****

**دانشگاه آزاد اسلامي - واحد قزوين**

**دانشکدهء مهندسي برق، کامپیوتر و فناوري اطلاعات**

**گروه کامپیوتر**

**پايان نامه کارشناسي ارشد مهندسی کامپيوتر**

**گرايش نرم‌افزار**

**يك مدل شبکه پتری تصادفی تطبیقی مبتنی بر اتوماتاهای یادگیر­ و­ کاربردهای آن درتخصیص منابع در شبکه­های گرید**

**نگارش: محمد شجاع‌فر**

**استاد راهنما: دكتر محمد‌رضا ميبدي**

**تابستان 1389**

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

**با تشکر از استاد گرانقدرم، جناب آقای دکتر محمدرضا ميبدی، که در طول تحصيل دوره‌ی کارشناسی ارشد به خصوص در انجام اين پايان‌نامه، از راهنمايي‌های بی‌دريغ ايشان بهره‌مند بودم.**

همچنين از دوستم سیامک برزگر که در ابتدای راه، مرا ياری نمود، سپاسگزاری می‌کنم.

**چکيده**

*شبكه­های­ پتري تصادفی­ وسيله‌اي براي مطالعه سيستم‌ها مي‌باشند. تئوري شبكه پتري تصادفی­ اجازه مي‌دهد كه يك سيستم بتواند بوسيله آن بصورت یک مدل ریاضی مدل­ شود. از رفتار پويا و ساختار سيستم مدل شده توسط آناليز شبكه پتري تصادفی­­، اطلاعات بسيار مفيدي اتخاذ مي‌گردد كه اين اطلاعات مي‌تواند جهت ارزشيابي­، حدسهاي برای­ ايجاد، بهبود يا تغييرات در سيستم استفاده شود. شبکه­های پتری تصادفی برای آنالیز سیستم­هایی گسترده کاربرد بسزایی دارند. يكي از مشكلات شبکه پتري تصادفی عدم تطبيق پذيري آن­ها مي­باشد و بهمين دليل در شبکه­هاي پتري تصادفی امکان دسترسي به ­اطلاعات قبلي وجود ندارد. اگر در هر زمان بيش از يك گذار فعال باشد، هر كدام مي­توانند به‌عنوان شليك بعدي محسوب ‌شوند. اين ويژگي شبكه پتري حقيقتي را تداعي مي‌كند كه چنانچه چندين واقعه همزمان اتفاق افتد و وقوع رويدادها يكسان نباشد، هريك از رويدادها مي‌تواند رخ دهد و وقوع رويدادها در طول زمان، تغيير نمي­کند و اين برخلاف دنياي واقعي و پويا مي­باشد، و شبيه سازي مشابه اجراي برنامه اصلي است، هدف آنست که از مدل شبيه سازي­شده براي بررسي عملکرد سيستم استفاده شود و بدين­وسيله مشکلات و نقاط ضعف مدل مشخص مي­گردد ولي ابزار شبکه پتري تصادفی به تنهايي نمي­تواند در جهت بهبود و رفع مشکلات کاري انجام دهد و وضعيت بهينه بعدي را نمي­توان پيشگويي نمود. در اين پايان‌نامه، 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 پتري تصادفی تطبيقي از طريق اطلاعات بدست آمده از حالات قبلي سيستم و واکنش­هاي محيط پويا، حالت بهينه بعدي را پيشگويي نموده و وضعيت جاري سيستم را بروز و احتمال وقوع رويدادها را در طول زمان تغيير مي­دهد و باعث مي­شود رويداد­ها بر اساس احتمال وقوعشان فعال ­شوند. بروز شدن وضعيت­هاي سيستم بر اساس واکنش محيط پويا کمک شاياني در يادگيري و آموزش شبکه­هاي پتري مي­کند 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،­ كاربرد مدل تطبيقي پيشنهاد شده در قسمت اول، در گريد محاسباتی مورد مطالعه و بررسي قرار مي­گيرد. در این بررسی ایده ای اکتشافی با توجه به الگوریتم های زمانبندی در گریدهای محاسباتی پیشنهاد می شود 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 هوشند اتوماتای یادگیر در حالتهای مختلف مورد بررسی قرار می گیرد. براي اين منظور در گرید اقتصادی از مدل تطبيقي ارائه شده براي تخصيص بهينه منابع در گريد با توجه به معيار زمان استفاده مي­گردد. 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 کليدی**: شبکه پتری تصادفی، تطبیقی بودن، گريد محاسباتی، زمانبندی، اتوماتای يادگير، بهينه‌سازی 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 benifit information from dynamic beheviour 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 thems.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 conccurrncy 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](#_Toc272055698)

[**1-1-** **مقدمه.** 2](#_Toc272055699)

[**1-2-** **ساختار پایان نامه** 3](#_Toc272055700)

فصل دوم: [کارهای پیشین 6](#_Toc272055701)

[**2-1-** **مقدمه.** 6](#_Toc272055702)

[**2-2-**  **شبكه‌هاي پتري تصادفي** 6](#_Toc272055703)

[2-2-1- تكنيكهاي تحليل در شبکه پتری تصادفی 10](#_Toc272055704)

[2-2-1-1- درخت دسترسي 10](#_Toc272055705)

[2-2-1-1-1- محدوديتهاي درخت دسترسي 13](#_Toc272055706)

[2-2-1-2- معادلات ماتريس 13](#_Toc272055707)

[2-2-2- بهبود كارايي در شبکه پتری تصادفی 14](#_Toc272055708)

[2-2-2-1- شاخص‌هاي كارايي 15](#_Toc272055709)

[2-2-3- بهبود کارائی در شبكه‌هاي پتري تصادفی عمومي 16](#_Toc272055710)

[2-2-3-1- ساختار شبکه پتری تصادفی عمومی 17](#_Toc272055711)

[2-2-3-2- توزيع عمومي تعداد دفعات فعال شدن 21](#_Toc272055712)

[2-2-3-3- روابط با مدل‌هاي ديگر 22](#_Toc272055713)

[2-2-3-4- تجزيه ساختاري شبكه‌هاي پتري تصادفی 24](#_Toc272055714)

[2-2-4- خلاصه و نتیجه گیری 25](#_Toc272055715)

[2-3-اتوماتای يادگير 27](#_Toc272055716)

[2-3-1- مقدمه 27](#_Toc272055717)

[2-3-2- تاريخچه‌ی اتوماتاي يادگير 28](#_Toc272055718)

[2-3-3- اتوماتاي يادگير تصادفي (SLA) 29](#_Toc272055719)

[2-3-3-1- اتوماتاي تصادفي 30](#_Toc272055720)

[2-3-3-2- محيط 30](#_Toc272055721)

[2-3-4- الگوريتم‌هاي يادگيري 32](#_Toc272055722)

[2-3-4-1- الگوريتم‌هاي يادگيري استاندارد 32](#_Toc272055723)

[2-3-4-2- الگوريتم‌هاي يادگيري مدل S 34](#_Toc272055724)

[2-3-4-2-1- الگوریتم  35](#_Toc272055725)

[2-3-4-2-2- الگوريتم  35](#_Toc272055726)

[2-3-4-2-3- الگوريتم  36](#_Toc272055727)

[2-3-5- الگوريتم‌هاي يادگيری با ساختار ثابت 36](#_Toc272055728)

[2-3-5-1- اتوماتاي دو حالته  36](#_Toc272055729)

[2-3-5-2- توسعه‌هاي اتوماتاي دوحالته  38](#_Toc272055730)

[2-3-5-3- اتوماتاي حافظه‌دار با دو عمل  38](#_Toc272055731)

[2-3-5-4- اتوماتاي کرينسکي  40](#_Toc272055732)

[2-3-5-5- اتوماتاي کرايلوف 41](#_Toc272055733)

[2-3-5-6- اتوماتاي  41](#_Toc272055734)

[2-3-5-7- اتوماتاي مهاجرت اشياء 41](#_Toc272055735)

[2-3-6- بازي‌هاي اتوماتا 42](#_Toc272055736)

[2-3-7- خلاصه و نتيجه‌گيري 43](#_Toc272055737)

[2-4- گريدهای محاسباتی 44](#_Toc272055738)

[2-4-1- مقدمه 44](#_Toc272055739)

[2-4-2- طبقه‌بندي سيستم‌هاي گريد 46](#_Toc272055740)

[2-4-3- توانمندی‌های گريد محاسباتی 47](#_Toc272055741)

[2-4-3-1- بهره برداری از منابع بدون استفاده 48](#_Toc272055742)

[2-4-3-2- موازی­سازی پردازنده­ها 49](#_Toc272055743)

[2-4-3-3- برنامه‌های کاربردی 50](#_Toc272055744)

[2-4-3-4- منابع مجازی و سازمانهای مجازی برای ايجاد همکاری 50](#_Toc272055745)

[2-4-3-5- دسترسی به منابع اضافی 51](#_Toc272055746)

[2-4-4- مديريت منابع در گريد 52](#_Toc272055747)

[2-4-4-1- تعاريف و نيازمندي‌ها 52](#_Toc272055748)

[2-4-4-2- مدل انتزاعي سيستم مديريت منبع 53](#_Toc272055749)

[2-4-4-3- سازماندهي ماشين 55](#_Toc272055750)

[2-4-4-4- منابع 55](#_Toc272055751)

[2-4-4-5- زمانبندي 57](#_Toc272055752)

[2-4-5- خلاصه و نتيجه‌گيری 60](#_Toc272055753)

فصل سوم: [شبکه پتری تصادفی تطابقی 62](#_Toc272055754)

[3-1- مقدمه 62](#_Toc272055755)

[3-2- کارهای انجام شده در یادگیری شبکه پتری 63](#_Toc272055756)

[3-2-1- ویژگی های تطابقی 65](#_Toc272055757)

[3-2-2- Fusion Hybrid 66](#_Toc272055758)

[3-2-3- Combination Hybrid 77](#_Toc272055759)

[3-3- خلاصه و نتیجه گیری 82](#_Toc272055760)

فصل چهارم: [کاربرد شبکه پتری تصادفی در گرید محاسباتی 84](#_Toc272055761)

[4-1- مقدمه 84](#_Toc272055762)

[4-2- مرور اجمالی بر گرید محاسباتی 84](#_Toc272055763)

[4-3- کارهای مرتبط در زمانبندی گرید محاسباتی به کمک شبکه پتری 85](#_Toc272055764)

[4-4- ایده پیشنهادی (HSPN) 86](#_Toc272055765)

[4-5- شبیه سازی ایده پیشنهادی 91](#_Toc272055766)

[4-6- خلاصه و نتیجه گیری و کارهای آینده 97](#_Toc272055767)

فصل پنجم: [پیاده سازی شبکه پتری تصادفی تطابقی در گرید اقتصادی 99](#_Toc272055768)

[5-1- مروری بر الگوریتم های موجود و الگوریتم پیشنهادی 99](#_Toc272055769)

[5-1-1- مراحل مشترک الگوريتم‌ها 100](#_Toc272055770)

[5-2-2- الگوریتم BTO و الگوریتم ABTO 100](#_Toc272055771)

[5-1-3- الگوریتم EBTO 105](#_Toc272055772)

[5-1-4- الگوریتم AEBTO 106](#_Toc272055773)

[5-1-5- الگوریتم LATO 108](#_Toc272055774)

[5-1-6- الگوریتم مینیمم هزینه محاسبات 111](#_Toc272055775)

[5-1-7- الگوریتم پیشنهادی ALATO 111](#_Toc272055776)

[5-2- شبیه سازی الگوریتم ها 116](#_Toc272055777)

[5-2-1- مقدمه 116](#_Toc272055778)

[5-2-2- مدل شبکه پتری تصادفی تطابقی در محیط گرید 117](#_Toc272055779)

[5-2-3- پیاده سازی الگوریتم ها 121](#_Toc272055780)

[5-2-4- محیط گرید 122](#_Toc272055781)

[5-2-5- مدل کاربر 122](#_Toc272055782)

[5-2-6- مدل برنامه 123](#_Toc272055783)

[5-2-7- مدل منابع 124](#_Toc272055784)

[5-2-8- شبیه سازی زمان 127](#_Toc272055785)

[5-2-8-1- بهینه سازی زمان در یک هزینه و زمان معین(حالت اول) 128](#_Toc272055786)

[5-2-8-1-1 مدل P-LRP 128](#_Toc272055787)

[5-2-8-1-2 مدل S-LRP 130](#_Toc272055788)

[5-2-8-1-3 مدل S-LRI 131](#_Toc272055789)

[5-2-8-1-4 مدل S-LReP 132](#_Toc272055790)

[5-2-8-2- بهینه سازی زمان در زمان معین با بودجه‌های مختلف (حالت دوم) 133](#_Toc272055791)

[5-2-8-3- بهينه‌سازی زمان در ناهمگونی‌های مختلف(حالت سوم) 139](#_Toc272055792)

[5-3- خلاصه و نتیجه گیری 144](#_Toc272055793)

فصل ششم: [نتيجه‌گيری و پيشنهادها 146](#_Toc272055794)

[فصل هفتم : پیوست ها 149](#_Toc272055795)

[7-1- واژه نامه ها 149](#_Toc272055796)

[7-2- مروری بر نرم افزار SPNP 155](#_Toc272055797)

[7-2-1- راه اندازی در ویندوز XP 155](#_Toc272055798)

[7-2-2- فایلهای خروجی در SPNP 155](#_Toc272055799)

[7-2-3- زبان CSPL 156](#_Toc272055800)

[7-2-4- طراحی در نرم افزار SPNP 160](#_Toc272055801)

[7-3- کدها... 164](#_Toc272055802)

[7-3-1- کد الگوریتم اول 164](#_Toc272055803)

[7-3-1-1- قسمت اصلی کد الگوریتم Min.min 164](#_Toc272055804)

[7-3-1-2- قسمت اصلی کدMax.Min و الگوریتم HSPN 166](#_Toc272055805)

[7-4- کد الگوریتم دوم و شبیه سازی آن در SPNP 167](#_Toc272055806)

[7-4-1- کد نزولی کردن کارها و کد تخصیص کارها در میان افزار 167](#_Toc272055807)

[7-4-2- کد الگوریتم های گرید اقتصادی در CSPL 170](#_Toc272055808)

[7-4-3- الگوریتم زمانبندی و تخصیص کارها به منابع مختلف 173](#_Toc272055809)

[مراجع 179](#_Toc272055810)

**فهرست شکل­ها**

[شکل ‏2‑1: مدل شبكه‌هاي پتري تصادفی برای مثال لامپ 8](#_Toc272054903)

[شکل ‏2‑2: نمایش شبكه‌هاي پتري تصادفی در صف 1/ M/M 9](#_Toc272054904)

[شکل ‏2‑3: يك شبكه پتري براي درخت دسترسي 11](#_Toc272054905)

[شکل ‏2‑4: مرحله اول درخت دسترسي 11](#_Toc272054906)

[شکل ‏2‑5: مرحله دوم درخت دسترسي 11](#_Toc272054907)

[شکل ‏2‑6: مرحله سوم درخت دسترسي 12](#_Toc272054908)

[شکل ‏2‑7: درخت دسترسي شبكه پتري 12](#_Toc272054909)

[شکل ‏2‑8: نمودار معادله ماتريسي 14](#_Toc272054910)

[شکل ‏2‑9: نمونه شبکه های پتری تصادفی عمومی از صف M/M/1 18](#_Toc272054911)

[شکل ‏2‑10: نمایش شبکه های پتری تصادفی عمومی صف M/M/2/5/10 19](#_Toc272054912)

[شکل ‏2‑11: سیستم دو پردازنده ای 20](#_Toc272054913)

[شکل ‏2‑12: شبکه های پتری تصادفی عمومی نمایش داده شده در سیستم دو پردازنده ای در شکل 2-11 21](#_Toc272054914)

[شکل ‏2‑13: نمایش مکانهای زمانبندی شده در چارچوب شبكه‌هاي پتري تصادفی عمومی 22](#_Toc272054915)

[شکل ‏2‑14: نمایش حالت زمانبندی شده سه مرحله ای در چارچوب شبكه‌هاي پتري تصادفی عمومی 24](#_Toc272054916)

[شکل ‏2‑15: رابطه‌ی بين اتوماتا و محيط احتمالي 31](#_Toc272054917)

[شکل ‏2‑16: نمودار عملكرد اتوماتاي دو حالته  37](#_Toc272054918)

[شکل ‏2‑17: گراف انتقال حالت اتوماتاي  37](#_Toc272054919)

[شکل ‏2‑18: گراف‌هاي انتقال حالت اتوماتاي 40](#_Toc272054920)

[شکل ‏2‑19: گراف‌هاي انتقال حالت اتوماتاي کرينسکي 40](#_Toc272054921)

[شکل ‏2‑20: انتقال حالات اتوماتاي کرايلوف هنگام دريافت پاسخ نامطلوب از محيط 41](#_Toc272054922)

[شکل ‏2‑21: نمودار تغيير وضعيت اتوماتا 41](#_Toc272054923)

[شکل ‏2‑22: نمودار تغيير وضعيت اتوماتاي مهاجرت اشياء 42](#_Toc272054924)

[شکل ‏2‑23: طبقه‌بندي سيستم‌هاي گريد 46](#_Toc272054925)

[شکل ‏2‑24: بازنمايي منابع ناهمگون و پراکنده به صورت يک سازمان مجازی واحد توسط گريد 51](#_Toc272054926)

[شکل ‏2‑25: زمينه‌ی سيستم سیستم مدیریت منابع 54](#_Toc272054927)

[شکل ‏2‑26: ساختار انتزاعي سيستم مديريت منابع 54](#_Toc272054928)

[شکل ‏2‑27: طبقه‌بندي سازماندهي ماشين‌ها در گريد 55](#_Toc272054929)

[شکل ‏2‑28: طبقه‌بندي مدل منابع 56](#_Toc272054930)

[شکل ‏2‑29: طبقه‌بندي سازماندهي فضاي نام 57](#_Toc272054931)

[شکل ‏2‑30: طبقه‌بندي يافتن منبع 57](#_Toc272054932)

[شکل ‏2‑31: طبقه‌بندي انتشار منبع 57](#_Toc272054933)

[شکل ‏2‑32: طبقه‌بندي سازماندهي زمانبند 58](#_Toc272054934)

[شکل ‏2‑33: طبقه‌بندي برآورد وضعيت 58](#_Toc272054935)

[شکل ‏2‑34: طبقه‌بندي زمانبندي مجدد 59](#_Toc272054936)

[شکل ‏2‑35: طبقه‌بندي سياست‌هاي زمانبندي 60](#_Toc272054937)

[شکل ‏3‑1: فرم استاندارد انتقال توکن بین مکانها و انتقالها با قوانین فازی تعریف شده 67](#_Toc272054938)

[شکل ‏3‑2: موتور استنتاج فازی لحاظ شده در شبکه پتری توسعه یافته 68](#_Toc272054939)

[شکل ‏3‑3: شبکه پتری عصبی-قسمت اول 69](#_Toc272054940)

[شکل ‏3‑4: شبکه پتری عصبی-قسمت دوم 69](#_Toc272054941)

[شکل ‏3‑5: شبکه پتری فازی عصبی برای نمایش سه قانون AND 71](#_Toc272054942)

[شکل ‏3‑6: شبکه پتری فازی عصبی برای نمایش سه قانون OR 72](#_Toc272054943)

[شکل ‏3‑7: مدل سه لایهای شبکه پتری فازی عصبی برای استنتاج فازی 72](#_Toc272054944)

[شکل ‏3‑8: یادگیری در شبکه پتری در چهار برش ورودی و خروجی شبکه عصبی 73](#_Toc272054945)

