انتخاب می‌کند. هنگامی که همه‌ی کارها به ترتیب، منابع مورد نظر خود را انتخاب کردند، از طرف محیط پاداش گرفته یا جریمه می‌شوند. سپس برای بار دیگر، همه‌ی کارها عمل انتخاب منبع را انجام می‌دهند تا پاسخ محیط را دریافت کنند. این عمل تا تعداد معینی تکرار می‌شود تا همه‌ی کارها منبع مناسب خود را بیابند و به آن واگذار شوند. الگوریتم LATO همانند الگوریتم‌های پیشنهادی دیگر، از روش زمانبندی یکباره استفاده می‌کند و صف کارها را از نظر طول به صورت نزولی مرتب می‌کند.

در الگوریتم LATO نیز در هر تکرار، زمانبندی ایجاد شده باید به طور موقت به منابع اعمال شود تا بتوان آن را ارزیابی کرد. برای این کار، هر منبع دارای یک صف موقت به نام صف انتساب است که در ابتدای هر تکرار خالی می‌شود و سپس هنگامی که کارها به ترتیب، در حال انتخاب منبع هستند، بلافاصله در صف انتساب منبع انتخاب شده قرار می‌گیرند.

پاسخ محیط به اتوماتاهای یادگیر متعلق به کارها، با توجه به شرایط ایده‌آل در یک بهینه‌سازی زمان انجام می‌گیرد. برای این منظور، محیط از سه نکته‌ی زیر برای دادن پاداش یا جریمه به کارها استفاده می‌کند:

1. مجموع هزینه‌ی اجرای کارها نباید از بودجه‌ی تعیین شده توسط کاربر بیشتر باشد.
2. زمان اجرای کارهای واگذار شده به پرکارترین منبع نباید با زمان اجرا در منابع دیگر اختلاف زیادی داشته باشد.

3. هر کار باید به منبعی واگذار شود که بتواند اجرای آن کار را زودتر از بقیه منابع به اتمام برساند.

به این ترتیب محیط با استفاده از دو نکته‌ی اول، کارها را جریمه می‌کند و با توجه به نکته‌ی سوم به کارها پاداش می‌دهد. با توجه به نکته‌ی اول، اگر مجموع هزینه‌های اجرای کارها بر روی منابع انتخاب شده توسط آنها از مقدار بودجه‌ی تعیین شده فرا تر رود محیط کارهایی را که هزینه‌ی زیادی برای اجرا لازم دارند جریمه می‌کند. در واقع عمل جریمه کردن از انتهای صف کارها در گران‌ترین منبع استفاده شده (گران‌ترین منبعی که حداقل یک کار به آن منتسب شده است) شروع می‌شود. با جریمه شدن هر کار و حذف آن از صف انتساب منبع انتخابی، اختلاف هزینه‌ی اجرای آن بر روی این منبع با هزینه‌ی اجرای آن بر روی ارزان‌ترین منبع موجود، از مقدار کل هزینه‌های اجرا کسر می‌شود. عمل جریمه دادن تا زمانی ادامه می‌یابد که مقدار کل هزینه‌ها از مقدار بودجه‌ی تعیین شده بیشتر باشد. در واقع با این جریمه، کارها به سمت استفاده از منابع ارزانتر هدایت می‌شوند.

با توجه به نکته‌ی دوم، محیط جریمه کردن آخرین کار در صف انتساب پرکارترین منبع را در نظر می‌گیرد. اگر یکی از منابع استفاده شده در زمانبندی (منابعی که حداقل یک کار منتسب شده دارند) به جز آخرین (گران‌ترین) منبع استفاده شده، بتواند این کار را زودتر از پرکارترین منبع پردازش کند، محیط این کار را جریمه می‌کند تا در تکرارهای بعدی منبع مناسب‌تری را برای خود انتخاب کند. این عمل باعث می‌شود کارها به صورت متعادل بین منابع توزیع شوند.

در نهایت محیط با توجه به نکته‌ی سوم، به کارها پاداش می‌دهد. کارهایی که توانسته‌اند زمان اتمام کمتر یا برابر نسبت به تکرار قبل برای خود بدست بیاورند پاداش می‌گیرند؛ البته به شرطی که در همین تکرار جریمه نشده باشند. به این ترتیب هر کار تشویق می‌شود منبعی را برای واگذاری خود انتخاب کند که در کمترین زمان، اجرایش را به اتمام برساند.

1. **مرتب‌سازی:** کارها را از نظر طول به صورت نزولی مرتب کن.
2. **یادگیری:** به تعداد 10000 بار تکرار کن:
 - a. صف‌های انتساب منابع را خالی کن.
 - b. **انتخاب:** برای هر کار به ترتیب انجام بده:
 - i. با استفاده از اتوماتای یادگیر مربوطه، منبعی را برای واگذاری، انتخاب کن.
 - ii. کار را در صف انتساب منبع انتخابی قرار بده.
 - c. **جریمه (1):** تا هنگامی که مجموع هزینه‌ی اجرای کارها بر روی منابع انتخاب شده از بودجه‌ی تعیین شده بیشتر است تکرار کن:
 - i. از انتهای صف انتساب در گران‌ترین منبع استفاده شده، یک کار را جدا کن.
 - ii. اختلاف هزینه‌ی اجرای کار بر روی این منبع با هزینه‌ی اجرای کار بر روی ارزان‌ترین منبع موجود را از مجموع هزینه‌ی اجرای کارها کم کن.
 - iii. کار را جریمه کن (مقدار جریمه: 0.01).
 - d. **جریمه (2):** آخرین کار منتسب شده به پرکارترین منبع را در نظر بگیر. اگر یکی از منابع استفاده شده به جز منبع آخر (گران‌ترین منبع استفاده شده) می‌تواند اجرای این کار را زودتر تمام کند این کار را جریمه کن (مقدار جریمه: 0.01).
 - e. **پاداش:** برای هر کار انجام بده:
اگر جریمه نشده است: اگر زمان اتمام اجرای آن بر روی منبع انتخابی از زمان اتمام در تکرار قبلی کمتر یا با آن برابر است به آن پاداش بده (مقدار

شکل 5-6: الگوریتم LATO

همانطور که در شکل 5-6 آمده است، الگوریتم LATO در ابتدا کارها را بر حسب طولشان به صورت نزولی مرتب می‌کند. سپس در تکرارهای زیاد، منبع مناسب برای اجرای هر کار را پیدا می‌کند. این الگوریتم در نهایت کارها را به آن منابع واگذار کرده و از روش زمان‌بندی یکباره استفاده می‌کند. محیط در هنگام یادگیری الگوریتم از دو پاسخ جریمه و یک پاسخ پاداش استفاده می‌کند. از آنجا که اگر یک کار در هیچ کدام از دو حالت جریمه نشده باشد و همچنین زمان اتمام کوچکتر یا برابر نسبت به تکرار قبلی بدست آورده باشد، احتمالاً منبع مناسبی را برای خود پیدا کرده است،

مقدار پاداش در این حالت (0.1) از مقادیر جریمه‌ها در دو حالت اول (0.01) بیشتر است.

5-1-6- الگوریتم مینیمم هزینه محاسبات

اجرای یک برنامه‌ی کاربردی بر روی گرید اقتصادی با صرف بودجه‌ای که توسط کاربر فراهم شده است، دارای یک کران پایین یا مینیمم برای زمان محاسبات می‌باشد. مینیمم نشان می‌دهد کاربر هنگامی که حاضر است بودجه‌ی معینی را برای اجرای برنامه‌ی کاربردی خود پرداخت کند، حداقل زمانی که باید برای دریافت نتیجه صبر کند چقدر است. البته در بیشتر موارد، هیچ زمانبندی وجود ندارد که بتواند کارها را در مدت زمان مینیمم به کاربر تحویل دهد و الگوریتم‌های زمانبندی سعی می‌کنند تا حد ممکن زمان محاسبات را به زمان مینیمم نزدیک کنند.

برای بدست آوردن مینیمم زمان محاسبات می‌توان از الگوریتم‌های ارائه شده برای بهینه‌سازی زمان استفاده کرد که به جای اینکه کارها را زمانبندی کنیم عمل زمانبندی را برای دستوالعمل‌ها به عنوان کوچکترین واحد اجرایی انجام می‌دهیم. یعنی به هر میلیون دستورالعمل، بودجه تخصیص می‌دهیم و آن را برای اجرا به منبع مناسب واگذار می‌کنیم. روش دیگر برای بدست آوردن مینیمم زمان محاسبات این است که از روش ارائه شده برای بدست آوردن مینیمم هزینه استفاده کنیم. برای این کار روش مینیمم هزینه را با در نظر گرفتن مهلت اجرا می‌کنیم و هزینه بدست آمده را بررسی می‌کنیم. سپس از مهلت مقداری کم می‌کنیم و دوباره روش مینیمم هزینه را اجرا می‌کنیم. این عمل را آنقدر تکرار می‌کنیم تا به مهلتی برسیم که مینیمم هزینه انجام کارها در آن مهلت با بودجه‌ی تعیین شده تقریباً برابر شود. در این حالت، مهلت استفاده شده در روش مینیمم هزینه، همان مینیمم زمان محاسبات خواهد بود. ما از این روش برای بدست آوردن مینیمم زمان اجرای کارها با صرف بودجه‌ی تعیین شده توسط کاربر، استفاده می‌کنیم.

5-1-7- الگوریتم پیشنهادی ALATO

الگوریتم پیشنهادی در این پایان‌نامه، از روش زمانبندی یکباره استفاده می‌کند و در هنگام ورود کاربر به سیستم، در صورتی که انجام برنامه‌ی کاربردی او در محدوده‌ی مهلت و بودجه‌ی تعیین شده، ممکن باشد، کاربر را می‌پذیرد و اتمام برنامه را با

رعایت این محدوده‌ها تضمین می‌کند. برای این منظور، در پایان الگوریتم پیشنهادی و پس از مشخص شدن زمانبندی (نگاشت کارها به منابع)، یک مرحله کنترل پذیرش²³² ساده وجود دارد که مجموع زمان انجام کارها و هزینه اجرای برنامه که به طور تقریبی در زمانبندی، بدست آمده‌اند را بررسی می‌کند. در صورت پذیرش کاربر، مرحله توزیع انجام می‌گیرد که در آن، کارها به منابع مشخص شده در زمانبندی، واگذار می‌شوند.

بنابر این، الگوریتم ALATO سعی می‌کند با استفاده از مجموعه‌ای از اتوماتاهای یادگیر عمل پیچیده‌ی بهینه‌سازی زمان را با کارایی بهتری نسبت به الگوریتم‌های مکاشفه‌ای انجام دهد. این الگوریتم از مجموعه‌ای از اتوماتاهای یادگیر استفاده می‌کند که به کارها نسبت داده می‌شوند. این اتوماتاهای یادگیر دارای ساختار متغیر بوده و اعمال آنها منابع گرید می‌باشد. در واقع اتوماتای یادگیر متعلق به یک کار، منبعی را که کار باید به آن واگذار شود، انتخاب می‌کند. هنگامی که همه‌ی کارها به ترتیب، منابع مورد نظر خود را انتخاب کردند، از طرف محیط پاداش گرفته یا جریمه می‌شوند. سپس برای بار دیگر، همه‌ی کارها عمل انتخاب منبع را انجام می‌دهند تا پاسخ محیط را دریافت کنند. این عمل تا تعداد معینی تکرار می‌شود تا همه‌ی کارها منبع مناسب خود را بیابند و به آن واگذار شوند. الگوریتم ALATO همانند الگوریتم‌های پیشنهادی دیگر، از روش زمانبندی یک‌باره استفاده می‌کند و صف کارها را از نظر طول به صورت نزولی مرتب می‌کند.

الگوریتم ALATO نیز همانند الگوریتم‌های دیگری که در بخش قبلی ارائه شد فرض می‌کند منابع به صورت فضا‌مشترک رفتار می‌کنند و عمل زمانبندی را به صورت یک‌باره انجام می‌دهد. همچنین، کارها را به صورت نزولی مرتب می‌کند تا زمان بیکاری منابع گران‌تر را کاهش دهد. الگوریتم ALATO نمی‌تواند تنها با یکبار بررسی کارها و منابع به زمانبندی نهایی برسد؛ بلکه با تکرارهای زیادی این عمل را انجام می‌دهد و در هر تکرار یک زمانبندی جدید را ایجاد می‌کند. محیط به کارها با توجه به منابعی که انتخاب کرده‌اند، پاسخ (پاداش یا جریمه) می‌دهد. کارها از این پاسخ محیط، در تکرار بعدی استفاده می‌کنند تا زمانبندی جدیدی که ایجاد می‌شود به زمانبندی بهینه نزدیکتر باشد. این تکرارها به تعداد زیادی

²³² Admission Control

انجام می‌شود تا همه‌ی کارها منبع مناسب خود را پیدا کنند و تغییر دیگری در تصمیم خود ندهند.

در واقع، تا جایی تکرارها ادامه می‌یابد که همه‌ی اتوماتاهای یادگیر همگرا شده و زمانبندی یکسانی در تکرارها ایجاد شود. این زمانبندی یکسان، همان زمانبندی نهایی است که بر اساس آن، کارها به منابع واگذار می‌گردد.

در الگوریتم ALATO نیز در هر تکرار، زمانبندی ایجاد شده باید به طور موقت به منابع اعمال شود تا بتوان آن را ارزیابی کرد. برای این کار، هر منبع دارای یک صف موقت به نام صف انتساب است که در ابتدای هر تکرار خالی می‌شود و سپس هنگامی که کارها به ترتیب، در حال انتخاب منبع هستند، بلافاصله در صف انتساب منبع انتخاب شده قرار می‌گیرند.

پاسخ محیط به اتوماتاهای یادگیر متعلق به کارها، با توجه به شرایط ایده‌آل در یک بهینه‌سازی زمان انجام می‌گیرد. برای این منظور، محیط از سه نکته‌ی زیر برای دادن پاداش یا جریمه به کارها استفاده می‌کند:

1. مجموع هزینه‌ی اجرای کارها نباید از بودجه‌ی تعیین شده توسط کاربر بیشتر باشد.
2. زمان اجرای کارهای واگذار شده به پرکارترین منبع نباید با زمان اجرا در منابع دیگر اختلاف زیادی داشته باشد.
3. هر کار باید به منبعی واگذار شود که بتواند اجرای آن کار را زودتر از بقیه‌ی منابع به اتمام برساند.

به این ترتیب محیط با استفاده از دو نکته‌ی اول، کارها را جریمه می‌کند و با توجه به نکته‌ی سوم به کارها پاداش می‌دهد. با توجه به نکته‌ی اول، اگر مجموع هزینه‌های اجرای کارها بر روی منابع انتخاب شده توسط آنها از مقدار بودجه‌ی تعیین شده فرا تر رود محیط کارهایی را که هزینه‌ی زیادی برای اجرا لازم دارند جریمه می‌کند. در واقع عمل جریمه کردن از انتهای صف کارها در گران‌ترین منبع استفاده شده (گران‌ترین منبعی که حداقل یک کار به آن منتسب شده است) شروع می‌شود. با جریمه شدن هر کار و حذف آن از صف انتساب منبع انتخابی، اختلاف هزینه‌ی اجرای آن بر روی این منبع با هزینه‌ی اجرای آن بر روی ارزان‌ترین منبع موجود، از مقدار کل هزینه‌های اجرا کسر می‌شود.

عمل جریمه دادن تا زمانی ادامه می‌یابد که مقدار کل هزینه‌ها از مقدار بودجه‌ی تعیین شده بیشتر باشد. در واقع با این جریمه، کارها به سمت استفاده از منابع ارزانتر هدایت می‌شوند. هدف در این مرحله، پایین آوردن سریع هزینه و رساندن آن به میزان مقدار کمتر از مهلت در نظر گرفته شده با انتقال کارها از گرانترین منبع به ارزانترین منبع می‌باشد.

با توجه به نکته‌ی دوم، محیط جریمه کردن کارها در صف انتساب پرکارترین منبع را در نظر می‌گیرد. این نکته را توجه داشته باشید که لزوماً پرکارترین منبع گرانترین منبع نیست. از انتهای صف پرکارترین منبع شروع می‌کنیم. به تعداد Y کار از انتهای صف پرکارترین منبع منابع را انتخاب می‌کنیم. اگر یکی از منابع استفاده شده در زمانبندی (منابعی که حداقل یک کار منتسب شده دارند) به جز آخرین (گران‌ترین) منبع استفاده شده، بتواند این کار را زودتر از پرکارترین منبع پردازش کند، محیط این کار را جریمه می‌کند تا در تکرارهای بعدی منبع مناسب‌تری را برای خود انتخاب کند، در غیر این صورت این کار هیچ جریمه‌ای نمی‌شود. این عمل برای همه Y کار از پرکارترین منبع اعمال می‌شود. این عمل باعث می‌شود کارها با تعادل و توازن بسیار بیشتری بین منابع توزیع گردند. مقدار Y با توجه به جدول 1-5 از رابطه‌های (1-5) و (2-5) محاسبه می‌شود:

جدول 1-5: علائم اختصاری در نظر گرفته شده برای ایده پیشنهادی

عنوان	علائم اختصاری
تعداد کارها	Tasks_No
تعداد منابع	Res_No
تعداد کارهای اختصاص داده شده به گران‌ترین منبع	A
تعداد کارهای اختصاص داده شده به پرکارترین منبع	B

$$X = (Task_No - A) / (Res_No - 1) \quad \text{رابطه (5-1)}$$

که X میزان کارهای متوسط اختصاص داده شده بازای $R-1$ منبع می‌باشد. در واقع تعداد کارهایی که باید بر روی همه منابع بغیر از گرانترین منبع پخش شود. (اختصاص دوباره کارها بر روی

منابه بغیر از گرانتترین منبع، چونکه، گرانتترین منبع هزینه اجرا را افزایش می دهد.)

$$Y = (B - X) * \frac{Res_No - 2}{Res_No - 1} \quad \text{رابطه (5-2)}$$

در اینجا مقدار Y بیانگر تعداد کارهایی است که از انتهای صف پرکارترین منبع باید مورد بررسی قرار بگیرند. مقدار B همواره از مقدار A بیشتر یا برابر می باشد، چونکه، X مقدار متوسط کارهای پخش شده بر روی منابع می باشد و تعداد کارهای صف پرکارترین منبع بیشتر از کارهای سایر منابع یا میزان متوسط کارها می باشد.

مقدار $Res_No - 2 / Res_No - 1$ میزان متوسط منابع قابل اجرا بغیر از گرانتترین منبع و پرکارترین منبع به منابع موجود غیر از گرانتترین منبع می باشد.

در نهایت محیط با توجه به نکته ی سوم، به کارها پاداش می دهد. کارهایی که توانسته اند زمان اتمام کمتر یا برابر نسبت به تکرار قبل برای خود بدست بیاورند پاداش می گیرند؛ البته به شرطی که در همین تکرار جریمه نشده باشند. به این ترتیب هر کار تشویق می شود منبعی را برای واگذاری خود انتخاب کند که در کمترین زمان، اجرای را به اتمام برساند.

- 1) **مرتب‌سازی:** کارها را از نظر طول به صورت نزولی مرتب کن.
- 2) **یادگیری:** به تعداد 10000 بار تکرار کن:
 - f. صف‌های انتساب منابع را خالی کن.
 - g. **انتخاب:** برای هر کار به ترتیب انجام بده:
 - iii. با استفاده از اتوماتای یادگیر مربوطه، منبعی را برای واگذاری، انتخاب کن.
 - iv. کار را در صف انتساب منبع انتخابی قرار بده.
 - h. **جریمه (1):** تا هنگامی که مجموع هزینه‌ی اجرای کارها بر روی منابع انتخاب شده از بودجه‌ی تعیین شده بیشتر است تکرار کن:
 - iv. از انتهای صف انتساب در گران‌ترین منبع استفاده شده، یک کار را جدا کن.
 - v. اختلاف هزینه‌ی اجرای کار بر روی این منبع با هزینه‌ی اجرای کار بر روی ارزان‌ترین منبع موجود را از مجموع هزینه‌ی اجرای کارها کم کن.
 - vi. کار را جریمه کن (مقدار جریمه: 0.01).
 - i. **جریمه (2):** به تعداد Y بار تکرار کن:
 - i. کار آخر منتسب شده به پرکارترین منبع را در نظر بگیر.
 - ii. اگر یکی از منابع استفاده شده به جز منبع آخر (گران‌ترین منبع استفاده شده) می‌تواند اجرای این کار را زودتر تمام کند این کار را جریمه کن (مقدار جریمه: 0.02).
 - j. **پاداش:** برای هر کار انجام بده:
 اگر جریمه نشده است: اگر زمان اتمام اجرای آن بر روی منبع انتخابی از زمان اتمام در تکرار قبل کمتر یا با آن برابر است به آن پاداش بده (مقدار پاداش: 0.1).

شکل 5-7: الگوریتم ALATO

همانطور که در شکل 5-7 آمده است، الگوریتم ALATO در ابتدا کارها را بر حسب طولشان به صورت نزولی مرتب می‌کند. سپس در تکرارهای زیاد، منبع مناسب برای اجرای هر کار را پیدا می‌کند. این الگوریتم در نهایت کارها را به آن منابع واگذار کرده و از روش زمان‌بندی یکباره استفاده می‌کند. برای بدست آمدن بهترین جواب انواع مدل اتوماتای یادگیر (مدل استاندارد و روش‌های مدل S) مورد بررسی قرار گرفته اند. در اتوماتای استاندارد، محیط در

هنگام یادگیری الگوریتم از دو پاسخ جریمه و یک پاسخ پاداش استفاده می‌کند. از آنجا که اگر یک کار در هیچ کدام از دو حالت جریمه نشده باشد و همچنین زمان اتمام کوچکتر یا برابر نسبت به تکرار قبل بدست آورده باشد، احتمالاً منبع مناسبی را برای خود پیدا کرده است، مقدار پاداش در این حالت (0.1) از مقادیر جریمه‌ها در حالت اول (0.01) و در حالت دوم (0.02) بیشتر است. علت بیشتر بودن مقدار جریمه در حالت دوم بعلاوه قرار گیری کارهای کوچکتر در پرکارترین منبع و افزایش تراکم در آن منبع می‌باشد. در واقع، با افزایش جریمه کارها به سمت قرار گیری در منابع کم کارتر و افزایش تعادل بار در هر تکرار گام بر می‌دارند و سرعت همگرا شدن اتوماتا به صورت چشمگیری افزایش می‌یابد. کد برنامه در ضمیمه آورده شده است.

اتوماتای یادگیر مدل S را با مقادیر مختلفی برای نرخ پاداش و جریمه تست کرده ایم، در روش‌های مدل S، اگر در تکرار $n^{\text{ام}}$ عمل α_i انتخاب شده باشد، پاسخ (پاسخ نامطلوب) محیط به آن برابر $\beta_i(n)=1$ می‌باشد و پاسخ (پاسخ مطلوب) محیط به آن عمل برابر می‌باشد، (یعنی به نسبت انتخاب منبع ارزان‌تر پاسخ $\beta_i(n)=\frac{1}{1+\frac{\text{PerviousTime}}{\text{NewTime}}}$)

بهتری از محیط دریافت می‌کند). که طبق نتایج که در فصل شبیه سازی مشاهده خواهیم نمود که روش S-LReP با نرخ پاداش برابر با 0.1 و نرخ جریمه برابر با 0.05 بهترین نتیجه را خواهد داشت.

شبیه سازی الگوریتم ها

5-2-1- مقدمه

در اینجا ابتدا به شرح کاربرد مدل شبکه پتری تصادفی تطابقی پیشنهاد در محیط گرید و بررسی و آنالیز آن می‌پردازیم، سپس کارایی الگوریتم‌های ارائه شده در بخش‌های قبلی را که زمانبندی کارها را با اتخاذ استراتژی بهینه سازی زمان در یک گرید اقتصادی انجام می‌دادند ارزیابی می‌کنیم. این الگوریتم‌ها سعی می‌کنند در یک محیط گرید اقتصادی، عمل زمانبندی تعدادی کار مستقل از هم را انجام دهند. این کارها متعلق به یک برنامه کاربردی است که کاربر آن را برای اجرا به گرید تحویل داده است. کاربر به همراه این برنامه، پارامترهای کیفیت سرویس مورد نظر خود یعنی مهلت زمانی، بودجه و استراتژی بهینه سازی را برای

سیستم مشخص می‌کند. الگوریتم‌های شبیه‌سازی شده، استراتژی بهینه‌سازی را به کار می‌برند. کاربر با انتخاب استراتژی بهینه‌سازی زمان برای زمانبندی، انتظار دارد که سیستم گرید، اجرای برنامه‌ی کاربردی او را قبل از مهلت تعیین شده، به پایان برساند و تا حد ممکن زمان اجرا را کاهش دهد. با انجام شبیه‌سازی می‌توان مشخص کرد که کدام یک از الگوریتم‌های پیشنهادی برای بهینه‌سازی زمان می‌توانند زمان کمتری را برای اجرای برنامه بدست آورند.

5-2-2- مدل شبکه پتری تصادفی تطابقی در محیط گرید

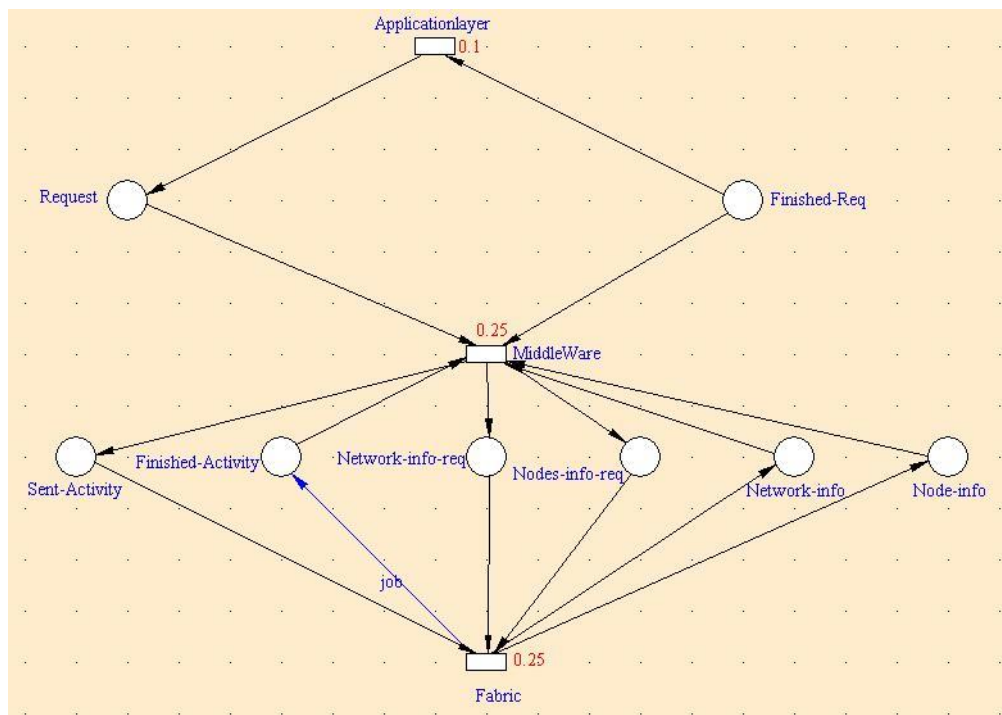
معماری گرید شامل 3 لایه است، در بالای لایه‌ی برنامه کاربردی²³³، محیط گردش کار²³⁴ است که در آن کاربران به شرح برنامه‌های خود می‌پردازند، در پایین، لایه‌ی فابریک²³⁵، شامل گره-های گرید متصل شده در شبکه است، در بین آن‌ها، لایه‌ی میان-افزار²³⁶ است که مسئول تخصیص منابع می‌باشد، شکل 5-8 سه لایه شبکه گرید را در شبکه پتری تصادفی نشان می‌دهد که در نرم افزار SPNP پیاده سازی شده است. هر درخواست در ابتدا از لایه برنامه به لایه میان افزار ارسال می شود و سپس پس از تحلیل و تجزیه شدن بصورت کارها و زیر کارهای مستقل به لایه فابریک که با سخت افزار و منابع گرید در ارتباط است فرستاده می شود. شکل 5-9 روال تخصیص کارها در لایه میان افزار نشان داده شده است. همانطور که در شکل 5-9 ملاحظه می شود این 200 کار مرتب شده به زمانبند داده می شود. این زمانبند کارها را بصورت اکشن‌های مختلف در اتوماتای یادگیر در نظر می گیرد. هر درخواست بصورت صفی از درخواست ها در نظر گرفته می شود و به منابع مختلف ارجاع می شود.