[شکل ‏3‑9: چگونگی فعالشدن و روشنشدن یک گذار در شبکه پتری یادگیر در هر برش زمانی 74](#_Toc272054946)

[شکل ‏3‑10: شبکه پتری عصبی دو لایه 74](#_Toc272054947)

[شکل ‏3‑11: مدل شبکه پتری خبره فازی 75](#_Toc272054948)

[شکل ‏3‑12: یک مدل شبکه پتری فازی عصبی برای کنترل سطوح ناهمواری در فرزکاری دستگاه CNC 78](#_Toc272054949)

[شکل ‏3‑13: نمودار شبکه پتری تصادفی مدل FMS 79](#_Toc272054950)

[شکل ‏3‑14: ساختار کلی ایده پیشنهادی آقای فاکودا 80](#_Toc272054951)

[شکل ‏3‑15: مدل وظیفه ایده پیشنهادی آقای فاکودا 80](#_Toc272054952)

[شکل ‏3‑16: مدل وظیفه برای یک وعده غذایی 81](#_Toc272054953)

[شکل ‏4‑1: مدل سه لایه ای زمانبندی کارها یا درخواست ها 87](#_Toc272054954)

[شکل ‏4‑2: مدل ایده پیشنهادی HSPN 90](#_Toc272054955)

[شکل ‏4‑3: زمان متوسط پاسخگویی برای هر درخواست بازای 100 درخواست 94](#_Toc272054956)

[شکل ‏4‑4: متوسط زمان تخصیص داده شده برای هر کار 95](#_Toc272054957)

[شکل ‏4‑5: هزینه متوسط پاسخگویی برای هر درخواست بازای 100 درخواست 96](#_Toc272054958)

[شکل ‏4‑6: متوسط هزینه تخصیص داده شده برای هر کار 96](#_Toc272054959)

[شکل ‏5‑1: الگوریتم BTO 102](#_Toc272054960)

[شکل ‏5‑2: کد برنامه الگوریتم BTO 103](#_Toc272054961)

[شکل ‏5‑3: کد قسمت دوم در ABTO 104](#_Toc272054962)

[شکل ‏5‑4: الگوريتم EBTO 106](#_Toc272054963)

[شکل ‏5‑5: الگوريتم AEBTO 108](#_Toc272054964)

[شکل ‏5‑6: الگوريتم LATO 110](#_Toc272054965)

[شکل ‏5‑7: الگوريتم ALATO 115](#_Toc272054966)

[شکل ‏5‑8: سه لایه شبکه گرید در شبکه پتری تصادفی 117](#_Toc272054967)

[شکل ‏5‑9: تخصیص کارها در لایه میان افزار 118](#_Toc272054968)

[شکل ‏5‑10: نزولی کردن کارها 119](#_Toc272054969)

[شکل ‏5‑11: انتساب کارها به منابع و اعمال پاداش و جریمه و اجرا به تعداد 10000 بار 120](#_Toc272054970)

[شکل ‏5‑12: الگوریتم زمانبندی 121](#_Toc272054971)

[شکل ‏5‑13: کد تعریف کارها 123](#_Toc272054972)

[شکل ‏5‑14: کد و منبعGR1 کد منبع GR2 125](#_Toc272054973)

[شکل ‏5‑15: توانمندی منابع در پيکربندی GR1 126](#_Toc272054974)

[شکل ‏5‑16: قيمت مفيد منابع در پيکربندی GR1 127](#_Toc272054975)

[شکل ‏5‑17: روند یادگیری الگوریتم های LATO و ALATO 129](#_Toc272054976)

[شکل ‏5‑18: مقایسه زمان اجرا الگوریتم ها در اتوماتای استاندارد با بودجه ثابت 150000 130](#_Toc272054977)

[شکل ‏5‑19: مقایسه زمان اجرا الگوریتم ها در اتوماتای S-LRP با بودجه ثابت 150000 131](#_Toc272054978)

[شکل ‏5‑20: مقایسه زمان اجرا اتوماتاها با بودجه ثابت 150000 در بهترین حالت هر اتوماتا 133](#_Toc272054979)

[شکل ‏5‑21: نتايج الگوريتم‌های بهينه‌سازی زمان با بودجه‌های مختلف 136](#_Toc272054980)

[شکل ‏5‑22: مقايسه‌ی الگوريتم‌های ALATO وLATO و Minimum با بودجه‌های مختلف 137](#_Toc272054981)

[شکل ‏5‑23: روند تغييرات زمان اجرای کارها در الگوريتم ALATO با بودجه‌های مختلف 138](#_Toc272054982)

[شکل ‏5‑24: نتايج الگوريتم‌های بهينه‌سازی زمان در ناهمگونی‌های مختلف 141](#_Toc272054983)

[شکل ‏5‑25: مقايسه‌ی الگوريتم‌های LATO و ALATOو Minimum در ناهمگونی‌های مختلف 142](#_Toc272054984)

[شکل ‏5‑26: اختلاف زمان اجرای کارها در الگوريتم‌های LATO و ALATOبا الگوریتم Minimum 143](#_Toc272054985)

[شکل ‏5‑27: نتايج الگوريتم‌های بهينه‌سازی زمان برای کارهای همگون و ناهمگون 144](#_Toc272054986)

[شکل ‏7‑1: قاب اصلي 161](#_Toc272054987)

[شکل ‏7‑2: پروژههای جدید 162](#_Toc272054988)

[شکل ‏7‑3 163](#_Toc272054989)

[شکل ‏7‑4 163](#_Toc272054990)

**فهرست جدول­ها**

[جدول ‏3‑1: جدول تغییرات فازی درمقادیر نشانه گذاری گذرهای شبکه پتری توسعه یافته 67](#_Toc272054991)

[جدول ‏3‑2: وضعیت مکان ها و گذر ها 82](#_Toc272054992)

[جدول ‏4‑1: نشانه های و توضیحات آن ها در HSPN 89](#_Toc272054993)

[جدول ‏4‑2: دسته بندی 100 کار به 4 گروه 25 تایی و شرایط لحاظ شده برای هر کار 92](#_Toc272054994)

[جدول ‏5‑1: علائم اختصاری در نظر گرفته شده برای ایده پیشنهادی 113](#_Toc272054995)

[جدول ‏5‑2: پيکربندی GR1 برای منابع 124](#_Toc272054996)

[جدول ‏5‑3: پيکربندی GR2 برای منابع 125](#_Toc272054997)

[جدول ‏5‑4 : شرايط آزمايش بهينه‌سازی زمان با بودجه‌ ثابت 128](#_Toc272054998)

[جدول ‏5‑5: تغییرات میزان پاداش و جریمه در مدل P-LRP و در الگوريتم‌های بهينه‌سازی زمان با بودجه‌ ثابت 150000 129](#_Toc272054999)

[جدول ‏5‑6: تغییرات میزان پاداش و جریمه در مدل S-LRP و الگوريتم‌های بهينه‌سازی زمان با بودجه‌ ثابت 150000 131](#_Toc272055000)

[جدول ‏5‑7: تغییرات میزان پاداش و جریمه در مدل S-LRI و الگوريتم‌های بهينه‌سازی زمان با بودجه‌ ثابت 150000 132](#_Toc272055001)

[جدول ‏5‑8: تغییرات میزان پاداش و جریمه در مدل S-LReP و الگوريتم‌های بهينه‌سازی زمان با بودجه‌ ثابت 150000 132](#_Toc272055002)

[جدول ‏5‑9: شرايط آزمايش بهينه‌سازی زمان با بودجه‌ متغییر 134](#_Toc272055003)

[جدول ‏5‑10: زمان اجرای برنامه در الگوريتم‌های بهينه‌سازی زمان با بودجه‌های مختلف 135](#_Toc272055004)

[جدول ‏5‑11: شرايط آزمايش بهينه‌سازی زمان در ناهمگونی‌های مختلف و بودجه 150000 139](#_Toc272055005)

[جدول ‏5‑12: زمان اجرای برنامه در الگوريتم‌های بهينه‌سازی زمان با ناهمگونی‌های مختلف 140](#_Toc272055006)

[جدول ‏7‑1: گزینه های موجود برای فایلهای میانی 157](#_Toc272055007)

[جدول ‏7‑2: راه حل تحلیلی و عددی option های موجود 157](#_Toc272055008)

[جدول ‏7‑3: Option های شبیهساز موجود 157](#_Toc272055009)

[جدول ‏7‑4: Option های متفرقه 158](#_Toc272055010)

فصل اول

**مقدمه**

# فصل اول

## مقدمه

شبكه­های­ پتري تصادفی­ وسيله‌اي براي مطالعه سيستم‌ها مي‌باشند. تئوري شبكه پتري تصادفی­ اجازه مي‌دهد كه يك سيستم بتواند بوسيله آن بصورت یک مدل ریاضی مدل­ شود. از رفتار پويا و ساختار سيستم مدل شده توسط آناليز شبكه پتري تصادفی­­، اطلاعات بسيار مفيدي اتخاذ مي‌گردد كه اين اطلاعات مي‌تواند جهت ارزشيابي­، حدسهاي برای­ ايجاد، بهبود يا تغييرات در سيستم استفاده شود. شبکه­های پتری تصادفی برای آنالیز سیستم­هایی گسترده کاربرد بسزایی دارند. . از جمله این سیستم­ها می­توان به نرم‌افزارهای توزيع شده، پروتكل ارتباطي، سيستم‌هاي توليد، سیستم­های پایگاهی، سیستم­های فهرست و آمارهای منطقی، سيستم‌هاي شی­گرا­، سيستم‌هاي زمان واقعي ­و شبکه­های محاسباتی گرید اشاره کرد. يكي از مشكلات شبکه پتري تصادفی عدم تطبيق پذيري آن­ها مي­باشد و بهمين دليل در شبکه­هاي پتري تصادفی امکان دسترسي به ­اطلاعات قبلي وجود ندارد. در شبکه پتری تصادفی اگر در هر زمان بيش از يك گذار فعال باشد، هر كدام مي­توانند به‌عنوان شليك بعدي محسوب ‌شوند. اين ويژگي شبكه پتري حقيقتي را تداعي مي‌كند كه چنانچه چندين واقعه همزمان اتفاق افتد و وقوع رويدادها يكسان نباشد، هريك از رويدادها مي‌تواند رخ دهد و وقوع رويدادها در طول زمان، تغيير نمي­کند و اين برخلاف دنياي واقعي و پويا مي­باشد، و شبيه سازي مشابه اجراي برنامه اصلي است، هدف آنست که از مدل شبيه سازي­شده براي بررسي عملکرد سيستم استفاده شود و بدين­وسيله مشکلات و نقاط ضعف مدل مشخص مي­گردد ولي ابزار شبکه پتري تصادفی به تنهايي نمي­تواند در جهت بهبود و رفع مشکلات کاري انجام دهد و وضعيت بهينه بعدي را نمي­توان پيشگويي نمود.

گريد محاسباتی يک زيربنای سخت­افزاری و نرم­افزاری می­باشد که دسترسی قابل­اعتماد، پايدار، فراگير و ارزان را به توانايي­های محاسباتی ديگران فراهم می­کند. يک گريد محاسباتی با مجموعه­ای از منابع در مقياس وسيع در ارتباط است. چنين مجموعه­ای به يک زيربنای سخت­افزاری برای دستيابی به ارتباطات لازم و يک زيربنای نرم‌افزاری برای بررسی و کنترل نتايج بدست آمده، نياز دارد. منابع در گريد، متعلق به سازمان‌های مختلف می‌باشد که دارای سياست‌های مديريتی و مدل‌های استفاده و هزينه‌ای مخصوص به خود برای کاربران مختلف در زمان‌های گوناگون می‌باشند. همچنين فراهم بودن منابع و ميزان بار بر روی آنها در طول زمان تغيير می‌کند.

صاحبان و کاربران منابع دارای اهداف، استراتژی‌ها و الگوهای عرضه و تقاضای متفاوتی می‌باشند. در مديريت چنين محيط پيچيده‌ای، نمی‌توان از رويکردهای رايج برای مديريت منابع که سعی می‌کنند ميزان کارايي را در سطح سيستم بهينه کنند، استفاده کرد. رويکردهای رايج از سياست‌های متمرکز استفاده می‌کنند که نيازمند اطلاعات کاملی از وضعيت‌ها و يک سياست مديريت زيربنايي مشترک يا يک سياست غيرمتمرکز توافقی می‌باشند. به دليل پيچيدگی‌های موجود در ايجاد محيط‌های گريد موفق، نمی‌توان يک ماتريس کارايي قابل قبول در سطح سيستم و همچنين يک سياست مديريتی مشترک را تعريف کرد.

رويکرد اقتصادی، يک مبنای عادلانه برای مديريت موفق ناهمگونی و غيرمتمرکز بودن که در مسائل اقتصادی بشری وجود دارد، فراهم کرده است. مدل‌های اقتصادی رقابتی، الگوريتم‌ها، سياست‌ها و ابزارهايي را برای اشتراک يا تخصيص منابع در سيستم‌های گريد فراهم می‌کنند. اين مدل‌ها می‌تواند بر مبنای مبادله‌ی پاياپای و يا قيمت‌گذاری باشد. در مدل مبتنی بر مبادله‌ی پاياپای، همه‌ی شرکت‌کننده‌ها بايد دارای منابعی باشند و عمل دادوستد منابع را با تبادل آنها انجام دهند (مثلاً مبادله‌ی زمان پردازنده با فضای ذخيره‌سازی). در مدل مبتنی بر قيمت، منابع دارای قيمتی هستند که بر اساس عرضه، تقاضا و ارزش در سيستم اقتصادی تعيين می‌گردد.

واسطه‌گر منبع گريد که وظيفه‌ی زمانبندی برنامه‌ی کاربردی متعلق به کاربر را بر عهده دارد، بايد دارای يک الگوريتم زمانبندی کارا باشد. اين الگوريتم می‌تواند با یک استراتژی عمده را برای زمانبندی اتخاذ کند. اين استراتژی که توسط بويا پيشنهاد شده است، عبارتست از:

* مينيمم کردن زمان با رعايت محدوديت بودجه (بهينه‌سازی زمان)

در اين پايان‌نامه، الگوريتم‌ جديدی برای استراتژی فوق در مدل اقتصادی بازار کالا، پيشنهاد می‌گردد. الگوريتم‌ پيشنهادیALATO بر اساس اتوماتاهای يادگير برای بهینه سازی زمانبندی می­باشد که نسبت به الگوريتم‌های گزارش شده، از کارايي بهتری برخوردارست و معايب آنها را برطرف می‌کنند. اين الگوريتم‌ در زمانی که ناهمگونی کارها زياد است، از نظر کارايي نسبت به الگوريتم‌های مکاشفه‌ای بهبود قابل ملاحظه‌ای دارند. الگوريتم‌ پيشنهادی و همچنين الگوريتم‌های گزارش شده، با استفاده از جعبه ابزار SPNP شبيه‌سازی شده و کارايي‌های آنها در شرايط گوناگون مورد بررسی قرار می‌گيرد.

## ساختار پایان نامه

در فصل دوم 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 يادگير و انواع آن معرفی می‌گردد. از اتوماتای يادگير در طراحی الگوريتم‌های کارا برای زمانبندی اقتصادی در اين پايان‌نامه استفاده می‌شود. در بخش سوم و انتهایی این فصل انواع طبقه‌بندی مفاهيم مرتبط با سيستم‌های گريد و همچنين تعاريف و مفاهيم مورد نياز، آورده شده است. همچنين مديريت منابع و زمانبندی کارها در گريد مورد بررسی قرار می‌گيرد.

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

در فصل چهارم یک شبکه پتری تصادفی بصورت 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 درخواست ها بررسی شده است و نتايج آنها در قالب نمودارهای مختلف آورده شده و توضيحات لازم ذکر می‌گردد.

فصل ششم شامل نتيجه‌گيری و پيشنهاد برای کارهای آينده می‌باشد.

فصل هفتم، پيوست پايان‌نامه را در بر می‌گيرد. در اين پيوست، واژه‌نامه‌ی فارسی به انگليسی برای معادل‌های به کار رفته در اين پايان‌نامه، راهنمای اجمالی از نرم افزار SPNP استفاده شده برای شبیه سازی الگوریتم ها و همچنين کدهای استفاده شده در روش­های پیشنهادی آورده شده است.

هر فصل شامل يک بخش مقدمه در ابتدا و يک بخش خلاصه و نتيجه‌گيری در انتهای فصل می‌باشد.

## 

فصل دوم

**کارهای پیشین**

# 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 يادگير و انواع آن معرفی می‌گردد. از اتوماتای يادگير در طراحی الگوريتم‌های کارا برای زمانبندی اقتصادی در اين پايان‌نامه استفاده می‌شود. در بخش سوم و انتهایی این فصل انواع طبقه‌بندی مفاهيم مرتبط با سيستم‌های گريد و همچنين تعاريف و مفاهيم مورد نياز، آورده شده است. همچنين مديريت منابع و زمانبندی کارها در گريد مورد بررسی قرار می‌گيرد.

## شبكه‌هاي پتري تصادفي

شبکه­هاي پتري تصادفی توسط مولُی[[1]](#footnote-2) و ناتکین[[2]](#footnote-3) [1-3] به عنوان يک مدل توسعه يافته از شبکه­هاي پتري معرفي شده است­. شبكه‌هاي پتري داراي تأخیرات در فعال شدن تصادفي با همكاري حالتها یا گذارها[[3]](#footnote-4) مي‌باشد كه عمل فعال شدن[[4]](#footnote-5) هر يك از اين حالات يك عمل اتميک و يكتا مي‌باشند که «شبكه‌هاي پتري تصادفی[[5]](#footnote-6)» نامیده مي‌شوند [4].

**تعريف** : شبكه‌هاي پتري تصادفی از شش بخش زیر تشکیل یافته است که ­بصورت زیر نشان داده می شود:

|  |  |
| --- | --- |
| **رابطه (2-1)** | (P, T, I, O, M0, Λ) |

که عبارت است از شبكه‌هاي پتري بدون زمان داراي علامت كه زير بناي شبكه‌هاي پتري تصادفی مي‌باشد. با توجه به مدل بالا شبكه‌هاي پتري تصادفی طبق معمول تشكيل شده‌ از:

* مجموعه از مكانها P=(p1,p2,…,pm)
* يك مجموعه از انتقال ‌ها T= (t1, t2, … tn)
* مجموعه ای از ورودي‌ها
* مجموعه ای از خروجي‌ها 
* يك نشانه گذاری[[6]](#footnote-7) اوليه M0=(m01, m02, … mom)
* = (λ1, λ 2 …λn)  يك رشته از (احتمالاً وابسته به نشانه گذاری ها) سرعتهاي فعال شدن مي‌باشد كه با انتقال­ها مرتبط است. فعال شدن يك انتقال، يك عمليات اتميک مي‌باشد كه علائم از مكان‌هاي ورودي خود جدا شده و در مكان‌هاي خروجي خود توسط عمليات يگانه و واحد، قرار مي‌گيرند (كه اين امر با مدلهاي شبكه‌هاي پتري را زماندار كه بعدها به آن اشاره مي‌كنيم و در آن عمليات فعال شدن که به سه مرحله جداگانه تقسيم مي‌شود، در تضاد مي‌باشد).

يك تأخير درفعال شدن با هر يك از انتقال‌ها همراه است كه مقدار زماني كه بايد قبل از فعال شدن انتقال گذرانده شود را تعيين مي‌كند. اين تأخيرفعال شدن يك متغير تصادفی و داراي تابع شدت احتمال نمايي منفي است. پارامتر تابع شدت احتمال[[7]](#footnote-8) كه به انتقال ti مرتبط است سرعت فعال شدن مي‌باشد كه با λi و ti ارتباط دارد. اين سرعت فعال شدن ممكن است علامتدار باشد در حقیقت بايد به صورت λi(Mj) نوشته شود. تأخيرفعال شدن متوسط انتقالti ، در علامتگذاري Mj عبارت است از [λi(Mj)]-1.

دو نوع توصيف از يك مدل شبكه‌هاي پتري تصادفی امکان پذیر مي‌باشد. اولین توصیف فرض مي‌كند كه هرگاه يك علامتگذاري جديد وارد شود هر كدام انتقال را قادر مي‌سازند تا يك نمونه از تأخير فعال شد تصادفی را از تابع شدت احتمال مربوطه، نشان دهند. انتقالي كه حداقل تأخيرفعال شدن را نشان دهد انتقالی خواهد بود كه فعال شدن آن تغيير در علامتگذاري را مشخص مي‌كند، بنابراين زمان توقف در علامتگذاري با حداقل ارزش نمونه نشانه برابر مي‌باشد. علامتگذاری جديد توسط قوانين زيربنايي شبكه‌هاي پتري غير زماني به دست مي‌آيد و آنگاه فرآيند دوباره آغاز مي‌شود.

به اين علت حداقل دو متغير تصادفی با تابع شدت احتمال نمايي منفي و داراي پارامترهاي و بعنوان متغيرهاي تصادفی مي‌باشند كه به‌صورت نمايي توزيع شده و پارامترهاي  را دارد، زمان توقف در علامتگذاري Mj يك متغير تصادفی با تابع شدت احتمال نمايي منفي است ،در واقع این امر بدين معني مي‌باشد:

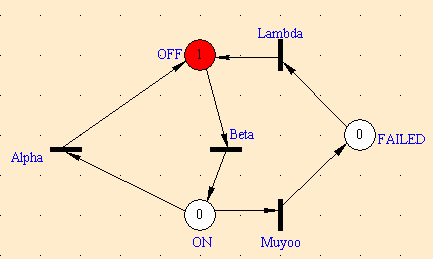
|  |  |  |
| --- | --- | --- |
| **رابطه(2-2)** | |  |
|  | كه در آن E(Mj)، مجموعه‌اي از تمام انتقال‌هاي فعال شده در Mj مي‌باشد. | |

اين واقعيت كه تمامي تأخيرهاي فعال شدن داراي تابع شدت احتمال نمايي هستند به ما امكان نوشتن عبارتي ساده براي تعيين تغيير علامتگذاري توسط فعال شدن را مي‌دهد.

|  |  |
| --- | --- |
| **رابطه(2-3)** |  |

ويژگي بدون حافظه تابع شدت احتمال نمايي منفي، امكان توصيف مفيد و متفاوت‌تري از مدلهاي شبكه‌هاي پتري تصادفی را مي‌دهد. هرگاه يك تغيير در علامتگذاري، انتقالهایی را فعال كند كه قبلاً از آخرين فعال شدن خود فعال نبوده، اين نمونه‌هاي انتقال، يك مدل از تأخيرفعال شدن از تابع شدت احتمال نمايي منفي مربوطه را نشان مي‌دهد و يك زمان سنج را براي ارزش نمونه مدلهاي تأخيري تنظيم مي‌كند. در زمان فعالسازي انتقال، زمان سنج[[8]](#footnote-9) با سرعتي ثابت كاهش مي‌يابد. اگر فعال شدن يك انتقال مخالف، آن را غير فعال كند زمان سنج متوقف شده و در حين فعالسازي مجدد انتقال ميزان كاهش احتمالاً با سرعتي متفاوت از سر گرفته مي‌شود. هنگامي كه زمان سنج به صفر نزديك مي‌شود، انتقال فعال مي‌گردد.

مفيد بودن توصيف دوم از اين واقعيت حاصل مي‌شود كه ممكن است فعاليتها را به انتقال‌ها مرتبط سازيم. اين فعاليتها مي‌توانند در طي پويايي‌هاي مدل آغازشده و در نهایت متوقف گردند و وقتي تكميل شدند، تغيير حالتي را باعث مي‌شوند. اين ارتباط در توصيف قبلي وجود نداشت چون نمونه‌برداري مجدد بدين معنا خواهد بود كه فعاليتها در هر علامتگذاري مجدداً آغاز گشته و در هر آغازشدنی ، مدت زمان متفاوت داشته باشد.



شکل ‏2‑1: مدل شبكه‌هاي پتري تصادفی برای مثال لامپ

به علت ويژگي بدون حافظه تابع شدت احتمال نمايي، توصيفات ديگر را نيز مي‌توان معادل دومي ‌دانست. هنگامي كه شبكه‌هاي پتري تصادفی را با شبکه­های توزيع شده عمومي نام مي‌بريم، اين معادل ديگر درست نخواهد بود. بايد خاطرنشان سازیم که همراه‌سازي يك تأخير درفعال شدن تصادفی با تابع شدت احتمال پيوسته در هر انتقال منجر به احتمالي با مقدار صفر[[9]](#footnote-10) براي دو انتقال در حال برنامه‌ريزي به‌منظور فعال شدن در همان لحظه مي‌گردد. بنابر این،يك مدل شبكه‌هاي پتري تصادفی توسط انتقال‌هاي فعال شده به ترتيب ظاهر مي‌شود.

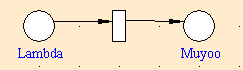
مجموعه فایل دسترس يك شبكه‌هاي پتري تصادفی، به علت حمایت غير زماني تابع شدت احتمال در تأخيرفعال شدن، مشابه يكي از شبكه‌هاي پتري غير زماني زير مي‌باشد. در واقع اين واقعيت كه نمونه‌هاي تأخير فعال شدن از مجموعه اعداد واقعي غير منفي نشان داده مي‌شود اين امر را ضمانت مي‌كند كه در هر علامتگذاري قابل دسترس، تمامي انتقال‌هاي فعال، احتمالي غير صفر از فعال شدن را خواهند داشت. هويت مجموعه قابل دسترس اين امر را مي‌رساند كه ويژگي‌هاي ساختاري بدست آمده براي شبكه‌هاي پتري براي شبكه‌هاي پتري تصادفی نيز معتبر هستند.

نمودار سرعت انتقال حالت زنجیره مارکو پیوسته زمانی[[10]](#footnote-11) كه مطابق با شبكه‌هاي پتري تصادفی مي‌باشد با ساخت گراف قابل دسترس و همچنين با نامگذاري کمان­ها با سرعت فعال سازی انتقال‌هايي كه فعال شدن آنها تغييرات علامتگذار را به‌وجود مي‌آورد، بدست مي‌آيد. بنابراين، به دست آوردن ژنراتور بسيار كوچك زنجیره مارکو پیوسته زمانی ، داراي پيچيدگي يكساني همچون ايجاد مجموعه قابل دسترسي شبكه‌هاي پتري را دارا می باشد. راه‌حل حالت ثابت اين مدل، با حل سيستم‌ معادلات خطي بدست مي‌آيد.

**رابطه(2-4)** 

دقت كنيد كه ، بر احتمال حالت ثابت علامتگذاري Mi دلالت دارد (و نيز در حالت i، يك تطابق يك به يك بين حالات و علامتگذاري‌ها وجود دارد) و ، تابع شدت احتمال تعادل در علامتگذاري‌هاي قابل دسترس مي‌باشد.

مدل شبكه‌هاي پتري تصادفی كه سيستم در نظر گرفته شده در شكل 2-1 به نمايش درآمده است. خواننده، لازم است مدل شبكه‌هاي پتري تصادفی و نمودارهاي سرعت انتقال حالت زنجیره مارکو پیوسته زمانی را با هم مقايسه كند و به شباهت در مكان‌شناسي دقت كند. علت آن به خاطر اين واقعيت است كه شبكه‌هاي پتري زيربناي مدل شبكه‌هاي پتري تصادفی و بصورت یک گراف علامتدار مي‌باشند [5]. در بسياري از حالات اين شباهت در مكان‌شناسي وجود ندارد. به عنوان مثال دوم، شبكه‌هاي پتري تصادفی شكل 2-2 را در نظر بگيريد. اين نمايش شبكه‌هاي پتري تصادفی از صف 1 / M/M مي‌باشد.



شکل ‏2‑2: نمایش شبكه‌هاي پتري تصادفی در صف 1/ M/M

در اين مدل‌ها توصيف در زمينه هایی همچون بهبود كارايي[[11]](#footnote-12) و همچنین توصیف آنکه تأخيرات تصادفي با توابع احتمالي گسترش يافته تواني منفي بکار می روند و نقاط ضعف و قوت آن‌ها چه هستند بحث شده است. تحقيقات زیادی روی این شبکه ها صورت گرفته است .همچنین امروزه توصيفات زیادی بر نتایج بدست آمده وجود دارد .در واقع، تعیین شده است که چه نتايجي به‌عنوان نتايج نامعتبر در نظر گرفته مي‌شوند و چه نتايجي بيشترين تأثير را در استفاده از اين تكنيك مدلسازي در رشته‌ي كاربردي دارند. شبكه‌هاي پتري، در شكل‌ها و اندازه‌هاي مختلف به منظور مطالعه ويژگي‌هاي كيفيتي سيستم‌هايي به‌كار مي‌روند كه داراي ويژگي‌هاي همروندي[[12]](#footnote-13) و همزماني[[13]](#footnote-14) مي‌باشند.

استفاده از تكنيك‌هاي مبتني بر شبكه‌هاي پتري به منظور تحلیل های كيفي سيستم ها ‌نياز به معرفي ويژگي‌هاي موقتی[[14]](#footnote-15) مدل‌هاي غير زمان و بنیادی دارند. اين موقعيت از زمان‌هاي بسيار دور شناخته شده و پيشنهادهاي بسيار متفاوتي براي معرفی ويژگي‌هاي موقتي شبكه‌هاي پتري در مطالعات ظاهر شده است.

پيشنهادهاي اصلي وجود دارد که ویژگیهای پيشنهادات مختلف را نشان مي‌دهدکه عبارتند از:

* عناصر شبكه‌هاي پتري كه زمان به آن‌ها مربوط مي‌شود (مكان‌ها يا تغيير حالات)
* مفهوم فعال شدن در مورد حالت زماني[[15]](#footnote-16) (فعال شدن اتميک يا فعال شدن در سه مرحله)
* ويژگي‌هاي موقتي معمولی (قطعي[[16]](#footnote-17) يا احتمالي)

در اين فصل، مدل‌هاي شبكه‌هاي پتري را در نظر مي‌گيريم كه با یک ويژگي موقتي همراهند . دراين ويژگي، فعال شدن تأخير درتغيير حالات همراه مي‌شود. فعال شدن تغيير حالت اتميک مي باشد.علائم و نشانه ها[[17]](#footnote-18) از مكان‌هاي ورودي خارج مي شود و در مكان‌هاي خروجي به همراه يك عمليات واحد قرار گیرند. ويژگي تأخير فعال شدن طبيعتي احتمالي دارد به‌طوري كه تابع شدت احتمال يا تابع توزيع احتمال[[18]](#footnote-19) كه توابع تاخیری مي‌باشند و همراه با يك تغيير حالت هستند بايد مشخص شوند.

در ساده‌ترين حالت فرض مي‌كنيم كه تمام تأخيرات داراي تابع شدت احتمال نمایی منفي هستند اما به‌صورت مختصر حالت تابع شدت احتمال عمومي را در نظر مي‌گيريم. كلاس مدلهايي كه در نظر مي‌گيريم اغلب با نام شبكه‌هاي پتري تصادفي ناميده مي‌شوند. هدف اين قسمت، به بحث گذاشتن چندين نكته‌ي مربوط به شبكه‌هاي پتري تصادفی مي‌باشند كه عبارتند از:

* علت معرفي شبكه‌هاي پتري تصادفی
* علت استفاده تابع شدت احتمال نمايي براي مشخص كردن زمان
* نكات قوت و ضعف شبكه‌هاي پتري تصادفی
* تلاش‌هاي تحقيقاتي موجود
* نتايج مهم شكست در زمينه شبكه‌هاي پتري تصادفی

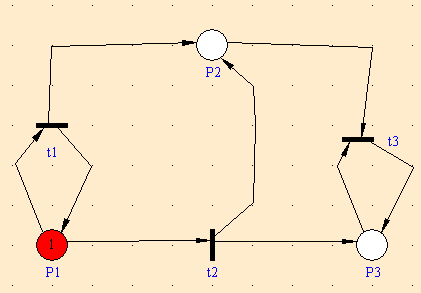
اين بحث متخصصين شبكه‌هاي پتري را مورد خطاب مي‌دهد كه با زمينه‌ي مدل‌سازي عملكرد تصادفي آشنايي ندارند به اين دليل يك مرور مختصر روش كلاسيك ارزيابي عملكرد سيستم ها در يك چارچوب احتمالي درقسمت بعدی آورده شده است كه در آن برخي از ايده‌هاي ابتلايي در مورد فرايندهاي تصادفي و تئوري صف[[19]](#footnote-20) خلاصه شده‌اند.

### تكنيك­هاي تحليل در شبکه پتری تصادفی

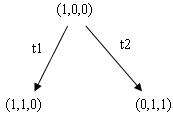
بهترين تكنيك تحليل شبكه پتري درخت دسترسي و تكنيك ديگر معاملات ماتريسي است.

#### درخت دسترسي

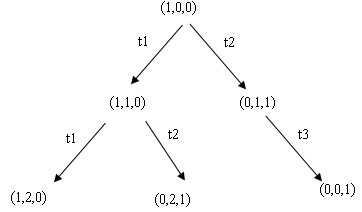
درخت دسترسي، سري دسترسي يك شبكه پتري ارائه مي‌كند كه جهت مطالعه شبكه بسيار مهم است.



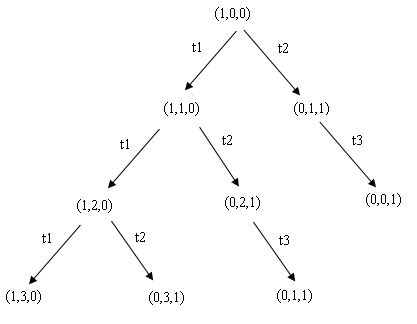
شکل ‏2‑3: يك شبكه پتري براي درخت دسترسي

******

شکل ‏2‑4: مرحله اول درخت دسترسي

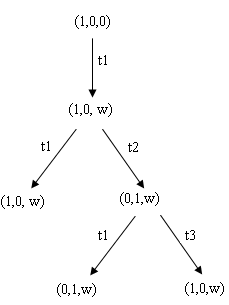
******

شکل ‏2‑5: مرحله دوم درخت دسترسي

****

شکل ‏2‑6: مرحله سوم درخت دسترسي

آن­هايي كه تغييري ندارند با w نشان مي‌دهيم.

****

شکل ‏2‑7: درخت دسترسي شبكه پتري

##### **محدوديت**‌**هاي درخت دسترسي**

درخت فوق براي حل مسائلي از قبيل بي­خطري[[20]](#footnote-21)، حفاظت و امثال آن مورد استفاده قرار مي‌گيرد ولي امكان حل توالي شليك‌ها را ندارد. اين امر به خاطر محدوديت وجود w است.

Submarking: بار تمام حالتهاي Pi، با حالت اوليه  (فقط يك  است)

Zero reachability: براي تمام دخالتهاي است و هيچگاه تغيير نمي‌كند (هميشه صفر است).

Single space zero reachability: براي يك حالت خاص pi،  است کهمی باشد.

Reachabililty:‌يعني چنانچه براي داشته باشيم  همان مقدار اوليه باشد (فقط  تغيير نكند).

#### معادلات ماتريس

راه ديگر تحليل ارائه ماتريسي وقايع است.

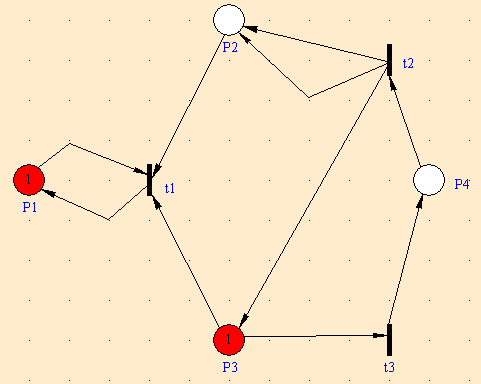
|  |  |  |
| --- | --- | --- |
|  |  | |
| **رابطه­ی (2-5)** | |  |
|  |
|  |  | |

دو ماتريس D- و D+ براي ورودي‌ها و خروجي‌ها تعريف مي‌شوند و توابع ورودي و خروجي‌ تعريف مي‌شوند و توابع ورودي و خروجي را ارائه مي‌كنند. هر ماتريس M رديف (براي هر گذار يك رديف) و N ستون (براي هر مكان يك ستون) دارد.

|  |  |
| --- | --- |
|  |  |
|  |  |
| **رابطه­ی (2-6)** |  |
|  |  |
|  |  |

مقدرا اوليه  فعال است و  بدست مي‌آيد.

|  |  |
| --- | --- |
| **رابطه­ی (2-7)** |  |
|  | |



شکل ‏2‑8: نمودار معادله ماتريسي

### بهبود كارايي در شبکه پتری تصادفی

در ابتدا بهبود كارايي مي‌تواند به دو زمينه كاري قسمت بندی شود. اولين زمينه به اندازه‌گيري[[21]](#footnote-22) بر‌مي‌گرددكه اندازه‌گيري بطور کلی به سه زمينه‌ي متمايز بخش‌بندي مي‌شود كه عبارتند از:

* اندازه‌گيري كردن
* الگوهاو شاخص‌هاي اندازه‌گيري [[22]](#footnote-23)
* پيش نوع‌ها[[23]](#footnote-24)

اندازه‌گيري كردن بر روي سيستم واقعي تحت شرايط اجرایی واقعی امکان پذیر است . نتايج اندازه‌گيري‌ها به جزئيات ویژگیهای سيستم‌هاي اندازه‌گيري وابستگي زیادی دارند و در خلال اندازه‌گيري بر روي سيستم به زمينه‌كاري خاصی اشاره مي‌كنند و عموميت خيلي كمي دارند.

هنگامي كه كارايي دو سيستم مقايسه شده‌اند، اندازه‌گيري‌ها به اندازه کفایت نخواهدداشت و اين از موقعي است كه هيچ تضميني بر شرايط اجرا بر اندازه‌گيري‌هايي كه در نظر گرفته و معادلات آن‌ها وجود ندارد، پس، با اين شرايط مقايسه كردن عادلانه نخواهد بود.

براي غلبه بر اين مشكل شاخص‌هاي اندازه‌گيري بوجود آمدند. آن‌ها كه يك بار كردن كاري[[24]](#footnote-25) بطور مصنوعي را براي سيستم‌ها مهيا مي‌كنند. مثلاً، مشاهدات مي‌تواند در يك شرايط اجرايي معادل اجرا شود و مقايسات معني داری بوجود آيد. اندازه‌گيري و شاخص‌هاي اندازه‌گيري موجوديتهاي يك سيستم می باشند كه بايد مورد مطالعه قرار بگيرند و بنابراين آن‌ها مي‌توانند در (بسياري) از حالت كه مطالعه كارايي مربوط به يك سيستم مهيا نباشند (ممكن است هنوز عملي و اجرايي نشده باشد) مشاهده شوند. لازم است كه يك چارچوب نمايشي از سيستم هم بصورت سخت‌افزاري و هم بصورت نرم افزاری نرم‌افزاري طراحي شود. چنين ساختاري مطمئناً نيازمند جزئياتی مي‌باشد كه به‌طور معمول به آن‌ها پيش‌نمونه گويیم. در واقع، اين يك ساختار اوليه است كه چارچوب كلي كار را نشان مي‌دهد. مشاهدات بر روي چنين پيش‌نمونه‌هايي اجرا مي‌شوند و استفاده از شاخص‌هاي كارايي را به‌عنوان بار كردن كارهاي مصنوعي امكان‌پذير مي‌‌كنند.