²³³ Application Layer

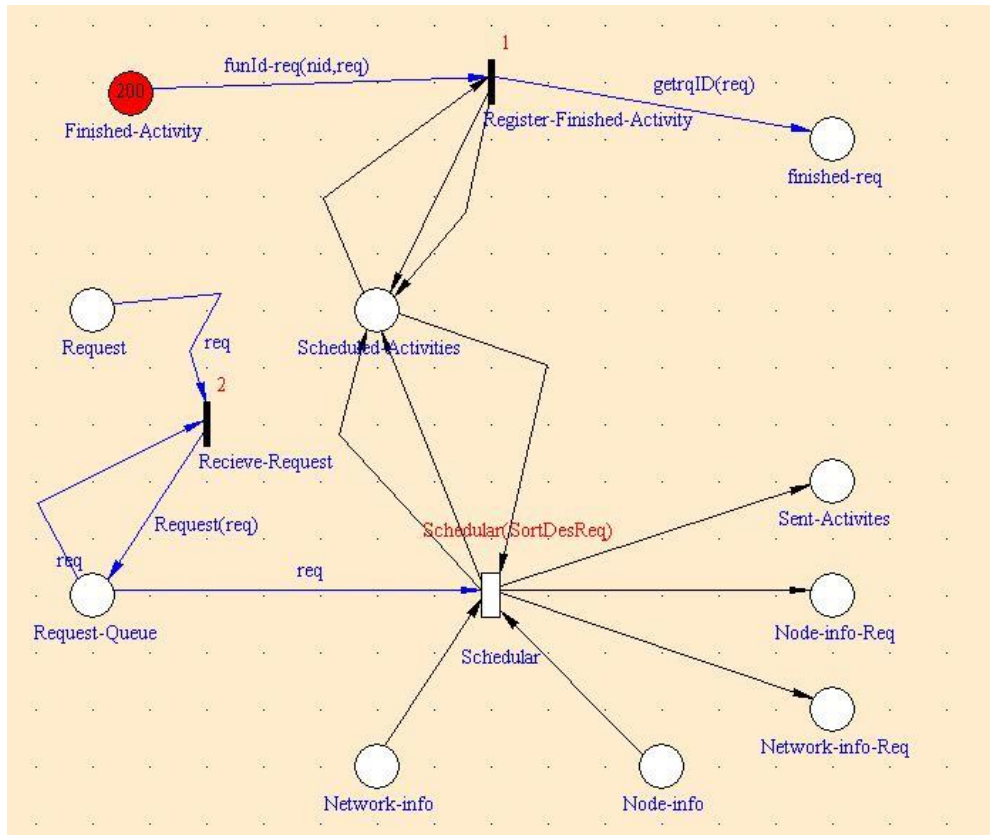
²³⁴ Workflow

²³⁵ Fabric Layer

²³⁶ Middleware Layer

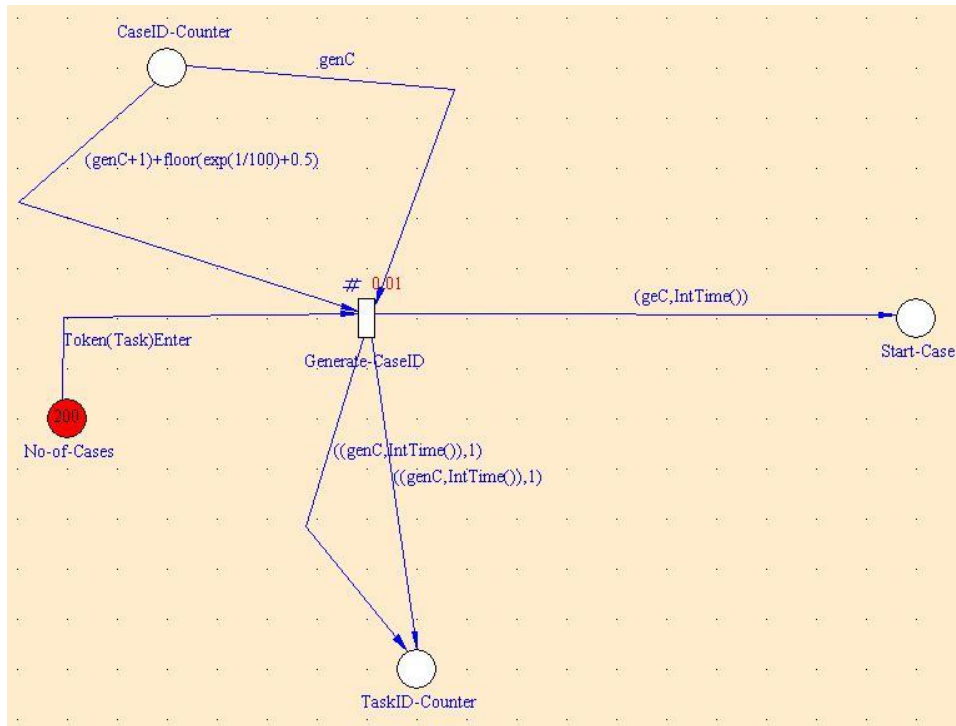


شکل 5-8: سه لایه شبکه گزید در شبکه پتری تصادفی



شکل 5-9: تخصیص کارها در لایه میان افزار

در لایه میان افزار، برنامه‌ی کاربر از طریق لایه‌ی برنامه کاربردی دریافت می‌شود و پس از گرفتن اطلاعات پویای منابع، درخواست زمان-بندی کار کاربر به ماژول زمان‌بند داده می‌شود، کارها بر اساس الگوریتم پیشنهادی ابتدا به صورت نزولی مرتب می‌شوند که در شکل 5-10 نشان داده شده است. در ابتدا مدل کلی سیستم گرید را با 200 کار در نظر گرفته ایم. همانطور که در شکل 5-10 ملاحظه می‌شود این 200 کار در یک حلقه بصورت نزولی مرتب شده اند. متغیر genC بعنوان متغیر حلقه در نظر گرفته شده است. شرط توقف در مکان CaseID-Counter قرار گرفته است. در حلقه دوم زمان نزولی کردن درخواست‌ها در نظر گرفته شده است. در نتیجه، کارهای مرتب شده بر حسب زمان تقاضاها بصورت نزولی مرتب می‌شود.

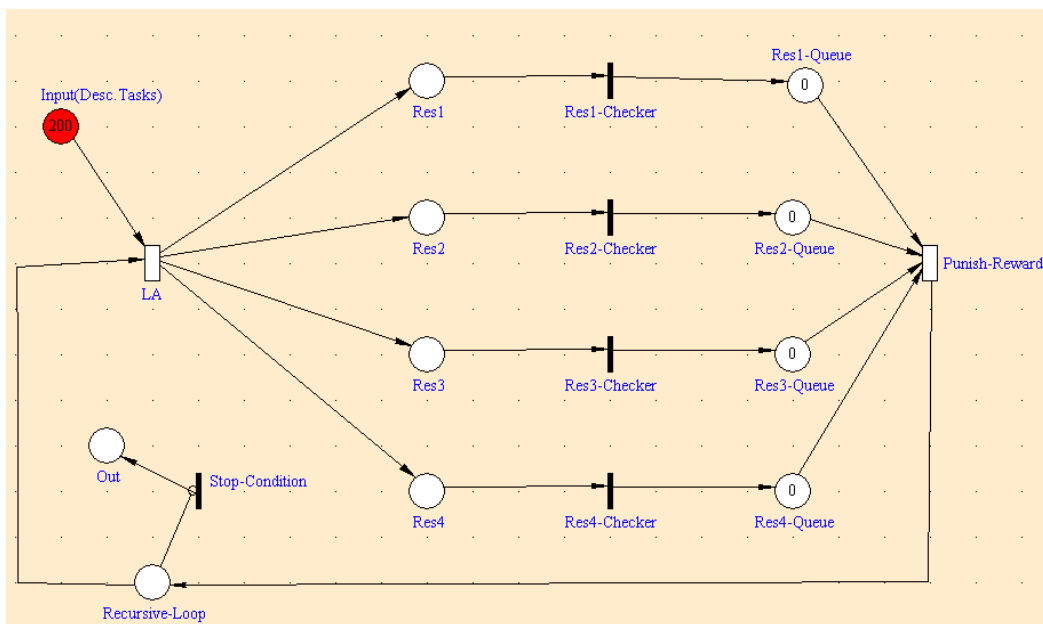


شکل 10-5: نزولی کردن کارها

هر توکن نشان دهنده‌ی یک کار از کارهای کاربر است، هر توکن شامل اطلاعاتی از جمله طول کار، شماره اولویت بر اساس طولش و اتوماتای یادگیر است. شبکه پتری تصادفی تطابقی ارائه شده، جزء دسته Fusion Hybrid می‌باشد، همانطور که در فصل‌های پیشین بیان شد، معایبی هم این دسته دارند. به علت ادغام شبکه‌های پتری با تکنیک‌های هوشمند، اغلب اوقات یادگیری در شبکه‌های پتری از این دسته باعث افزایش پیچیدگی عملیات و افزایش سربار محاسباتی می‌شود. در شبکه‌های پتری تطابقی ارائه شده تاکنون که در فصل دوم هم مورد بررسی قرار گرفته، از تکنیک‌های هوشمند نظیر شبکه‌های عصبی مصنوعی، منطق فازی و سیستم‌های مبتنی بر دانش‌بهره برده‌اند، ولی شبکه پتری تصادفی تطابقی که ما ارائه کرده ایم مبتنی بر اتوماتای یادگیر و همچنین ترکیب آن با منطق فازی است، شبکه پتری تصادفی تطابقی مبتنی بر اتوماتای یادگیر بر خلاف شبکه‌های پتری تطابقی که از شبکه‌های عصبی مصنوعی استفاده می‌کند، نیاز به آموزش دیدن با یکسری الگوهای از قبل مشخص شده نیست، در سیستم‌هایی که محیط کاملاً ناشناخته و در هر لحظه در حال تغییر است، نمی‌توان الگویی داشت تا بتوان شبکه عصبی مصنوعی را با آن آموزش داد ولی مدل تطبیقی پیشنهاد شده در این محیط‌ها کاملاً قابل انعطاف بوده چون اتوماتای یادگیر نیازی به آموزش دیدن با

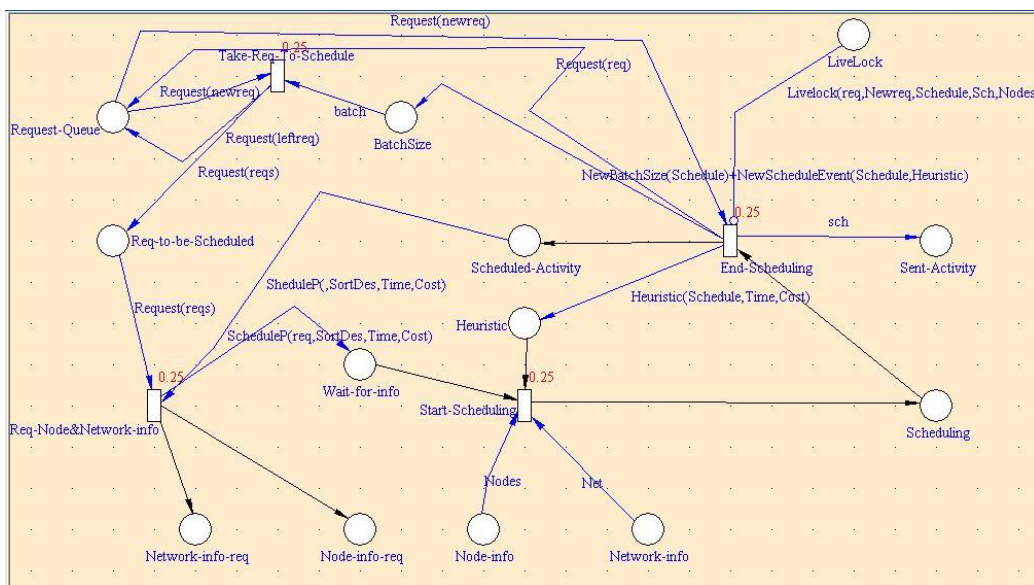
داده‌های از قبل ندارند، به همین دلیل هم مشکل دسته Fusion Hybrid که افزایش سربار محاسباتی است را ندارد، بعلاوه، در شبکه پتری تصادفی تطابقی مبتنی بر اتوماتای یادگیر، هر توکن اتوماتای مخصوص بخودش را دارد و برخلاف مدل‌های تطابقی قبلی که نیاز به ایجاد مکان‌ها و گذرهای اضافه‌تری برای ایجاد شبکه عصبی مصنوعی داشتند، نیازی به مکان و گذر جدیدتری نیست و این باعث عدم پیچیدگی در طراحی و افزایش سربار محاسباتی می‌گردد.

در شکل 5-11، هر توکن بر اساس بردار احتمالاتش، یک عمل (- یکی از منابع موجود در شبکه گرید است) را انتخاب می‌کند. سپس تمام کارهای انتساب شده در هر یک از منابع بر اساس ظرفیت آن منبع، مهلتی که کاربر برای انجام آن کار تعیین کرده است و الگوریتمی که در فصل قبل پیشنهاد داده شده، جریمه یا پاداش به آن عمل تعلق می‌گیرد. منابع Res1, Res2, Res3 و Res4 منابع چهار گانه ما برای انجام عمل تخصیص می‌باشند. که هر کدام دارای یک صف مجزا می‌باشند که به ترتیب در مکان‌های Res1_Queue, Res2_Queue, Res3_Queue و Res4_Queue قرار گرفته‌اند. در هر بار تخصیص کارها به هر منبع شماره کارها به ترتیب تخصیص در این صف‌ها قرار می‌گیرند و گذار Punish_Reward عمل پاداشی جریمه را به هر منبع اختصاص می‌دهد. این روال بصورت بازگشتی تا 10000 بار بر روی منابع انجام می‌شود و نتایج بدست آمده هر بار اجرا بعد از ذخیره شدن در مکان Recursive_Loop به اجرا مرحله بعدی می‌رود. کدها و توابع پیاده سازی بطور کامل در ضمیمه دوم آورده شده است.



شکل 5-11: انتساب کارها به منابع و اعمال پاداش و جریمه و اجرا به تعداد 10000 بار

در شکل 5-12 الگوریتم تخصیص و زمانبندی درخواست ها نشان داده شده است.



شکل 5-12: الگوریتم زمانبندی

3-2-5- پیاده سازی الگوریتم ها

در این بخش کارایی الگوریتم ارائه شده در بخش‌های قبلی را که زمانبندی کارها را با اتخاذ استراتژی‌های مختلف بهینه‌سازی در یک گرید اقتصادی انجام می‌دادند ارزیابی می‌کنیم. این الگوریتم‌ها سعی می‌کنند در یک محیط گرید اقتصادی، عمل زمانبندی تعدادی کار مستقل از هم را انجام دهند. این کارها متعلق به یک برنامه‌ی کاربردی است که کاربر آن را برای اجرا به گرید تحویل داده است. کاربر به همراه این برنامه، پارامترهای کیفیت سرویس مورد نظر خود یعنی مهلت زمانی، بودجه و استراتژی بهینه‌سازی را برای سیستم مشخص می‌کند. الگوریتم‌های شبیه‌سازی شده، استراتژی بهینه‌سازی هزینه یا زمان یا هر دو را به کار می‌برند. کاربر با انتخاب استراتژی بهینه‌سازی زمان برای زمانبندی، انتظار دارد که سیستم گرید، اجرای برنامه‌ی کاربردی او را کمتر از بودجه تعیین شده، به پایان برساند و تا حد ممکن زمان را کاهش دهد. با انجام شبیه‌سازی می‌توان مشخص کرد که کدام یک از الگوریتم‌ها برای بهینه‌سازی زمان می‌توانند زمان کمتری را برای اجرای برنامه بدست آورند. کاربر همچنین می‌تواند با انتخاب استراتژی بهینه‌سازی زمان از سیستم گرید بخواهد تا برنامه‌ی کاربردی او را با صرف بودجه‌ی مشخص شده و در کمترین زمان ممکن اجرا کند. با شبیه‌سازی می‌توان نشان داد که کارایی الگوریتم‌های بهینه‌سازی زمان در مقایسه با یکدیگر چگونه است.

3-2-4- محیط گرید

محیط شبیه‌سازی شده برای گرید شامل یک یا چندین کاربر و تعدادی منبع است. کاربران در هر زمانی ممکن است وارد گرید شوند و برنامه‌ی کاربردی خود را برای اجرا عرضه کنند. این برنامه شامل تعدادی کار مستقل از هم است که هر کدام می‌تواند بر روی منبع دلخواه اجرا شود. نهاد کاربر پس از ورود به سیستم، دارای یک نهاد واسط‌گر مخصوص به خود می‌شود که عمل زمانبندی برنامه‌ی او را انجام می‌دهد و پس از واگذاری کارها به منابع و اتمام اجرای آنها، نتایج را به کاربر برمی‌گرداند.

5-2-5- مدل کاربر

اغلب اوقات در آزمایش‌های²³⁷ شبیه‌سازی، محیط به صورت تک‌کاربره مدل می‌شود؛ مگر اینکه بخواهیم یک محیط چندکاربره را مدل کرده و الگوریتم‌ها را در چنین وضعیتی با هم مقایسه کنیم. منظور از مدل تک‌کاربره این نیست که از آغاز تا پایان شبیه‌سازی تنها یک کاربر وارد سیستم می‌شود؛ بلکه بدین معنی است که زمانبندی هیچ دو کاربری با هم تداخل پیدا نمی‌کند. برای نمونه برای مقایسه‌ی الگوریتم‌های LATO و ELATO باید دو کاربر را با برنامه‌های کاربردی یکسان طوری وارد سیستم کنیم که بر هم تأثیر نگذارند (مثلاً کاربر اول با الگوریتم LACO در زمان 0 وارد و در زمان 2000 خارج شود و سپس کاربر دوم با الگوریتم ALATO در زمان 3000 وارد شود و همان برنامه‌ی کاربردی کاربر قبلی را به سیستم عرضه کند). البته نمی‌توانیم از دو آزمایش شبیه‌سازی جداگانه استفاده کنیم؛ زیرا باید برنامه‌های کاربردی از نظر ناهمگونی کارها یکسان باشند تا بتوان الگوریتم‌ها را با هم مقایسه کرد.

کاربر در هنگام ورود، مقادیر بودجه و مهلت زمانی را مشخص می‌کند که در آزمایش‌های گوناگون، مقادیر مختلفی را برای آن‌ها قرار می‌دهیم. فرض می‌کنیم برنامه‌ی کاربردی ارائه شده، همواره در مهلت و بودجه‌ی تعیین شده قابل اجرا است. اگر کاربر خواستار بهینه‌سازی هزینه باشد آنگاه پس از مشخص شدن امکان اجرا، سیستم زمانبندی، باید برنامه‌ی کاربر را در قبل از مهلت تعیین شده و با حداقل هزینه، تمام کند. برای نمونه، اگر کاربر مقدار 200 هزار را برای بودجه و 10 هزار را برای مهلت تعیین کند و در زمان 5000 وارد سیستم شود آنگاه سیستم باید بتواند برنامه‌ی کاربر را قبل از زمان 15000 و با حداقل هزینه‌ی ممکن (که البته کمتر از مقدار بودجه خواهد بود) به کاربر تحویل دهد.

5-2-6- مدل برنامه

برنامه‌ی کاربردی، از تعدادی کار مستقل از هم تشکیل شده است. در آزمایش‌ها، در اغلب مواقع، تعداد کارها را 200 عدد در نظر می‌گیریم. هر کار دارای طول مخصوص به خود بر حسب میلیون دستورالعمل (MI) می‌باشد. طول کارها را متغیر در نظر می‌گیریم؛ به این صورت که یک محدوده‌ی (حداکثر... حداقل) برای طول کارها

²³⁷ Experiment

انتخاب می‌شود؛ سپس برای طول هر کار، یک مقدار به صورت تصادفی از این محدوده انتخاب می‌شود، به طوری که توزیع طول کارها یکنواخت باشد. برای همگون‌ترین حالت، محدوده‌ی توزیع طول کارها را $(100000 \dots 110000)$ و برای ناهمگون‌ترین حالت محدوده‌ی $(10000 \dots 200000)$ را در نظر می‌گیریم. میانگین محدوده را در هر حالتی مقدار 105000 قرار می‌دهیم تا بتوان یک الگوریتم را در ناهمگونی‌های مختلف کارها با هم مقایسه کرد. برای راحتی، واحد طول را در بعضی اوقات 1000MI قرار می‌دهیم و برای نمونه، محدوده‌ی ناهمگون‌ترین حالت را به صورت $(10 \dots 200)$ بیان می‌کنیم.

در ابزار GridSim هر کار به صورت یک Gridlet مدل می‌شود که دارای پارامترهای طول و اندازه‌ی فایل‌های ورودی و خروجی می‌باشد. از آنجا که در الگوریتم‌های زمان‌بندی، فقط زمان اجرای کارها بر روی منابع برای ما مهم است و تأخیر شبکه برای جابجایی فایل‌ها را در نظر نمی‌گیریم مقدار 0 را برای پارامتر اندازه‌ی فایل‌ها قرار می‌دهیم. البته در واقعیت نیز اندازه‌ی فایل‌ها آنقدری نیست که تأخیر شبکه در مقابل زمان اجرای کارها قابل ملاحظه باشد.

همچنین کارهای مورد مطالعه کاملاً محاسباتی هستند و فقط در ابتدا مقادیر مورد نیاز را از فایل ورودی دریافت می‌کنند و در پایان اجرا نتایج بدست آمده را در فایل خروجی قرار می‌دهند. از این تعامل داده‌ای اندک کارها نیز صرف نظر می‌کنیم و تنها به جنبه‌ی محاسباتی آنها می‌پردازیم.

در این پیاده سازی، یک مدل کار در زبان CSPL نرم افزار SPNP تعریف کردیم. کد آن در شکل 5-13 آورده شده است.

```
struct task{
    double r_prob[4]; // ehtemale gharar giri har task dar 4 manbah-dar ebteda yeksan va 0.25 mibashad
    char name; // nami baraye task
    double t_cost; // dar LATO va ALATO estefadeh nemishavad
    double t_time; // bar hasb MI
    double punishment; // mizan PUNISHMENT har task ke dar meghdar dehi avalie har charkhesh LA 0 dar nazr gerefte mishavad.
    double reward; // mizan REWARD har task ke dar meghdar dehi avalie har charkhesh LA 0 dar nazr gerefte mishavad.
}t[200];
```

شکل 5-13: کد تعریف کارها

5-2-7- مدل منابع

ما از یک پیکربندی نمونه با نام GR1 برای منابع گرید در آزمایش‌ها استفاده می‌کنیم که البته در جهان خارج وجود ندارد.

یک نمونه از پیکربندی واقعی با نام WWG در [133] آمده است که ما از آن نیز در آزمایشها استفاده کرده ایم؛ ولی فقط نتایج حاصل با پیکربندی GR1 و GR2 را آورده ایم. خصوصیات منابع در GR1 و GR2 طوری است که بتوان زمانبندیهای مختلف را به خوبی با هم مقایسه کرد. همه این منابع دارای یک پردازنده هستند و به صورت فضا مشترک رفتار میکنند. این پیکربندیها را میتوان در جدول 2-5 و 3-5 مشاهده کرد. در این جداول، منابع بر حسب قیمت به صورت صعودی مرتب شده اند و به همین ترتیب نیز نامگذاری شده اند تا در هنگام شبیه سازی بتوان بار کاری هر کدام از منابع و کارهای واگذار شده را راحت تر بررسی کرد (البته برای مرتب سازی منابع، از قیمت مفید ($G\$/MI$) آنها استفاده میشود که در اینجا با ترتیب قیمت های اعلام شده ($G\$/sec$) یکسان است). این همان ترتیبی است که الگوریتم های بهینه سازی هزینه به آن علاقه دارند. این الگوریتم ها برای واگذاری کارها از ارزان ترین منبع یعنی R1 شروع میکنند.

همراه با افزایش قیمت اعلام شده برای منابع، توانمندی آنها (نرخ اجرای دستورالعملها در واحد زمان) نیز زیاد میشود. در ستون آخر، قیمت مفید منابع را میتوان دید که ترکیبی از قیمت و توانمندی هر منبع است. در واقع این قیمت از تقسیم قیمت اعلام شده بر توانمندی منبع بدست می آید که سپس برای این که مقادیر خیلی کوچک نباشد در عدد 1000 ضرب میشود. لازم به ذکر است که هر جا از قیمت (ارزانی یا گرانی) صحبت میشود منظور قیمت مفید ($G\$/MI$) است و قیمت اعلام شده ($G\$/sec$) ارزش خاصی ندارد زیرا نمیتوان از آن نتیجه ای گرفت.

جدول 2-5: پیکربندی GR1 برای منابع

نام منبع	نرخ اجرا (MI/sec)	قیمت (G\$/sec)	قیمت مفید (G\$/1000MI)
R1	100	0.5	5
R2	240	1.5	6.25
R3	300	2.5	8.33
R4	500	6	12

و همچنین پیکربندی دیگری تعریف می کنیم که در شبیه سازی را با هر دو انجام می دهیم و نتایج را به هم مقایسه می کنیم.