#### شاخص‌هاي كارايي

عموماً ،هدف تجزيه مدل شبكه‌هاي پتري تصادفی، محاسبه نمايه‌هاي عملكردي متراكم‌تر و منسجم تر مي‌باشد تا محاسبه احتمالات علامت‌گذاري فردي. چندين نوع از نتايج به هم پيوسته را مي‌توان به راحتي از توزيع‌ حالت ثابت در علامتگذاري‌هاي قابل دسترس بدست آورد. در اين بخش، برخي از پارامترهاي عملكردي حالات ثابت بهم پيوسته را كه به سادگي قابل محاسبه مي‌باشند را معرفي مي‌كنيم[6].

* احتمال يك پيشامد كه از طريق علامت‌گذاري‌هاي مكاني تعريف شده (مثلاً، بدون علامت در مجموعه مكانها، و حداقل يك علامت در يك مكان، در حالي كه مكان ديگر خالي مي‌باشد) را مي‌توان با اضافه كردن احتمال هاي تمام علامتگذاري‌ها مشخص كرد و در آنها شرايط مشابه تعريف پيشامد،صحیح مي‌باشد. بنابراين، مثلاً، احتمال حالت ثابت پيشامد A، كه از طريق حالتي تعريف شده كه براي علامتگذاري‌هاي  درست مي‌باشد، از طريق  بدست مي‌آيد.
* *pmf* عدد علامت‌ها در يك مكان، مثل pi ، را مي توان با محاسبه احتمالات تكي در pmf مانند احتمالات پيشامد «مكان pi كه شامل علائم k مي‌شود» بدست آورد.
* تعداد متوسط علائم يك مكان را مي‌توان با محاسبه pmf علائم در آن مكان به دست آورد.
* بسامد فعال شدن يك انتقال يعني متوسط دفعاتي كه انتقال در واحد زمان فعال مي‌شود را مي‌توان توسط مجموع سرعت فعال سازی انتقال محاسبه كرد.

|  |  |
| --- | --- |
| **رابطه (2-8)** |  |

كه در آن ، بسامد فعال سازی tj مي‌باشد، E(Mi) مجموعه انتقال‌هاي فعال شده در Mi و  ،سرعت فعال سازی tj در Mi مي‌باشد.

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

|  |  |
| --- | --- |
| **رابطه (2-9)** |  |

كه در آن E[T] تأخير متوسط، E[N] تعداد متوسط علائم در فرآيند عبور از شبكه فرعي و  سرعت ورودي متوسط علائم در شبكه فرعي مي‌باشد. اين روند را هرگاه كه امكان شناسايي علائم جالب در داخل شبكه فرعي وجود داشته باشد (كه همچنين مي‌تواند شامل نشانه‌هاي ديگر باشد كه شرايط داخلي آن را تعريف مي‌كند 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).

### بهبود کارائی در شبكه‌هاي پتري تصادفی عمومي[[25]](#footnote-26)

عامل اساسي كه كاربرد مدلهاي شبكه‌هاي پتري تصادفی را محدود مي‌سازد، پيچيدگي تجزيه آنهاست. اين امر به علت عوامل زيادي مي‌باشد. احتمال تعداد بسيار زياد نشانه‌گذاري‌ها اصلي‌ترين آنها است. جنبه‌هاي ديگر، ممكن است به پيچيدگي راه حل مدل اضافه كند. يكي از آنها به علت حضور در مدل فعاليتها كه نسبت به آنهايي كه مربوط به پيشامدهايي هستند و نقش حياتي در عملكرد كلي بازي مي‌كنند، در مقياس زماني بسيار سريع‌تري رخ مي‌دهند. اين منجر به پيدايش سيستم‌هاي محاسبه‌اي طولي كه تجزیه و تحلیل آن دشوار مي‌باشند مي‌شود، يعني، حل آنها با مقدار قابل قبولي از دقت توسط تكنيكهاي عددي معمول سخت مي‌باشد. از طرفي ديگر، ناديده پنداشتن فعاليت‌هاي آنی[[26]](#footnote-27) ممكن است منجر به مدلهايي شود كه از لحاظ منطقي غير صحيح هستند. حتي ممكن است در ساخت مكان‌شناسي يك مدل شبكه‌هاي پتري تصادفی، تجزيه‌گر انتقال‌هايي را وارد كند كه با جنبه‌هاي كاملاً منطقي رفتار سيستم مطابق است[9,10].. بطوري كه، هيچ زمان‌گذاري را نمي‌توان به آنها مرتبط دانست. مدلهاي شبکه های پتری تصادفی عمومی از دو نوع انتقال تشكيل شده‌اند که عبارتند از:

1. انتقال زمان بندي شده كه، مرتبط هستند با تأخيرهاي فعال شدن توزيع شده نمايي مانند شبكه‌هاي پتري تصادفی
2. انتقال‌هاي بلافاصل كه در زمان صفر و با تقدم بر انتقال‌هاي زمان‌بندي شده فعال شده مي‌باشند.

علاوه بر آن، کمان ها مجاز می باشند، سطوح تقدم متعددي ازانتقال‌هاي بلافاصل را مي‌توان استفاده كرد و وزن‌ها با انتقال‌هاي بلافاصل مرتبط هستند.

#### ساختار شبکه پتری تصادفی عمومی

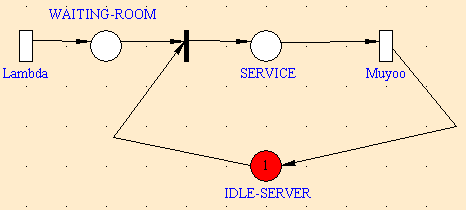
يك شبکه های پتری تصادفی عمومی ، يك هشت تايي بصورت زیر مي‌باشد:

|  |  |
| --- | --- |
| **را بطه (2-10)** | (P, T, ρ, I, O, H, M0, W) |

كه در آن (P, T, *ρ*, I, M0, O, H) مدل غير زماني اساسي است كه شامل:

* شبكه‌هاي پتري اساسي (P, T, I, O, M0)
* يك مجموعه از کمانهاي مانع بصورت 
* تقدم‌هاي انتقال به انتقال‌ها، *ρ* كه با تقدم پايين‌تر (0) داراي انتقال‌هاي زماني و تقدم‌هاي بالاتر (1 ≥) با انتقال‌هاي بلافاصل ارتباط دارد و  ، يك سري است با مدخل 
* يك پارامتر تابع شدت احتمال نمايي منفي از تأخير فعال سازی انتقال است اگر tiيك انتقال زماني باشد.
* Wيك وزن است كه براي محاسبه احتمالات فعال سازی انتقال‌هاي بلافاصل استفاده مي‌شود اگر ti يك انتقال بلافاصل باشد.

در نمونه گرافيكي شبکه های پتری تصادفی عمومی ، انتقال‌هاي بلافاصل به‌عنوان بخش‌ها و انتقال ‌هاي زماني به‌عنوان مربع های چهارگوش‌ طراحي شده‌اند. تفسير يك مدل شبکه های پتری تصادفی عمومی مشابه تفسير مدل شبكه‌هاي پتري تصادفی است كه تغييراتي براي توضيح انتقال‌هاي بلافاصله لازم مي‌باشد. هنگامي كه علامتي وارد مي شود، ابتدا لازم است كه اطمينان حاصل شود که فقط انتقال‌هاي زماني يا حداقل يك انتقال بلافاصل را فعال مي‌سازد. علامت‌گذاري‌هاي نوع پيشين را محسوس[[27]](#footnote-28) و علامتگذاري‌هاي نوع دوم را در نامحسوس مي‌ناميم.



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

در حالت علامت‌گذاري محسوس، زمان سنجهاي انتقال‌هاي زماني فعال شده كاهش خود را از سرگرفته يا دوباره تنظيم شده و سپس كاهش مي‌يابند تا زماني كه يك انتقال زماني درست مثل حالت شبكه‌هاي پتري تصادفی فعال شود.

در حالت علامتگذاري نا­محسوس، انتخاب اينكه كدام انتقال فعال شده، نمي‌تواند براساس توصيف موقتي باشد زيرا تمام انتقال‌هاي بلافاصل دقيقاً در زمان صفر فعال می شوند . پس اين انتخاب براساس تقدمات و وزنها می باشد. مجموعه انتقال‌هاي داراي بيشترين سطح تقدم در ابتدا شناسايي مي‌شود و اگر شامل بيش از يك انتقال شود، انتخاب بعدي، از نوع احتمالي و براساس وزنهاي انتقال طبق عبارت زير خواهد بود:

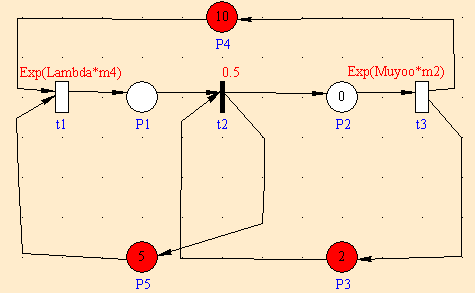
|  |  |
| --- | --- |
| **رابطه (2-11)** |  |

كه در آن E(M) عبارت است از مجموعه انتقال‌هاي بلافاصل‌ فعال شده در علامتگذاري Mيعني انتقال‌هاي داراي تقدم با سطح بالا.دقت كنيد كه معناشناسي مدل شبکه های پتری تصادفی عمومی هميشه بر اين فرض استوار نيست كه انتقال‌ها، حتي در علامتگذاري غير محسوس كه شامل انتقال‌هاي بلافاصل فعال شده يكسان است، يك به يك فعال ‌شوند. معادله اين رفتار با معادله‌اي كه از فعال شدن همزمان برخي از انتقال‌هاي بلافصل در اين مدل ناشي مي‌شود را مي‌توان براي كاهش پيچيدگي الگوريتم‌هاي راه حل بكار برد [11].

وجود کمان هاي مانع وتقدمات اولیه تعداد علامتگذاري‌هاي قابل دسترس را با توجه به شبكه‌هاي پتري بنيادي كاهش مي‌دهد. مجموعه قابل دسترس شبکه های پتری تصادفی عمومی ، برابر است با يكي از شبكه‌هاي پتري بنيادي كه شامل کمان ‌هاي مانع و تقدمات مي‌باشد. تجزيه يك مدل از شبکه های پتری تصادفی عمومی باز به حل يك سيستم معادلات خطي دارد كه شامل تعداد زيادی معادلات مانند علامتگذاري‌هاي محسوس[[28]](#footnote-29) قابل دسترس مي‌شود. ژنراتور بسيار ريز[[29]](#footnote-30) زنجیره مارکو پیوسته زمانی كه به مدل شبکه های پتری تصادفی عمومی مرتبط است با كاهش گراف قابل دسترس كه تحت عنوان سرعتها يا وزنهاي انتقال‌هاي باعث تغيير علامت قرار گرفته است، بوجود مي‌آيد.

در شكل 2-9 به ‌عنوان اولين نمونه از مدل شبکه های پتری تصادفی عمومی ، نمايشي از صف M/M/1 را نشان داده‌ايم. احتمال استفاده از يك انتقال بلافاصله نمایش آشكار[[30]](#footnote-31) اتاق انتظار، از ايستگاه سرويس و از سِروِر بلااستفاده را ممكن مي‌سازد. يك مدل نسبتاً پيچيده‌تر، نمونه شبکه های پتری تصادفی عمومی از صف M/M/2 است كه داراي اتاق انتظار محدود (5 جايگاه) و جمعيت كاربري محدود (10 مشتري) مي‌باشد (مانند شكل 2-10).

اين سيستم صفي عموماً M/M/2/5/10 نام دارد. اتاق انتظار توسط مكان p1 نشان داده مي‌شود، سِروِرهاي بلا استفاده توسط نشانه‌هاي p3 در آن مكان به نمايش درمي‌آيند؛ سِروِرهاي مشغول نشانه p2 در محل را دارند. مكان p4، داراي مشترياني است كه ممكن است به صف برسند. مكان p5 داراي نشانه‌هايي است كه مكان­هاي خالي را در اتاق انتظار نشان مي‌دهد و باعث ايجاد اعتبار براي پذيرش در صف مي‌گردد. علامتگذاري p4، سرعت فعال سازی ورود مشتريان مدل انتقال و علامتگذاري p2، سرعت فعال سازی خروج مشتريان مدل انتقال را تعيين مي‌كند.



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

**GB**

**PM1**

**CM1**

**CM2**

**PM2**

**LB2**

**LB1**

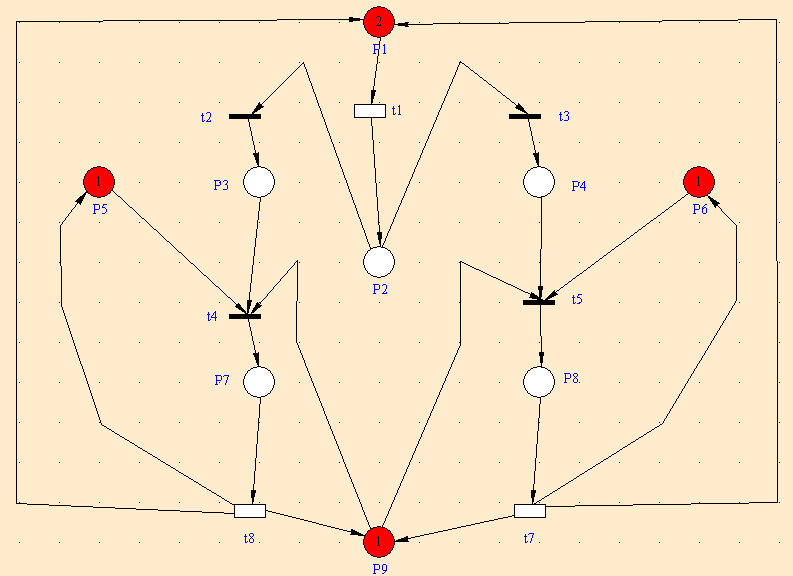
شکل ‏2‑11: سیستم دو پردازنده ای

در واقع، سيستم‌هاي صفي جمعيت محدود عموماً با استفاده از يك فرآيند ورودي به همراه زمانهاي ورودي مياني توزيع شده به صورت نهايي كه سرعت آن با توجه به تعداد مشتريان خارج از صف نسبي است، مدل‌بندي مي‌شوند.

به عنوان سومين نمونه از يك مدل شبکه های پتری تصادفی عمومی ، عملكرد سيستم داراي 2 پردازشگر شكل2-10 را مورد تجزيه قرار مي‌دهيم[12].

واحدهاي داراي دو پردازشگر، از يك پردازشگر (P1, P2) و يك حافظه اختصاصي (PM1, PM2) تشكيل شده كه توسط دو باس محلی(,LB21LB) به هم مرتبط مي‌شوند و از دو ماژول حافظة معمولی(CM1,CM2) توسط يك باس كلي و سراسری(GB) به هم مرتبط مي‌شوند. پردازشگرها يا مي‌توانند فعال باشند يعني در حافظه اختصاصي عمل كنند يا سعي كنند به يك واحد حافظه مشترك دست يابند. تنها زماني تقاضاها پاسخ داده مي‌شوند كه باس سراسری و حافظه عمومي تقاضا شده در دسترس باشند (توجه كنيد كه در اين حالت، در دسترس بودن باسهای عمومي بدين معناست كه حافظه‌هاي كلي هر دو در دسترس هستند، طوري كه مدل را مي‌توان ساده نمود) زمانهاي فعال پردازشگر و زمانهاي دستيابي به حافظه را متغيرهاي تصادفی با تابع شدت احتمال نمايي منفي در نظر مي‌گيريم.

مدل شبکه های پتری تصادفی عمومی بدست آمده از اين سيستم داراي دو پردازشگرمی باشد که در شكل­2-12 نشان داده شده است. نشانه‌ها در مكان p1، پردازشگرهاي فعال را به نمايش درمي‌آورند. انتقال t1، مورد يك تقاضا دسترسي به حافظه كلي را نمايش مي‌دهد. انتقال‌هاي t1 و t3، انتخاب يكي از دو واحد حافظه كلي را مدل‌گذاري مي‌كنند. نشانه‌ها در مكان‌هاي p5 و p6، در دسترس بودن دو واحد حافظه كلي را نيز نشان مي‌دهد. يك نشانه در مكان p9، حاكي از در دسترس بودن باس عمومي مي‌باشد. انتقال‌های t4 و t5 آغاز دسترسي به حافظه كلي را مدل‌گذاري مي‌كنند كه مدت آن با تأخير در انتقال‌هاي t6 و t7 نشان داده مي‌شود، سرعتهاي فعال شدن t6, t1 و t7، عبارتند از λ، و.



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

كاربرد تكنيك‌هاي مدلسازي شبكه‌هاي پتري تصادفی و شبکه های پتری تصادفی عمومی در زمينه‌هاي مختلف فاصله زیادی با هم دارند. عاملي كه پذيرش آن‌ها را به‌عنوان ابزار‌هاي مدلسازي محدود كرده است در پيچيدگي واقع گرایانه[[31]](#footnote-32) مدل وجود دارد یعنی اين عاملي كه در اين سيستم‌ها باعث محدود بودن شده است. مسأله پيچيدگي را به‌طور دقيق‌تر در بخش بعدي مورد بحث و بررسي قرار مي‌دهيم. همچنين بايد در اين مسأله تأكيد نمود كه كاربرد شبكه‌هاي پتري تصادفی و شبکه های پتری تصادفی عمومی وابستگی شدیدی به ابزارهاي كافي نرم‌افزاري موجود برمي‌گردد كه ساخت مدل و راه‌حل‌ها فقط به مثال‌هاي بسيار ريز امكان‌پذير خواهند بود.

#### توزيع عمومي تعداد دفعات فعال شدن

برخي از تحقيقات تلاش بر اين است تا تلاش را به اين اختصاص دهد تا قدرت مدلسازي شبكه‌هاي پتري تصادفی و شبکه های پتری تصادفی عمومی را با اجازه دادن تأخيرات فعال شدن تا متغيرهاي تصادفي تابع شدت احتمال مطلق مجاز بشمرد. اين نتايج در تعاريف شبكه‌هاي پتري تصادفی توزیع شده[[32]](#footnote-33) يا شبكه‌هاي پتري تصادفی گسترش یافته[13][[33]](#footnote-34)، شبكه‌هاي پتري تصادفی معين[14,15][[34]](#footnote-35) يا تاخیر انتشار در فاز گسترش آن ها بیان شده است. ذکر این نکته ضروری است که موجود بودن در اين مدل‌ها داراي تأخير در فعال‌سازي تصادفي با تابع شدت احتمال با تأخيرهاي نمايي منفي در روند تعريف اجراي شبكه‌هاي پتري تصادفی مي‌باشیم.در واقع تابع شدت احتمال نمايي آن‌ها منفي است یا به تعبير ديگر، هرگاه تغيير علامتگذاري اتفاق بيافتد حالتي را فعال مي‌كند كه از آخرين فعالسازي خود فعال نشده بودند. اين حالت نمونه‌اي از تأخير فعالسازي را از تابع شدت احتمال عمومي مرتبط نشان مي‌دهد و يك زمان سنج ارزش يك نمونه‌ي تأخير در نظر گرفته شده را تنظيم مي‌كند[16,17].