جدول 3-5: پیکربندی GR2 برای منابع

نام منبع	نرخ اجرا (MI/sec)	قیمت (G\$/sec)	قیمت مفید (G\$/1000MI)
R1	100	0.5	5
R2	180	1.0	5.55
R3	240	1.5	6.25
R4	280	2	7.14
R5	300	2.5	8.33
R6	400	4	10
R7	500	6	12
R8	600	9	15

کد این منبع در شکل 5-14 نشان داده شده است.

```
void Res(GR1 R[4]) //policy baraye manabeh(FIX)
{
    R[1].benf_cost=5;      R[1].cost=0.5;  R[1].Run_Rate=100; R[1].no=1;
    R[2].benf_cost=6.25;   R[2].cost=1.5;  R[2].Run_Rate=240; R[2].no=2;
    R[3].benf_cost=8.33;   R[3].cost=2.5;  R[3].Run_Rate=300; R[3].no=3;
    R[4].benf_cost=12;     R[4].cost=6;    R[4].Run_Rate=500; R[4].no=4;
} //GR1 Difinition finished
//GR2 Difinition
void Res(GR1 R[4]) //policy baraye manabeh(FIX)
```

```

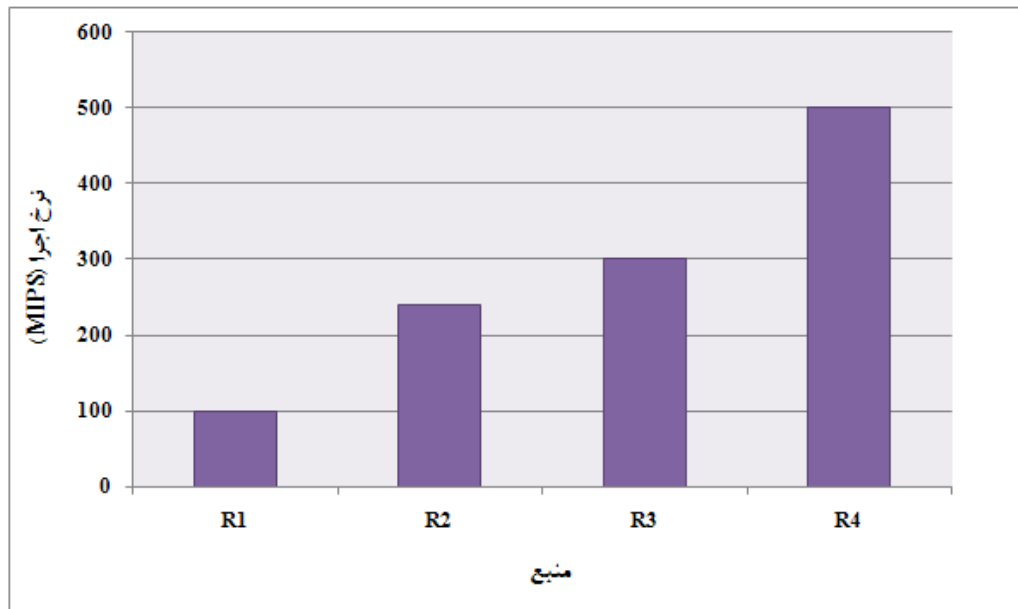
{      R[1].benf_cost=5;      R[1].cost=0.5;  R[1].Run_Rate=100; R[1].no=1;
      R[2].benf_cost=5.55;    R[1].cost=1;    R[1].Run_Rate=180; R[2].no=2;

      R[3].benf_cost=6.25;    R[3].cost=1.5;  R[3].Run_Rate=240; R[3].no=3;
      R[4].benf_cost=7.14;    R[4].cost=2;    R[4].Run_Rate=280; R[4].no=4;
      R[5].benf_cost=8.33;    R[5].cost=2.5;  R[5].Run_Rate=300; R[5].no=5;
      R[6].benf_cost=10;      R[6].cost=4;    R[6].Run_Rate=400; R[6].no=6;
      R[7].benf_cost=12;      R[7].cost=6;    R[7].Run_Rate=500; R[7].no=7;
      R[8].benf_cost=15;      R[8].cost=9;    R[8].Run_Rate=600; R[8].no=8;
} //GR1 Difinition finished

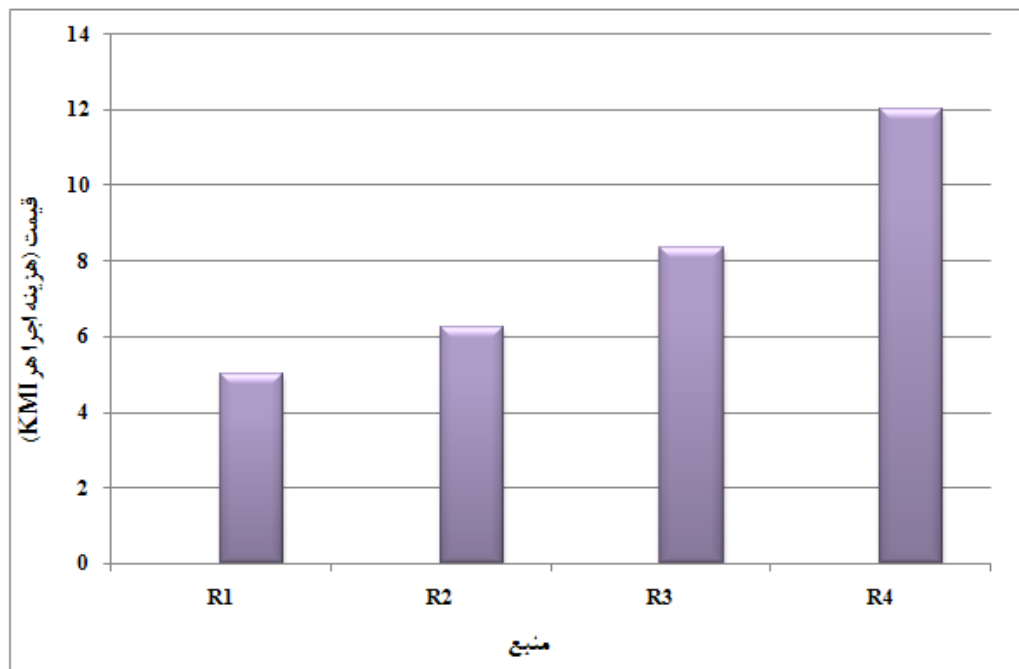
```

شکل 5-14: کد و منبع GR1 کد منبع GR2

شاید اگر کسی تنها به قیمت اعلام شده‌ی منابع توجه کند تصور کند که منابعی که توانمندی زیادی دارند بسیار گران‌تر از منابع دارای توانمندی پایین هستند و استفاده از آنها برای کاربر بسیار پرهزینه است. مشاهده می‌شود که به نظر می‌آید استفاده از منبع R4 در مقابل منبع R1 اصلاً مقرون به صرفه نیست. نکته اینجاست که ما باید پارامتر توانمندی منبع را همزمان با قیمت آن در نظر بگیریم. برای مثال، اگر قیمت اعلام شده برای یک منبع، دو برابر قیمت منبع دیگر باشد، ولی توانمندی آن نیز دو برابر باشد، آنگاه این دو منبع از نظر یک الگوریتم بهینه‌سازی هزینه، دارای ارزش یکسانی هستند. توانمندی‌های منابع GR1 در شکل 5-15 نشان داده شده است. مشاهده می‌شود که منابع گران‌تر دارای توانمندی بالاتری هستند؛ ولی این افزایش توانمندی تقریباً به صورت خطی است و رشد توانمندی از رشد قیمت اعلام شده کمتر است. بنابراین انتظار داریم قیمت مفید منابع نیز به همین ترتیب افزایش یابد ولی رشد آن از رشد قیمت اعلام شده کمتر باشد. این مسأله در شکل 5-16 تأیید می‌شود. در این شکل می‌توان ارزش واقعی منابع و هزینه‌ی اجرای کارها بر روی آنها را با هم مقایسه کرد. می‌بینیم بر خلاف آنچه قبلاً دیده بودیم، اختلاف قیمت منبع R4 و R1 خیلی زیاد نیست و در بعضی مواقع در صورت موجود بودن بودجه و کوتاه بودن مهلت کاربر، استفاده از R4 منطقی به نظر می‌آید.



شکل 5-15: توانمندی منابع در پیکربندی GR1



شکل 5-16: قیمت مفید منابع در پیکربندی GR1

5-2-8- شبیه سازی زمان

الگوریتم‌های زمانبندی که دارای هدف بهینه‌سازی زمان هستند باید بتوانند برنامه‌ی کاربردی کاربر مراجعه کننده را با صرف بودجه‌ی تعیین شده، بر روی گریدی که از مدل اقتصادی بازار کالا استفاده می‌کند، اجرا کرده و نتایج را در کمترین زمان ممکن به کاربر ارائه کنند. از نظر صاحبان و مشتریان منابع، الگوریتمی از بقیه کاراتر است که بتواند کارهای مستقل از هم که برنامه را تشکیل داده‌اند را طوری به منابع موجود در گرید واگذار کند که زمان لازم برای اجرای کارها در پرکارترین منبع نسبت به سایر الگوریتم‌ها کمتر باشد. یک الگوریتم خوب باید بتواند نتایج قابل قابولی را در شرایط گوناگون مانند مهلت‌های کم و زیاد، حضور کاربران مختلف به صورت همزمان در گرید، ناهمگونی کارها و منابع به کاربر ارائه کند. در ادامه الگوریتم‌های BTO, AEBTO, LATO, Minimum و الگوریتم پیشنهادی ALATO در حالت‌های مختلف مورد بررسی قرار گرفته است. حالت اول حالتی است مقدار 200 کار با طول ثابت که مقدار هزینه و زمان توسط کاربر معین می‌باشد. حالت دوم مقدار 200 کار با طول ثابت در یک زمان تعیین شده توسط کاربر در ناهمگونی‌های مختلف بودجه برای 4 منبع مورد بررسی قرار گرفته است. حالت سوم، مقدار 200 کار با طول متغیر در یک زمان تعیین شده توسط کاربر در ناهمگونی‌های مختلف بودجه برای 4 منبع مورد بررسی قرار گرفته است. حالت چهارم، مقدار 400 کار در حالت‌های سه گانه بالا ذکر شده‌اند. هر چهار حالت در اتوماتای یادگیر استاندارد و حالت‌های سه گانه مدل S مورد آزمایش قرار گرفته‌اند و نتایج آورده شده‌اند. توجه داشته باشید که الگوریتم Minimum بهترین حالت اجرا و پیاده‌سازی می‌باشد که طریقه بکاربردن آن در فصل قبل شرح داده شد و مقدار آن در حالت‌های مختلف اتوماتا تغییری نمی‌کند.

5-2-8-1- بهینه سازی زمان در یک هزینه و زمان معین (حالت اول)

الگوریتم پیشنهادی را با انواع مدل‌های اتوماتای یادگیر با توجه به شرایط جدول 5-4 مورد آزمایش قرار داده ایم، برای بررسی مناسب طول کارها در تست تمامی مدل‌ها برابر در نظر گرفته شده است. پارامتر محدودی طول کارها نشان‌دهنده‌ی محدوده‌ی توزیع یکنواخت طول کارها می‌باشد. پارامتر تعداد آزمایش‌ها نشان می‌دهد که برای بدست آوردن زمان اجرای برنامه با استفاده از

الگوریتم‌های موجود در یک مهلت معین، تعداد 20 آزمایش انجام شده است و سپس میانگین مقادیر برای بررسی انتخاب می‌شود.

جدول 4-5: شرایط آزمایش بهینه‌سازی زمان با بودجه ثابت

پارامتر	الگوریتم	تعداد کاربر	تعداد کارها	مهلت	بودجه	محدوده‌ی طول کارها	پیکربندی منابع	تعداد آزمایش‌ها
مقدار	متغیر	1	200	50000	150000	[10...200]	GR1	20

حال شرایط زیر را در حالت‌های مختلف اتوماتای استاندارد (مدل P-LRP)، اتوماتای مدل S (S-LRP، S-LRI و S-LReP) مورد ارزیابی قرار می‌دهیم.

5-2-8-1-1 مدل P-LRP

برای این مدل نرخ پاداش و جریمه را برابر با مقادیر مختلفی قرار داده ایم و نتایج را با هم تست کرده ایم، در این مدل (P)، پاسخ $\beta_i(n)$ با احتمال c_i برابر با 1 (پاسخ مطلوب) و با احتمال $1-c_i$ برابر با صفر است (پاسخ نامطلوب). از آنجایی که تنها دو الگوریتم LATO و ALATO الگوریتم‌های پیاده سازی شده بر اساس اتوماتای یادگیر می‌باشند، بنابر این، میزان تغییرات پاداش و جریمه در بودجه 150000 برای این الگوریتم‌ها مد نظر قرار می‌دهیم. این نتایج در اتوماتای استاندارد در جدول 5-5 نشان داده شده است.

از آنجاکه نرخ پاداش و جریمه برابر هستند، باید مقادیر کمی داشته باشند تا الگوریتم بتواند به تدریج و به آهستگی، زمانبندی مناسب را بدست آورد. در بودجه ثابت 150000 میزان نرخ پاداش و جریمه یکسان و 0.01 بهترین حالت برای الگوریتم‌های بر پایه اتوماتای یادگیر می‌باشند.

جدول 5-5: تغییرات میزان پاداش و جریمه در مدل P-LRP و در الگوریتم‌های بهینه‌سازی زمان با بودجه ثابت 150000

الگوریتم‌ها	نرخ	نرخ
-------------	-----	-----

		BTO	AEBTO	LATO	ALATO	Minimum
b=0.01	a=0.01	17,440	17,240	17,240	17,175	17,150
b=0.05	a=0.05	17,900	17,440	17,314	17,275	17,150
b=0.1	a=0.1	17,950	17,645	17,420	17,354	17,150
b=0.5	a=0.5	18,543	17,840	17,580	17,408	17,150

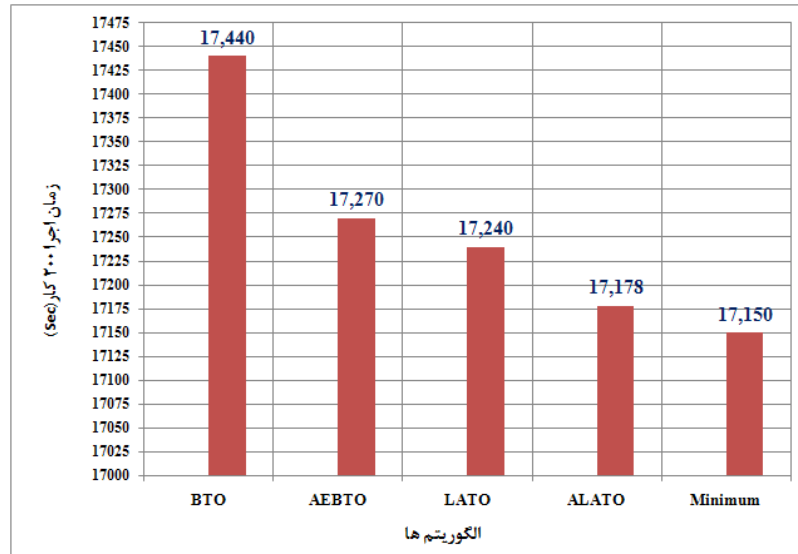
از آنجا که الگوریتم ALATO حالت های بیشتری از کارها را در نظر می گیرد بنابر این میزان سرعت یادگیری اتوماتا ارائه شده ALATO از LATO بیشتر می باشد. این مطلب در شکل 5-17 نشان داده شده است. در اینجا با افزایش مقدار a و b به تدریج میزان رشد زمان در الگوریتم ALATO از الگوریتم LATO کمتر می باشد.



شکل 5-17: روند یادگیری الگوریتم های LATO و ALATO

همانطور که در شکل بالا دیده می شود متغیر Delta-ALATO نشاندهنده میزان اختلاف زمانی الگوریتم ALATO با مقدار الگوریتم Minimum در پاداش های (جریمه ها) متفاوت می باشد و همچنین Delta-LATO میزان اختلاف زمانی الگوریتم ALATO با مقدار الگوریتم Minimum در پاداش ها (جریمه ها) متفاوت می باشد. این Linear (Delta-ALATO) از میزان شیب Linear (Delta-LATO) کمتر می باشد. بدان معنی است که میزان یادگیری الگوریتم ALATO نسبت به الگوریتم LATO بیشتر است. درواقع، الگوریتم ALATO سریعتر از الگوریتم LATO کارها را به منابع اختصاص می دهد. در شکل 5-18 مدت زمان اجرا در حالت قابل قبول اتوماتای استاندارد یعنی حالتی که جریم و پاداش 0.01 هستند نشان داده شده است. همانطور

که می بینید الگوریتم ALATO تقریباً 60 واحد از الگوریتم LATO بهتر عمل می کند و تنها 30 واحد با حالت بهینه اختلاف دارد.



شکل 5-18: مقایسه زمان اجرا الگوریتم ها در اتوماتای استاندارد با بودجه ثابت 150000

مدل S-LRP

5-2-1-8-2

در روش های مدل S، اگر در تکرار n ام عمل α_i انتخاب شده باشد، پاسخ (پاسخ نامطلوب) محیط به آن برابر $\beta_i(n)=1$ می باشد و پاسخ (پاسخ مطلوب) محیط به آن عمل برابر $\beta_i(n)=\frac{1}{1+\frac{PreviousTime}{NewTime}}$ می باشد،

(یعنی به نسبت انتخاب منبع ارزان تر پاسخ بهتری از محیط دریافت می کند). که طبق نتایج که در فصل شبیه سازی مشاهده خواهیم نمود که روش S-LRP با نرخ پاداش برابر با 0.1 و نرخ جریمه برابر با 0.05 بهترین نتیجه را خواهد داشت. این نتایج در جدول 5-6 نشان داده شده است. از آنجاکه نرخ پاداش و جریمه برابر هستند، باید مقادیر کمی داشته باشند تا الگوریتم بتواند به تدریج و به آهستگی، زمان بندی مناسب را بدست آورد. در بودجه ثابت 150000 میزان نرخ پاداش و جریمه یکسان و 0.05 بهترین حالت برای الگوریتم های بر پایه اتوماتای یادگیر می باشند.

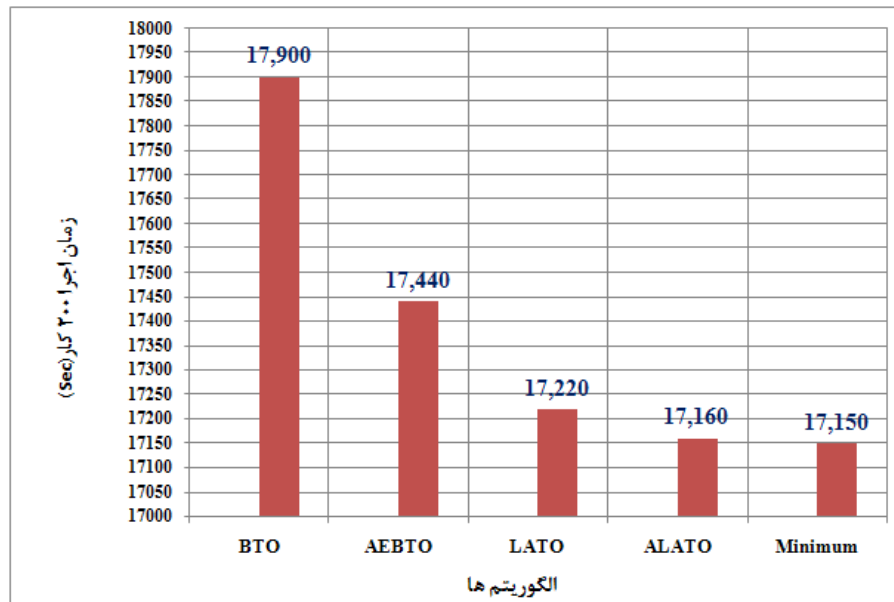
همانطور که گفته شد، نرخ پاداش و جریمه با هم برابر هستند، پس باید مقادیر کمی داشته باشند تا الگوریتم بتواند به تدریج

و به آهستگی، زمانبندی مناسب را بدست آورد. به علت اینکه در این مدل پاسخ محیط (پاسخ مطلوب) بستگی به میزان ارزانی منبع انتخابی نسبت به تکرار قبل دارد، نتایج مدل S-LRP بهتر از نتایج مدل P-LRP شده است.

جدول 5-6: تغییرات میزان پاداش و جریمه در مدل S-LRP و الگوریتم‌های بهینه‌سازی زمان با بودجه ثابت 150000

نرخ جریمه	نرخ پاداش	الگوریتم‌ها				
		BTO	AEBTO	LATO	ALATO	Minimum
b=0.01	a=0.01	17,440	17,240	17,230	17,168	17,150
b=0.05	a=0.05	17,900	17,440	17,220	17,160	17,150
b=0.1	a=0.1	17,950	17,645	17,440	17,370	17,150
b=0.5	a=0.5	18,543	17,840	17,460	17,400	17,150

در شکل 5-19 مدت زمان اجرا در حالت قابل قبول اتوماتای S-LRP و حالتی که جریم و پاداش 0.05 هستند نشان داده شده است. همانطور که می بینید الگوریتم ALATO تقریباً 60 واحد از الگوریتم LATO بهتر عمل می کند و تنها 10 واحد با حالت بهینه اختلاف دارد.



شکل 5-19: مقایسه زمان اجرا الگوریتم‌ها در اتوماتای S-LRP با بودجه ثابت 150000

3-1-8-2-5 مدل S-LRI

در این مدل نرخ جریمه برابر با صفر ($b=0$) است و 4 مقدار مختلف را برای نرخ پاداش آزمایش کرده ایم. این نتایج در جدول 7-5 نشان داده شده است.

به علت اینکه نرخ جریمه برابر با صفر است و در نتیجه کارهایی که منبع اشتباهی را انتخاب می‌کنند، جریمه نمی‌شوند، اتوماتای یادگیر نمی‌تواند به خوبی الگوریتم پیشنهادی را به زمانبندی مناسب هدایت کند.

جدول 7-5: تغییرات میزان پاداش و جریمه در مدل S-LRI و الگوریتم‌های بهینه‌سازی زمان با بودجه ثابت 150000

نرخ جریمه	نرخ پاداش	الگوریتم‌ها				
		BTO	AEBTO	LATO	ALATO	Minimum
b=0	a=0.01	17,440	17,240	17,238	17,178	17,150
b=0	a=0.05	17,900	17,440	17,290	17,196	17,150
b=0	a=0.1	17,950	17,645	17,320	17,240	17,150
b=0	a=0.5	18,543	17,840	17,400	17,310	17,150

4-1-8-2-5 مدل S-LReP

در این مدل نرخ جریمه بسیار کوچکتر از نرخ پاداش است و 4 مقدار مختلف را برای نرخ پاداش و جریمه آزمایش کرده ایم. این نتایج در جدول 8-5 نشان داده شده است.

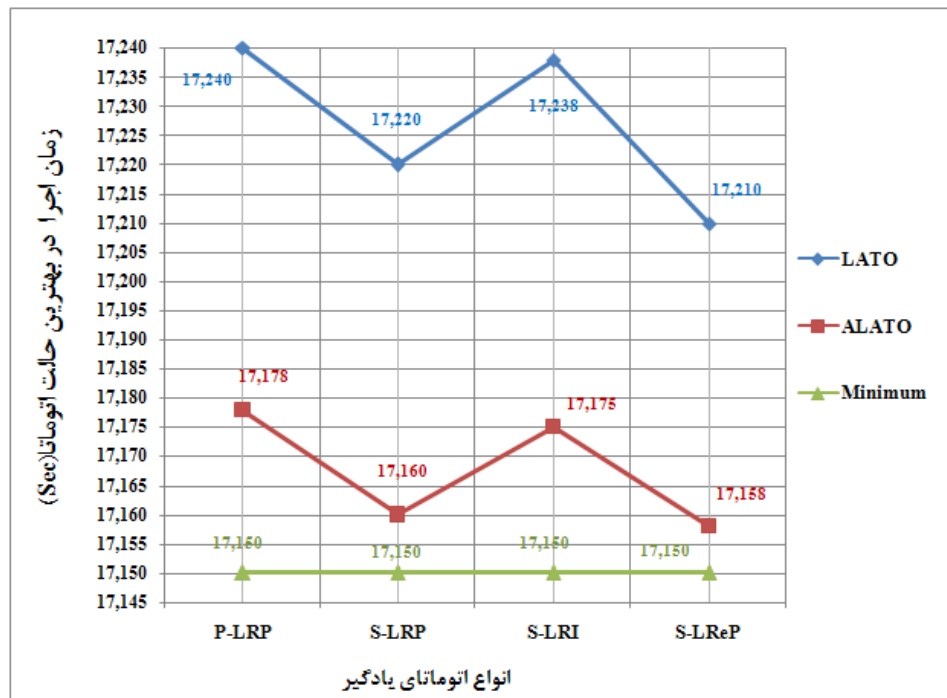
جدول 8-5: تغییرات میزان پاداش و جریمه در مدل S-LReP و الگوریتم‌های بهینه‌سازی زمان با بودجه ثابت 150000

نرخ جریمه	نرخ پاداش	الگوریتم‌ها				
		BTO	AEBTO	LATO	ALATO	Minimum
b=0.01	a=0.1	17,440	17,240	17,232	17,183	17,150
b=0.05	a=0.1	17,900	17,440	17,210	17,158	17,150
b=0.01	a=0.5	17,950	17,645	17,383	17,365	17,150
b=0.05	a=0.5	18,543	17,840	17,470	17,440	17,150

با توجه به جدول 8-5، بهترین نرخ جریمه و پاداش برای مدل S-LReP برابر با $a=0.1$ و $b=0.05$ است، با توجه به اینکه پاسخ محیط (پاسخ مطلوب) در تکرار n ام به عمل انتخابی (α_i) متناسب با میزان ارزانی منبع انتخاب شده نسبت به تکرار قبل می‌باشد و پاداش برابر 0.1 است، هرچه منبع ارزان‌تر انتخاب شود احتمال آن

عمل نیز بیشتر می‌گردد و الگوریتم را به سوی زمان‌بندی بهتر با بهینه‌سازی هزینه هدایت می‌کند.

در شکل 5-20 مقایسه ای بین اتوماتای های ذکر شده در الگوریتم های LATO و ALATO به همراه الگوریتم Mnimun صورت گرفته است. در اینجا بهترین زمان اجراهای هر اتوماتا برای بودجه 150000 در نظر گرفته شده است. حد پایین و بالای نمودار نشان داده شده 17175 و 17240 در نظر گرفته شده و گام رشد داده در آن 5 در نظر گرفته شده است. همانطور که مشاهده می کنید الگوریتم مینیمم در هر چهار اتوماتا یکسان می باشد. الگوریتم ALATO نسبت به الگوریتم LATO بسیار از نظر زمان بهبود پیدا کرده است و بسیار به حالت بهینه نزدیک شده است. مدل S-LReP نسبت به سایر اتوماتاها زمان کمتری را صرف می کند. این به این خاطر است که هرچه میزان پاداش نسبت به جریمه بیشتر باشد مدت زمان برای بروز رسانی و انتخاب منبع با تکرار کمتر بیشتر امکان پذیر می شود. مدل S نسبت به مدل P نتیجه بهتری داده است. چونکه با لحاظ کردن میزان تغییرات جزئی تر در β_i دقت اتوماتا را بیشتر کرده و زمان ثابت شدن احتمالات انتخاب منابع با تکرار کمتر میسر می شود و در نتیجه زمان پایدار شدن اتوماتا نسبت به حالت P بیشتر است.



شکل 5-20: مقایسه زمان اجرا اتوماتاها با بودجه ثابت 150000 در بهترین حالت هر اتوماتا

5-2-8-2- بهینه سازی زمان در زمان معین با بودجه های مختلف (حالت دوم)

در این بخش، الگوریتم های زمانبندی که هدفشان بهینه سازی زمان است را با فراهم کردن بودجه های مختلف با هم مقایسه می کنیم. همچنین نشان داده می شود که هر کدام از الگوریتم ها در شرایط یکسان (کاربر، برنامه ی کاربردی و منابع)، اجرای برنامه ی کاربر را طی چه زمانی به پایان می رسانند. مطمئناً الگوریتمی برای محیط گزید مناسبتر است که عمل زمانبندی کارها را طوری روی منابع انجام دهد که در کمترین زمان، نتایج اجرای برنامه را به کاربر ارائه کند.