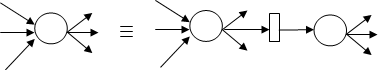
وقتي كه انتقال فعال مي‌شود زمان سنج با سرعت يكساني كاهش مي‌يابد و اگر حالت انتقال با فعال شدن يك حالت متضاد غير فعال شود آنگاه زمان سنج متوقف مي‌شود و زماني كه انتقال دوباره فعال مي‌شود زمان سنج دوباره فعال مي‌شودو کارش را ادامه می دهد. هنگامي كه زمان سنج به مقدار 0 مي‌رسد انتقال فعال مي‌شود. روندهاي فعال‌سازي ديگر نيز مانند رقابت با حافظه فعال‌سازي كه در آن موقع كه يك انتقال فعال مي‌شود نمونه‌گيري مجدد صورت مي‌گيرد و پيش انتخاب كه در آن يك انتقال‌كه بايد فعال شود و مستقل از تأخيرهاي فعالسازي انتخاب مي‌شود.

تفاوت بين حالات رقابت ميان طول عمر و حافظه فعال كننده از اين واقعيت ناشي مي‌شود كه درحالت توزيع‌هاي عمومي ويژگي بدون حافظه وجود ندارد بنابراين، زمان باقي مانده قبل از فعالسازي هنگامي كه انتقالی كه غير فعال شده بود دوباره فعال مي‌شود توزيعي متفاوت از زمان فعالسازي نمونه‌گيري شده از توزيع مربوط به انتقالی غير فعال فعال شده بود ایجاد می کند . متأسفانه قدرت مدلسازي كاهش يافته‌ي اين مدل‌هاي شبكه‌هاي پتري تصادفی توسعه يافته، هزينه اش با يك پيچيده افزايش يافته راه‌حل‌شان پرداخت مي‌شود. طوري كه، اين نوع مدل‌ها اغلب به منظور شبيه ‌سازي بافت‌ها استفاده مي‌شود برخي از نتايج جديد نشان داده‌اند كه در حالت شبكه‌هاي پتري تصادفی گسترش یافته امكان آن وجود دارد كه در ساخت از مدل شبكه‌هاي پتري بنيادي استفاده كرد تا كاهش‌هاي قابل توجهي را در كاهش پيچيدگي راه‌حل به‌دست آورد[18]. علاوه بر آن،در مقاله[19]هندرسون و لوسیک در بحثي شرايطي كه تحت آن ها توزيع حالت پایدار[[35]](#footnote-36) بر نشانه‌ها تنها بستگي به تأخير فعالسازي متوسط يك انتقال دارد و وابسته به نوع توزيع نیست تعريف کردند. در بخش 2-1-3-2 در شبکه پتری تصادفی ‌به اين نتايج برمي‌گرديم.

#### روابط با مدل‌هاي ديگر

پيشنهادات مختلفي متفاوت از آنچه در اين بحث در نظر گرفته شده‌اند به منظور معرفي ويژگي‌هاي موقتي احتمالي در درون يك مدل معرفي شده‌اند و در محافل علمي منتشر شده است. دو جريان اصلي را مي‌توان شناسايي نمود :

* مدل‌هاي شبكه‌هاي پتري كه در آن‌ها زمان‌گذاري با مكان‌ها ارتباط دارد[20,21].
* مدل‌هاي شبكه‌هاي پتري كه در آن‌ها انتقال‌هاي زمان‌گذاري شده در سه زمان فعال مي‌شوند[22-25].



شکل ‏2‑13: نمایش مکانهای زمانبندی شده در چارچوب شبكه‌هاي پتري تصادفی عمومی

ما بدون در نظر گرفتن فرضيه‌ي برتري هر يك از مدلهاي شبكه‌هاي پتري زمان‌گذاري شده با توجه به بقيه توضيح مي دهيم كه چگونه معناي حالت بالا (شکل 2-13) را مي‌توان با نشانه‌ي شبكه‌هاي پتري تصادفی عمومی ارائه داد. هنگامي كه زمان‌گذاري با مكان‌ها مرتبط مي‌شود در حالت فعال شدن نشانه‌هاي انتقال فعال شده از مكان‌هاي ورودي خارج مي‌شوند و در خروجي با يك عمليات مستقل قرار مي‌گيرند. اگر چه، نشانه‌ها در مكان‌هاي خروجي انتقال قرار مي‌گيرند بلافاصله براي فعال كردن ساير انتقال‌ها آماده‌و و مهيا نمي‌باشند. فقط آن‌ها بعد تأخيري كه در ارتباط با مكان‌هاي گذشته دارند فعال مي‌شوند. پس امكان آن وجود دارد با جايگزيني هر مكان زمان‌گذاري شده با يك مكان استاندارد و يك انتقال زمان‌گذاري‌شده و يك انتقال استاندارد ديگر ، شبكه‌هاي پتري زماندار مرتبط با مكان‌ها را تبديل به يك فرمول شبكه‌هاي پتري تصادفی عمومی بكنيم. پيكان‌هاي ورودي مكان زمان‌گذاري شده بر اولين مكان غير زماني مرتبط هستند و پيكان‌هاي خروجي با مكان زمان‌گذاري شده از مكان غير زمان‌گذاري شده دومي جدا مي‌شوند. (طبق شكل 2-13)

توجه كنيد كه طول عمر نشانه‌هايي كه در مكان زمانگذاري قرار دارند به‌صورت موازي پيش مي‌روند به طوري كه انتقال نشان داده شده در شكل 2-13 مستلزم آنست كه سرعت فعالسازي انتقال زماندار به تعداد نشانه‌ها در مكان ورودي نسبي باشند (انتقال كه عبارت‌ صف‌بندي را قرض گرفته گفته مي‌شود از نوع سِروِر نامعین مي‌باشد).

زمان‌گذاري به انتقال‌هايي مربوط است كه در سه مرحله فعال مي‌شوند، يك انتقال فعال شده بلافاصله شروع به فعال شدن مي‌كند و نشانه‌ها را از مكان هاي ورودي خود حذف مي‌كند. نشانه‌ها، به هر حال، در مكانهاي خروجي انتقال قرار نمي‌گيرند تا زماني كه تأخير انتقال سپری شود. آنها براي كل تأخير انتقال در انتقال نگهداري مي‌شوند. به عبارت ديگر، فعال‌سازي يك انتقال از سه مرحله تشكيل شده است که عبارتست از:

1. يك مرحله شروع فعال‌سازي كه در آن نشانه‌ها از مكانهاي ورودي خارج مي‌شوند.

2. يك فعال‌سازي در مرحله پيشرفت، كه تأخير زماني به آن مرتبط است.

3. يك مرحله فعال‌سازي نهايي، كه در آن نشانه ها در درون مكانهاي خروجي مرتبط هستند.

انتقال يك شبكه‌هاي پتري زماندار با انتقال‌هايي كه در سه مرحله به فرمول شبكه‌هاي پتري تصادفی عمومی تبديل مي‌شوند، به راحتي با جايگزيني هر انتقال سه مرحله‌اي با انتقال بلافصل، يك مكان و يك انتقال زماندار صورت مي‌گيرد. پيكانهاي ورودي انتقال سه مرحله‌اي، به انتقال بلافصل مرتبطند و پيكانهاي خروجي انتقال سه مرحله‌اي از انتقال زماندار جدا مي‌شوند( مانند شكل 2-14).

عموماً، انتقال‌هاي زماندار سه مرحله‌اي مي‌توانند يك فعالسازي را از ابتدا آغاز كنند و اين در حالي است كه فعالسازي ديگر در حال اجرا شدن می باشد. انتقال زماندار در شكل 2-14، از نوع سرور نامعین مي‌باشد. هنگامي كه انتقال زماندار سه مرحله‌اي مي‌تواند تنها زماني كه يك فعالسازي قبلي تكميل نشده، فعالسازي جديد را آغاز كند (سرعت فعالسازي انتقال زماندار شكل 2-14 ثابت است گفته مي‌شود انتقال از نوع سِروِر واحد[[36]](#footnote-37) مي‌باشد). يك مانع مي تواند از آن به مكان انتقال بلافصل اضافه شود و اين در صورتي است كه در طي تاخير انتقال نشانه‌ها بايد در مكانهاي ورودي انتقال سه‌مرحله‌اي زماندار حفظ شوند.



شکل ‏2‑14: نمایش حالت زمانبندی شده سه مرحله ای در چارچوب شبكه‌هاي پتري تصادفی عمومی

بايد خاطر نشان ساخت كه در حالت انتقال‌هاي زماندار كه در سه مرحله فعال مي‌شوند، تضادها، به انتقال‌هايي محدود مي‌شوند كه همزمان فعال شده‌اند. در واقع، ممكن نيست كه انتقالاي كه فعال شده در زمان ديرتري با فعالسازي يك انتقال فعال شده قبلي، غير فعال گردد. به‌منظور شامل كردن اين ويژگي كه ممكن است در برخي از حالات براي مدلسازي سيستم‌ها مفيد باشد، مدلهاي شبكه‌هاي پتري زمانی كه در آنها انتقال در سه مرحله فعال مي‌شود، پيكانهاي خاصي كه انتقال خاصي را غير فعال[[37]](#footnote-38) مي‌كنند پيشنهاد شده‌اند[26].

#### تجزيه ساختاري شبكه‌هاي پتري تصادفی

همانطور كه در 2-2-3-1. گفته شد، شبكه‌هاي پتري تصادفی از ابتدا، توسط محققيني پيشنهاد شد كه در زمينه مدلسازي تصادفي كاربردي فعال بودند و آن را به‌عنوان يك نشان گرافيكي مناسب براي تعريف انتزاعي مدل‌هاي ماركوين معرفي كردند. بنابراين، احتمال تعريف سرعتهاي فعالسازي مستقل از علامتگذاري در شبكه‌هاي پتري تصادفی بر اين فرضيه بود كه تجزيه‌گر از تمام علائم قابل دسترس آگاهي داشته باشد تا بدرستي وابستگي علامتگذاري را مشخص کند.

حتی، تعریف ابتدایی شبكه‌هاي پتري تصادفی عمومی نیز موجود بوده که در آن انتخاب انتقال بلافصل که باید در یک علامتگذاری ناپدید شود فعال می شود، براساس يك توزيع راه گزيني كاربردي قرار دارد. كه، آن را مي‌توان با دانستن مجموعه قابل دسترسي بكه‌هاي پتري تصادفی عمومی تعريف نمود. با تعريف اصلاح شده شبكه‌هاي پتري تصادفی عمومی،تلاشي به منظور تلفيق ويژگي‌هاي ساختاري و اتفاقي مدل صورت گرفت طوري كه تعريف مدل در سطح شبكه و بدون نياز به ايجاد مجموعه قابل دسترس، ممكن شد.

يكي از مزاياي اصلي مدل شبكه‌هاي پتري تصادفی بنيادي، علاوه بر سادگي و سهولت استفاده، هويت قابل دسترس گراف آن به همراه يكي از شبكه‌هاي پتري بنيادي مي‌باشد. اين بدان معناست كه تمام نتايج موجود براي تجزيه ساختاري شبكه‌هاي انتقال را بتوان بسادگي در محيط شبكه‌هاي پتري تصادفی بكار برد.

شبكه‌هاي پتري تصادفی عمومی، در عوض، شبكه قابل دسترس، مشابه يكي از شبكه‌هاي انتقال بنيادي به همراه پيكانهاي مانع و اولویت‌ها مي‌باشند، يك مدل غير زماني كه هر گونه توجه كوچك را از تئوري‌پردازان شبكه دريافت مي‌دارد. خوشبختانه، وجود پيكانهاي مانع و اولویت‌ها، تنها مجموعه قابل دسترس را با توجه به يكي از شبكه‌هاي بنيادي اساسي كه از طريق حذف اولویت‌ها، پيكانهاي مانع و زمانگذاري بدست مي‌آيد محدود مي‌كند. اين بدان معناست كه برخي از ويژگي‌هاي ساختاري معتبر براي شبكه‌هاي پتري بنيادي اساسي توسط شبكه‌هاي پتري تصادفی عمومی بدست مي‌آيد. در واقع، محدودسازي شبكه قابل دسترس، ضمانت مي‌كند كه تمام متغيرهاي مكاني كه در مطالعه شبكه‌هاي پتري بنيادي اساسي بدست آمده، همچنان براي شبكه‌هاي پتري تصادفی عمومی وجود داشته باشند. در اصل، متغيرهاي مكاني ديگري موجود است كه در مورد شبكه‌هاي پتري تصادفی عمومی صدق مي‌كند اما، براي شبكه‌هاي پتري بنيادي اساسي معتبر نمي‌باشد. تجربه نشان داده است كه اين مسئله عموماً رخ نمي‌دهد زيرا، آنچه مربوط به متغيرهاي انتقال است يعني وجود پيكانهاي مانع و اولویت‌ها، ممكن است متغيري كه قابل فعالسازي در شبكه‌هاي پتري بنيادي اساسي است را غير محتمل كند. در حالتي كه در تابع شدت احتمال تأخير عمومي ممكن است مشكل تجزيه ساختاري مشكل‌تر باشد، زيرا شايد اين اتفاق بيفتد كه ويژگي‌هاي احتمالي اين مدل تأثيري بر رفتار كيفي داشته باشد تأثيري به اين صورت كه اگر چه دو انتقال همزمان فعال شوند، يكي از آنها نمي‌تواند به‌علت محدوديتهای زماني فعال گردد، يعني ويژگي‌هاي ساختاري شبكه را نمي‌توان كاملاً بررسي نمود حتي با در نظر گرفتن پيكانهاي مانع و اولویت‌هاي ممكن. بنابر این، زمانگذاري بايد به‌عنوان يك جز ساختاري محسوب گردد. يك شرط كافي به‌منظور اجتناب از اين تأثير نامطلوب ويژگي موقتي بر رفتار كيفي مدل اين است. تمام تابع شدت احتمال تأخير فعال سازي میزانی نامعین داشته باشند. به هر حال، اين محدوديت، استفاده از تأخيرهاي فعالسازي سريع را كه از ديدگاه مدلسازي اهميت خيلي دارد، ممكن نمي‌سازد. به هرحال، بايد تأكيد نمود كه امروزه ويژگي‌هاي ساختاري مدلهاي شبكه‌هاي پتري تصادفی در هر حالت براي تعريف مدل يا محاسبه نتايج نسبي استفاده مي‌شوند[27,28]. چيزي در مورد نقش ويژگي‌هاي ساختاري مدل بر كاهش پيچيدگي محاسبه pmf برعلامتگذاري‌هاي قابل دسترس وجود ندارد.

### خلاصه و نتیجه گیری

اكثر تحقيقات كنوني در زمينه شبكه‌هاي پتري تصادفی، به كاربرد شبكه‌هاي پتري تصادفی و مدلهاي مربوط به عملكرد و مطالعات معتبر محدوده وسيعي از سيستم‌ها می شود كه شامل زمینه های زیر می باشد که عبارتست از: معماري‌هاي سيستم‌هاي محاسبه توزيع شده، نرم‌افزار توزيع شده، پروتكل ارتباطي، سيستم‌هاي توليد،VLSI، پایگاه داده، فهرست و آمارهای منطقی، سيستم‌هاي شی گرا[[38]](#footnote-39) و سيستم‌هاي زمان واقعي[[39]](#footnote-40) .

در برخي از اين مطالعات كاربردي، جنبه تجزيه عملكرد با اثبات رسمي درست بودن سيستم تحت بررسي تلفيق شده كه اين كار را با بهره‌گيري از توصيف سيستم رسمي بدست آمده با فرمول شبكه‌هاي پتري صورت داده‌اند[29]. همانطور كه گفتيم، مشكل اساسي در استفاده از تكنيكهاي شبكه‌هاي پتري تصادفی براي تجزيه سيستم‌هاي واقعي از پيچيدگي راه‌حل سيستم نشأت مي‌گيرد. معمولاً اين مدلها شامل فضاهاي عظيمي هستند كه توليد مجموعه قابل دسترس را براي اجرا بسيار هزينه‌بر مي‌سازد (هم از لحاظ زماني و هم فضايي). چندين تحقيق به تلاش‌هايي به‌منظور كاهش هزينه راه‌حل اختصاص يافته است .

يك راه حل، استفاده از تكنيكهاي شبيه سازي مي‌باشد. با انجام اين كار، مشكلاتي كه از پيچيدگي فضايي ناشي مي‌شوند از بين مي‌روند. زيرا، ايجاد مجموعه قابل دسترس ديگر لازم نيست، اما، مشكلات مربوط به پيچيدگي زمان همچنان به قوّت خود باقي است (مثل مورد شبيه سازي). از سوی ديگر، هاس[[40]](#footnote-41) و شیلدر[[41]](#footnote-42)[12,30,31] نشان داده‌اند كه قدرت مدلسازي شبكه‌هاي پتري تصادفی در چهارچوب شبيه سازي بسيار برجسته‌است زيرا آنها معادل فرآيندهاي نيمه ـ ماركو عمومي هستند.

روش‌هاي ديگر را كه مي‌توان براي كاهش پيچيدگي راه‌حل مدل استفاده نمود که مربوط به استفاده از ويژگي‌هاي ساختاري مدلهاي شبكه‌هاي پتري براي ايجاد الگوريتم‌هاي راه‌حل مؤثر مي‌باشد. متغيرهاي مكاني براي بهبود اثربخشي فضايي ايجاد مجموعه قابل دسترسي مفيد مي‌باشد. ارتباط اتفاقي و حذف دو جانبه به همراه متغيرها را مي‌توان براي شناسايي زير مدلهايي بكار برد كه در حالات خاص قابل مطالعه به‌صورت مجزا مي‌باشند و نتايجی ارائه مي‌دهد كه تركيب بعدي آنها، راه‌حل كلی مطلوبي را ارائه مي‌دهند. تلاش به‌منظور استفاده از ويژگي‌هاي ساختاري مدلهاي شبكه‌هاي پتري بنيادي در فازهاي مختلف يك مطالعه عملكردي با استفاده از متغيرهاي مكان و انتقال می باشد. مدل و فرآيند اشکال زدائی[[42]](#footnote-43) مدل توصیف شده است[32]. استفاده از متغيرهاي مكاني، ارتباط اتفاقي[[43]](#footnote-44)، اجراي دوجانبه[[44]](#footnote-45) براي بررسي اينست كه آيا مدل را مي‌توان در كلاسي خاص طبقه‌بندي نمود؟ استفاده از متغيرهاي انتقال براي شناسايي راه‌‌هاي باريك، استفاده از تضادها به‌منظور سهولت تعريف مدل شبكه‌هاي پتري تصادفی عمومی می باشد. پيچيدگي گرافيكي مدلهاي شبكه‌هاي پتري تصادفی را مي‌توان در صورت استفاده از توصيف سطح بالا كاهش داد اما، تحقيقات به‌منظور دستيابي به تعادل كافي حداكثر ميان قدرت مدلسازي و سهولت استفاده ادامه دارد.

در نهايت، برخي از محققان در حال بررسي احتمال تركيب شبكه‌هاي پتري تصادفی با صف‌ها هستند تا روش ارزيابي عملكرد تلفيقي را بدست آورند كه از ويژگي‌هاي مطلوب هر دو روش استفاده كند .

هدف اين بحث، معرفي شبكه‌هاي پتري تصادفي براي متخصصين شبكه پتري مي‌باشد كه با رشته مدلسازي عملكرد تصادفي آشنايي ندارند. بدين علت، يك مرور مختصر از رشته تجزيه عملكرد، از فرآيندهاي ماركوين و از مدلهاي صف بندي ارائه شد. ويژگي‌هاي مدلهاي شبكه‌هاي پتري تصادفی و انگيزه‌‌اي كه به تعريف آنها منجر مي‌شود توصيف شد. برخي از نمونه‌ها، همچنين براي توضيح كاربرد احتمالي بیان شدو از روش شبكه‌هاي پتري تصادفی براي تجزيه عملكرد سيستم‌هاي استفاده شد. خلاصه مختصري از فعاليتهاي تحقيقاتي موجود و بارزترين مشكلات ارائه شده است. با اين اميد كه برخي از خوانندگان به رشته شبكه‌هاي پتري تصادفی علاقه‌مند شوند و نتايج جديدي كه كاربرد روش شبكه‌هاي پتري تصادفی را گسترش مي‌دهد، بدست آورند [33, 34].