الگوریتم پیشنهادی را با انواع مدل های اتوماتای یادگیر با توجه به شرایط جدول 5-9 مورد آزمایش قرار داده ایم، برای بررسی مناسب طول کارها در تست تمامی مدل ها برابر در نظر گرفته شده است. پارامتر محدودی طول کارها نشان دهنده ی محدوده ی توزیع یکنواخت طول کارها می باشد. پارامتر تعداد آزمایش ها نشان می دهد که برای بدست آوردن زمان اجرای برنامه با استفاده از الگوریتم های موجود در یک مهلت معین، تعداد 20 آزمایش انجام شده است و سپس میانگین مقادیر برای بررسی انتخاب می شود.

جدول 5-9: شرایط آزمایش بهینه سازی زمان با بودجه متغیر

پارامتر	الگوریتم	تعداد کاربر	تعداد کارها	مهلت	بودجه	محدوده ی طول کارها	پیکربندی منابع	تعداد آزمایش ها
مقدار	متغیر	1	200	50000	متغیر	[10...200]	GR1	20

از آنجایی که اتوماتای مدل S زمان بهتری را نسبت به اتوماتای استاندارد ارائه می کند. این اتوماتا برای بررسی انتخاب می شود. در انواع مدل S اتوماتای S-IReP نتیجه بهتری را ارائه کرده است. بنابر این، تنها نتایج را در این اتوماتا مورد ارزیابی قرار می دهیم.

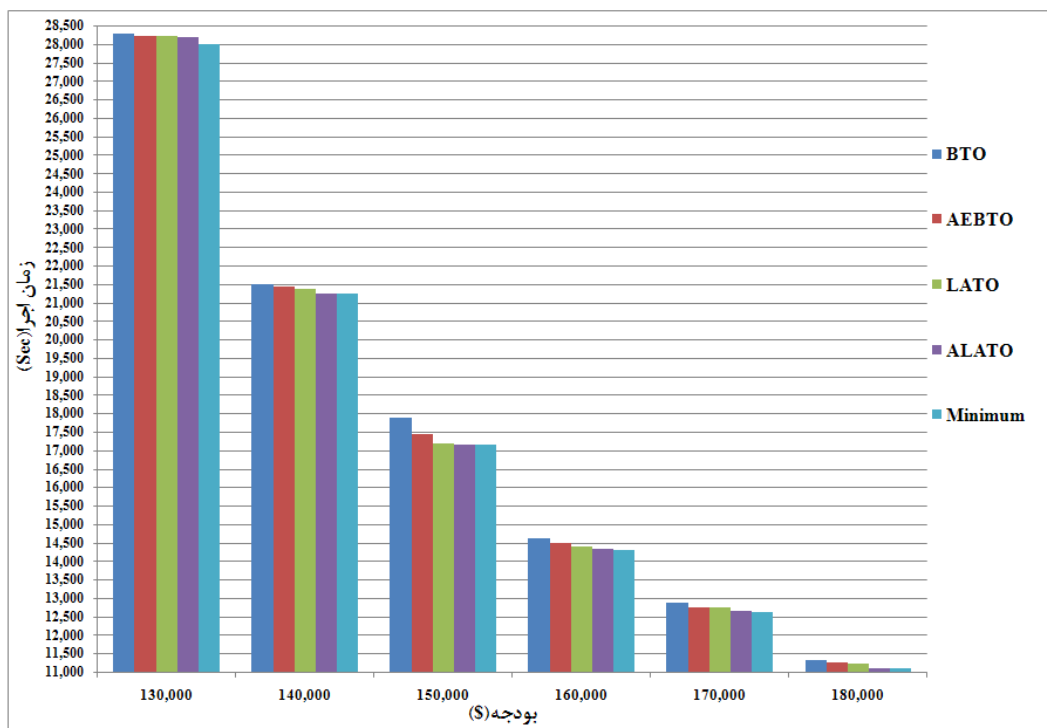
نتایج آزمایش های انجام شده برای مقایسه ی الگوریتم های بهینه سازی زمان در جدول 5-10 آورده شده است. این نتایج در اتوماتای مدل S-IReP که بهترین زمان را ارائه کرده است لحاظ شده است. ستون اول، بودجه ی تعیین شده توسط کاربر را نشان می دهد.

مقدار بودجه از 130000 تا 180000 تغییر می‌کند تا بتوانیم کارایی الگوریتم‌ها را با بودجه‌های مختلف، ارزیابی کرده و با یکدیگر مقایسه کنیم. هر کدام از ستون‌های بعدی نمایانگر یکی از الگوریتم‌ها می‌باشند و ستون آخر، مقدار مینیمم زمان محاسبات را با صرف بودجه‌های مختلف نشان می‌دهد که این مینیمم در بیشتر موارد برای هیچ الگوریتمی دست یافتنی نیست. هر کدام از خانه‌های جدول مقدار زمان اجرای برنامه‌ی کاربر را برای الگوریتم زمانبندی مربوطه و با صرف بودجه‌ی تعیین شده مشخص می‌کند. برای مثال، با در اختیار داشتن بودجه‌ی 150000، الگوریتم زمانبندی BTO اجرای کارهای برنامه‌ی کاربردی متعلق به کاربر را بعد از مدت زمان 17900 واحد به اتمام می‌رساند و الگوریتم LATO و ALATO برای اجرای همین برنامه و با صرف بودجه‌ی فراهم شده، به ترتیب به مدت زمانهای 17210 و 17158 واحد نیاز دارد. برای هر کدام از بودجه‌ها تعداد 20 آزمایش انجام داده‌ایم و میانگین مقادیر بدست آمده را برای هر کدام از الگوریتم‌ها به عنوان زمان متوسط اجرای کارها با بودجه‌ی تعیین شده در نظر گرفتیم. اگر تنها از مقادیر بدست آمده از یک آزمایش استفاده کنیم، آنگاه اگرچه ممکن است بتوانیم الگوریتم‌ها را با بودجه‌ی 140000 با هم مقایسه کنیم ولی نمی‌توان مقادیر را در کنار مقادیر بدست آمده با یک بودجه‌ی دیگر مانند 150000 قرار داد و تغییرات را بررسی کرد. مقادیر مختلفی برای زمان اجرای کارها با استفاده از هر کدام از الگوریتم‌ها بدست آمده است که مجبور می‌شویم در نهایت پس از انجام آزمایشات متعدد، عمل میانگین‌گیری زمان اجرا را برای هر کدام از الگوریتم‌ها انجام دهیم. این اختلافات به علت توزیع تصادفی طول کارها در محدوده‌ی تعیین شده بوجود می‌آید و هر چه کارها ناهمگون‌تر باشند (محدوده‌ی توزیع طول کارها بزرگتر باشد) این اختلافات نیز بیشتر خواهد بود.

جدول 5-10: زمان اجرای برنامه در الگوریتم‌های بهینه‌سازی زمان با بودجه‌های مختلف

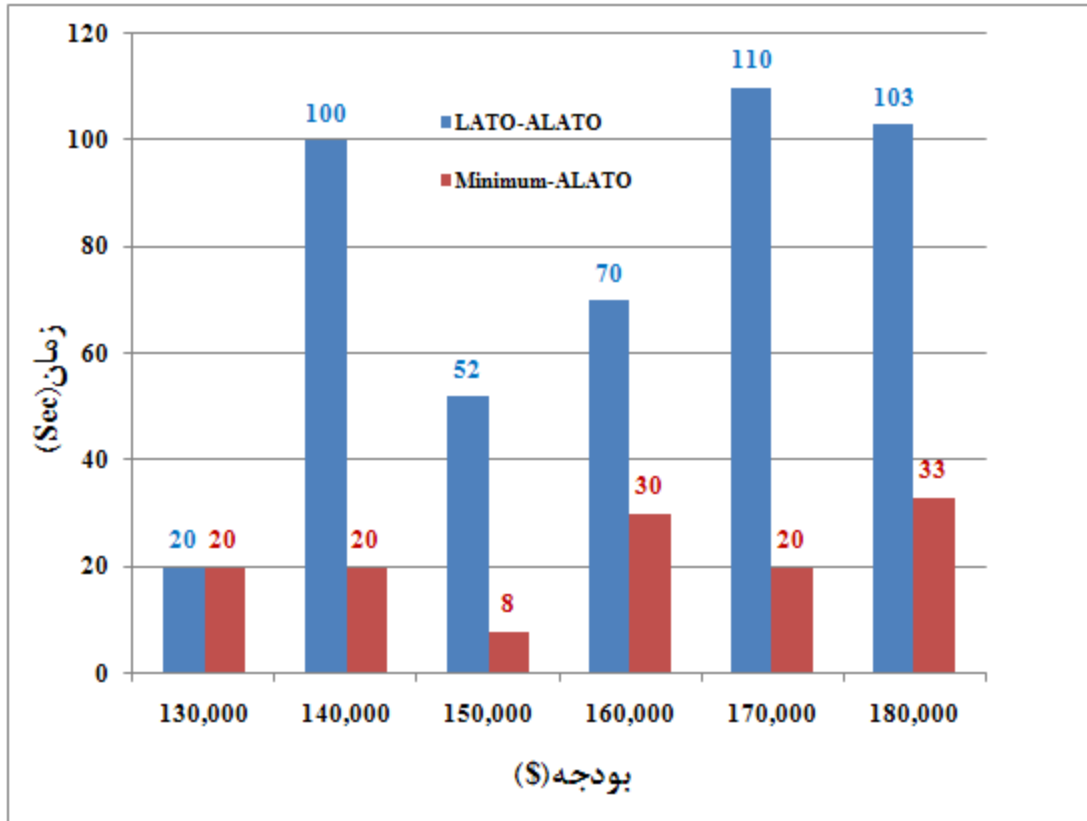
بودجه	الگوریتم‌ها				
	BTO	AEBTO	LATO	ALATO	Minimum
130,000	28,300	28,220	28,220	28,200	28,020
140,000	21,500	21,450	21,370	21,305	21,250
150,000	17,900	17,440	17,210	17,158	17,150
160,000	14,610	14,490	14,400	14,305	14,300
170,000	12,890	12,760	12,750	12,642	12,620
180,000	11,320	11,270	11,220	11,120	11,090

شکل 5-21 الگوریتم‌های بهینه سازی با بودجه‌های مختلف را نشان می‌دهد. حد پایین و بالای قسمت زمان در نمودار نشان داده شده 11000 و 28500 در نظر گرفته شده و گام رشد داده در آن 500 در نظر گرفته شده است. همانطور که دیده می‌شود. در همه حالات الگوریتم ALATO از سایر الگوریتم‌ها بهتر عمل کرده و بسیار به الگوریتم ایده آل Minimum نزدیک شده است. الگوریتم AEBTO از BTO بهتر عمل کرده است ولی از الگوریتم‌های پیاده سازی شده با اتوماتا زمان بیشتری را صرف می‌کند. همانطور که در نمودار مشاهده می‌شود، مدت زمانی که همه‌ی الگوریتم‌ها برای اجرای کارها بدست آورده‌اند تقریباً نزدیک به هم است. با افزایش بودجه همواره مقدار زمان اجرا کاهش پیدا می‌کند و همواره به مینیمم نزدیکتر می‌شوند در واقع، هر چه قدر که بودجه‌ی تعیین شده توسط کاربر بیشتر باشد، الگوریتم‌ها به زمان کمتری برای اجرای کارهای برنامه‌ی کاربردی نیاز دارند؛ زیرا هنگامی که کاربر بودجه‌ی فراوانی را برای اجرای برنامه‌ی کاربردی خود بر روی گرید در نظر می‌گیرد، الگوریتم‌های زمانبندی می‌توانند گستره‌ی وسیع‌تری از منابع را به کار گرفته و با اجرای کارها بر روی منابع گران و توانمند زمان اجرا را کاهش دهند.



شکل 5-21: نتایج الگوریتم‌های بهینه‌سازی زمان با بودجه‌های مختلف

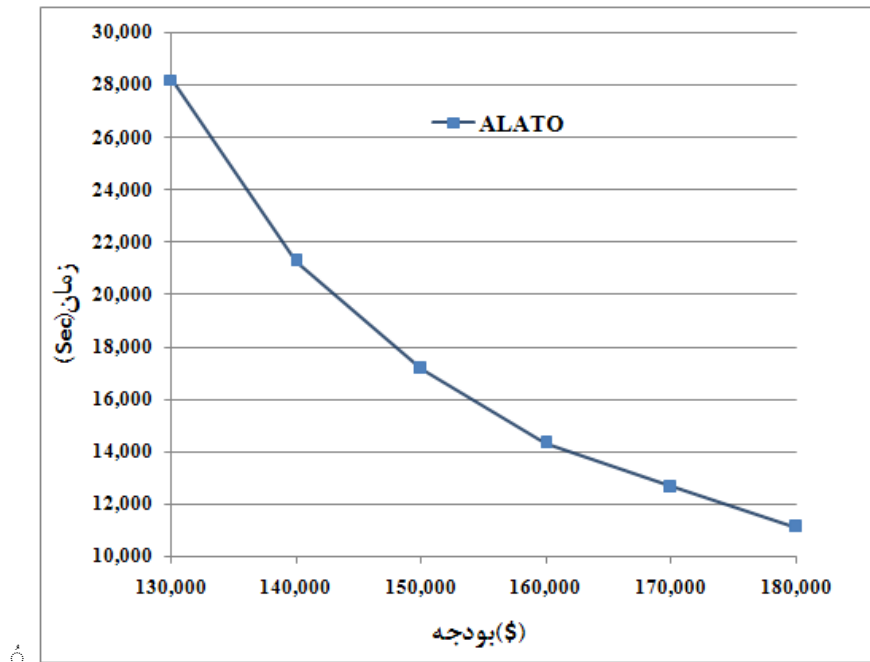
از آنجا که هدف ما بهبود بهترین الگوریتم گزارش شده برای بهینه‌سازی زمان تا قبل از نگارش این پایان‌نامه الگوریتم LATO بود، در اینجا الگوریتم ALATO را که از اتوماتاهای یادگیر استفاده می‌کند، با این الگوریتم مقایسه می‌کنیم. شکل 5-22 اختلاف نتایج این دو الگوریتم را در بودجه‌های مختلف نشان می‌دهد. مشاهده می‌شود که زمان اجرای بدست آمده با استفاده از الگوریتم ALATO از زمان اجرا در الگوریتم LATO کمتر است. می‌بینیم که الگوریتم ALATO در بودجه‌های کم توانسته است تا 100 واحد در زمان صرفه‌جویی کند. البته این میزان صرفه‌جویی در بودجه‌های زیاد به حدود 110 واحد می‌رسد.



شکل 5-22: مقایسه‌ی الگوریتم‌های ALATO و LATO و Minimum با بودجه‌های مختلف

برای اینکه ببینیم چقدر امکان بهبود الگوریتم ALATO که بهترین الگوریتم پیشنهاد شده می‌باشد وجود دارد باید این الگوریتم را با مینیمم زمان اجرای کارها مقایسه کنیم. اختلاف نتایج بدست آمده از الگوریتم ALATO با مینیمم زمان محاسبات در بودجه‌های مختلف در شکل 5-22 نشان داده شده است. مشاهده می‌شود که این اختلاف بسیار ناچیز و در همه‌ی موارد کمتر از 8 واحد است. این مقدار در حالی که زمان اجرای کارها در همه‌ی بودجه‌ها بیش از 10 هزار واحد می‌باشد اصلاً قابل ملاحظه نیست. بنابراین می‌توان گفت که الگوریتم ALATO تا حد ممکن به زمان مینیمم نزدیک شده است و از آنجا که می‌دانیم این مینیمم در اغلب موارد توسط هیچ الگوریتمی قابل دستیابی نیست، ارائه‌ی یک الگوریتم جدید حداکثر تا کمتر از 8 واحد می‌تواند بهبود حاصل کند که اصلاً نمی‌توان آن را در نظر گرفت.

همانطور که قبلاً دیده شد، همه‌ی الگوریتم‌های پیشنهادی روند یکسانی را در تغییرات زمان بر حسب بودجه دارند. برای نمونه، این روند برای الگوریتم LATO در شکل 5-23 نشان داده شده است. مهمترین نکته در این تغییرات، کاهش زمان اجرای کارها در اثر افزایش بودجه است. علت این امر این است که با افزایش بودجه‌ی فراهم شده توسط کاربر، الگوریتم‌ها می‌توانند با به کار بردن منابع گرانتر، از منابع بیشتری برای اجرای کارها استفاده کرده و زمان اجرا را کاهش دهند. این روند کاهش زمان در همه‌ی الگوریتم‌ها دیده می‌شود. البته این کاهش در ابتدا سرعت بیشتری دارد و هر چه بودجه بیشتر شود این کاهش نیز کمتر می‌شود. در واقع تغییرات زمان بر حسب بودجه همانطور که دیده می‌شود به صورت غیرخطی است. در ابتدا که بودجه کم است الگوریتم‌ها مجبورند برای اینکه هزینه‌ی اجرای کارها از بودجه فراتر نرود، از تعداد اندکی از منابع ارزان استفاده کنند؛ ولی با زیاد شدن بودجه، الگوریتم‌ها می‌توانند از منابع گران‌تر نیز برای اجرای کارها استفاده کرده و زمان را کاهش دهند. البته به تدریج منابع اضافه شده گرانتر می‌شوند و به کار بردن آنها به هزینه‌ی بیشتری نیاز دارد و بنابراین تأثیر افزایش بودجه در کاهش زمان محاسبات، کمتر می‌شود. در نهایت، هنگامی که با صرف بودجه‌ی تعیین شده بتوان از همه‌ی منابع برای زمانبندی کارها استفاده کرد، آنگاه دیگر افزایش بودجه، تغییری در زمان محاسبات ایجاد نمی‌کند و نمودار به صورت یک خط افقی در می‌آید. از این طرف هم نقطه‌ی شروع نمودار کمترین بودجه‌ای است که با صرف آن می‌توان کارها را قبل از مهلت تحویل داد که در واقع مینیمم هزینه‌ی محاسبات می‌باشد. برای مهلت 100 هزار که در این آزمایش‌ها استفاده می‌شد، مقدار مینیمم هزینه حدود 115 هزار محاسبه شده بود و ما آزمایش‌ها را از بودجه‌ای بیشتر از آن یعنی مقدار 120000 شروع کرده ایم.



شکل 5-23: روند تغییرات زمان اجرای کارها در الگوریتم ALATO با بودجه های مختلف

3-8-2-5- بهینه سازی زمان در ناهمگونی های مختلف (حالت سوم)

در این بخش، الگوریتم های زمانبندی که هدفشان بهینه سازی زمان است برای زمانبندی کارهایی با ناهمگونی های مختلف با هم مقایسه می شوند. همچنین بررسی می کنیم که هر الگوریتم در ناهمگونی های مختلف کارها، چه زمانی برای اجرای آنها بدست می آورد. همانطور که قبلاً گفته شد، برنامه ی کاربردی عرضه شده توسط کاربر از تعدادی کار مستقل از هم تشکیل شده است. هر کار، دارای طول مخصوص به خود است که بر حسب میلیون دستورالعمل (MI) بیان می شود. همیشه طول کارها را متغیر در نظر می گیریم؛ به این صورت که یک محدوده ی (حداکثر... حداقل) برای طول کارها انتخاب می شود. سپس برای طول هر کار، یک مقدار به صورت تصادفی از این محدوده انتخاب می شود به طوریکه توزیع طول کارها یکنواخت باشد. برای همگون ترین حالت، محدوده ی توزیع طول کارها را (100000...110000) و برای ناهمگون ترین حالت محدوده ی (10000...200000) را در نظر می گیریم. میانگین محدوده را در هر حالتی مقدار 105000 قرار می دهیم تا بتوان نتایج یک الگوریتم

معین را در ناهمگونی‌های مختلف کارها بررسی کرد. برای راحتی واحد طول را 1000MI قرار می‌دهیم و برای مثال، محدوده ناهمگون‌ترین حالت را به صورت (10...200) بیان می‌کنیم.

شرایط انجام آزمایش شبیه‌سازی در جدول 5-11 آمده است. پارامتر محدوده طول کارها، متغیر است و آزمایش‌ها در محدوده‌های مختلف (یعنی ناهمگونی‌های مختلف) انجام می‌شود. پارامتر تعداد آزمایش‌ها مشخص می‌کند که برای بدست آوردن مقادیر زمان اجرای الگوریتم‌ها در یک ناهمگونی معین تعداد 20 آزمایش انجام شده و سپس میانگین مقادیر برای بررسی، انتخاب شده است.

جدول 5-11: شرایط آزمایش بهینه‌سازی زمان در ناهمگونی‌های مختلف و بودجه 150000

پارامتر	الگوریتم	تعداد کاربر	تعداد کارها	مهلت	بودجه	محدوده طول کارها	پیکربندی منابع	تعداد آزمایش‌ها
مقدار	متغیر	1	200	20000	150000	متغیر	GR1	20

از آنجایی که اتوماتای مدل S زمان بهتری را نسبت به اتوماتای استاندارد ارائه می‌کند. این اتوماتا برای بررسی انتخاب می‌شود. در انواع مدل S اتوماتای S-LReP نتیجه بهتری را ارائه کرده است. بنابر این، تنها نتایج را در این اتوماتا مورد ارزیابی قرار می‌دهیم. برای هر ناهمگونی، تعداد 20 آزمایش انجام دادیم و میانگین مقادیر بدست آمده را برای هر کدام از الگوریتم‌ها، به عنوان زمان متوسط اجرای کارها در نظر گرفتیم. این عمل، همانند آنچه در آزمایش‌های بودجه‌ی متغیر در بخش قبل آمد، انجام می‌شود.

نتایج آزمایش‌های انجام شده برای مقایسه‌ی الگوریتم‌های بهینه‌سازی زمان، در جدول 5-12 آورده شده است. ستون اول، محدوده‌ی توزیع طول کارها را نشان می‌دهد. این محدوده از همگون‌ترین حالت یعنی (100...110) شروع می‌شود و با افزایش 10 هزار واحد از هر دو طرف در هر مرحله، به ناهمگون‌ترین حالت یعنی محدوده‌ی (10...200) می‌رسد. ستون‌های بعدی هر کدام نمایانگر یکی از الگوریتم‌ها می‌باشد و ستون آخر، مقدار مینیمم را برای زمان محاسبات در آن ناهمگونی نشان می‌دهد. این مینیمم در بیشتر موارد برای هیچ الگوریتمی دست‌یافتنی نیست. هر کدام از خانه‌های جدول، مقدار زمان اجرای کارها را برای الگوریتم

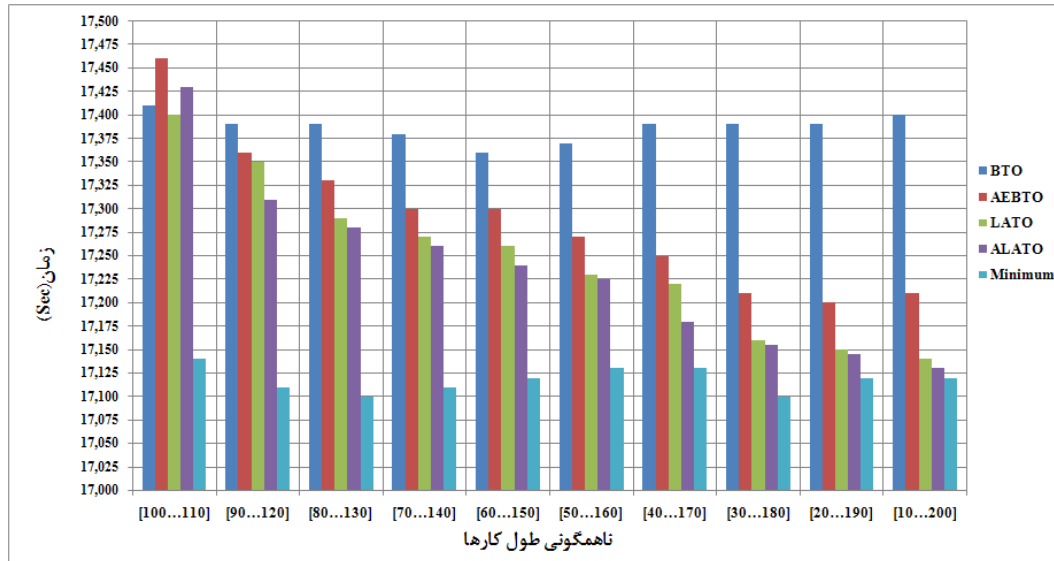
زمانبندی مربوطه و در ناهمگونی مورد نظر مشخص می‌کند. برای مثال، الگوریتم زمانبندی AEBTO، کارهایی را که دارای ناهمگونی (170...40) می‌باشند (تعداد 200 کار)، در مدت زمان 17250 واحد انجام می‌دهد و الگوریتم ALATO در همین شرایط، برای اجرای کارها به زمان 17220 واحد نیاز دارد.

جدول 5-12: زمان اجرای برنامه در الگوریتم‌های بهینه‌سازی زمان با ناهمگونی‌های مختلف

ناهمگونی	الگوریتم‌ها				
	BTO	AEBTO	LATO	ALATO	Minimum
[100...110]	17,410	17,460	17,400	17,430	17,140
[90...120]	17,390	17,360	17,350	17,310	17,110
[80...130]	17,390	17,330	17,290	17,280	17,100
[70...140]	17,380	17,300	17,270	17,260	17,110
[60...150]	17,360	17,300	17,260	17,240	17,120
[50...160]	17,370	17,270	17,230	17,225	17,130
[40...170]	17,390	17,250	17,220	17,180	17,130
[30...180]	17,390	17,210	17,160	17,155	17,100
[20...190]	17,390	17,200	17,150	17,145	17,120
[10...200]	17,400	17,210	17,140	17,130	17,120

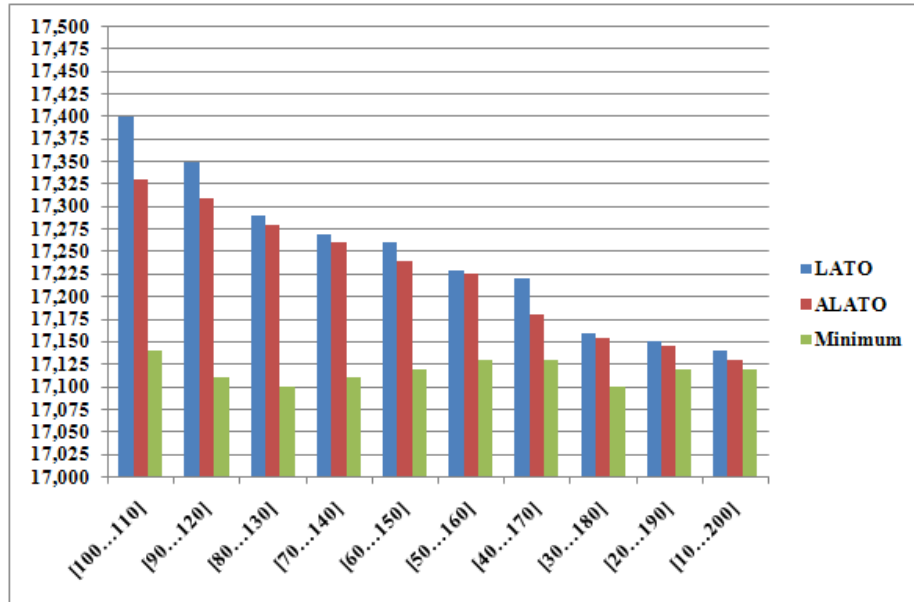
نتایج بدست آمده از الگوریتم‌های زمانبندی با استراتژی بهینه‌سازی زمان، در شکل 5-24 نشان داده شده است. همانطور که در نمودار مشاهده می‌شود، تغییرات زمان در اثر افزایش ناهمگونی کارها، در هر کدام از الگوریتم‌ها روند متفاوتی دارد. زمان اجرای بدست آمده در الگوریتم BTO، بسیار کم تحت تأثیر تغییرات ناهمگونی قرار گرفته است. در واقع پارامتر ناهمگونی، تأثیری بر کارایی این الگوریتم ندارد که این اصلاً مطلوب نیست. ما از یک الگوریتم خوب انتظار داریم که بتواند از ناهمگونی کارها استفاده کند و زمان اجرای کارها را کاهش دهد الگوریتم AEBTO در ناهمگونی کم از همه الگوریتم‌ها بدتر عمل کرده است ولی با افزایش ناهمگونی توانسته است کارایی خود را افزایش دهد و به عنوان بهترین الگوریتم مکاشفه‌ای زمان اجرای خوبی را بدست آورد. الگوریتم LATO و ALATO که از اتوماتاهای یادگیر استفاده می‌کنند توانسته اند با افزایش ناهمگونی در کارها، کاهش قابل قبولی را در زمان محاسبات بدست آورند. زمان مینیمم، همانطور که مشاهده می‌شود، در ناهمگونی‌های مختلف تغییرات کمی دارد؛ زیرا میانگین محدوده‌هایی که در هر ناهمگونی برای توزیع تصادفی و یکنواخت طول کارها مورد استفاده قرار می‌گیرند یکسان و برابر با 105 kMI (هزار میلیون دستورالعمل) می‌باشد. بنابراین با توجه به اینکه تعداد کارها 200 عدد می‌باشد، مجموع طول کارها در همه حالات برابر با 105×200 kMI می‌باشد و به این ترتیب تغییر خاصی

نباید در زمان مینیمم ایجاد شود. حد پایین و بالا در نمودار عمودی که معرف زمان اجرای الگوریتم ها می باشد به ترتیب 17000 و 17500 با گتم رشد 25 گام می باشند. نمودار افقی هر دسه از ناهمگونی ها آورده شده اند.