## اتوماتای يادگير

### مقدمه

فرآيندي که طي آن، موجودات زنده مطالب مختلف را ياد مي‌گيرند، مدت‌هاي مديدي مورد علاقه متخصصان بوده است. تحقيقاتي که در اين زمينه انجام شده است، در دو شاخه‌ی کلي متمرکز مي‌باشد:

1. درک فرآيندي که موجودات زنده در طي آن، عمل به يادگيري نمايند.
2. بدست آوردن روش‌هايي که با استفاده از آنها بتوان اين قابليت يادگيري را در ماشين‌ها ايجاد نمود.

يادگيري به روش‌هاي مختلفي تعريف شده است. بنابراين تعريف يادگيري، اِعمال يك سري تغييرات در کارايي سيستم بر اساس تجارب قبلي مي‌باشد. مهم‌ترين مشخصه‌ی اين سيستم يادگير، قابليت آن در افزايش کارايي در طول زمان مي‌باشد. اگر بخواهيم کاملاً به صورت رياضي اين تعريف را بيان نماييم، مي‌گوييم: هدف يك سيستم يادگير، بهينه‌سازي يك عملكرد مي‌باشد که امکان شناسايي آن به طور کامل وجود ندارد. با اين تعريف مي‌توان وظيفه‌ی سيستم يادگير را به يك مسئله‌ی بهينه‌سازي که بر روي يك مجموعه از پارامترها تعريف شده است کاهش داد و با بدست آوردن مجموعه‌ی پارامترهاي بهينه، از طريق روش‌هاي موجود آن را حل نمود. در اين بخش روش‌هاي يادگير که براساس اتوماتاهاي يادگير تصادفي[[45]](#footnote-46) (SLA) مي‌باشند، بررسي مي‌شود.

مفهوم اتوماتاي تصادفي اولين بار توسط گروه تحقيقاتي تستلين[[46]](#footnote-47) در اوايل سال 1960 در اتحاد جماهير شوروي مطرح گرديد. پس از آن در تحقيقات بعدي، نمونه‌هاي مختلفي از کاربرد روش‌هاي يادگيري در سيستم‌هاي مهندسي بوجود آمد که از جمله‌ی آنها مي توان کاربرد در مسيريابي تلفن،‌ شناسايي الگو، تقسيم‌بندي اشياء[[47]](#footnote-48) و کنترل قابل تطبيق[[48]](#footnote-49) را نام برد.

اتوماتاي يادگير[[49]](#footnote-50) [35, 36] را مي‌توان يک شيء مجرد در نظر گرفت که داراي تعداد محدودي عمل[[50]](#footnote-51) مي‌باشد. عملکرد اين شيء به اين صورت است که در هر زمان، يک عمل از بين مجموعه‌ی اَعمال انتخاب مي‌گردد و سپس در يک محيط تصادفي مورد ارزيابي قرار مي‌گيرد. جوابي که از محيط دريافت مي‌شود، توسط اتوماتا براي انتخاب عمل بعدي به کار گرفته مي‌شود و به اين ترتيب اتوماتا به تدريج، عمل بهينه را شناسايي مي‌نمايد. روشي که اتوماتا با استفاده از آن پاسخ محيط را براي انتخاب عمل بعدي به کار مي‌برد، توسط الگوريتم يادگيري که استفاده شده است، تعيين مي‌گردد. در بخش‌هاي بعدي جزئيات بيشتري در مورد اجزاي SLA مطرح خواهيم نمود.

### تاريخچه‌ی اتوماتاي يادگير

در دهه‌ی 1960 ميلادي، سيپکين[[51]](#footnote-52) روشي براي ساده كردن مسائل فوق به مسأله‌اي براي تشخيص پارامترهاي بهينه و به كار بردن روش‌هاي تپه نوردی براي حل آن معرفي كرد[37]. ستلين و همكارانش[38] در همان زمان، كار بر روي اتوماتاي يادگير را آغاز كردند. مفهوم اتوماتاي يادگير، نخستين بار توسط وي مطرح شد. وی به مدل‌سازي رفتارهاي سيستم‌هاي بيولوژيكي علاقمند بود و يك اتوماتاي قطعي را كه در محيطي تصادفي فعاليت مي‌كرد، به عنوان مدلي براي يادگيري، معرفي نمود. در تحقيقات انجام شده‌ی بعدي، استفاده از يادگيري در سيستم‌هاي مهندسي نيز در نظر گرفته شد. رويكرد ديگري كه توسط نارندرا[[52]](#footnote-53) و ویسواناتان[[53]](#footnote-54) ارائه شد[39]، در نظر گرفتن مسأله به صورت پيدا كردن عمل بهينه از ميان مجموعه‌اي از عمل‌هاي مجاز برای يك اتوماتاي تصادفي بود. تفاوت دو روش اخير در اين است كه در روش اول در هر تكرار، فضاي پارامترها بِروز مي‌شود، اما در روش دوم، فضاي احتمال بِروز مي‌گردد. پس از آن، اكثر كارهاي انجام شده در تئوري اتوماتاي يادگير، در ادامه‌ی مسير معرفي شده، توسط ستلين انجام شدند. وارشاوسکی[[54]](#footnote-55) و وورونتسوا[[55]](#footnote-56) [40]اتوماتاي يادگير با ساختار متغير را مطرح كردند كه احتمالات عمل‌هاي[[56]](#footnote-57) خود را بِروز مي‌كرد و در نتيجه باعث كاهش تعداد حالات[[57]](#footnote-58)، نسبت به اتوماتاي قطعي مي‌شد.

نخستين تلاش‌ها براي استفاده از اتوماتاي يادگير در كاربردهاي كنترلي، توسط فو[[58]](#footnote-59) و همكارانش انجام شد[41]. از جمله‌ی اين كارها مي‌توان به كاربردهايي از اتوماتاي يادگير در تخمين پارامترها[[59]](#footnote-60)، تشخيص الگو[[60]](#footnote-61) و تئوري بازي[[61]](#footnote-62) اشاره كرد[42-44]. مک لارن[[62]](#footnote-63) روش‌هاي بِروزرساني خطي و ويژگي‌هاي آنها را بررسي نمود[45] و پس از آن چاندراسِکار[[63]](#footnote-64) و شِن[[64]](#footnote-65) روش‌هاي بِروزرساني غيرخطي را بررسي كردند[46]. كتاب نارندرا و تاتاچار[[65]](#footnote-66) تحت عنوان اتوماتای یادگیر ، مقدمه‌اي به تئوري اتوماتا است كه كليه‌ی تحقيقات انجام شده تا اواخر دهه‌ی 1980 را در بر مي‌گيرد[47] مثال‌ها و كاربردهاي متعدد ديگري از اتوماتاي يادگير نيز توسط نَجیم[[66]](#footnote-67) و پوژنیاک[[67]](#footnote-68) در كتابي با عنوان"اتوماتای یادگیر: تئوری و کاربردها "[[68]](#footnote-69) ارائه شده است[48] .

نتايج اين تحقيقات، به استفاده از اتوماتا به عنوان مدلي از يادگيري با كاربرد‌هايي در مسير‌يابي شبكه‌‌هاي تلفن، تشخيص الگو و كنترل تطبيق‌پذير منجر شد[49-52]. علاوه بر اين، از كاربردهاي اخير اتوماتاي يادگير در مسائل واقعي مي‌توان به تشخیص الگو[53] جداسازی گراف[54][[69]](#footnote-70) و برنامه‌ريزي مسير[55][[70]](#footnote-71) اشاره كرد. دیگر کاربردهای اتوماتای یادگیر در[56-60] ارائه شده اند.

يك اتوماتاي يادگير را مي‌توان به صورت يك شیء مجرد كه داراي تعداد متناهي عمل است، در نظر گرفت. نحوه‌ی کار يک اتوماتاي يادگير بدين صورت است که يك عمل از مجموعه‌ی عمل‌هاي خود را انتخاب کرده وآن را به محيط اِعمال می‌کند. سپس عمل مذكور توسط يك محيط تصادفي ارزيابي مي‌شود و اتوماتا از پاسخ محيط براي انتخاب عمل بعدي خود استفاده مي‌كند. در طي اين فرآيند، اتوماتا ياد مي‌گيرد كه چگونه عمل بهينه را انتخاب نمايد. نحوه‌ی استفاده از پاسخ محيط به عمل انتخابي اتوماتا كه در جهت انتخاب عمل بعدي اتوماتا استفاده مي‌شود، توسط الگوريتم يادگيري اتوماتا مشخص مي‌گردد. در بخش بعد جزئيات قسمت‌هاي يك اتوماتاي با ساختار متغير[[71]](#footnote-72) معرفي مي‌شود.

### اتوماتاي يادگير تصادفي (SLA)

هر SLA از دو جزء عمده تشكيل شده است.

1. يک اتوماتاي تصادفي که داراي تعداد محدودي عمل مي‌باشد و با يک محيط تصادفي در حال تعامل مي‌باشد.
2. الگوريتم يادگيري که با استفاده از آن، اتوماتا عمل بهينه را یاد مي‌گیرد.

#### اتوماتاي تصادفي

هر اتوماتا را مي‌توان يک ماشين حالت متناهي[[72]](#footnote-73) در نظر گرفت که به وسيله‌ی پنج‌تايي زير قابل نمايش است:

|  |  |
| --- | --- |
| رابطه‌ی (2-12) |  |

پارامترهای موجود در اين پنج‌تايي عبارتند از:

* مجموعه‌ی عمل‌هاي اتوماتا 
* مجموعه‌ی ورودي‌هاي اتوماتا
* تابعي که ورودي و حالت جاري را به حالت بعدي نگاشت مي‌کند
* تابع خروجي که حالت جاري را به خروجي بعدي نگاشت مي‌کند
* مجموعه‌ی حالت‌هاي داخلي اتوماتا در لحظه‌ی -nام 

مجموعه‌ی شامل اعمال اتوماتا مي‌باشد که اتوماتا در هر بار تکرار، يكي از آنها را انتخاب مي‌کند. مجموعه‌ی ورودي، ورودي‌هاي اتوماتا را تعريف مي‌کند که جزئيات بيشتر مربوط به آن را در بخش بعدي بيان خواهيم کرد. نگاشت‌هاي و حالت جاري و ورودي فعلي را به خروجي (عمل) بعدي که توسط اتوماتا انتخاب شده است تبديل مي‌کنند. حال اگر نگاشت‌هاي و قطعي باشند، اتوماتا نيز اتوماتاي قطعي[[73]](#footnote-74) گفته مي‌شود. در چنين حالتي با در دست داشتن حالت اوليه و ورودي، خروجي و حالت بعدي به صورت منحصر به فردي بدست مي‌آيد. حال اگر نگاشت‌هاي و تصادفي باشند، اتوماتا نيز اتوماتاي تصادفي گفته مي‌شود. در چنين حالتي فقط مي‌توان احتمالات مربوط به حالت بعدي و نيز خروجي‌هاي مربوطه را مشخص نمود. اتوماتاي تصادفي خود به دو دسته تقسيم مي‌شود:

1. اتوماتاي با ساختار ثابت[[74]](#footnote-75)
2. اتوماتای با ساختار متغير[[75]](#footnote-76)

در نوع اول، احتمالات مربوط به عمل‌هاي مختلف ثابت هستند؛ اما در نوع دوم احتمالات در هر بار تکرار بهنگام مي‌شوند. تعريف و انواع مختلف اتوماتای احتمالي با ساختار ثابت و متغير در ادامه آمده است.

#### محيط

محيط احتمالي را مي‌توان به صورت رياضي با سه‌تايي E بيان نمود:

|  |  |
| --- | --- |
| رابطه‌ی (2-13) |  |

 مجموعه‌ی ورودي‌هاي محيط و  مجموعه‌ی خروجي‌هاي آن مي‌باشد. مدل‌هاي مختلفي براي محيط‌هاي احتمالي تعريف شده است. در P-Model، محيط مقادير صفر و يك را به عنوان خروجي اختيار مي‌كند. صفر به معني پاداش[[76]](#footnote-77) و يك به‌ معني جريمه[[77]](#footnote-78) مي­باشد. در Q-Model، مقادير خروجي به صورت اعداد گسسته‌ی بين صفر و يك هستند و در S-Model، خروجي محيط، مقدار پيوسته­اي بين صفر و يك مي­باشد.  نيز مجموعه‌ی احتمال جريمه‌شدن عمل‌هاي اتوماتاي احتمالي است كه به صورت زير تعريف مي­شود:

|  |  |
| --- | --- |
| رابطه‌ی (2-14) |  |

 يعني احتمال آنكه عمل نتيجه‌ی نامطلوب داشته باشد. در محيط‌هاي پايدار[[78]](#footnote-79) مقادير به صورت ثابت باقي مي­ماند؛ در حالی که در محيط‌هاي ناپايدار[[79]](#footnote-80) اين مقادير در طي زمان تغيير مي­كنند. ها معمولاً مشخص نيستند و دانستن آنها به معني شناخت كامل محيط احتمالي است كه در اكثر كاربردها امكان ندارد و هدف اتوماتاي يادگير، مشخص كردن اين مقادير است.

رابطه‌ی بين محيط احتمالي و يك اتوماتای احتمالي در شکل 2-15 نشان داده شده است.

**محیط**

**اتوماتای تصادفی**

{α}

{β}

شکل ‏2‑15: رابطه‌ی بين اتوماتا و محيط احتمالي

بنابراين در نهايت SLA را مي‌توان توسط پنج‌تايي زير نمايش داد.

|  |  |
| --- | --- |
| رابطه‌ی (2-15) |  |

که در آن داريم:

* مجموعه‌ی خروجي‌هاي اتوماتا/ مجموعه‌ی ورودي‌هاي محيط 
* مجموعه‌ی ورودي‌هاي اتوماتا/ مجموعه‌ی خروجي‌هاي محيط
* بردار احتمالات 
* الگوريتم يادگير 
* مجموعه‌ی احتمالات جريمه‌ی تعريف شده براي محيط

همان طور که قبلاً بيان شد، جهت سادگي رياضي، هر يك از حالت‌هاي داخلي اتوماتا را بر يك عمل منحصر به فرد از اتوماتا منطبق مي‌نمايند.

### الگوريتم‌هاي يادگيري

#### الگوريتم‌هاي يادگيري استاندارد

يک الگوريتم يادگيري را مي‌توان به صورت زير بيان نمود:

|  |  |
| --- | --- |
| رابطه‌ی (2-16) |  |

اگر عملگر T خطي باشد، الگوريتم يادگيري خطي و در غير اين صورت غيرخطي خوانده مي‌شود. مي‌توان گفت که ايده‌ی اصلي تمام الگوريتم‌هاي يادگيری به صورت زير است: اگر SLA در تکرار -nام عمل را انتخاب نمايد و پاسخ مطلوبي براي آن از محيط دريافت نمايد، در آن صورت احتمال مربوط به عمل افزايش يافته و احتمالات مربوط به ساير اعمال کاهش مي‌يابد. در صورت دريافت جواب نامطلوب، کاهش يافته و احتمالات مربوط به ساير اعمال افزايش مي‌يابد. بنابراين داريم:

براي پاسخ مطلوب[[80]](#footnote-81):

|  |  |
| --- | --- |
| رابطه‌ی (2-17) |  |

براي پاسخ نامطلوب[[81]](#footnote-82):

|  |  |
| --- | --- |
| رابطه‌ی (2-18) |  |

توابع  و  توابعي غير منفي مي‌باشندکه به ترتیب توابع پاداش و جريمه نامیده مي‌شود. هم اکنون مطالعات بسياري بر روي الگوريتم‌هاي يادگير خطي به علت قابليت انعطاف بالاي آنها انجام مي‌شود. براي اين نوع از الگوريتم‌ها، توابع پاداش و جريمه به صورت زير تعريف مي‌شوند[61]:

|  |  |
| --- | --- |
| رابطه‌ی (2-19) |  |
| رابطه‌ی (2-20) |  |

در روابط فوق داريم:

 تعداد اعمال يک اتوماتا

 پارامتر پاداش

 پارامتر جريمه

براساس مقادير مختلفي که براي پارامتر‌هاي يادگيري a و b در نظر گرفته مي‌شود، الگوريتم‌هاي مختلفي بدست مي‌آيد که در زير مواردي از آنها را نام مي‌بريم :

روش: اگر a و b مساوي باشند، روش جريمه‌ی پاداش خطي[[82]](#footnote-83) حاصل مي‌شود.

روش : اگر b چندين برابر از a کوچکتر باشد، روش يادگيري حاصل، روش جريمه‌ی جزئي پاداش خطي[[83]](#footnote-84) خواهد بود.

روش : اگر پارامتر جريمه (b) صفر باشد، روش حاصل، روش عدم پاداش خطي[[84]](#footnote-85) ناميده مي‌شود.

اگر رابطه­ی (2-19) و رابطه­ی (2-20) براي توابع و به کار برده شوند، شكل عمومي الگوريتم يادگيری به صورت زير خواهد بود:

اگر در تکرار n عمل انتخاب شود، در اين صورت در تکرار (n+1) داريم:

دريافت پاسخ مطلوب از محيط:

|  |  |
| --- | --- |
| رابطه‌ی (2-21) |  |
| رابطه‌ی (2-22) |  |

دريافت پاسخ نامطلوب از محيط:

|  |  |
| --- | --- |
| رابطه‌ی (2-23) |  |
| رابطه‌ی (2-24) |  |

اگر در معادلات فوق a و b برابر باشند، روش بدست مي‌آيد و اگر b = 0 باشد، روش  حاصل می‌گردد و اگر  اتوماتای یادگیر حاصل می شود. لازم است در اينجا يادآور شويم که روش‌هاي بهنگام‌سازي غيرخطي نيز توسط محققين دنبال مي شود؛ اما استفاده از اين روش‌ها منجر به افزايش قابل ملاحظه‌اي در کارايي نسبت به روش‌هاي خطي نشده است. مهم‌ترين عاملي که منجر به محدوديت کاربرد SLA مي‌شود، نرخ کند همگرايي آن مي‌باشد. اين امر به خصوص زماني كه تعداد عمل‌ها زياد باشد و SLA در هر بار تکرار بايد تعداد اعمال زيادي را بهنگام نمايد، بيشتر احساس مي‌شود. با توجه به اين مشكل به بيان الگوريتم‌هاي غيرخطي نخواهيم پرداخت.

#### الگوريتم‌هاي يادگيري مدل S

در محيطهاي S پاسخ محيط به عملهاي اتوماتاي يادگير، يک متغير تصادفي در بازه [1،0] مي‌باشد. بنابراين خروجي محيط (ورودي اتوماتا) بصورت زير تغيير مي‌کند:

|  |  |
| --- | --- |
| رابطه­ی(‏2‑25) |  |

از آنجاييکه پاسخ محيط از نوع S يک متغير تصادفي در بازه [1،0] است، استفاده از مدلS براي سيستمهاي يادگير نياز به اطلاعات اوليه‌اي از کران بالا و پايين شاخصهاي کارايي سيستم دارد تا بتواند پاسخهاي محيط را در مقياس [1،0] تنظيم نمايد. کارايي مناسب مدلS در [62] نشان داده شده است. در[63] يک الگوريتم غير خطي اما بهينه براي يک اتوماتا با دو عمل در محيط مدل S معرفي شده است. يک روش زيربهينه براي يک اتوماتا با چند عمل در محيط مدل S نيز در[64] معرفي شده است.