شکل 5-24: نتایج الگوریتم های بهینه سازی زمان در ناهمگونی های مختلف

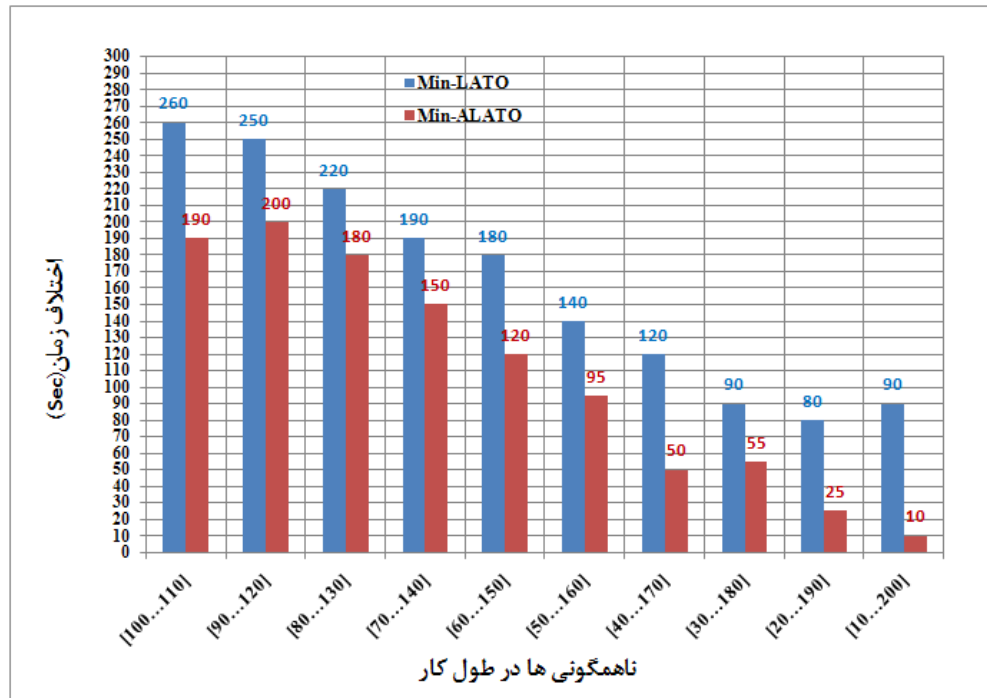
الگوریتم های LATO و ALATO که از اتوماتاهای یادگیر استفاده می کنند و ALATO به عنوان بهترین الگوریتم پیشنهادی، قصد دارد تا الگوریتم LATO را بهبود دهد. در اینجا تأثیر ناهمگونی کارها بر میزان بهبود حاصل از الگوریتم LATO و ALATO را بررسی می کنیم. شکل 5-25 نتایج این دو الگوریتم به همراه الگوریتم Minimum را در کنار هم و در ناهمگونی های مختلف نشان می دهد. مشاهده می شود که بهبود حاصل از زمان بندی الگوریتم های LATO و ALATO با افزایش ناهمگونی کارها بیشتر می شوند. روند کاهش زمان در الگوریتم ALATO از الگوریتم LATO بیشتر است. درواقع، سرعت به مینیمم رسیدن در ALATO از LATO بیشتر است.



شکل 5-25: مقایسه‌ی الگوریتم‌های LATO و ALATO و Minimum در ناهمگونی‌های مختلف

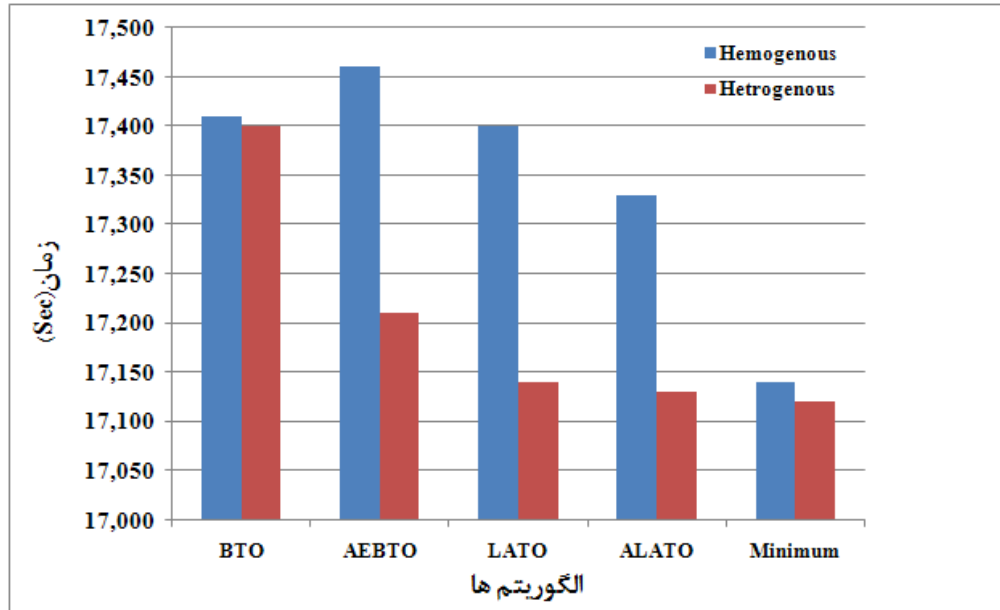
برای مقایسه‌ی دقیق‌تر و مشاهده‌ی میزان بهبود حاصل شده، به نمودار اختلاف زمان اجرا هرکدام از این دو الگوریتم با الگوریتم مینیمم نیاز داریم. این نمودار در شکل 5-26 نشان داده شده است. مشاهده می‌کنیم که هنگامی که کارها هم‌گون هستند الگوریتم ALATO تقریباً زمان زیادی مصرف می‌کند و فاصله‌ی نتیجه‌ی این الگوریتم با زمان مینیمم زیاد است. علت این امر این است که چون طول کارها تقریباً به هم نزدیک است اختلاف زمان اجرای کارها بر روی پرکارترین منبع با منابع دیگر زیاد است و نمی‌توان آن را کاهش داد. با افزایش ناهمگونی کارها، الگوریتم ALATO به سرعت خود را به مینیمم نزدیک می‌کند و در ناهمگون‌ترین حالت، اختلاف خود را به کمترین میزان ممکن می‌رساند. در این روند به تدریج با افزایش فاصله‌ی طولی بین کوچکترین و بزرگترین کارها، الگوریتم می‌تواند با تهیه‌ی یک زمانبندی مناسب و توزیع متعادل کارها بین منابع، زمان اجرای کارها بر روی منابع را به هم نزدیک کرده و زمان اجرا بر روی پرکارترین منبع (زمان اجرای برنامه‌ی کاربردی از نظر کاربر) را کاهش دهد ولی به تدریج با ناهمگون شدن کارها توانسته است افزایش کارایی بیشتری را داشته باشد به طوری که در ناهمگونی (10...200)، میزان نزدیک شدن ALATO به الگوریتم مینیمم که بهترین حالت اجرای کارهاست در حدود 10

واحد می باشد. در حالی که در این حالت LATO به میزان 90 واحد تا حالت بهینه اش فاصله دارد.



شکل 5-26: اختلاف زمان اجرای کارها در الگوریتم های LATO و ALATO با الگوریتم Minimum

در شکل 5-27 نتیجه ی بدست آمده از الگوریتم های بهینه سازی زمان در دو حالت همگون و ناهمگون آورده شده است. در حالت همگون، محدوده ی توزیع تصادفی و یکنواخت طول کارها محدوده ی (100...110) و در حالت ناهمگون این محدوده برابر با (10...200) می باشد. به جز الگوریتم BTO که در هر دو حالت تقریباً یکسان عمل می کند، در بقیه ی الگوریتم ها مقداری بهبود در حالت ناهمگون نسبت به حالت همگون بوجود آمده است. این بهبود در الگوریتم های LATO و ALATO نسبت به بقیه قابل ملاحظه تر می باشد. همچنین می توان مشاهده کرد که در حالت همگون، نتیجه ی بدست آمده از الگوریتم ها به هم نزدیک است؛ در حالی که در وضعیت ناهمگون، زمان اجرای کارها در الگوریتم ها با هم اختلاف زیادی پیدا می کند.



شکل 5-27: نتایج الگوریتم‌های بهینه‌سازی زمان برای کارهای همگون و ناهمگون

خلاصه و نتیجه گیری

در این فصل، عمل شبیه‌سازی الگوریتم پیشنهادی و الگوریتم‌های قبلی ارائه شده در فصل قبل را انجام دادیم. این کار با استفاده از جعبه‌ابزار GridSim و نرم افزار SPNP که امکانات لازم برای شبیه‌سازی گریدهای محاسباتی اقتصادی و سیستم‌های مدیریت منابع در آنها را فراهم می‌کند، انجام گرفت. همچنین از بسته‌ی نرم افزاری gridbroker که توسط بویا برای مدل‌سازی وظایف واسطه‌گر منبع عرضه شده است، به صورت تغییر یافته و با افزودن امکانات لازم استفاده کردیم.

با استفاده از شبیه‌سازی الگوریتم‌های بهینه‌سازی زمان مشخص شد که الگوریتم یادگیر ALATO در ناهمگونی زیاد کارها، نسبت به الگوریتم‌های مکاشفه‌ای نتایج بهتری را بدست می‌آورد و می‌تواند برنامه‌ی کاربردی متعلق به کاربر را با صرف بودجه‌ی تعیین شده، در مدت زمان کمتری نسبت به الگوریتم‌های دیگر، بر روی گرید اجرا کند. همچنین با توجه به اختلاف اندک زمان بدست آمده از این الگوریتم با مینیمم زمان محاسبات، امکان بهبود قابل ملاحظه در الگوریتم ALATO وجود ندارد.

فصل ششم

نتیجه
گیری

6 نتیجه گیری و پیشنهادها

هدف اصلی از این پروژه ارائه یک شبکه پتری تصادفی تطبیقی مبتنی بر اتوماتای یادگیری می باشد، شبکه پتری تصادفی تطبیقی از طریق اطلاعات بدست آمده از حالات قبلی سیستم و واکنش های محیط پویا، حالت بهینه بعدی را پیشگویی نموده و وضعیت جاری سیستم را بروز و احتمال وقوع رویدادها را در طول زمان تغییر می دهد و باعث می شود رویدادها بر اساس احتمال وقوعشان فعال شوند. بروز شدن وضعیت های سیستم بر اساس واکنش محیط پویا کمک شایانی در یادگیری و آموزش شبکه های پتری می کند. در قسمت دیگری از این پروژه کاربرد مدل تطبیقی پیشنهاد شده، در گرید محاسباتی و گرید اقتصادی مورد مطالعه و بررسی قرار می گیرد

کارهای انجام شده در زمینه شبکه پتری تطابقی به دو دسته ی Fusion Hybrid و Combination Hybrid تقسیم شده اند. شبکه پتری تصادفی تطابقی ارائه شده برای بررسی در گرید محاسباتی، جزء دسته Fusion Hybrid می باشد، همانطور که بیان شد. این دسته معایبی دارند. به علت ادغام شبکه های پتری با تکنیک های هوشمند، اغلب اوقات یادگیری در شبکه های پتری از این دسته باعث افزایش پیچیدگی عملیات و افزایش سربار محاسباتی می شود. در شبکه های پتری تطابقی ارائه شده تاکنون، از تکنیک های هوشمند نظیر شبکه های عصبی مصنوعی، منطق فازی و سیستم های مبتنی بر دانش بهره برده اند، ولی شبکه پتری تصادفی تطابقی که ما ارائه کرده ایم مبتنی بر اتوماتای یادگیر و همچنین ترکیب آن با منطق فازی است، شبکه پتری تصادفی تطابقی مبتنی بر اتوماتای یادگیر بر خلاف شبکه های پتری تطابقی که از شبکه های عصبی مصنوعی استفاده می کند، نیاز به آموزش دیدن با یکسری الگوهای از قبل مشخص شده نیست، در سیستم هایی که محیط کاملاً ناشناخته و در هر لحظه در حال تغییر است، نمی توان الگویی داشت تا بتوان شبکه عصبی مصنوعی را با آن آموزش داد ولی مدل تطبیقی پیشنهاد شده در این محیط ها کاملاً قابل انعطاف بوده چون اتوماتای یادگیر نیازی به آموزش دیدن با داده های از قبل ندارند، به همین دلیل هم مشکل دسته Fusion Hybrid که افزایش سربار محاسباتی است را ندارد، بعلاوه در شبکه پتری تصادفی تطابقی مبتنی بر اتوماتای یادگیر، هر توکن اتوماتای مخصوص بخودش را دارد و برخلاف مدل های تطابقی قبلی که نیاز به ایجاد مکان ها و گذرهای اضافه تری برای ایجاد شبکه عصبی مصنوعی

داشتند، نیازی به مکان و گذر جدیدتری نیست و این باعث عدم پیچیدگی در طراحی و افزایش سربار محاسباتی می‌گردد. شبکه پتری تصادفی فازی تطابقی ارائه شده از تکنیک هوشمند اتوماتای یادگیر بعلاوه سیستم‌های مبتنی بر دانش در شبکه پتری تصادفی استفاده شده است، که جزء هر دو دسته Fusion Hybrid و Combination Hybrid می‌باشد، که سعی شده هم از مزایای دسته دوم استفاده بشود و هم معایب دسته اول را رفع بکند.

همانطور که گفته شد، کاربرد مدل‌های تطبیقی پیشنهاد شده، در گرید محاسباتی و گرید اقتصادی مورد مطالعه و بررسی قرار گرفت. در گرید اقتصادی از مدل تطبیقی ارائه شده برای تخصیص بهینه منابع در گرید با توجه به معیار زمان استفاده شد، الگوریتم پیشنهادی (ALATO) با الگوریتم‌های مکاشفه‌ای (BTO) (AEBTO) و الگوریتم یادگیری تقویتی (LATO) مقایسه گردید، که در مقایسه با الگوریتم LATO تنها مراحل جریمه و پاداش تغییر کرد. با تغییر مهمی که در مرحله جریمه دادن به کارها، ایجاد شد، کارایی الگوریتم پیشنهادی نسبت به الگوریتم LATO بهبود قابل ملاحظه‌ای پیدا کرد. شبیه‌سازی‌های انجام شده نشان داد که نتایج الگوریتم یادگیر در استراتژی بهینه‌سازی زمان اختلاف کمی با مینیمم زمان محاسبات دارد. این اختلاف اندک برای الگوریتم یادگیر ارائه شده برای بهینه‌سازی زمان نیز نسبت به مینیمم زمان محاسبات وجود دارد.

از کارهایی که می‌توان در ادامه‌ی کار این پایان‌نامه در آینده انجام داد، ارائه‌ی استراتژی‌های جدید برای زمانبندی اقتصادی می‌باشد. باید توجه کرد که یک استراتژی جدید باید برای کاربر گرید قابل فهم بوده و بتواند در صرفه‌جویی زمان و هزینه، نقش داشته باشد.

یکی از پیشنهادهایی که می‌توان در مورد الگوریتم‌های زمانبندی اقتصادی مطرح کرد، بررسی و ارائه‌ی روش‌های زمانبندی جدید در مدل‌های اقتصادی غیر از مدل بازار کالا (که در این پایان‌نامه، مد نظر بود) می‌باشد. از رایج‌ترین آنها می‌توان به مدل‌های اقتصادی حراجی و چانه‌زنی اشاره کرد.

به عنوان آخرین پیشنهاد می‌توان به مبحث کاملاً جدید بیمه اشاره کرد. در این پیشنهاد بیمه در برابر تأخیر زمانی مورد بررسی قرار خواهد گرفت که در آن کاربر در برابر تأخیر در تحویل نتیجه‌ی برنامه‌ی خود از سیستم خسارت دریافت کند. تأخیر

در انجام درخواست مشتریان می‌تواند به دلایل مختلفی مانند خرابی شبکه، قطع برق، از کار افتادن بخشی از یک منبع و حوادث احتمالی دیگر بوجود بیاید که سیستم از قبل از رخ دادن آنها اطلاع ندارد. بنابراین گریه می‌تواند شبکه و اجزای مختلف منابع و تجهیزات دیگر خود را توسط شرکت‌های بیمه در مقابل این حوادث بیمه کرده و در مقابل، خود، مشتریان را با دریافت مبلغی اضافه در برابر تأخیرهای احتمالی بیمه کند. برای این کار باید تحلیل جامعی بر روی اتفاقاتی که ممکن است بر روند اجرای برنامه‌ها در گریه تأثیر بگذارد انجام شود و با توجه به اطلاعات گذشته و هزینه‌های بیمه‌ای و خسارت‌های احتمالی وارد شده، میزان افزایش هزینه‌ی اجرای برنامه‌ها محاسبه گردد و از کاربرانی که خواستار بیمه در برابر همه یا بخشی از این حوادث هستند، دریافت گردد.

همچنین، می‌توان از بکاربردن سایر الگوریتم‌های هوشمند همچون شبکه عصبی، الگوریتم‌های ژنتیک در شبکه پتری تصادفی تطابقی در سایر کاربردها نام برد.

فصل هفتم

پیوست ها

7 پیوست‌ها

واژه‌نامه‌ها

Action	عمل
Action Probability	احتمالات عمل‌ها
Adaptability	سازگارپذیری
Adaptive Control	کنترل قابل تطبیق
Adaptive Stochastic Petri Net(ASPN)	شبکه پتری تصادفی تطبیقی
Administrative Domain	حوزه‌ی مدیریتی
Admission Control	کنترل پذیرش
Agent	عامل
Artificial Neural Network (ANN)	شبکه‌های عصبی مصنوعی
Asynchronous	ناهمزمان
Autonomous	خودمختار
Average Response Cost per Request (ART)	هزینه متوسط پاسخگویی بازای هر درخواست
Average Response Time per Request (APT)	زمان متوسط پاسخگویی بازای هر درخواست
Axons	آکسون‌ها
Back propagation	انتشار به عقب
Batch Application	برنامه‌ی دسته‌ای
Benchmarks	شاخص اندازه‌گیری-شاخص سنجیدن
Binary Pattern Classification	دسته‌بندی الگوهای دودویی
Boundness	حد و مرز پذیری
Broker	واسطه‌گر-دلال
Buyya	بویا
Casual Connection	ارتباط اتفاقی
Certainty Factor	فاکتور قطعیت
Chandrasekhar	چاندراسکار
Class Membership Value	مقدار عضویت کلاس
Cluster Computing (CC)	محاسبات کلاستری
Collaborative	همکاری کننده
Colored Petri Net(CPN)	شبکه پتری رنگی
Completion Time	زمان اتمام-زمان کامل شدن
Complex	پیچیده
Computational Grids	گریدهای محاسباتی
Computational Network	شبکه‌ی محاسباتی
Concurrency	همروندی
Continues-Time Markov Chain(CTMC)	زنجیره مارکو پیوسته زمانی
Cycle-Stealing	حلقه ربایی
Data Grids	گریدهای داده‌ای
Data Mining	داده‌کاوی
Data Warehouse	انبارهی داده
Deadlock	بن بست
Debugging	اشکال زدایی
Decay Factor	فاکتور تضعیف کننده
Dendrite	دندریت (شاخه‌های متعدد سلول عصبی)
Deterministic	قطعی
Deterministic Automata	اتوماتای قطعی

Deterministic Stochastic Petri Net(DSPN)	شبکه‌های پتری تصادفی معین
Directed Acyclic Graph(DAG)	گراف جهت‌دار غیر حلقوی
Distributed Computing Environments (DCE)	محیط‌های محاسباتی توزیع شده
Distributed SPN(DSPN)	شبکه‌های پتری تصادفی توزیع شده
Domain Knowledge	حوزه دانش
DVD	دی وی دی
Electrocardiograph	الکتروکاردیوگراف
Electromyogram	الکترومایگرام
Element	المنت-عنصر
Entity	نهاد
European Data Grid (EDG)	گرید داده‌ای اروپایی
Excitatory	توابع تحریک
Explicit Representation	نمایش آشکار
Experiment	آزمایش
Extended Petri Net(EPN)	شبکه پتری توسعه یافته‌ای
Extended SPN(ESPN)	شبکه‌های پتری تصادفی گسترش یافته
Extensibility	توسعه‌پذیری
External Events	رخدادهای خارجی
Failures	شکست‌ها
Favorable	مطلوب
Finite State Machine (FSM)	ماشین حالت متناهی
Fire	فعال شدن
Fixed	ثابت
Fixed Structure	اتوماتای با ساختار ثابت
Flat	مسطح
Flexible Manufacturing System(FMS)	سیستم‌های تولید قابل انعطاف صنعتی
Fu	فو
Fuzzy inference	استنتاج فازی
Fuzzy inference engine	موتور استنتاج فازی
Fuzzy Logic (FL)	منطق فازی
Fuzzy Neural Petri Net(FNPN)	شبکه پتری عصبی فازی
Fuzzy production Rules	تولید قوانین فازی
Fuzzy Reasoning Algorithm	الگوریتم‌های استنتاج فازی
Fuzzy Reasoning Petri Net(FRPN)	شبکه پتری استدلال فازی
Game Theory	تئوری بازی
Gaussian Mixed Network	شبکه مرکب گوسین
General Stochastic Petri Net(GSPN)	شبکه پتری تصادفی عمومی
Generic Sigmoid Transfer Function	توابع انتقال کلی حلقوی یا مارپیچی
Global Computing (GC)	محاسبات سراسری
Globus	گلوباس
Graph partitioning	جداسازی گراف
Grid Computing	گرید محاسباتی
Grid Layer	لایه گرید
Gridlet	گریدلت
Hass	هاس
HEBB	هب
Heterogeneous	ناهمگون
Heuristic	مکاشفه‌ای

Hierarchical	سلسله‌مراتبی
Hierarchical Petri Net(HPN)	شبکه پتری سطح بالا
Hierarchical Stochastic Petri Net(HSPN)	شبکه پتری تصادفی که بصورت سلسله‌مراتبی
Hierarchical Timed Petri Net(HTPN)	شبکه پتری زماندار سلسله‌مراتبی
High level Reasoning	استنتاج سطح بالا
Home Layer	لایه خانگی
Hopfield	هاپ‌فیلد
Human Cognition	شناخت انسان
Infinitesimal Generator	ژنراتور بسیار ریز
Inhibitory	توابع بازداشت
Integrated	بطور مجتمع
Integrity Constraints	محدودیت‌های جامعیت
Interface	رابط - واسط
Internal Events	رخدادهای داخلی
Iris Data set	عه داده های آیریس
Job Centric	کار محور
Knowledge Base System(KBS)	سیستم های مبتنی بر دانش
Knowledge Representation	بازنمایی دانش
Learning Automata (LA)	اتوماتای یادگیر
Learning Automata: Theory and Application	اتوماتای یادگیر:تئوری و کاربردها
Learning Petri Network(L.P.N)	شبکه پتری یادگیر
License	جوازها -مجوزها
Linear Reward Inaction (LRI) Scheme	روش عدم پاداش خطی
Linear Reward Penalty (LRP) Scheme	روش جریمه‌ی پاداش خطی
Linear Reward Penalty Epsilon Penalty (LReP) Scheme	روش جریمه‌ی جزئی پاداش خطی
Liveness	تحریک پذیری
Local Layer	لایه محلی
Log Linear Model	مدل ثبت وقایع بصورت خطی
Marking	نشانه گذاری-علامت گذاری
McLaren	مک لارن
Measurement	اندازه گیری
Meta Computing	فرامحاسبات
Metal Surface Roughness	ناهموار سطوح فلزات
Methodology	متودولوژی
Milling	فرز-ذوب
Molloy	مولی
Multimedia	چند رسانه‌ای
Mutual Exclusion	اجرای دوجانبه
Najim	نجیم
Namespace	فضای نام
Narendra	نارندرا
Natkin	ناتکین
Network Application	برنامه‌ی کاربردی شبکه
Neural Fuzzy Expert Petri Net(NFEPN)	شبکه پتری خبره عصبی فازی
Neural Petri Net(NPN)	شبکه پتری عصبی
Non-predictive	غیرپیش‌گویانه
non-stationary	محیط‌های ناپایدار
Null	خالی - صفر
Object Migration Automata (OMA)	اتوماتای مهاجرت اشیاء

Object Model	مدل شیء
Object Oriented Systems	سیستم‌های شی گرا
Object Partitioning	تقسیم‌بندی اشیاء
Off-line	غیربرخط
On-line	بر خط
Overhead	سربار-سرایند
Parameter Estimation	تخمین پارامترها
Parameter Sweep	پارامترروب
Path planning	برنامه‌ریزی مسیر
Pattern Classification	طبقه‌بندی الگوها
Pattern Recognition	تشخیص الگو
Penalty	جریمه
Performance Evaluation	ارزیابی کارایی
Pre-Synaptic Clefts	شکاف‌های پس‌سیناسپی
probability density function (pdf)	تابع شدت احتمال
Probability Distribution Function(PDF)	تابع توزیع احتمال
Propositions	گزاره‌ها
Prototype	پیش‌نوع‌ها
Pznyak	پوژنیاک
Queuing Theory	تئوری صف
Reachability	قابلیت دسترسی
Real Time	آنی-بلا درنگ
Realistic	واقع‌گرایانه
Real-Time Systems	سیستم‌های زمان واقعی
Remote	راه دور
Rescheduling	زمان‌بندی مجدد
Resource	منبع
Resource Consumer	مصرف‌کننده‌ی منبع
Resource Discovery and Dissemination (RDD)	یافتن و انتشار منبع
Resource Management System (RMS)	سیستم مدیریت منبع
Resource Manager Peer Interface (RMPI)	رابط هم‌پای مدیر منبع
Resource Manager Support Interface (RMSI)	رابط پشتیبانی مدیر منبع
Resource Provider (RP)	فراهم‌کننده‌ی منبع
Reusability	استفاده‌ی مجدد
Reward	پاداش
Safeness	بی‌خطری -امن بودن
Scalability	مقیاس‌پذیری
Scheduling policy	سیاست زمان‌بندی
Schema	شما-الگوی خاص
Sensitive	محسوس
Service Grid	گرید خدماتی
Shen	شین
Shielder	شیلدر
Single Server	سرور واحد
Site Autonomy	خودمختاری سایت
State Estimation	برآورد وضعیت
States	حالات-وضعیت
Stationary	محیط‌های پایدار
Steady-State	پایدار
Stochastic Learning Automata (SLA)	اتوماتاهای یادگیر تصادفی
Stochastic Petri Net (SPN)	شبکه‌های پتری تصادفی
Subtasks	زیر کارها

Successes	موفقیت‌ها
Supercomputing	ابر محاسبات
Synchronization	همزمانی
System Centric	سیستم محور
Tangible Marking	علامتگذاری‌های محسوس
Task	کار - درخواست
Temporal	موقتی
Thathachar	تاتچار
Thick Request	درخواست پر محتوا
Thin Request	درخواست کم محتوا
Throughput	توان عملیاتی
Time Transition	حالت زمانی - گذار زمانی
Timer	زمان سنج
Tokens	علائم - نشانه‌ها
Toolkits	جعبه ابزارها
Transition matrix	ماتریس انتقال
Transitions	گذارها - حالت‌ها
Transparent	شفاف
Triggers	راه اندازی کردن
Tsetlin	تستلین
Tuple	تاپل - ستون جدول
Unfavorable	نامطلوب
Unfired	غیر فعال
Variable Structure	اتوماتای با ساختار متغیر
Variable structure learning Automata	اتوماتای با ساختار متغیر
Varshavski	وارشائوسکی
Virtual Enterprises (VE)	بنگاه‌های مجازی
Virtual Organizations (VOs)	سازمان‌های مجازی
Virtualization	مجازی‌سازی
Viswanatan	ویواناتان
Vorontsova	وورونتسودا
Weighted Fuzzy Production Rule	قانون تنظیم شده فازی
Word Processing	پردازش کلمه - پردازش لغت
Workload	بار کاری
Workstations	ایستگاه‌های کاری
Y.Z. Tsyarkin	سیپکین

مروری بر نرم افزار SPNP

نرم افزار Stochastic Petri Net Package (SPNP) یک ابزار مدلسازی برای مدل‌های شبکه پتری می‌باشند. مدل‌های شبکه پتری تصادفی با زبان CSPL یا زبان C در SPN توصیف می‌شوند. این زبان یک زبان گسترش یافته بر اساس زبان برنامه نویسی C با اضافه کردن ساختارها و توانمندی‌های قابل استفاده در مدل SPN می‌باشد. در ادامه به توصیف کلی برنامه نویسی زبان CSPL و نحوه اجرا و راه اندازی این نرم افزار می‌پردازیم.

7-2-1- راه اندازی در ویندوز XP

یک کاربر باید یک دیسک فشرده ای که شامل کتاب راهنما شارپ رابط کاربری گرافیکی و یا فایل شارپ Gui.zip می باشد را دریافت کند. برای شروع رابط ، شما باید رابط کاربری گرافیکی را از روی دیسک فشرده در پوشه

C:\Spnp-Gui کپی کنید. سپس دستور زیر را در مسیر مورد نظر که اغلب درایو C می باشد اجرا کنید. توجه داشته باشد که filename همان فایل CSPL شما می باشد.

```
make -f /PATH_TO_SPNP/spnp/obj/Makerun SRN=filename
```

سپس بعد از نصب و کپی کردن فایل ها در محل در نظر گرفته شده در بالا به آدرس Control panel>System option> Environment Variables در User variable یک متغیر با نام SPNP_DIRECTORY و با مقدار C:\Spnp-Gui\spnp ایجاد کنید و همچنین در قسمت System Variable در قسمت Path در انتها عبارت C:\Spnp-Gui\spnp را به مقدار متغیر Path اضافه کنید. حال در ادامه با زدن دکمه OK به آدرس C:\Spnp-Gui\spnp بروید و فایل Interface SPNP.exe را اجرا کنید.

7-2-2- فایل‌های خروجی در SPNP

فایل‌های میانی توسط پکیج (بسته) تولید میشوند و نتایج نهایی در همان مسیری که filename.c هست ، خواهد بود (جایی که شما دستور را از آنجا صادر کردید). فایل‌ها بر اساس نوع اطلاعاتی که آنها منتقل میکنند مصادیق مختلفی دارند. اگر فایل CSPL شما test.c نامگذاری شده است ، بنابراین فایل‌های زیر تولید خواهند شد :

- test.o : هنگام ترکیب test.c بدست می‌آید.
- test.spn : توسط فایل‌های موضوعی پکیج (بسته) یکدیگر با test.o فایل‌های اجرایی بدست می‌آید.
- test.rg : شامل اطلاعات نمودار قابل لمس میشود: ترکیب هر نشانه گذاری ، توصیف انتقال میان آنهاست و غیره.
- test.mc : شامل (عددی) CTMC/DTMC متناظر با SRN.
- test-paramname.mc : شامل (عددی) مشتق از CTMC با در نظر گرفتن پارامتر paramname (یک فایل که تولید شده برای هر پارامتر تولید شده در test.c)
- test.prb : شامل (عددی) نتایج تجزیه CTMC پایین: احتمال شرایط ثابت یا گذرا برای هر نشانه ملموس ، تجمیع زمان موقت در

وضعیت انتقال به زمان حل و مشتقات (با در نظر گرفتن پاترامترها) از مقیاس مذکور.

- test.prbdtmc: شامل (عددی) نتایج محاط کردن DTMC.
- test.out: شامل درخواست خروج (براساس چیزی که در test.c مشخص شده توابع مهیا شده استفاده میشوند).
- test.log: شامل همه پیغامهای خروجی تولید شده توسط پکیج (بسته) مدل حل.
- test.dot: شامل توصیف شبکه در زبان نموداری dot.

3-2-7- زبان CSPL

زبان CSPL میتواند بعنوان يك سیستم برای توصیف تعریف شده باشد، پردازش و تحلیل تصادفی شبکه ها (SRNs). نحو و معانی CSPL بر اساس زبان ANSI C است. بعلاوه يك فایل CSPL صحیح همچنین يك فایل ANSI C صحیح است. چیزی که CSPL را از ANSI C تمیز میدهد مجموعه ای از توابع از پیش تعریف شده برای تعیین و تحلیل SRNs هاست. آشنایی با زبان ANSI C مزیت بسیار بزرگی برای يك کاربر برای بیشترین کار برجسته از ترکیب زبان CSPL خواهد بود. بنابراین آن باید برای کاربر مشکل نباشد توانایی تعریف و حل SRNs خودش بموجب مطالعه قسمت کمک کتاب مبانی، مخصوصاً " با مثالهای درون آن. هر ساختار قانونی زبان ANSI C براساس اینکه نیاز شده در زبان CSPL منظور شده است. توابع و متغیر های تعریف شده کاربر می‌تواند در فایل CSPL استفاده شده باشد. خصوصاً " همه توابع استاندارد C، از قبیل `fbrintf()`, `fscanf()`, `log()`, `exp()` و غیره، در صورت نیاز میتواند در CSPL نامگذاری شود. يك فایل CSPL باید شامل توابع پایه زیر باشد:

- Options()
- Net()
- Assert()
- Ac_init()
- Ac_reach()
- Ac_final()

هر تابع فهرست شده برای اجرای يك (یا چند) وظیفه (وظائف) توسط بعضی از زیرروالها ی (توابع) مربوطه طراحی شده است. این وظایف مربوط به تعاریف و تحلیل پردازش بجای شبکه ها هستند.

تابع Option()

Void options (void);

يك فایل معتبر CSPL باید شامل تابع options() باشد. تابع options با زیرروالهای زیر:

- Iopt()
- Fopt()
- Input()
- Finput()

برای تنظیم اختیاراتی که از طریق توصیف و تحلیل SRN تأثیر خواهد داشت نامیده می‌شوند. لیست‌های زیر در Option می‌آید. جدول‌های 1-7 تا 4-7 در زیر آمده است.

جدول 1-7: گزینه‌های موجود برای فایل‌های میانی

نام	ارزش‌ها	پیش‌فرض
IOP PR RSET	VAL YES, VAL NO, VAL TAN	VAL NO
IOP PR RGRAPH	VAL YES, VAL NO	VAL NO
IOP PR MARK ORDER	VAL CANONIC VAL LEXICAL VAL MATRIX	VAL CANONIC
IOP PR MERG MARK	VAL YES, VAL NO	VAL YES
IOP PR FULL MARK	VAL YES, VAL NO	VAL NO
IOP USENAME	VAL YES, VAL NO	VAL NO
IOP PR MC	VAL YES, VAL NO	VAL NO
IOP PR DERMIC	VAL YES, VAL NO	VAL NO
IOP PR MC ORDER	VAL FROMTO, VAL TOFROM	VAL FROMTO
IOP PR PROB	VAL YES, VAL NO	VAL NO
IOP PR PROBDTMC	VAL YES, VAL NO	VAL NO
IOP PR DOT	VAL YES, VAL NO	VAL NO

جدول 2-7: راه حل تحلیلی و عددی option‌های موجود

نام	ارزش‌ها	پیش‌فرض
IOP MC	VAL CTMC, VAL DTMC	VAL CTMC
IOP SSMETHOD	VAL SSSOR, VAL GASEI, VAL POWER	VAL SSSOR
IOP SSDETECT	VAL YES, VAL NO	VAL YES
FOP SSPRES	non-negative double	0.25
IOP TSMETHOD	VAL TSUNIF, VAL FOXUNIF	VAL FOXUNIF
IOP CUMULATIVE	VAL YES, VAL NO	VAL YES
IOP SENSITIVITY	VAL YES, VAL NO	VAL NO
IOP ITERATIONS	non-negative int	2000
FOP PRECISION	non-negative double	0.000001

جدول 3-7: Option‌های شبیه‌ساز موجود

نام	ارزش‌ها
IOP SIMULATION	VAL YES, VAL NO
IOP SIM RUNS	non-negative int
IOP SIM RUNMETHOD	VAL REPL, VAL BATCH, VAL RESTART VAL SPLIT, VAL IS, VA
IOP SIM SEED	non-negative int
IOP SIM CUMULATIVE	VAL YES, VAL NO
IOP SIM STD REPORT	VAL YES, VAL NO
IOP SPLIT LEVEL DOWN	non-negative double

IOP SPLIT PRESIM	VAL YES, VAL NO
IOP SPLIT NUMBER	non-negative double
IOP SPLIT RESTART FINISH	VAL YES, VAL NO
IOP SPLIT PRESIM RUNS	non-negative double
FOP SIM LENGTH	non-negative double
FOP SIM CONFIDENCE	non-negative double
FOP SIM ERROR	non-negative double

جدول 4-7: Option های متفرقه

نام	ارزش‌ها	پیش‌فرض
IOP ELIMINATION	VAL REDONTHEFLY, VAL REDAFTERRG, VAL REDNE	VAL REDONTHEFL
IOP OK ABSMARK	VAL YES, VAL NO	VAL NO
IOP OK VANLOOP	VAL YES, VAL NO	VAL NO
IOP OK TRANS M0	VAL YES, VAL NO	VAL YES
IOP OK VAN M0	VAL YES, VAL NO	VAL YES
FOP ABS RET M0	non-negative double	0.0
IOP DEBUG	VAL YES, VAL NO	VAL NO
FOP FLUID EPSILON	non-negative double	0.000001
FOP TIME EPSILON	non-negative double	0.000001

تابع Net()

Void net (void);

یک فایل CSPL معتبر باید شامل تابع net() باشد. تابع net() مجموعه‌ای از توابع تعریف شده در SRN نامیده می‌شود.

توابع پایین کافی هستند برای تعریف یک SRN ساده بدون ترکیب توسعه یافته از نشانه‌های وابسته فعال، نرخ/وزن و اعداد شمارشی و غیره.

تابع Place() و init()

Void place (char*p);

یک مکان با نام p تعریف می‌شود.

یک اسم قانونی است اگر: (1) طول رشته آن بین 1 و MAX_NAME_LENGTH بر اساس تعریف شده در فایل const.h باشد (مثلاً "20)، (2) آن فقط از کاراکترهای { A ... Z ، a ... z ، 9 ... 0 } حروفچینی شده است، (3) اولین کاراکتر در مجموعه { a..z ، A..Z } است. همه اسم‌ها باید متمایز باشد، که هست. داشتن دو مکان اشتباه است، دو ارتباط، یا یک مکان و ارتباط با نام یکسان.

تابع Void init (char*p, int n)

شماره آغازین اعتبار در مکان p به n بودن را نشان میدهد. بر اساس پیش فرض مکانها از جاتی دیگر در ابتدا خالی هستند.

تابع Imm()

تحولات و تغییرات بصورت اتوماتیک تعریف میشود وقتی که مقدار دهی آنها (یا توزیع پذیری آنها) تعریف شد (پایین را ببینید) . بنابراین تغییرات بی واسطه باید تعریف شوند توسط :

Void imm (char*t);

توابع Rateval() and probval()

Void rateval (char *t, doubl Val) ;

Void probval (char*t, doubl Val);

تعریف میزان انتقال زمان سوخت شده t و وزن تغییرات بی واسطه سوخت شده (احتمالاً" غیر معمول) t ، بر اساس یک ارزش ثابت val . تابع $rateval$ (یا بیشتر در نسخه های عمومی آن در بخش زیرین) ضمناً " تعریف میشود انتقالات زمانی باید تعریف شده باشد برای هر تغییرات . تابع $probval$ احتیاج به نامگذاری دارد فقط اگر وزن سوخت شده متفاوت است با ارزش پیش فرض 0.1 که با تعریف تغییرات بی واسطه توسط imm می آید.

تابع Assert()

int assert(void);

یک CSPL معتبر باید شامل تابع $assert()$ باشد. تابع $assert$ یک تابع مستقل از علامت گذاری بولی است که طی ساخت نمودار دستیابی برای بررسی اعتبار هر علامت یافت شده جدید، از سوی SPNP فراخوانی می شود. اگر علامت گذاری غیرقانونی باشد، به خطای راه اندازی مجدد و اگر (فرضاً) قانونی باشد، فاقد خطای راه اندازی مجدد مراجعه می کند.

بررسی قانونی بودن علامت گذاری با استفاده از توابع مشابه به کار رفته برای کسب استقلال علامت گذاری صورت می گیرد، یعنی علامت و نشانه فعال یا غیرفعال. معمولاً تا زمانی که امکان تعیین همه شرایطی که در علامت گذاری باید حفظ شود (یا نشود) وجود ندارد، بررسی ذاتاً ناقص است. اما هرچه دقت در تنظیم شرایط بیشتر باشد، اطمینان بیشتری از تطابق نمودار دستیابی یا سیستم حقیقی خواهید داشت.

اما این نوع از بررسی محدود است. این به شناسایی علامت گذاری یا نتایج آغاز به کار غیر قانونی کمک می کند اما نمی تواند غیاب علامت گذاری ها یا نتایج آغاز به کار قانونی را شناسایی کند (که به چیدمان دستیابی یا نمودار مربوط است و کلاً نمی تواند در زمانی که نمودار دست یابی ایجاد می شود، مورد بررسی قرار گیرد). در عمل، توانایی ایجاد هرچه زودتر بررسی موارد غیرقانونی- نوعاً عیب یابی شبکه ای فرض می شود محدود است - مهم است ما در می یابیم که اینگونه نیست. زمانیکه برنامه ارائه پیام برای حافظه ناکافی را متوقف می کند، آزمودن نمودار دستیابی کل (نامحدود) جای سئوالی باقی نمی گذارد.

توابع `ac_int()` و `ac_reach()`

**`void ac_init(void);`
`void ac_reach(void);`**

یک فایل CSPL باید شامل `ac_init()` و `SPNP.ac_reach` پیش از شروع ساختار نمودار دسترسی، `ac_init()` را فرا می خواند. تابع `void pr_net_info(void)` همیشه در تابع `ac_init()` برای ایجاد اطلاعات در خصوص مدل تحت بررسی فایل "out" فراخوانی می شود. این مخصوصاً زمانی مفید است که تعداد مکان ها یا گذارها در زمان اجرای برنامه تعریف شده است. فراخوانی اتصال در اینجا نیز می تواند برای اختصاص مقادیر عددی به پارامترها به همان روش که در بخش قبل تشریح شد اسفاده شود؛ «تعریف و استفاده از پارامترها» به کار می رود.

تابع `ac_reach()` برای جایگزین فراخوانی، بعد از ساخت نمودار دستیابی کامل می شود. تابع `void pr_rg_info(void)` می تواند در تابع `ac_reach()` برای تولید اطلاعات در خصوص نمودار دستیابی فایل "out" استفاده شود (توجه: این بر تولید فایل «rg» تأثیر می گذارد). فراخوانی اتصال در اینجا نیز می تواند برای اختصاص مقادیر عددی به پارامترها به همان روش که پیش تر تشریح شد استفاده شود. علاوه بر این، تابع `void pr_parms(void)` می تواند در اینجا برای ارائه نام گذارهای پارامتری در امتداد نام و مقدار کنونی فایل "out" فراخوانی شود.

تابع `ac_final()`

`void ac_final(void) ;`

فایل CSPL باید شامل تابع ac final() باشد. تابع ac final() برای جهت منعطف ساختن خروجی های تعریف شده توسط کاربر طراحی شده است. CSPL دسته ای از تابع را برای این منظور ارائه می کند.

7-2-4- طراحی در نرم افزار SPNP

با اجرای برنامه SPNP شکل 1-7 ظاهر می شود.



شکل 1-7: قاب اصلی

کاربر باید با کلیک بر روی "فایل" و سپس منوی "جدید" برای ایجاد پروژه جدید و یا استفاده از میانبر برای این سایت متعلق به عمل است. میانبر: Ctrl+N (در هر زمانی در طول جلسه). هنگامی که اطلاعات " برای پروژه جدید" و هنگامی که جعبه گفتگو ظاهر شد، فیلدهای مختلف را پر و بروی Ok کلیک کنید. فیلدهای متفاوت که باید پر شوند نام پروژه (name.rgl) خواهد شد پروژه های حاوی فایل اطلاعات برای همه مدل ها ایجاد شده). و نوع مدل مثل مدل 1 که در این پروژه اجرا شود. قاب اکنون بر روی صفحه نمایش ظاهر می شود پس از اعتبار سنجی از کاربران است. قاب شامل 2 بخش است:

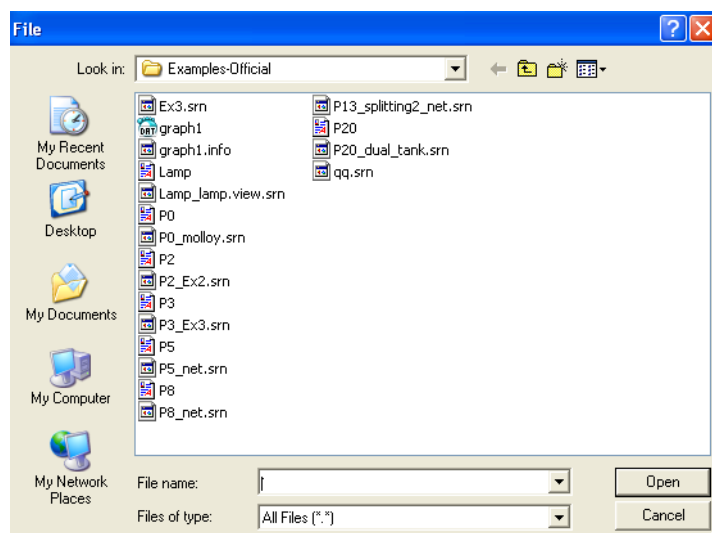
- a. قسمت سمت چپ که شامل لیست اسامی مدل های موجود در این پروژه است.
- b. سمت راست پانل مورد استفاده برای طراحی مدل فعلی (انتخاب شده از لیست سمت چپ).

برای ذخیره پروژه کاربر می تواند پروژه فعلی را با تمام مدل های اجرا شده در داخل آن ذخیره کند. همچنین می تواند یک مدل تک را ذخیره کند (نگاه کنید به "ذخیره مدل"). پروژه ها می توانند با فرمت rgl و مدلها می توانند با فرمت SPNP ذخیره شوند (شکل 7-2). برای اصلاح از پروژه فعلی خواص با کلیک بر روی "مدل کردن ویرایش گر" سپس بر روی "اطلاعات". پروژه ها " و مدل اطلاعات "جعبه گفتگو باید ظاهر شود. در این کادر محاوره ای، کاربر می تواند تغییر دهد: نام پروژه فعلی-نام مالک. اصلاح از پروژه فعلی خواص شرح پروژه برای بروز رسانی ارزش باید "معتبر" را فشار دهید. برای تغییر نام پروژه با کلیک بر روی "مدل کردن ویرایش گر" منو و سپس بر روی "اطلاعات" منو کلیک کنید و بعد

بروي نوار پروژه کلیک کنید. شما همچنین می توانید توضیحات مالک و ویرایش را تفسیر کنید. برای تغییر نام یک مدل از طرح در حال حاضر با کلیک بروي "مدل کردن ویرایش گر" و سپس بر روي "اطلاعات" کلیک کنید وبعد بروي نوار مدلها کلیک کنید.

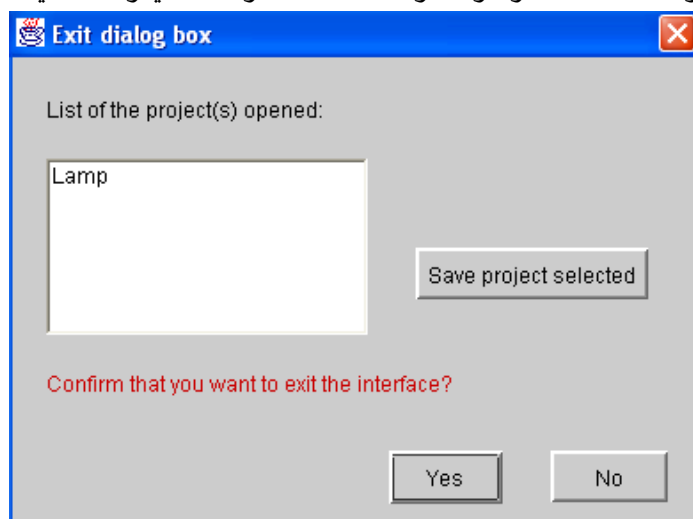
شکل 2-7: پروژه‌های جدید

شما می توانید رابطه بین تمام مدل های ایجاد شده در پروژه کنونی را تجسم کنید. هنگامی که کادر محاوره ای زیر ظاهر می شود فایل Rgl را انتخاب نموده و بروي Ok کلیک کنید. پسوند پروژه در فایل های رابط همیشه Rgl است مانند شکل 3-7.



شکل 3-7

برای بستن پروژه هنگامی که کاربر می‌خواهد پروژه را ببندد کادرمحاوره ای ظاهر می‌شود که از شما سوال می‌کند که قبل از بسته شدن می‌خواهید فایل یا فایل‌های باز شده‌ای که در لیست پروژه‌های باز مانده قرار گرفته‌اند را ذخیره کنید (شکل 4-7).