##### **الگوریتم**

در محيط مدلP محيط توسط احتمال جريمه‌ها تعريف مي‌شود. براي هر عمل مانند ، محيط پاسخي با يک مقدار تصادفي  به اتوماتا مي‌دهد که ورودي اتوماتا را تشکيل مي‌دهد. در مدل P، پاسخ  با احتمال  برابر 1 (پاسخ نامطلوب) و با احتمال -1 صفر(پاسخ مطلوب) در نظر گرفته مي‌شود. اما در محيط مدل**S**، محيط بصورت زير تعريف مي‌شود:

|  |  |
| --- | --- |
|  |  |
| رابطه­ی(‏2‑26) |  |

 مقدار ميانگين پاسخ  براي عمل  مي‌باشد و در واقع ها بعنوان شدت جريمه‌ها محسوب مي‌شوند. فرض کنيد در گام nاُم عمل  انتخاب شده باشد و پاسخ محيط به آن  بوده باشد. در چنين حالتي احتمال عملهاي اتوماتاي  بصورت زير بروز مي‌شوند]47[ :

|  |  |
| --- | --- |
|  | رابطه­ی(‏2‑27) |

که ***a*** پارامتر يادگيري و 1<***a***<0 مي‌باشد.

##### **الگوريتم**

الگوريتم يادگيري تقويتي اتوماتاي يادگير در مدل  بصورت زير بردار احتمالات عملهاي اتوماتا را بروز مي‌کند[47]. در اتوماتايي با *r* عمل، اگر در تکرار *n*اُم عمل  انتخاب شده باشد و پاسخ محيط به آن  باشد، بردار احتمالهاي اتوماتا طبق رابطه زیر بروز مي‌شود.

|  |  |
| --- | --- |
|  | رابطه­ی(‏2‑28) |

##### **الگوريتم**

اتوماتاي يادگيري  با r عمل و پارامتر پاداش ***a*** و پارامتر جريمه ***b***، بصورت زير بردار عملهاي خود را بروز مي‌کند. اگر در تکرار *n*اُم عمل  انتخاب شده باشد و پاسخ محيط به آن  باشد، بردار احتمالهاي اتوماتا طبق رابطه زیر بروز مي‌شود.

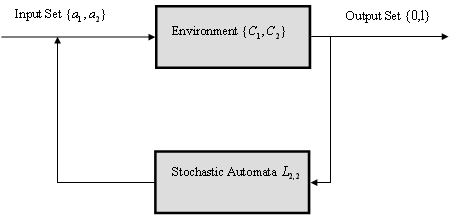
|  |  |
| --- | --- |
|  | رابطه­ی(‏2‑29) |

### الگوريتم‌هاي يادگيری با ساختار ثابت

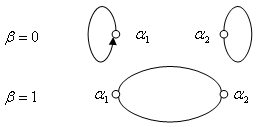
الگوريتم‌هاي يادگيری را مي‌توان برحسب نوع بردار احتمالات نيز تقسيم‌بندي نمود. بر اساس اين نوع از تقسيم‌بندي، اگر بردار احتمالات در طول مدت زمان اجرا ثابت باشد، اتوماتا از نوع ساختار ثابت خواهد بود و اگر اين بردار در طول زمان اجرا تغيير نمايد، اتوماتای با ساختار متغير حاصل خواهد شد. در هر دو نوع اتوماتاي ذکر شده، احتمالات در بازه‌ی [1و0] قرار دارند. الگوريتم‌هاي يادگيری که پيش از اين در مورد آن بحث گرديد، مربوط به اتوماتای از نوع ساختار متغير مي‌باشند. در ادامه به بررسي انواع ديگري از اتوماتاي يادگير با ساختار ثابت مي‌پردازيم و تعدادي از بهترين ايده‌هاي معرفي شده در سال 1960 را معرفي مي‌نماييم.

#### اتوماتاي دو حالته

اين اتوماتا داراي دو حالت  و  و دو خروجي  و  مي‌باشد(شکل 2-16). ورودي‌هاي پذيرفته شده توسط اتوماتا مجموعه‌ی {1و0} می‌باشد. اتوماتا هرگاه ورودي 1 را دريافت نمايد (پاسخ نامطلوب) حالت خود را عوض مي‌کند و در صورت دريافت ورودي 0 (پاسخ مطلوب) در حالت قبلي باقي مي‌ماند.



شکل ‏2‑16: نمودار عملكرد اتوماتاي دو حالته 



شکل ‏2‑17: گراف انتقال حالت اتوماتاي 

در نتيجه، ماتريس انتقال[[85]](#footnote-86) اين اتوماتا به صورت زير مي‌باشد که گراف مرتبط با آن نيز در شکل 2-17 نمايش داده شده است. هر اتوماتايي که منطبق بر استراتژي فوق باشد  ناميده مي‌شود. در اينجا انديس اول، تعداد حالت‌ها و انديس دوم نيز تعداد اَعمال را مشخص مي‌نمايد. خصوصيات محيط، توسط مجموعه‌ی احتمالات جريمه  بيان مي‌گردد. در اينجا هر  بيانگر احتمال دريافت پاسخ  براي ورودي  است.

|  |  |
| --- | --- |
| رابطه‌ی (2-30) |  |

استراتژي اتوماتا به شكلي طراحي شده است که تا زمان دريافت پاسخ مطلوب از محيط، در يك عمل باقي مي‌ماند و به محض آنکه پاسخ نامطلوبي دريافت شود به سوي عمل ديگر حرکت مي‌نمايد.

#### توسعه‌هاي اتوماتاي دوحالته

از آنجايي که تمامي سطرهاي  فقط داراي مقدار صفر و يا يک هستند و در هر سطر نيز يک عنصر منفرد وجود دارد ماتريس انتقال موجود دررابطه­ی (2-30) قطعي مي‌باشد. حال اين سوال مطرح مي شود که آيا همگرايي اتوماتا را مي توان با تصادفي نمودن ماتريس‌هاي انتقال به صورت زير بهبود بخشيد؟

|  |  |
| --- | --- |
| رابطه‌ی (2-31) |  |

در اين مورد هنگامي كه يك عمل منجر به پاسخ مطلوب گردد، اتوماتا از همان عمل با احتمال  استفاده نموده و به عمل ديگر با احتمال  سوئيچ مي‌نمايد. همچنين هنگامي که خروجي محيط نامطلوب باشد، اين عمل را با احتمالات  و انجام مي‌دهد و مي‌توان اثبات نمود که اين نوع از اتوماتا، قطعي می‌باشد.

#### اتوماتاي حافظه‌دار با دو عمل

اتوماتاي عموماً نمی‌تواند بهينه باشد. به همين دليل، تستلين نوع تغييريافته‌اي از اتوماتاي  را پيشنهاد نموده است که اگر احتمالات جريمه‌ی محيط داراي شرايط خاص باشند، مي‌تواند بهينه گردد. چنين اتوماتايي  ناميده مي‌شود و داراي 2N حالت و دو عمل مي‌باشد. اين نوع از اتوماتا سعی می‌کند که رفتار گذشته‌ی سيستم را جهت تصميم‌گيري در مورد چگونگي انتقال از اعمال، مورد استفاده قرار دهد. اتوماتاي هنگامي كه يك جواب نامطلوب از محيط دريافت نمايد، از عملي به عمل ديگر سوئيچ مي‌نمايد؛ در حالي که اتوماتاي اقدام به نگهداري تعداد موفقيت‌ها[[86]](#footnote-87) و شكست‌هاي[[87]](#footnote-88) دريافت شده به ازاي هر عمل مي‌نمايد و فقط زماني که تعداد شكست‌ها بيش از تعداد موفقيت‌ها گردد (يا از يك مقدار حداکثرN تجاوز نمايد) به عمل ديگر سوئيچ مي‌نمايد. اين کار با توسعه دادن فضاي حالت به 2N و تعيين قواعد انتقال از يك حالت به حالت ديگر صورت مي‌پذيرد.

اتوماتاي  داراي 2N حالت  و دو عمل و مي‌باشد.

حالت‌هاي مربوط به عمل و حالات  نيز متعلق به عمل  مي‌باشند.

بنابراين:

|  |  |
| --- | --- |
| رابطه‌ی (2-32) |  |

که در آن  تابع نگاشت خروجي مي‌باشد.

اگر اتوماتا در يک حالت  باشد، عمل  را انجام مي‌دهد. حال اگر اين عمل منجر به پاسخي نامطلوب شود، حالت به صورت زير تغيير مي‌کند:

|  |  |
| --- | --- |
| رابطه‌ی (2-33) |  |

و

|  |  |
| --- | --- |
| رابطه‌ی (2-34) |  |

به صورت مشابه اگر عمل انتخابي منجر به پاسخ مطلوب شود، انتقال حالت‌ها به صورت زير صورت مي گيرد:

|  |  |
| --- | --- |
| رابطه‌ی (2-35) |  |

و

|  |  |
| --- | --- |
| رابطه‌ی (2-36) |  |

به شكلي مشابه مي‌توان انتقال حالت را براي هنگامي كه و عمل در حال انجام است به شكل زير بيان نمود:

 منجر به يک پاسخ نامطلوب شده است. 

 منجر به يک پاسخ مطلوب شده است. 

گراف‌هاي انتقال حالت در شکل 2-18 نمايش داده شده‌اند و ماتريس‌هاي انتقال آنها نيز توسط  و  بيان شده است.



پاسخ مطلوب 



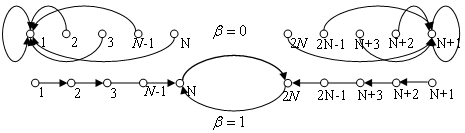
پاسخ نا مطلوب 

شکل ‏2‑18: گراف‌هاي انتقال حالت اتوماتاي

هر عمل با N حافظه مرتبط مي‌باشد و در مجموع، اتوماتا داراي 2N حافظه است. هنگام دريافت هر پاسخ مطلوب، حالت اتوماتا به حالت عميق‌تري در حافظه تغيير کرده و اگر پاسخ نامطلوبي دريافت گردد، حالت آن به عمق كمتري در حافظه‌ی اتوماتا تغيير مي‌يابد. عمق يك حالتدر ارتباط با عمل را مي‌توان به صورت قراردادي به شكل زير تعريف نمود: حداقل طولي از سيگنال‌هاي ورودي که را از مجموعه‌ی حالت‌هاي متعلق به عمل () بيرون مي‌آورد. همچنين عمق يك اتوماتا عبارت است از بزرگترين عمق حالت‌هاي آن که در اين مثال N حالت می‌باشد.

#### اتوماتاي کرينسکي

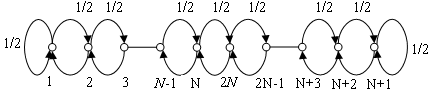
نمودار انتقال حالت نمايش داده شده در شکل 2-19 نشان می‌دهد که رفتار اتوماتاي کرينسکي براي پاسخ نامطلوب از محيط همانند  می‌باشد. به هنگام دريافت پاسخ مطلوب، هر حالت  به حالت  رفته و حالت  نيز به حالت  مي‌رود. در مجموع، برای رفتن از يك عمل به عمل ديگر نياز به دريافت N پاسخ نامطلوب متوالي دارد.



شکل ‏2‑19: گراف‌هاي انتقال حالت اتوماتاي کرينسکي

#### اتوماتاي کرايلوف

اين اتوماتا براي پاسخ مطلوب محيط، نمودار تغيير حالتي مانند اتوماتای دارد. زماني كه پاسخ محيط نامطلوب است، اتوماتا با احتمال 0.5 از حالت به حالت‌هاي يا  مي‌رود (شکل 2-20). همچنين زماني كهيا باشد، حالت بعدي اتوماتا، با احتمال 0.5، همان حالت يا حالت خواهد بود. وقتي كه، با احتمال 0.5 حالت بعدي،يامي‌باشد و هنگامي كه ، با احتمال 0.5 به يا تغيير حالت مي‌دهد.



شکل ‏2‑20: انتقال حالات اتوماتاي کرايلوف هنگام دريافت پاسخ نامطلوب از محيط

#### اتوماتاي

در اين اتوماتا بر خلاف، عمل حداقل N بار انجام مي­گردد (پس از گرفتن N جريمه) تا اينكه در نهايت عمل دوباره انتخاب شود. گراف تغيير وضعيت اين اتوماتا براي پاسخ مطلوب مانند اتوماتاي بوده و براي پاسخ نامطلوب مطابق شکل 2-21 می باشد.



|  |  |  |
| --- | --- | --- |
| G2N,2 | پاسخ نا مطلوب |  |

شکل ‏2‑21: نمودار تغيير وضعيت اتوماتا

#### اتوماتاي مهاجرت اشياء*[[88]](#footnote-89)*

نمودار تغيير وضعيت در اين اتوماتا، براي پاسخ مطلوب مانند اتوماتاي بوده و براي پاسخ نامطلوب مطابق شکل 2-22مي‌باشد.



پاسخ نا مطلوب 

شکل ‏2‑22: نمودار تغيير وضعيت اتوماتاي مهاجرت اشياء

### بازي‌هاي اتوماتا

در مسائل مختلف اجتماعي و اقتصادي که تداخل علايق در بين عوامل تصميم‌گيري نقش مهمي را در تصميم‌گيري نهايي بازي مي‌کند، تئوري بازي‌ها نقشي بسيار عمده ايفا مي‌نمايد. در زماني كه هر دو طرف، عملي را انتخاب مي‌کنند و به ازاي آن جوابي از محيط دريافت مي‌دارند، مي‌گوييم که يک بازي در حال انجام است. هر يك از بازيگران مي‌تواند اطلاعات کاملي از طرف مقابل و گزينه‌‌هاي او داشته يا نداشته باشد و هر بازيگر، حرکت بعدي خود را بر پايه‌ی جواب بدست آمده از محيط استوار مي‌سازد.

در يك بازي اتوماتا، تعدادي اتوماتا در يك محيط، بدون آنکه دانش کاملي از ديگران در اختيار داشته باشند عمل مي‌کنند. هر يك از آنها مي‌تواند داراي تعداد محدودي اَعمال و قواعد متفاوت با ديگران باشد. براي بازي‌هاي اتوماتا يك قاعده‌ی عمومي به صورت زير مي‌توان بيان نمود: اگر N اتوماتا در بازي داشته باشيم و هر اتوماتاي دلخواه به وسيله‌ی پنج‌تايي توصيف گردد که در آنها تعاريف زير را داشته باشيم:

* مجموعه‌ی اعمال
* مجموعه‌ی ورودي‌ها 
* مجموعه‌ی حالت‌هاي داخلي 
* قواعد بهنگام‌سازي اتوماتا 

در آن صورت يك بازي اتوماتا  به عنوان مجموعه‌اي از استراتژي‌هاي انتخاب شده توسط اتوماتا در تکرار –nام تعريف شده و به صورت زير بيان مي‌گردد:

|  |  |
| --- | --- |
| رابطه‌ی (2-37) |  |

نتيجه‌ی بازي نيز با نام  به صورت زير تعريف مي‌شود:

|  |  |
| --- | --- |
| رابطه‌ی (2-38) |  |

حال اگر احتمال نتيجه‌ی وابسته به بازي باشد گفته مي‌شود که N اتوماتا در بازي، شريك هستند.

### خلاصه و نتيجه‌گيري

الگوريتم‌هاي يادگيری همواره سعي دارند با شناسايي مشخصات محيطي که با آن در ارتباط هستند، عملكرد و خصوصيات خود را در جهت رسيدن به يك هدف خاص بهينه نمايند. اتوماتاي يادگير نيز نوعي از اين الگوريتم‌ها است که سعي دارد بر حسب پاسخي که از محيط دريافت مي‌کند احتمال مربوط به حالات خود را تغيير دهد و در هر حالت نيز عملكرد خاص خود را اجرا نمايد. اتوماتاهاي يادگير بر حسب ساختارشان به دو دسته‌ی عمده تقسيم مي‌شوند. در دسته‌ی اول، ساختار اتوماتا ثابت مي‌باشد و بر حسب يك توالي خاص از پيش تعريف شده، انتقال حالات صورت مي‌گيرد. اين نوع از اتوماتاها را اتوماتاي ساختار ثابت مي‌نامند. در نوعي ديگر از اتوماتاها، هنگامي كه اتوماتا در حال فعاليت است، احتمالات مربوط به هر يک از حالات دائماً در حال تغيير مي‌باشد؛ لذا اين نوع از اتوماتا را اتوماتاي ساختار متغير مي‌نامند. اتوماتاهايي که در اين دسته قرار می‌گيرند، برحسب چگونگي تعيين احتمال جديد بر حسب احتمالات گذشته، به دو نوع خطي و غيرخطي طبقه‌بندي مي‌شوند. اتوماتاهاي نوع غيرخطي به علت پيچيدگي‌هايي که در عملكرد و تحليل رفتار خود دارند، کمتر مورد استفاده‌ی عملي قرار مي‌گيرند.

## گريد­های محاسباتی

### مقدمه

گريدهای محاسباتی[[89]](#footnote-90) به عنوان يک رويکرد جديد برای حل مسائل در مقياس بزرگ در زمينه‌های علمی، مهندسی و تجاری بوجود آمده‌اند [65-68]. آنها اين قابليت را بوجود می‌آورند که بتوان بنگاه‌های مجازی[[90]](#footnote-91) (VE) را به منظور اشتراک و يکپارچه‌سازی ميليونها منبع که از نظر جغرافيايي در سطح سازمان‌ها و حوزه‌های مديريتی پراکنده‌اند، ايجاد کرد (مانند SETI@Home ). آنها از يک سری منابع ناهمگون (کامپيوترهای شخصی، ايستگاه‌های کاری[[91]](#footnote-92)، کلاسترها و ابرکامپيوترها)، سيستم‌های مديريت زيربنايي (سيستم عامل واحد، سيستم‌های صف و غيره)، سياست‌ها و برنامه‌های کاربردی (علمی، مهندسی و تجاری) با نيازمندی‌های مختلف (پردازنده، ورودی و خروجی، حافظه و شبکه) تشکيل شده‌اند. منابع، متعلق به سازمان‌های مختلف می‌باشد که دارای سياست‌های مديريتی و مدل‌های استفاده و هزينه‌ای مخصوص به خود برای کاربران مختلف در زمان‌های گوناگون می‌باشند. همچنين فراهم بودن منابع و ميزان بار بر روی آنها در طول زمان تغيير می‌کند.

محاسبات گريدی[[92]](#footnote-93) در اوايل دهه‌ی 90 در انجمن ابرمحاسبات[[93]](#footnote-94) مطرح شد و هدف آن به کار گرفتن منابع محاسباتی موجود به شکلي آسان براي محاسبات پيچيده توسط سايت‌هايي بود كه به صورت جغرافيايي توزيع شده مي‌باشند. ايده‌ی اصلي گريد اين‌ است که منابع ماشين‌هاي شركت‌كننده، از طريق يك لايه‌ی نرم‌افزاري به صورت شفاف[[94]](#footnote-95) و مطمئن مورد استفاده قرار بگيرد. اين لايه‌ی نرم‌افزاري، وظايفي همچون مجازي‌سازي[[95]](#footnote-96) منابع، يافتن و جستجوي منابع، و مديريت برنامه‌هاي درحال اجرا را بر عهده دارد. محاسبات گريدی، در ساده‌ترين بيان، محاسبات توزيع شده‌ای است که به سطح تکامل بالاتری رسيده است. هدف اين است که تصوری از يک کامپيوتر مجازی ساده و در عين حال بزرگ و پرقدرت که توانايي مديريت خودش را دارا می‌باشد از يک مجموعه‌ی وسيعی از کامپيوترها بوجود بيايد. اين مجموعه از يک سری سيستم‌های ناهمگون متصل به هم که ترکيبات مختلفی از منابع را به اشتراک گذاشته‌اند، تشکيل شده است.