شکل 4-7

کدها

7-3-1- کد الگوریتم اول

7-3-1-1- قسمت اصلی کد الگوریتم Min.min

```

package minmin;
import java.util.ArrayList;
import java.util.List;

public class Main {
    public static List<Task> taskList = new ArrayList<Task>();

    public static void findMinInMin() { //this method return a list of Jobs

        List<Job> numOfNodes = new ArrayList<Job>();
        List<Integer> sortedNodes = new ArrayList<Integer>();

        for (int i = 0; i < taskList.size(); i++) {

            int minTime = 100;
            int numOfTask = -1;
            Node node = new Node();

            for (int j = 0; j < taskList.get(i).nodeList.size(); j++) {
                if (taskList.get(i).nodeList.get(j).time < minTime) {
                    numOfTask = j;
                    minTime = taskList.get(i).nodeList.get(j).time;
                    node = taskList.get(i).nodeList.get(j);
                }
            }

            System.out.println(node.time + " in node " + numOfTask);
            numOfNodes.add(new Job(node, taskList.get(numOfTask), node.time));
            sortedNodes.add(node.time);
            for (int i1 = 0; i1 < taskList.size(); i1++) {
                taskList.get(i1).nodeList.get(numOfTask).time += minTime;
            }
        }

        int [] returnArr = new int [sortedNodes.size()];
        for (int i = 0; i < sortedNodes.size(); i++) {
            returnArr[i] = sortedNodes.get(i);
        }

        for (int i = 0; i < returnArr.length; i++) {
            int temp;
            for (int j = i; j < returnArr.length; j++) {
                if (returnArr[j] < returnArr[i]) {
                    temp = returnArr[i];
                    returnArr[i] = returnArr[j];
                    returnArr[j] = temp;
                }
            }
        }
    }
}

```

```

    }

    System.out.println("Sorted time is : ");
    for (int i = 0; i < returnArr.length; i++) {
        System.out.println(returnArr[i]);
    }
}

public static void main(String[] args) { // main method

    Task t1 = new Task();
    Node n1 = new Node(8,0);
    t1.nodeList.add(n1);
    Node n2 = new Node(1,0);
    t1.nodeList.add(n2);
    Node n3 = new Node(5,0);
    t1.nodeList.add(n3);

    taskList.add(t1);

    t1 = new Task();
    n1 = new Node(6,0);
    t1.nodeList.add(n1);
    n2 = new Node(9,0);
    t1.nodeList.add(n2);
    n3 = new Node(8,0);
    t1.nodeList.add(n3);

    taskList.add(t1);

    t1 = new Task();
    n1 = new Node(2,0);
    t1.nodeList.add(n1);
    n2 = new Node(3,0);
    t1.nodeList.add(n2);
    n3 = new Node(4,0);
    t1.nodeList.add(n3);

    taskList.add(t1);

    findMinMin();

}

}

}

```

2-1-3-7- قسمت اصلی کد Max.Min و الگوریتم HSPN

```

package minmax;
import java.util.ArrayList;

```

```

import java.util.List;

public class Main {
    public static List<Task> taskList = new ArrayList<Task>();

    public static void findMaxMin() { //this method return a list of Jobs

        List<Job> numOfNodes = new ArrayList<Job>();
        List<Integer> sortedNodes = new ArrayList<Integer>();

        for (int i = 0; i < taskList.size(); i++) {
            int minTime = 100;
            int numOfTask = -1;
            Node node = new Node();

            for (int j = 0; j < taskList.get(i).nodeList.size(); j++) {
                if (taskList.get(i).nodeList.get(j).time < minTime) {
                    numOfTask = j;
                    minTime = taskList.get(i).nodeList.get(j).time;
                    node = taskList.get(i).nodeList.get(j);
                }
            }

            System.out.println(node.time + " in node " + numOfTask);
            numOfNodes.add(new Job(node, taskList.get(numOfTask), node.time));
            sortedNodes.add(node.time);
            for (int i1 = 0; i1 < taskList.size(); i1++) {
                taskList.get(i1).nodeList.get(numOfTask).time += minTime;
            }
        }

        int [] returnArr = new int [sortedNodes.size()];

        for (int i = 0; i < sortedNodes.size(); i++) {
            returnArr[i] = sortedNodes.get(i);
        }

        for (int i = 0; i < returnArr.length; i++) {
            int temp;

            for (int j = i; j < returnArr.length; j++) {
                if (returnArr[j] > returnArr[i]) {
                    temp = returnArr[i];
                    returnArr[i] = returnArr[j];
                    returnArr[j] = temp;
                }
            }
        }

        System.out.println("Sorted time is : ");

        for (int i = 0; i < returnArr.length; i++) {
            System.out.println(returnArr[i]);
        }
    }
}

```

```

    }

    public static void main(String[] args) { // main method

        Task t1 = new Task();
        Node n1 = new Node(8,0);
        t1.nodeList.add(n1);
        Node n2 = new Node(1,0);
        t1.nodeList.add(n2);
        Node n3 = new Node(5,0);
        t1.nodeList.add(n3);

        taskList.add(t1);

        t1 = new Task();
        n1 = new Node(6,0);
        t1.nodeList.add(n1);
        n2 = new Node(9,0);
        t1.nodeList.add(n2);
        n3 = new Node(8,0);
        t1.nodeList.add(n3);

        taskList.add(t1);

        t1 = new Task();
        n1 = new Node(2,0);
        t1.nodeList.add(n1);
        n2 = new Node(3,0);
        t1.nodeList.add(n2);
        n3 = new Node(4,0);
        t1.nodeList.add(n3);

        taskList.add(t1);

        findMaxMin();

    }

}

```

کد الگوریتم دوم و شبیه سازی آن در SPNP

7-4-1- کد نزولی کردن کارها و کد تخصیص کارها در میان افزار

```

format 8
factor on

var sch double sch;
func ScheduleP(Schedule, Time, Cost, req)
Double ScheduleP(int schedule,int Time, int Cost,int req)
{ char chr;
if (Schudule==1)

```

```

chr="FCFS";
else

if (Schudule==2)
chr="LCS";
else

if (Schudule==3)
chr="RR";
else

if (Schudule==4)
chr="MFA";
else
return(-1.0);
Random(selceton algorithm);
return((double)time*cost);
end

func Request(req)
int Request(int req)
{ int req[];
if (req[req]==0)
returne(0);
else
return(1);

end
func NewBatchSize(schedule)
double NewBatchSize(int *schedule)
{
if (chedule==1) return(12.00);
else if (chedule==2) return(10.00);
else if (chedule==3) return(8.00);
else if (chedule==4) return(6.00);
else return(-1.00);

end
func funreq(reqno, reqtime, reqcost)
int funreq(int reqno,double reqtime,double reqcost)
{

return(reqno);

end
func funid-req(nid, req)
int funid-req(int nid,int req)
{ int a,b;
a=funid(nid);
b=funreq(req);
return(a,b);

end
func Output(req, leftreq)
Void Output(int req, int leftreq)
{
fprintf("this is req status ",req);

```

```

fprintf("the req that come back is"leftreq);

end
func Livelock(req[], newreq, schedule, sch, Nodes)
boolean Livelock(int req[],int newreq,int *schedule,int *sch,int *Nodes)
{
if (newreq not in req[])
return(false);
else if (schedule not in[1,2,3,4])
return(false);
else if (Nodes>200)
return(false);
else if (sch>100)
return(false);
else
return(True);

end
func Input(newreq, batch)
void Input(int newreq,double batch[])
{
fprintf("Please Enter newreq");
scanf(newreq);
fprintf("Please Enter batch group");
scanf(batch[]);

end
func getreqID(req)
void getreqID(int req)
{
if requestISDel(req)
empty
else
printf("request have been Finished");

end
func Heuristic(Time, Schedule, Cost)
void Heuristic(int Schedule,Double Time,Double Cost)
{ char chr;
if (Schudule==1)
chr="FCFS";
else

if (Schudule==2)
chr="LCS";
else

if (Schudule==3)
chr="RR";
else
if (Schudule==4)
chr="MFA";
Randomselection;

end
func newSchedulingEvent(Schedule, Time)
double newSchedulingEvent(int *schedule,double Time)

```

```

{
return(SchedulingEvent);

end

srn General-Cases(genC, Token, Task, Enter, IntTime, floor, exp, geC)
* == PLACE ==
CaseID-Counter 0
Start-Case 0
TaskID-Counter 0
No-of-Cases 200
end
* == Timed Transitions ==
Generate-CaseID placedep CaseID-Counter 0.01
end
* == Immediate Transitions ==
end
* == ARC ==
* Input Arcs
CaseID-Counter Generate-CaseID genC
No-of-Cases Generate-CaseID Token(Task)Enter
CaseID-Counter Generate-CaseID (genC+1)+floor(exp(1/100)+0.5)
end
* Output Arcs
Generate-CaseID TaskID-Counter ((genC,IntTime()),1)
Generate-CaseID Start-Case (geC,IntTime())
Generate-CaseID TaskID-Counter ((genC,IntTime()),1)
end
* Inhibitor Arcs
end

echo *****
echo ***** Outputs asked for the model: General-Cases *****

end

```

7-4-2- کد الگوریتم های گرید اقتصادی در CSPL

```

#include "stdafx.h"
#include "stdio.h"
#include "conio.h"
#include "targetver.h"
#include "math.h"

struct task{
    double r_prob[4];// ehtemale gharar giri har task dar 4 manbah-dar ebteda yeksan va 0.25 mibashad
    char name; // 191ari baraye task
    double t_cost;// dar LATO va ALATO estefadeh nemishavad
    double t_time;// bar hasb MI
    double punishment;// mizan PUNISHMENT har task ke dar meghdar dehi avalie har charkhesh LA 0 dar nazr
gerefte mishavad.
    Double reward;// mizan REWARD har task ke dar meghdar dehi avalie har charkhesh LA 0 dar nazr gerefte
mishavad.
}t[4];
//dar inja noe format bandi manbah 191ariff mishavad(GR1)
struct GR1{
    double benf_cost; // ($/1000MI)- mizan HAZINEH MOFID har manbah

```

```

double cost;          //($/Sec) –MIZAN HAZINE har manbah
double no;             //shomare manbah
int q_end;             //shomare ENTEHAY har saf manbah –teedad task haye Queue manbah
double Run_Rate;       //(M1/Sec) mizan NERKH EJRA har manbah
double name;           // name har manbah
double Q[4][3];        //[tehdad fazaye saf baraye task][p(ti) and ti(MI) and ti(name)] saf manbah
}R[4];
double User_Cost,User_Time;
double find_last_task(GR1 R) // SHOMARE Task akhare mojud dar sa R ra barmigardanaad
{
int end;
double k;
end= R.q_end;
k=R.Q[end][3];
return(k);
} //find_last_task
void Insert_task(task T,GR1 R) // Ezafe kardan Task T dar manbah R
{
int k,end;
k= R.q_end;
if (k>5) printf("error");
k++;
int b= (double)R.no;
int c= (char)T.name;
R.Q[k][1]=T.r_prob[b];
R.Q[k][2]=T.t_time;
R.Q[k][3]=c;
} //Insert_task
void Delete_task(task T,GR1 R) // HAZF kardan Task T AZ saf manbah R
{
double k,end;
k= R.q_end;
if (k=0) printf("error");
k--;
int a= (double)end;
int b= (double)R.no;
R.Q[a][1]=0.0;
R.Q[a][2]=0.0;
R.Q[a][3]=0.0;
} //Delete_task
double find_first_task(GR1 R) // SHOMARE Task akhare mojud dar sa R ra barmigardanaad
{
double first;
first=R.Q[1][3];
int a= (double)first;
return(first);
} //find_first_task
double find_current_time(GR1 R[4]) // peyda kardan hazine zamani EJRA Kol karhaye mojud dar safhaye manabaeh
{
double temp1=0.0,temp2=0.0,temp=0.0,Rate=0.0;
int i,j;
for(i=1;i<=4;i++)
{
for(j=R[i].q_end;j<=1; j--)
{ int q_end;
q_end=R[i].q_end;
temp2=R[i].Q[q_end][2];

```

```

        Rate=R[i].Run_Rate;
        temp1=temp2/Rate+temp1;
    }//for-j
    temp=temp1+temp;
} //for-i
printf("current_Time_calculated",temp);
return(temp);
} //find_current_time
double find_current_cost(GR1 R[4])// peyda kardan hazine mali EJRA Kol karhaye mojud dar safhaye manabaeh
{
    double temp1=0.0,temp2=0.0,temp=0.0,Rate=0.0,b_cost=0.0;
    int i,j;
    for(i=1;i<=4;i++)
    {
        for(j=R[i].q_end;j<=1; j--)
        {
            int q_end;
            q_end=R[i].q_end;
            temp2=R[i].Q[q_end][2];
            Rate=R[i].Run_Rate;
            b_cost=R[i].benf_cost;
            temp1=(temp2/Rate)*b_cost+temp1;
        } //for-j
        temp=temp1+temp;
    } //for-i
    printf("current_cost_calculated",temp);
    return(temp);
} //find_current_cost
void resetdata(GR1 R[4],task T[4])//reset kardan kole dade ha(punishment ya Reward task/ehtemalat/khali kardan saf)
{
    int i,j;
    for (i=1; i<=4;i++)
        for (j=1;j<=4;j++)
        {
            T[i].reward=0.0;
            T[i].punishment=0.0;
            T[i].t_time=0.0;
            T[i].t_cost=0.0;
            T[i].r_prob[j]=0.25;
            R[i].no=0.0;
            R[i].name=0.0;
            R[i].Q[j][1]=0.0;
            R[i].Q[j][2]=0.0;
            R[i].Q[j][3]=0.0;
            //R[i].benf_cost=0.0;
            //R[i].cost=0.0;
            R[i].q_end=0.0;
            //R[i].Run_Rate=0.0;
        }
}

} //resetdata
void Res(GR1 R[4]) //policy baraye manabeh(FIX)
{
    R[1].benf_cost=5; R[1].cost=0.5; R[1].Run_Rate=100; R[1].no=1;
    R[2].benf_cost=6.25; R[2].cost=1.5; R[2].Run_Rate=240; R[2].no=2;
    R[3].benf_cost=8.33; R[3].cost=2.5; R[3].Run_Rate=300; R[3].no=3;
    R[4].benf_cost=12; R[4].cost=6; R[4].Run_Rate=500; R[4].no=4;
}
} //double uniform_fun(long int min,long int max)// UNIFORM SELECTION [min,max]

```

```
//{double u;
//u=rand()/(RAND-max+1)*(max-min);
//return (u);
//}
void tasks_Uniform(task t[4])//moshakhasat manabeh tozih yeknavakht
{
}
}
```

7-4-3- الگوریتم زمانبندی و تخصیص کارها به منابع مختلف

```
func Livelock(req[], newreq, schedule, sch, Nodes)
boolean Livelock(int req[],int newreq,int *schedule,int *sch,int *Nodes)
{
if (newreq not in req[])
return(false);
else if (schedule not in[1,2,3,4])
return(false);
else if (Nodes>200)
return(false);
else if (sch>100)
return(false);
else
return(True);

end
func Input(newreq, batch)
void Input(int newreq,double batch[])
{
fprintf("Please Enter newreq");
scanf(newreq);
fprintf("Please Enter batch group");
scanf(batch[]);

end
func getreqID(req)
void getreqID(int req)
{
if requestISDel(req)
empty
else
printf("request have been Finished");

end
func newSchedulingEvent(Schedule, Time)
double newSchedulingEvent(int *schedule,double Time)
{
return(SchedulingEvent);

end
srn Scheduler(newreq, leftreq, batch, reqs, req, SortDes, Time, Cost, SheduleP, Schedule, NewScheduleEvent, Newreq,
Sch, Nodes, Net)
* == PLACE ==
Request-Queue 0
BatchSize 0
```

```

Network-info-req 0
Node-info-req 0
Node-info 0
Network-info 0
Scheduling 0
Heuristic 0
Wait-for-info 0
Req-to-be-Scheduled 0
Scheduled-Activity 0
Sent-Activity 0
LiveLock 0
end
* == Timed Transitions ==
Take-Req-To-Schedule ind 0.25
End-Scheduling ind 0.25
Req-Node&Network-info ind 0.25
Start-Scheduling ind 0.25
end
* == Immediate Transitions ==
end
* == ARC ==
* Input Arcs
Request-Queue Take-Req-To-Schedule Request(newreq)
BatchSize Take-Req-To-Schedule batch
Req-to-be-Scheduled Req-Node&Network-info Request(reqs)
Wait-for-info Start-Scheduling 1
Heuristic Start-Scheduling 1
Scheduled-Activity Req-Node&Network-info SheduleP(,SortDes,Time,Cost)
Scheduling End-Scheduling 1
Request-Queue End-Scheduling Request(newreq)
Node-info Start-Scheduling Nodes
Network-info Start-Scheduling Net
end
* Output Arcs
Take-Req-To-Schedule Request-Queue Request(leftreq)
Take-Req-To-Schedule Req-to-be-Scheduled Request(reqs)
Req-Node&Network-info Network-info-req 1
Req-Node&Network-info Node-info-req 1
Req-Node&Network-info Wait-for-info ScheduleP(req,SortDes,Time,Cost)
End-Scheduling Scheduled-Activity 1
Start-Scheduling Scheduling 1
End-Scheduling Sent-Activity sch
End-Scheduling BatchSize NewBatchSize(Schedule)+NewScheduleEvent(Schedule,Heuristic)
End-Scheduling Request-Queue Request(req)
End-Scheduling Heuristic Heuristic(Schedule,Time,Cost)
end
* Inhibitor Arcs
LiveLock End-Scheduling Livelock(req,Newreq,Schedule,Sch,Nodes)
end
/* This example models a Multi-server FCFS queue with ?nite buffer */
/* An M/M/m/b queue */
# include "user.h"
/* Global variables */
double lambda;
double mu;
int b;
int m;

```

```

int method;
void options(){
method = input("Input 0/1 for Steady-state/Transient analysis");
if ( method == 0 )
iopt(IOP SSMETHOD,VAL SSSOR);
else if ( method == 1 )
iopt(IOP TSMETHOD,VAL TSUNIF);
else
{
fprintf(stderr,"ERROR: Illegal method specification");
exit(1);
}
iopt(IOP PR FULL MARK,VAL YES);
iopt(IOP PR MARK ORDER,VAL CANONIC);
iopt(IOP PR MC ORDER,VAL TOFROM);
iopt(IOP PR MC,VAL YES);
iopt(IOP MC,VAL CTMC);
iopt(IOP PR PROB,VAL YES);
iopt(IOP PR RSET,VAL YES);
iopt(IOP PR RGRAPH,VAL YES);
iopt(IOP ITERATIONS,20000);
iopt(IOP CUMULATIVE,VAL NO);
fopt(FOP ABS RET M0,0.0);
fopt(FOP PRECISION,0.00000001);
lambda = ?nput("Enter lambda");
mu = ?nput("Enter mu");
b = input("Enter the number of buffers");
m = input("Enter the number of servers");
}
/* Marking dependent ?ring rate */
double rate serv(){ if(mark("buf")< m ) return( mark("buf")_mu);
else return(m_mu);}
void net(){
place("buf");
rateval("trin",lambda);
ratefun("trserv",rate serv);
oarc("trin","buf"); mharc("trin","buf",b);
iarc("trserv","buf");
}
int assert(){
/* Make sure that the number of tokens in buf does not exceed the buffer size */
if ( mark("buf")> b)
return(RES ERROR);
else
return(RES NOERR);
}
void ac init(){
fprintf(stderr,"A model of the M/M/m/b Queue");
pr net info();
}
void ac reach(){
pr rg info();
}
double qlength(){ return(mark("buf"));}
double util(){ return(enabled("trserv"));}
double tput(){ return(rate("trserv"));}
double probrej(){ if ( mark("buf") == b ) return(1.0);

```

```

else return(0.0);}
double probempty(){ if ( mark("buf") == 0 ) return(1.0);
else return(0.0);}
double probhalffull(){ if ( mark("buf") == b/2 ) return(1.0);
else return(0.0);}
void ac ?nal(){
double time pt;
/* measures related to the queue */
if ( method == 0 )
{
solve(INFINITY);
pr_expected("Average Queue Length", qlength);
pr_expected("Average Throughput", tput);
pr_expected("Utilization", util);
/* this case corresponds to buf having b tokens */
pr_expected("Probability of rejection", probrej);
/* this case corresponds to buf having zero tokens */
pr_expected("Probability that queue is empty", probempty);
/* this case corresponds to buf having b/2 tokens */
pr_expected("Probability that queue is half full", probhalffull);
}
else
{
for (time pt = 0.1; time pt < 1.0; time pt += 0.1)
{
solve(time pt);
pr_expected("Average Queue Length", qlength);
pr_expected("Average Throughput", tput);
pr_expected("Utilization", util);
/* this case corresponds to buf having b tokens */
pr_expected("Probability of rejection", probrej);
/* this case corresponds to buf having zero tokens */
pr_expected("Probability that queue is empty", probempty);
/* this case corresponds to buf having b/2 tokens */
pr_expected("Probability that queue is half full", probhalffull);
}
for ( time pt = 1.0; time pt < 10.0; time pt += 1.0)
{
solve(time pt);
pr_expected("Average Queue Length", qlength);
pr_expected("Average Throughput", tput);
pr_expected("Utilization", util);/* this case corresponds to buf having b tokens */
pr_expected("Probability of rejection", probrej);/*k this case corresponds to buf having zero tokens */
pr_expected("Probability that queue is empty", probempty);/* this case corresponds to buf having b/2 tokens */
pr_expected("Probability that queue is half full", probhalffull);
}
}
} //End Program

```

مر اجمع

مراجع

- [1] M. K. Molloy, *On the Integration of Delay and Throughput Measures in Distributed Processing Models*, PhD thesis, UCLA, Los Angeles, CA, 1981.
- [2] S. Natkin, *Les Reseaux de Petri Stochastiques et leur Application a l'Evaluation des Systemes Informatiques*. PhD thesis, CNAM, Paris, France, 1980.
- [3] F. J. W. Symons, *Introduction to numerical Petri nets, a general graphical model of concurrent processing systems*, Australian Telecommunications Research, Vol. 14, Issue 1, 1980, pp. 28–33.
- [4] M. A. Marsan, *Stochastic Petri nets: an elementary introduction*, Lecture Notes in Computer Science, Advances in Petri nets 1989, Springer-Verlag, NY, USA, ISBN: 0-387-52494-0, 1990, pp. 1-29. DOI: <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.110.2081&rep=rep1&type=pdf>
- [5] W. Reisig, *Petri nets: an introduction*, Springer-Verlag, ISBN: 0-387-13723-8, New York, 1985, Pages: 161, DOI: <http://www.amazon.com/exec/obidos/ASIN/0387137238/acmorg-20>
- [6] M. A. Marsan, A. Bobbio, G. Conte, A. Cumani, *Performance Analysis of Degradable Multiprocessor Systems Using Generalized Stochastic Petri Nets*, Distributed Processing T-C Newsletters, IEEE Computer Society IEEE Distributed Processing, Technical Committee Newsletter, Vol. 6, No. SI-1, January 1984, pp.47-54.
- [7] L. Kleinrock, *Queueing Systems*, Vol. I: Theory, John Wiley, New York, NY, 1975. (Published in Russian, 1979, Published in Japanese, 1979, Published in Hungarian, 1979, Published in Italian 1992.)
- [8] J. D. C. Little, *A Proof of the Queueing Formula $L = \lambda w$* , Operations Research, Vol. 9, 1961, pp. 383-387. DOI: <http://dx.doi.org/10.2307/167570>
- [9] M. A. Marsan, G. Balbo, G. Conte, *A Class of Generalized Stochastic Petri Nets for the Performance Analysis of Multiprocessor Systems*, ACM Transactions Journal on Computer Systems (TOCS), ISSN: 0734-2071, Vol. 2, Issue 1, May 1984, pp. 93-122. DOI: <http://doi.acm.org/10.1145/190.191>
- [10] M. A. Marsan, G. Balbo, G. Chiola, G. Conte, "Generalized Stochastic Petri Nets Revisited: Random Switches and Priorities", Proceedings of the International Workshop on Petri Nets and Performance Models, IEEE Computer Society Press, Madison, WI, USA, August 1987, pp.44-53.
- [11] G. Balbo, G. Chiola, G. Franceschinis, G. Molinar Roet, "On the Efficient Construction of the Tangible Reachability Graph of Generalized Stochastic Petri Nets", Proceedings of the International Workshop on Petri Nets and Performance Models, Madison, WI, USA, ISBN:0-8186-0796-3, August 1987, pp.136-145.
- [12] P. J. Haas, G. S. Shedler, *Stochastic Petri Nets with Timed and Immediate Transitions*, Communications in Statistics – Stochastic Models, Vol. 5, Issue 4, 1989, pp.563-600. DOI: [10.1080/15326348908807125](http://dx.doi.org/10.1080/15326348908807125)
- [13] J. B. Dugan, K. S. Trivedi, R. M. Geist, V. F. Nicola, *Extended Stochastic Petri Nets: Applications and Analysis*, Proceedings of PERFORMANCE '84 and Technical Report: DUKE--TR--1984--16, Paris, France, Dec. 1984, Pages: 25.
- [14] M. A. Marsan, G. Chiola, "On Petri Nets with Deterministic and Exponential Transition Firing Times", Proceedings of the 7th European Workshop on Application and Theory of Petri Nets, Oxford, England, June 1986. Reprinted in Advances on Petri Nets '87 G. Rozenberg editor, LNCS 266, Springer Verlag, 1987.

- [15] M. A. Marsan, G. Chiola, *On Petri Nets with Deterministic and Exponentially Distributed Firing Times*, in: G. Rozenberg (editor), *Advances in Petri Nets 1987*, Lecture Notes on Computer Science, LNCS 266, Springer Verlag, 1987, pp.132-145.
- [16] M. A. Marsan, G. Balbo, A. Bobbio, G. Chiola, G. Conte, A. Cumani, "*On Petri Nets with Stochastic Timing*", Proceedings of the International Workshop on Timed Petri Nets, Torino, Italy, IEEE Computer Society, ISBN: 0-8186-0674-6, 1985, pp. 80 - 87. DOI: <http://portal.acm.org/citation.cfm?id=645809.670488>
- [17] M. A. Marsan, G. Balbo, A. Bobbio, G. Chiola, G. Conte, A. Cumani, *The Effect of Execution policies on the Semantics and Analysis of Stochastic Petri Nets*, IEEE Transactions Journal on Software Engineering, NJ, USA, ISSN: 0098-5589, Vol. 15, Issue 7, July 1989, pp. 832-846. DOI: <http://dx.doi.org/10.1109/32.29483>
- [18] M. A. Marsan, G. Chiola, A. Fumagalli, "*Improving the Efficiency of the Analysis of DSPN Models*", Proceedings of the 9th European Workshop on Application and Theory of Petri Nets - selected paper, Lecture Notes In Computer Science, Venezia, Italy, ISBN:3-540-52494-0, Vol. 424, June 1988, pp.30-50. DOI: <http://www.springerlink.com/content/p32u474722752226/fulltext.pdf>
- [19] W. Henderson, D. Lucic, "*Application of Generalized Semi Markov Processes to Stochastic Petri Nets*", Proceedings of the International Seminar on Performance of Distributed and Parallel Systems, Kyoto, Japan, December 1988.
- [20] J. Sifakis, "*Performance Evaluation of Systems Using Petri Nets*", Proceedings of the Advanced Course on General Net Theory of Processes and Systems: Net Theory and Applications, Lecture Notes in Computer Science, edited by G. Goos and J. Hartmanis, Springer Verlag, New York, NY, ISBN: 3-540-10001-6, Vol. 84, 1979, pp. 307-319. DOI: <http://www.springerlink.com/content/y7327m3086510100/fulltext.pdf>
- [21] C. Y. Wong, T. S. Dillon, K. E. Forward, "*Timed Places Petri Nets with Stochastic Representation of Place Time*", Proceedings of the International Workshop on Timed Petri Nets, IEEE Computer Society, Torino, Italy, ISBN: 0-8186-0674-6, July 1985, pp.96-103.
- [22] R. R. Razouk, C. V. Phelps, "*Performance Analysis using Timed Petri Nets*", Proceedings of the International Conference on Distributed Computing Systems, Technical Report 206, University of California, Irvine, August 1984.
- [23] M. A. Holliday, M. K. Vernon, *A Generalized Timed Petri Net Model for Performance Analysis*, IEEE Transactions Journal on Software Engineering, IEEE Press, NJ, USA, ISSN: 0098-5589, Vol. 13, Issue 12, July 1987, pp. 1297-1310. DOI: <http://dx.doi.org/10.1109/TSE.1987.233141>
- [24] W. M. Zuberek, "*Timed Petri Nets and Preliminary Performance Evaluation*", International Symposium on Computer Architecture, Proceedings of the 7th Annual Symposium on Computer Architecture in ACM Press, NJ, USA, May 1980, pp. 88-96. DOI: <http://doi.acm.org/10.1145/800053.801913>
- [25] W. M. Zuberek, "*Performance Evaluation using Unbounded Timed Petri Nets*", Proceedings of the Third International Workshop on Petri Nets and Performance Models (PMPM'89), Torino, Italy, July 1989. pp. 180-186. DOI: <http://dx.doi.org/10.1109/PNPM.1989.68551>
- [26] W. M. Zuberek, "*M-Timed Petri Nets, Priorities, Preemptions, and Performance Evaluation of Systems*", In *Advances on Petri Nets '85* edited by G. Rozenberg, LNCS 222, Springer Verlag, 1986, pp. 478-498. DOI: <http://www.springerlink.com/content/5890555581480548/fulltext.pdf>
- [27] M. K. Molloy, "*Fast Bounds for Stochastic Petri Nets*", Proceedings of the International Workshop on Timed Petri Nets, Published in IEEE Computer Society, Torino, Italy, ISBN:0-8186-0674-6, July 1985, pp.244-249. DOI: <http://portal.acm.org/citation.cfm?id=670482>

- [28] M. K. Molloy, "*Structurally Bounded Stochastic Petri Nets*", Proceedings of the International Workshop on Petri Nets and Performance Models, published in IEEE Computer Society, Madison, WI, USA, ISBN: 0-8186-0796-3, 1987, pp. 156-163. DOI: <http://portal.acm.org/citation.cfm?id=645810.670635>
- [29] G. Balbo, G. Chiola, S. C. Bruell, P. Chen, *An example of modeling and evaluation of a concurrent program using colored stochastic Petri nets: Lamport's fast mutual exclusion algorithm*, IEEE Transactions Journal on Parallel and Distributed Systems, ISSN : 1045-9219 ,Vol. 3, Issue 2, 1992, pp. 221-240. DOI: <http://doi.ieeecomputersociety.org/10.1109/71.127262>
- [30] P. J. Haas, G. S. Shedler, "*Regenerative Simulation of Stochastic Petri Nets*", Proceedings of the International Workshop on Timed Petri Nets, Published in IEEE Computer Society, Torino, Italy, ISBN: 0-8186-0674-6, July 1985, pp. 14-21. DOI: <http://www.cs.wm.edu/~ciardo/pubs/2003PMCCS-NonMarkovSimul.ps>
- [31] P. J. Haas, G. S. Shedler, *Regenerative Stochastic Petri Nets*, Journal of Performance Evaluation, Published in Elsevier Science Publishers B. V. Amsterdam, the Netherlands, Vol.6, Issue 3, September 1986, pp. 189-204. DOI: [http://dx.doi.org/10.1016/0166-5316\(86\)90017-9](http://dx.doi.org/10.1016/0166-5316(86)90017-9)
- [32] G. Chiola, "*Compiling Techniques for the Analysis of Stochastic Petri Nets*", Proceedings of the 4th International Conference on Modeling Techniques and Tools for Computer Performance Evaluation, Palma de Madorca, Spain, September 1989.
- [33] G. Balbo, S. C. Bruell, S. Ghanta, *Combining Queueing Network and Generalized Stochastic Petri Net Models for the Analysis of Some Software Blocking Phenomena*, IEEE Transaction Journal on Softwar Engineering, Published By IEEE Press Piscataway, NJ, USA, Vol. SE-12, Issue 4, April 1986, pp. 561-576. DOI: <http://dx.doi.org/10.1109/12.5986>
- [34] G. Balbo, *Introduction to stochastic Petri nets*, Springer Lectures On Formal Methods and Performance Analysis, Lectures on formal methods and performance analysis: first EEF/Euro summer school on trends in computer science, Published By Springer-Verlag New York, New York, NY, USA, ISBN: 3-540-42479-2, 2002, pp. 84-155. DOI: <http://www.springerlink.com/content/0dg47hmt2ynjvk2m/fulltext.pdf>
- [35] R. Boppana, M. M. Halldorsson, *Approximating maximum independent sets by excluding subgraphs*, BIT Magazine, Published By BIT Computer Science and Numerical Mathematics, ISSN:0006-3835, Vol. 32, Issue 2, 1992, pp.180-196. DOI: <http://dx.doi.org/10.1007/BF01994876>
- [36] T. N. Bui, P. H. Eppley, "*A hybrid genetic algorithm for the maximum clique problem*", Proceedings of 6th International Conference on Genetic Algorithms, Published By Morgan Kaufmann Publishers Inc. San Francisco, CA, USA, ISBN: 1-55860-370-0, 1995, pp. 478-484.
- [37] Ya. Z. Tsytkin, *Adaptation and learning in automatic systems*, ISBN: 0127020500, Published By Academic Press, FL, USA, 1971. DOI: <http://www.amazon.com/exec/obidos/ASIN/0127020500/acmorg-20>
- [38] M. L. Tsetlin, *Automaton theory and modeling of biological systems*, published By Academic Press, FL, USA, 1973. DOI: <http://www.amazon.com/Automation-modeling-biological-Mathematics-Engineering/dp/0127016503>
- [39] K. S. Narendra, R. Viswanathan, "*Learning models using stochastic automata*", Proceedings of International Conference of Cybernetics and Society, Washington DC, October 1972.
- [40] V. I. Varshavsky, I. P. Vorontsova, *On the behavior of stochastic automata with variable structure*, Avtomatika & Telemekhanika, Vol. 24, 1963, pp. 353-360.
- [41] K. S. Fu, G. J. McMurtry, *An application of stochastic automata to the synthesis learning systems*, Technical Report TR-EE 65-17, Purdue University, 1965.

- [42] K. S. Fu, T. J. Li, *Formulation of learning automata and automata games*, Journal of Information Science, Vol. 1, Issue 3, 1969, pp.237-256.
- [43] K. S. Fu, T. J. Li, *On stochastic automata and languages*, Journal of Information Science, Vol. 1, Issue 4, 1969, pp. 403-419.
- [44] K. S. Fu, *Stochastic automata as models of learning systems*, In Computer and Information Sciences II, Edition Lou J.T., New York: Academic Press, 1967.
- [45] R. W. McLaren , *A stochastic automaton model for synthesis of learning systems*, IEEE Transactions Journal on System Science and Cybernetics, Vol. 2, 1966, pp. 109-114.
- [46] B. Chandrasekharan, D. W. C. Shen, *On expediency and convergence in variable structure stochastic automata*, IEEE Transactions Journal on System Science and Cybernetics, Vol. 5, 1968, , pp. 145-149.
- [47] K. S. Narendra, M. A. L. Thathachar, *Learning automata: An introduction*, Prentice Hall, 1989.
- [48] K. A. Najim, S. Poznyak, *Learning automata: theory and application*, Tarrytown, New York: Elsevier Science Publishing Ltd., 1994.
- [49] K. S. Fu, G. J. McMurtry, *A study of stochastic automata as models of adaptive and learning controllers*, IEEE Transactions on Automatic Control, Vol. 11, 1966, pp. 379-387.
- [50] S. Lakshmivarahan, *Learning algorithms: theory and applications*, New York: Springer-Verlag, 1981.
- [51] K. S. Narendra, M. A. L. Thathachar, *Learning automata a survey*, IEEE Transactions on Systems, Man and Cybernetics, vol. 4, Issue 4, July 1974.
- [52] B. J. Oommen, D. C. Y. Ma, *Deterministic learning automata solutions to equi-partitioning problem*, IEEE Transactions on Computers, Vol. 37, January 1988, pp. 2-14.
- [53] B. J. Oommen, St. Croix, E. V. De , *String taxonomy using learning automata*, IEEE Transactions on Systems, Man and Cybernetics, Vol. 27, April 1997,, pp. 354-365.
- [54] B. J. Oommen, St. Croix, E. V. De, *Graph partitioning using learning automata*, IEEE Transactions on Computers, Vol. 45, Issue 2, 1996, pp. 195-208.
- [55] A. Tsoularis, C. Kambhampati, K. Warwick, "*Path planning of robots in noisy workspaces using learning automata*", Proceedings of the IEEE International Symposium of Intelligent Control, Chicago, 1993.
- [56] K. Najim, *Modeling and self-adjusting control of an absorption column*, International Journal of Adaptive Control and Signal Processing, Vol. 5, 1991, pp. 335-345.
- [57] V. Gilbert, J. Thibault, K. Najim, "*Learning automata for the control and optimization of a continuous stirred tank fermenter*", IFAC Symposium on Adaptive Systems in Control and Signal Processing, Grenoble, France, July 1992.
- [58] J. Sequeira, C. Bispo, J. Sentieiro, "*Distributed control of a flexible manufacturing plant using learning automata*", Proceedings of the IMACS/IFAC International Symposium on Parallel and Distributed Computing in Engineering Systems, Corfu, Greece, June 1992, pp.327-332.
- [59] E. Ikonen, K. Najim, "*Use of learning automata in distributed fuzzy logic processor training*", IEEE Proceedings of Control Theory and Applications, 1997, pp. 255-262.
- [60] T. Aoki, T. Suzuki, S. Okuma , "*Acquisition of optimal action selection in autonomous mobile robot using learning automata (experimental evaluation)*", Proceedings of the IEEE Conference on Fuzzy Logic and Neural Networks/Evolutionary Computation, Nagoya, Japan, November 1995, pp. 56-63.

- [61] K. S. Narendra, M. A. L. Thathachar, *Learning automata: An introduction*, Prentice Hall, 1989.
- [62] S. Lakshmivarahan and M. A. L. Thathachar, *Absolute expediency of Q and S-model learning algorithm*, *IEEE Transactions on Systems, Man and Cybernetics*, vol. 6, 1976, pp. 222-226.
- [63] R. Viswanathan and K. S. Narendra, *Expedient and Optimal Variable Structure Stochastic Automata*, Technical report CT-31, Dunham Lab., Yale University, New Haven, Connecticut, U.S.A., April 1970.
- [64] L.G. Mason, *An Optimal Learning Algorithm for S-model Environments*, *IEEE Transactions on Automatic Control*, October 1973, pp. 493-496.
- [65] I. Foster, C. Kesselman, *the Grid 2: Blueprint for a New Computing Infrastructure*, Morgan Kaufmann, San Francisco, 2003.
- [66] I. Foster, C. Kesselman, S. Tuecke, *The anatomy of the Grid: Enabling scalable virtual organizations*, *International Journal of Supercomputer Applications*, 2001.
- [67] M. Baker, R. Buyya, D. Laforenza, *Grids and Grid technologies for wide-area distributed computing*, *The Journal of Concurrency and Computation: Practice and Experience*, Vol. 14, Issue 13-15, Nov. 2002.
- [68] V. Berstis, *Fundamentals of Grid Computing*, IBM Redbook series, November 2002, <http://ibm.com/redbooks>
- [69] W. T. Sullivan, D. Werthimer, S. Bowyer, J. Cobb, D. Gedy, D. Anderson, "A new major SETI project based on Project Serendip data and 100,000 personal computers", *Proceedings of the Fifth International Conference on Bioastronomy*, 1997, <http://setiathome.ssl.berkeley.edu/>
- [70] M. Baker, R. Buyya, D. Laforenza, "The Grid: International Efforts in Global Computing", *International Conference on Advances in Infrastructure for Electronic Business, Science, and Education on the Internet (SSGRR 2000)*, Rome, Italy, August 2000.
- [71] R. Buyya, "The Virtual Laboratory Project: Molecular Modeling for Drug Design on Grid", *IEEE Distributed Systems Online*, Vol. 2, No. 5, 2001, <http://www.buyya.com/vlab/>
- [72] S. Smallen, W. Cime, J. Frey, F. Berman, R. Wolski, M. Su, C. Kesselman, S. Young, M. Ellisman, "Combining Workstations and Supercomputers to Support Grid Applications: The Parallel Tomography Experience", *Proceedings of the 9th Heterogeneous Computing Workshop*, May 2000.
- [73] A. Chervenak, I. Foster, C. Kesselman, C. Salisbury, S. Tuecke, *The Data Grid: Towards an Architecture for the Distributed Management and Analysis of Large Scientific Datasets*, *Journal of Network and Computer Applications*, Vol. 23, No. 3, July 2000.
- [74] I. Foster, "An Open Grid Services Architecture", *IBM OMG Web Services Workshop*, March 2002.
- [75] I. Ekmecic, I. Tartalja, V. Milutinovic, "A taxonomy of heterogeneous computing", *IEEE computer*, Vol. 28, No.12, 1995, pp. 68-70.
- [76] R. Buyya, K. Krauter, M. Maheswaran, *A taxonomy and survey of grid resource management systems for distributed computing*, *Journal of Software - Practice and Experience*, Vol. 32, No. 2, 2002, pp. 135-164.
- [77] C. S. Yeo, R. Buyya, *A taxonomy of market-based resource management systems for utility-driven cluster computing*, Technical Report GRIDS-TR-2004-12, Grid Computing and Distributed Systems (GRIDS) Laboratory, University of Melbourne, Melbourne, Australia, Dec. 2004.
- [78] T. L. Casavant, J. G. Kuhl, *A taxonomy of scheduling in general-purpose distributed computing systems*, *IEEE Transactions on Software Engineering*, Vol. 14, No. 2, 1998, pp. 141-154.

- [79] T. D. Braun, H. J. Siegel, "A taxonomy for describing matching and scheduling heuristics for mixed-machine heterogeneous computing systems", Proceedings of the 17th IEEE Symposium on Reliable Distributed Systems, 1998, pp. 330-335.
- [80] I. Ekmecic, I. Tartalja, V. Milutinovic, "A taxonomy of heterogeneous computing", IEEE computer, Vol. 28, No.12, 1995, pp. 68-70.
- [81] J. Peterson, *Petri Net Theory and the Modeling of Systems*, Perantic Hall, Englewood Cliffs, NJ, Vol. 1, 1981.
- [82] W. Reisig, *Petri Nets: An Introduction*, EATCS Monographs on Theoretical Computer Science, Springer Verlag, Vol. 1, 1985.
- [83] R. Mirchandaney, J.A. Stankovic, *Using Stochastic Learning Automata for Job Scheduling in Distributed Processing Systems*, Journal of Parallel and Distributed Computing, 1986, pp. 527-55.
- [84] C. Hirel, S. Wells, R. Fricksy, K. S. Trivedi, "ISPN: An Integrated Environment for Modeling Using Stochastic Petri Nets", Center for Advanced Computing and Communication Department of Electrical and Computer Engineering Duke University Durham, NC 27708-0291, 2000.
- [85] K. Takahashi, M. Yamamura, S. Kobayashi, *A GA Approach to Solving Reach Ability Problems for Petri Nets*, IEICE Transaction on Fundamentals in Electronics Communications and Computer Science, Vol. 79, No. 11, 1996, pp. 1774-1780.
- [86] R. D. Venkataramana, N. Ranganathan, "Multiple Cost Optimization for Task Assignment in Heterogeneous Computing Systems Using Learning Automata", Heterogeneous Computing Workshop (HCW'99), 1999, pp. 137-145.
- [87] J. Moore, L. Hahn, "Petri Net Modeling of High-Order Genetic Systems Using Grammatical Evolution", Bio Systems 72, 2003, pp. 177-186.
- [88] C. Hirel, B. Tun, K. S. Trivedi, "SPNP: Stochastic Petri Nets. Version 6.0", Center for Advanced Computing and Communication Department of Electrical and Computer Engineering Duke University, Durham, NC 27708-0291, U.S.A, 2000.
- [89] E. Schwardy, "Optimization of Petri Nets Structure Using Genetic Programming", Dept. of Cybernetics and Artificial Intelligence, Faculty of Electrical Engineering and Informatics, University of Technology Koice, Slovakia, 2000.
- [90] H. H. Mohamed, D. H. J. Epema, "An Evaluation of the Close-To-Files Processor and Data Co-Allocation Policy in Multiclusters", International Conference on Cluster Computing, Los Alamitos, California, IEEE Computer Society, 2004, pp. 287-298.
- [91] M. LaLena, "Travelling Salesman Problem Using Genetic Algorithms", <http://www.lalena.com/ai/tsp/>, 2007.
- [92] R. Al-Ali, O. Rana, D. Walker, S. Jha, S. Sohail, *G-QOSM: Grid Service Discovery Using QOS Properties*, Computing and Informatics Journal, Special Issue on Grid Computing, Vol. 21, No. 4, 2002, pp. 363-382.
- [93] R. Buyya, *Economic-Based Distributed Resource Management and Scheduling for Grid Computing*, Ph.D. Thesis, School of Computer Science and Software Engineering, Monash University, Melbourne, Australia, 2002.
- [94] R. Buyya, M. Murshed, D. Abramson, "A Deadline and Budget Constrained Cost-Time Optimization Algorithm for Scheduling Task Farming Applications on Global Grids", Proceeding of International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA2002), Las Vegas, NV, 2002.

- [95] K. Czajkowski, I. Foster, C. Kesselman, V. Sander, S. Tuecke, "SNAP: A Protocol for Negotiating Service Level Agreements and Coordinating Resource Management in Distributed Systems", 8th Workshop on Job Scheduling Strategies for Parallel Processing, 2002.
- [96] Y. MahdaviFar, M. R. Meybodi, "Time Optimization in Economic Computational Grids Using Learning Automata", Proceedings of the First Iranian Data Mining Conference, Amirkabir University of Technology, Tehran, Iran, 2007.
- [97] Y. MahdaviFar, M. R. Meybodi, "Cost-Time Optimization in Economic Computational Grids", Proceedings of the Third Information and Knowledge Technology, Ferdowsi University of Mashad, Mashad, Iran, 2007.
- [98] T. Murata, "Some recent applications of high-level Petri nets", IEEE International Symposium on Circuit and System, ISBN: 0-7803-0050-5, Vol. 2, Old Version (1991), 2002, pp. 818-821. DOI: <http://dx.doi.org/10.1109/ISCAS.1991.176488>
- [99] M. C. Zhou, M. D. Jeng, *Modeling, analysis, simulation, scheduling, and control of semiconductor manufacturing systems: A Petri net approach*, IEEE Transaction on Semiconductor Manufacturing, ISSN: 0894-6507, Vol. 11, Issue 3, Old Version (1998), 2002, pp. 333-357. DOI: <http://dx.doi.org/10.1109/66.705370>
- [100] R. Khosla, T. Dillon, "Intelligent hybrid multi-agent architecture for engineering complex systems", International Conference on Neural Networks, Houston, TX, and ISBN: 0-7803-4122-8, Vol. 4, Old Version (1997), 2002, pp. 2449-2454. DOI: <http://dx.doi.org/10.1109/ICNN.1997.614540>
- [101] L. C. Jain, N. M. Martin, *Fusion of Neural Networks, Fuzzy Sets, and Genetic Algorithms: Industrial Applications*, ISBN: 0849398045, First edition, CRC Press, FL, USA, 1998. DOI: <http://www.amazon.com/exec/obidos/ASIN/0849398045/acmorg-20>
- [102] L. Rutkowski, *Flexible Neuro-fuzzy Systems: Structures, Learning and Performance Evaluation (Kluwer International Series in Engineering and Computer Science)*, ISBN: 1402080425, Kluwer Academic Publishers Norwell, MA, USA, 2004. DOI: <http://www.amazon.com/exec/obidos/ASIN/1402080425/acmorg-20>
- [103] P. Stavroulakis, *Neuro-Fuzzy and Fuzzy-Neural Applications in Telecommunications (Signals and Communication Technology)*, ISBN: 3540407596, Springer-verlag, 2004. DOI: <http://www.amazon.com/Neuro-Fuzzy-Fuzzy-Neural-Applications-Telecommunications-Communication/dp/3540407596>
- [104] K. Tsuji, T. Matsumoto, "Extended Petri net Models for Neural Networks and Fuzzy Inference Engines -their net structural Properties", IEEE International Symposium on Circuits and Systems, New Orleans, LA, Vol. 4, 2002, pp. 2670-2673. DOI: <http://dx.doi.org/10.1109/ISCAS.1990.112558>
- [105] M. G. Hadjinicolaou, M. B. E. Abdelrazik, G. Musgrave, "Structured analysis for neural networks using Petri Nets", Proceedings by the 33rd IEEE Midwest Symposium on Circuit and Systems, Calgary, Canada, ISBN: 0-7803-0081-5, Vol. 2, 2002, pp. 770-773. DOI: <http://dx.doi.org/10.1109/MWSCAS.1990.140834>
- [106] K. Venkatesh, O. Masory, A. Pandya, "A High level Petri net Model of Olfactory bulb", In Proceedings of the IEEE International conference on Neural Networks, Sanfrancisco, CA, Vol. 2, 2002, pp. 766-771, DOI: <http://dx.doi.org/10.1109/ICNN.1993.298652>
- [107] S. I. Ahson, *Petri Net Models of Fuzzy Neural Networks*, Proceedings in IEEE Transactions on Systems, Man and Cybernetics, Vol. 35, Issue 6, 2002, pp. 926-932. DOI: <http://dx.doi.org/10.1109/21.384255>

- [108] K. Hirasawa, S. Oka, S. Sakai, M. Obayashi, J. Murata, "*Learning Petri network with route control*", IEEE International Conference on Man and Cybernetics, Intelligent Systems for the 21st Century, Vancouver, BC, Vol. 3, 2002, pp. 2706-2711, DOI:<http://dx.doi.org/10.1109/ICSMC.1995.538192>
- [109] K. Hirasawa, M. Ohbayashi, S. Sakai, J. Hu, "*Learning Petri network and its application to nonlinear system control*", Proceedings in IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics, ISSN: 1083-4419, Vol. 28, Issue 6, 2002, pp. 781-789. DOI: <http://dx.doi.org/10.1109/3477.735388>
- [110] X. F. Zha, S. Y. E. Lim, S. C. Fok, "*Integration of knowledge-based systems and neural networks: neuro-expert Petri net models and applications*", Proceedings in IEEE International Conference on Robotics and Automation, ISBN: 0-7803-4300-X, Vol. 2, 2002, pp. 1423-1428. DOI:<http://dx.doi.org/10.1109/ROBOT.1998.677304>
- [111] T. Sang, C. C. Eric, S. Daniel, Yeung, J. W. T. Lee, "Learning capability in fuzzy Petri nets", Proceedings in IEEE International Conference on Systems, Man and Cybernetics, ISBN: 0-7803-5731-0, Vol. 3, 2002, pp. 355-360. DOI: <http://dx.doi.org/10.1109/ICSMC.1999.823230>
- [112] X. Li, F. Lara-Rosano, "A weighted fuzzy petri net model for knowledge learning and reasoning", Proceedings in IEEE International Joint Conference on Neural Networks, ISBN: 0-7803-5529-6, Vol. 4, 2002, pp. 2368-2372. DOI: <http://dx.doi.org/10.1109/IJCNN.1999.833436>
- [113] X. Li, W. Yu, F. L. Rosano, "*Dynamic knowledge inference and learning under adaptive fuzzy Petri net framework*", Proceedings in IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews, Vol. 30, Issue 4, 2002, pp. 442-450. DOI: <http://dx.doi.org/10.1109/5326.897071>
- [114] X. Chen, D. Jin, L. I. Zhijian, "*Fuzzy Petri nets for rule-based pattern classification*", Proceedings in IEEE International Conference on Communications, Circuits and Systems and West Sino Expositions, ISBN: 0-7803-7547-5, Vol. 2, 2003, pp. 1218-1222. <http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=1179002>
- [115] I. R. Jones, D. P. Tracy, "*A multi-threaded simulator for a distributed control system*", Proceedings in IEEE International Conference on Systems, Man and Cybernetics, ISBN: 0-7803-7952-7, Vol. 3, 2003, pp. 2272-2277. <http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=1244222>
- [116] M. Gao, M. C. Zhou, X. Huang, Z. Wu, "*Fuzzy reasoning Petri nets*", Proceedings in IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans, ISSN:1083-4427, Vol. 33, Issue 3, 2003, pp. 314-324. DOI: <http://dx.doi.org/10.1109/TSMCA.2002.804362>
- [117] M. Hanna, A. Buck, R. Smith, "*Fuzzy Petri nets with neural networks to model products quality from a CNC-milling machining centre*", Proceedings in IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans, ISSN:1083-4427, Vol. 26, Issue 5, 2002, pp. 638-645. DOI: <http://dx.doi.org/10.1109/3468.531910>
- [118] A. H. Song, J. T. Ootsuki, W. K. Yoo, Y. Fujii, T. Aekigchu, "*FMS's Scheduling by colored Petri Net Model and Hopfield Neural Network Algorithm*", Proceedings in International Session Papers of the 34th SICE Annual Conference, ISBN: 0-7803-2781-0, 2002, pp.1285- 1290. DOI: <http://dx.doi.org/10.1109/SICE.1995.526696>
- [119] O. Fukuda, T. Tsuji, K. Takahashi, M. Kaneko, "*Skill assistance for myoelectric control using an event-driven task model*", Proceedings in IEEE/RSJ International Conference on Intelligent Robots and Systems, Lausanne, Switzerland, ISBN: 0-7803-7398-7, Vol. 2, 2002, , pp. 1445- 1450. DOI: <http://dx.doi.org/10.1109/IRDS.2002.1043958>
- [120] J. Yu, R. Buyya, "*A taxonomy of Scientific Workflow Systems for Grid Computing*", Special Issue on scientific Workflows, SIGMOD Records, ISSN:0163-5808, Vol. 34, Issue 3, 2005, pp. 44-49.

- DOI: <http://www.buyya.com/papers/workflow-sigmod05.pdf>
- [121] J. Yu, R. Buyya, "A budget Constrained Scheduling of workflow Application on Utility Grids using Genetic Algorithms", Workshop on Workflows in Support of Large-Scale Science, Proceedings High Performance Distributed Computing Paris, ISSN:1058-9244, Vol. 14, Issue 3-4, 2006, pp. 217-230, DOI: <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.63.7273&rep=rep1&type=pdf>
- [122] K. Jensen, L. Michael Kristensen, L. Wells, *Coloured Petri Nets and CPN Tools for modelling and validation of concurrent systems*, International Journal of Software Tools Technology Transfer, 2007, pp. 1T213-2541T.
- [123] H., Yaojun, J. Changjun, L. Xuemei, "Resource Scheduling Model for Grid Computing Based on Sharing Synthesis of Petri net", Proceedings of the 9th International Conference on Computer Supported Cooperative Work in Design, CSCWD 2005, Vol. 1, UK, pp. 367-372, 2005. DOI: <http://doi.ieeecomputersociety.org/10.1109/CSCWD.2005.194199>
- [124] H., Yaojun, L. Xuemei, "Modeling and Performance Analysis of Grid Task Scheduling Based on Composition and Reduction of Petri nets", Proceedings of the Fifth International Conference on Grid and Cooperative Computing (GCC'06), ISBN: 0-7695-2694-2, pp. 331-334, 2006. DOI: <http://doi.ieeecomputersociety.org/10.1109/GCC.2006.62>
- [125] X. Zhao, B. Wang, Li. Xu, "Grid Application Scheduling Model Based on Petri Net with Changeable Structure", Proceeding of 6th International Conference on Grid and Cooperative Computing (GCC 2007), ISBN: 0-7695-2871-6, 2007, pp. 733-736, DOI: <http://doi.ieeecomputersociety.org/10.1109/GCC.2007.71>
- [126] Y. Han, Ch. Jiang, X. Lue, "Resource Scheduling for Grid Computing and Its Petri Net Model and Analysis", LCNS 3759, Springer 2005, 2005, pp.530-539.
- [127] H. Zhi-gang, H. Rong, G. Wei-hua, *General Scheduling Framework in computational Grid Based on Petri net*, Journal on technology central university in Changsha china, 1005-9784(2005)S1-0232-06, 2005.
- [128] B. SenthilKumar, P. Chitra, G. Prakash, *Robust Task Scheduling on Heterogeneous Computing Systems using Segmented MaxR-MinCT*, International Journal of Recent Trends in Engineering, Vol. 1, No.2, 2009, pp.63-65. DOI: <http://www.academypublisher.com/ijrte/vol01/no02/ijrte0102063065.pdf>
- [129] WU. Min-you, SH. Wei., "Segmented Min-Min: a static mapping algorithm for meta-tasks on heterogeneous computing systems", Proceedings of 9th Heterogeneous Computing Workshop (HCW 2000) [C], 2000, pp.375 –385.
- [130] V. subramani, R. kettimuthu. "Distributed Task Scheduling on Computational Grids Using Multiple Simulations Requests", Proceeding of 11th IEEE International Symposium on High Performance Distributed Computing (HPDC'02), Edinburgh, Scotland, pp. 359-366, 2002.
- [131] M. Murshed, R. Buyya, D. Abramson, "GridSim: A Portable and Scalable Toolkit for Modeling and Simulation of Application Scheduling for Parallel and Distributed Computing", Technical Report, Monash University, Oct. 2001. Available at: <http://www.buyya.com/gridsim/>.
- [132] R. Buyya and M. Murshed, "GridSim: A Toolkit for the Modeling and Simulation of Distributed Resource Management and Scheduling for Grid Computing", Journal of Concurrency and Computation: Practice and Experience, pp. 1-32, May 2002.
- [133] R. Buyya, "The World-Wide Grid (WWG)", <http://www.buyya.com/ecogrid/wwg/>, 2002.



دانشگاه آزاد اسلامی - واحد قزوین

دانشکده مهندسی برق، کامپیوتر و فناوری اطلاعات
گروه کامپیوتر

پایان نامه کارشناسی ارشد مهندسی کامپیوتر
گرایش نرم افزار

یک مدل شبکه پتری تصادفی تطبیقی مبتنی بر اتوماتاهای یادگیر و
کاربردهای آن در تخصیص منابع در شبکه‌های گرید

نگارش: محمد شجاع‌فر

استاد راهنما: دکتر محمدرضا میبیدی

تابستان 1389