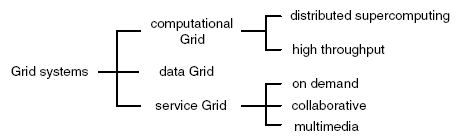
دانشمندان در اواسط دهه‌ی 90 تحقيقاتی را بر روی طراحی و پياده­سازی گريدهای محاسباتی آغاز کردند تا امکان انجام محاسبات موازی و توزيع شده در يک ناحيه‌ی گسترده فراهم شود. انگيزه‌ی رويکرد گريدهای محاسباتی در ابتدا وجود برنامه­های کاربردی علمی در مقياس بزرگ بود که منابع (محاسباتی و داده­ای) زيادی را بيشتر از آنچه که می­توانست در يک کامپيوتر (کامپيوتر شخصی، ايستگاه کاری، ابرکامپيوتر يا کلاستر) قرار بگيرد، نياز داشتند. يک گريد محاسباتی می­تواند منابع مورد نياز برای اين برنامه­های کاربردی را در يک حوزه‌ی مديريتی واحد فراهم کند. محاسبات گريدی به عنوان يک روش کارآمد برای نسل جديد محاسبات مطرح شده است. اين روش امکان به اشتراک گذاشتن، انتخاب و مجتمع کردن منابع ناهمگونی را می­دهد که از نظر جغرافيايي توزيع شده هستند. به اين ترتيب، مسائل علمی، مهندسی و تجاری در مقياس بزرگ، اکنون با استفاده از گريدهای محاسباتی قابل حل هستند. تلاش‌های زيادی در قالب پروژه‌های گوناگون در زمينه‌ی گريد (سيستم‌های گريد يکپارچه، ميان‌افزارهای درونی و در سطح کاربر، و برنامه‌های کاربردی) تاکنون انجام شده است که در شکل 2-23 طبقه‌بندی شده و اطلاعات مربوط به کشور و سازمان انجام دهنده و کاربردهای آنها آورده شده است.

معمولاً اصطلاحاتي مانند محاسبات توزيع شده، محاسبات سراسري[[96]](#footnote-97)، فرامحاسبات[[97]](#footnote-98) و محاسبات کلاستري[[98]](#footnote-99) با محاسبات گريدی اشتباه مي‌شوند. اگر چه همه‌ی اينها شباهت‌هايي با هم دارند، ولی متفاوت هستند. محاسبات توزيع شده يک اصطلاح عام است که همه‌ی موارد ذکر شده را شامل مي‌شود؛ در حالي که محاسبات گريدی يا کلاستري و سايرين، زيرمجموعه‌اي از آن هستند.

در محاسبات کلاستري تأکيد بر بسترهايي متشکل از گره‌هاي همگون متصل به هم است و معمولاً در يک حوزه‌ی مديريتی[[99]](#footnote-100) قرار دارند. کلاسترها عموماً از کامپيوترهاي شخصي و ايستگاه‌هاي کاري[[100]](#footnote-101) و شبکه‌هاي پرسرعت و مطمئن تشکيل شده‌اند. تأکيد برنامه‌ها روي محاسبات حلقه ربایی[[101]](#footnote-102)، محاسبات با توان عملياتی[[102]](#footnote-103) بالا و محاسبات توزيع شده می‌باشد. محاسبات گريدی بر مجموعه‌اي از منابع ناهمگون توزيع شده تأکيد دارد که از آنها به منظور ايجاد يک بستر براي محاسبات با کارايي بالا استفاده مي‌شود. بستر ارتباطي و محاسباتی به صورت نامطمئن و پويا مي‌باشد و منابع به شدت ناهمگون مي‌باشند.

### طبقه‌بندي سيستم‌هاي گريد

اهداف مورد نظر در طراحی سيستم گريد و همچنين کاربردهايي که برای آن در نظر گرفته شده است، از عوامل تعيين‌کننده در نوع کاربری گريد می‌باشند. اهداف طراحي را می‌توان به سه گروه عمده تقسيم‌بندي کرد که عبارتند از: افزايش کارايي برنامه‌ی کاربردي، دسترسي به داده‌ها، و افزايش و بهبود سرويس‌ها. با توجه به اين اهداف می‌توان طبقه‌بندی شکل 2-23 را برای سيستم‌های گريد ارائه کرد.



شکل ‏2‑23: طبقه‌بندي سيستم‌هاي گريد

گريد‌هاي محاسباتی به دو دسته‌ی عمده تقسيم مي‌شوند: 1- ابرمحاسبات توزيع شده و 2- گريد داراي توان عملياتي بالا. گريد ابرمحاسبات توزيع شده، برنامه را به صورت موازي روي چندين ماشين اجرا مي‌کند و هدفش کاهش زمان اتمام[[103]](#footnote-104) يک کار است. نمونه‌ی برنامه‌هايي‌ که نياز به ابرمحاسبات توزيع شده دارند، مسائل بزرگ و پرچالشي مانند شبيه‌سازي هسته‌اي يا مدل‌سازي آب و هوا مي‌باشد. گريد با توان عملياتي بالا، نرخ اتمام يک جريان از کارها را افزايش مي‌دهد و براي کاربردهاي پارامترروب[[104]](#footnote-105) مانند شبيه‌سازي مونت کارلو مناسب است. برنامه‌های کاربردی پارامترروب از تعداد زيادی کار مستقل از هم تشکيل شده‌اند که برای آنها پارامترهای يکسان و مشخصی وجود دارد. هر کدام از کارها تنها در مورد مقادير اين پارامترها با کارهای ديگر متفاوت است. در دنيای واقعی، از برنامه‌های کاربردی پارامترروب که با استفاده از گريدهای محاسباتی به نتيجه رسيده‌اند، می‌توان به مدل‌سازی مولکول‌ها برای طراحی دارو و تشخيص بيماری‌ها از روی پرتونگاری اشعه‌ی X اشاره کرد.

گريدهاي داده‌اي[[105]](#footnote-106) ، يک بستر براي ساختن اطلاعات جديد از منابع داده‌اي توزيع شده همچون کتابخانه‌هاي ديجيتال يا انباره‌ی داده[[106]](#footnote-107) را فراهم می‌کنند. البته گريدهاي محاسباتی نيز به فراهم نمودن خدمات داده‌اي نياز دارند، ولی عمده تفاوت يک گريد داده‌اي و گريد محاسباتی، بستر ويژه‌اي است که براي کاربردهاي مديريت ذخيره‌سازي و دستيابي داده، فراهم شده است. در گريد محاسباتی، برنامه‌ی کاربردي، خود، شِماي مديريت ذخيره‌سازي را پياده‌سازي مي‌کند. نمونه‌اي از کاربردهاي اين سيستم‌ها (گريد داده‌اي) داده‌کاوی[[107]](#footnote-108) برای اهداف خاص است که اطلاعات را از منابع داده‌اي متفاوت به هم مربوط مي‌سازد. پروژه‌هاي گرید داده ای اروپایی[[108]](#footnote-109) و گلوباس[[109]](#footnote-110) نمونه‌‌هايي هستند که در حال کار بر روي توسعه دادن سازمان‌هاي داده‌اي با مقياس بزرگ هستند.

سيستم‌هايي که در يک گريد خدماتي[[110]](#footnote-111) حضور دارند، خدماتي را فراهم مي‌کنند كه توسط يک ماشين واحد قابل ارائه نيست. گريدهاي ‌خدماتي به سه دسته تقسيم مي‌شوند:

1. **سيستم‌هاي گريد on-demand:** منابع مختلف را به صورت پويا با يکديگر ترکيب مي‌کنند تا يک سرويس جديد ارائه دهند.
2. ***سيستم‌هاي گريد همکاري‌کننده[[111]](#footnote-112):*** *کاربران و بر*نامه‌هاي کاربردي را به گروه‌هاي کاري همکاري‌کننده متصل مي‌کنند. اين سيستم‌ها امکان تعامل بلادرنگ بين کاربران و برنامه‌هاي کاربردي را از طريق يک فضاي کاري مجازي فراهم مي‌کنند.
3. **سيستم‌هاي گريد چندرسانه‌اي[[112]](#footnote-113):** يک بستر براي برنامه‌هاي بلادرنگ چندرسانه‌اي فراهم مي‌کنند.

اکثر پژوهش‌هاي فعال در توسعه‌ی سيستم‌هاي گريد، در يکي از دسته‌بندي‌هاي فوق قرار مي‌گيرد. برای توضيحات بيشتر می‌توان به مراجعه کرد.

### توانمندی‌های گريد محاسباتی

هنگامی که يک گريد محاسباتی سازماندهی می­شود، بايد يک سری از نيازمندي‌های کاربران را برطرف کند. برای اينکه نيازهای موجود به طور بهتری با توانمندي‌های يک گريد منطبق گردد، بايد دلايل استفاده از محاسبات گريدی را در نظر داشته باشيم. در اين بخش به بررسی توانمندي‌های مهم محاسبات گريدی می‌پردازيم.

#### بهره­برداری از منابع بدون استفاده

ساده­ترين راه استفاده از محاسبات گريدی اين است که يک برنامه‌ی کاربردی موجود در يک ماشين را بر روی ماشين ديگری اجرا کنيم. ماشينی که برنامه‌ی کاربردی در حالت عادی بر روی آن در حال اجرا می­باشد، ممکن است به علت مواجه شدن با تعداد زيادی فعاليت، به طور غيرعادی درگير محاسبات شود. اين برنامه‌ی کاربردی می­تواند بر روی يک ماشين بدون استفاده که در جای ديگری در گريد موجود است، اجرا شود.

حداقل دو پيش­نياز برای اين مسأله وجود دارد: اول اينکه برنامه‌ی کاربردی بايد بتواند بدون ايجاد کردن سربار[[113]](#footnote-114) غيرضروری، در يک ماشين غير از ماشينی که در آن قرار دارد، اجرا شود. ثانياً، ماشين مقصد بايد سخت­افزار و نرم­افزار و منابع مورد نياز برنامه را دارا باشد.

برای مثال يک برنامه‌ی دسته­ای[[114]](#footnote-115) که زمان قابل ملاحظه­ای را صرف پردازش يک مجموعه از داده­های ورودی می­کند تا يک سری خروجی را توليد کند، ممکن است ايده­آل­ترين و ساده­ترين کاربرد برای يک گريد محاسباتی باشد. اگر تعداد ورودي‌ها و خروجي‌ها زياد باشد، طراحی و تفکر بيشتری برای استفاده‌ی بهينه از گريد برای اين کار مورد نياز است. معمولاً اجرای يک برنامه‌ی پردازش کلمه[[115]](#footnote-116) بر روی يک ماشين ديگر در گريد بی­معنی است؛ زيرا باعث بوجود آمدن تأخيرها و خطاهای بالقوه‌ی زيادی می­شود. در بيشتر سازمانها، تعداد زيادی منابع پردازشی بدون استفاده وجود دارد. بيشتر کامپيوترهای روميزی، کمتر از پنج درصد زمان، فعاليت می­کنند. در بعضی از سازمانها حتی ماشين سرور نيز نسبتاً بيکار است. محاسبات گريدی چارچوبی را برای بهره­برداری از اين منابع بيکار فراهم می­کند و بنابراين اين امکان را بوجود می‌آورد که به مقدار زيادی، استفاده از منابع افزايش پيدا کند.

منابع پردازشی تنها منابعی نيستند که ممکن است بدون استفاده باشند. بعضی اوقات، ماشينها دارای فضای ذخيره­سازی زيادی هستند که استفاده نمی­شود. محاسبات گريدی و به طور خاص يک گريد داده­ای می­تواند امکان جمع نمودن اين فضاها و تبديل آن به يک انبار داده‌ی مجازی بسيار بزرگتر را فراهم کند. به اين ترتيب، موجب می‌شود که کارايي و قابليت اطمينان نسبت به يک ماشين واحد بهبود يابد. اگر يک کار دسته­ای بخواهد مقدار زيادی داده را از ورودی دريافت کند، می­توان نسخه­های يکسانی از اين داده­ها را در نقاط مهم گريد قرار داد. بنابراين اگر يک برنامه بخواهد بر روی يک ماشين دور[[116]](#footnote-117) اجرا شود، لازم نيست داده­های مورد نياز آن نيز جابجا شوند، بلکه داده­ها در نزديکی و يا در خود آن ماشين موجود خواهند بود.

وظيفه‌ی ديگر گريد محاسباتی بهبود توازن در استفاده از منابع می­باشد. در يک سازمان ممکن است در زمانهايي مقدار فعاليتها به اوج خود برسد که باعث افزايش تقاضای منابع می­شود. اگر برنامه­های کاربردی، قابليت اجرا در گريد را داشته باشند، در چنين مواقعی می­توان آنها را برای اجرا به ماشين‌هاي بدون استفاده منتقل نمود. در واقع، بعضی از پياده­سازی­های گريد محاسباتی می­توانند کارهای نيمه­تمام را بين ماشينها جابجا کنند. در حالت کلی، يک گريد می­تواند توازن بار محاسباتی را در مجموعه­های گسترده­تری از منابع انجام دهد. اين منابع می­تواند شامل پردازنده، حافظه‌ی جانبی و انواع ديگری از منابع موجود در گريد باشد.

#### موازی­سازی پردازنده­ها

ظرفيت انبوه پردازنده­های موازی يکی از ويژگي‌های جذاب گريدهای محاسباتی می­باشد. علاوه بر نيازهای کاملاً علمی، اين قدرت محاسباتی می­تواند باعث بوجود آمدن تحول بزرگی در صنايعی مانند حوزه‌ی پزشکی و زيست‌شناسی، مدل‌سازی مالی، اکتشاف نفت، متحرک­سازی تصاوير و بسياری ديگر شود.

ويژگی مشترک چنين استفاده­هايي، استفاده از الگوريتم‌هايي برای تقسيم برنامه­های کاربردی به بخش‌های اجرايي مستقل می­باشد. يک برنامه‌ی کاربردی قابل اجرا در گريد که کار محاسباتی دارد را می­توان به صورت تعداد زيادی کارهای کوچک در نظر گرفت که هر کدام بر روی ماشين متفاوتی در گريد اجرا می­شوند. هر اندازه که اين کارها به ارتباط با يکديگر نياز نداشته باشند، برنامه‌ی کاربردی قابليت مقياس­پذيری[[117]](#footnote-118) بيشتری خواهد داشت. يک برنامه‌ی کاربردی کاملاً مقياس­پذير، اگر از ده پردازنده برای کارهای خود استفاده کند، اجرای آن ده برابر زودتر تمام می­شود.

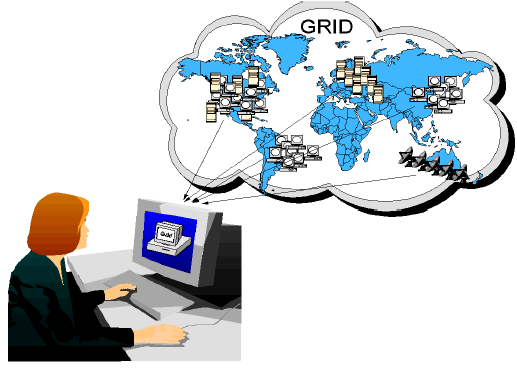
موانعی در برابر مقياس­پذيری کامل وجود دارد. اولين مانع، الگوريتم‌هايي هستند که برای تقسيم برنامه­های کاربردی بين تعداد زيادی پردازنده استفاده می­شوند. اگر الگوريتم فقط به تعداد محدودی بخش‌های مستقل اجرايي قابل تقسيم باشد، مانعی در برابر مقياس­پذيری خواهد بود. دومين مانع هنگامی بوجود می­آيد که بخشهای اجرايي کاملاً مستقل از هم نباشند. اين باعث درگيری کارها با هم می­شود و مقياس­پذيری را محدود می­کند. برای مثال، اگر لازم باشد همه‌ی کارها عمل نوشتن و خواندن از يک فايل يا پايگاه داده‌ی مشترک را انجام دهند، محدوديت‌های دسترسی به آن فايل يا پايگاه داده، يک عامل محدودکننده برای مقياس­پذيری برنامه‌ی کاربردی خواهد بود. دلايل ديگر ايجاد درگيری بين کارها عبارتند از: تأخير در تبادل پيام بين کارها، ظرفيت ارتباطات در شبکه، پروتکل‌های همزمان­سازی، پهنای باند ارتباط با حافظه­های جانبی و وسايل ديگر، و تأخيرهايي که با نيازهای بلادرنگ تداخل پيدا می­کند.

#### برنامه‌های کاربردی

برای اينکه يک برنامه‌ی کاربردی بتواند بر روی گريد اجرا شود، عوامل زيادی را بايد در نظر گرفت. يکی از آنها اين است که بايد قبول کرد که همه‌ی برنامه‌های کاربردی اين قابليت را ندارند که به صورت موازی بر روی گريد محاسباتی اجرا شوند و در عين حال، مقياس‌پذيری قابل توجهی نيز بدست بيايد. علاوه بر آن، در عمل، هيچ ابزار خودکاری وجود ندارد که برنامه‌های کاربردی را طوری تغيير دهد تا بتوانند از قابليت‌های موازی گريدهای محاسباتی استفاده کنند. ابزارهايي وجود دارند که اين مهارت را به طراحان برنامه‌های کاربردی می‌دهند که بتوانند برنامه‌های موازی را به مرحله‌ی پياده‌سازی برسانند. البته دانش تبديل خودکار برنامه‌های کاربردی در مراحل ابتدايي قرار دارد. اين کار واقعاً مشکل است و حتی در شرايطی که امکان‌پذير باشد به دانش رياضيات و مهارت‌های بالای برنامه‌نويسی نياز دارد. اغلب برنامه‌های کاربردی دارای محاسبات زياد که امروزه پياده‌سازی می‌شوند، دارای ساختار موازی هستند و اگر استانداردها و پروتکل‌های جديد گريد را در خود نداشته باشند، به راحتی می‌توان قابليت اجرا بر روی گريدهای محاسباتی را در آنها بوجود آورد.

#### منابع مجازی و سازمان­های مجازی برای ايجاد همکاری

کمک مهم ديگری که يک گريد محاسباتی به ما می‌کند اين است که همکاری بين تعداد زيادی سيستم‌های متصل به هم در يک محدوده‌ی وسيع را امکان‌پذير و ساده می‌سازد. در گذشته، محاسبات توزيع شده در حال رسيدن به اين مرحله از همکاري‌ها بود و تا حدی نيز به آن دست يافته بود. اکنون محاسبات گريدی اين توانايي‌ها را حتی در مقياس وسيع‌تری نسبت به آنچه تصور می‌شد در اختيار قرار می‌دهد. همچنين همراه با آن، استانداردهای مهمی را ارائه می‌کند که سيستم‌های بسيار ناهمگون را قادر می‌سازد تا با يکديگر همکاری کنند و به اين ترتيب، تصويری از يک سيستم محاسباتی مجازی بزرگ که منابع مجازی بسياری را فراهم می‌کند ايجاد شود. اين ديد يکپارچه در شکل 2-24 نشان داده شده است.



شکل ‏2‑24: بازنمايي منابع ناهمگون و پراکنده به صورت يک سازمان مجازی واحد توسط گريد

#### دسترسی به منابع اضافی

علاوه بر منابع پردازشی و ذخيره‌سازی اطلاعات، يک گريد می‌تواند دسترسی به تعداد زيادی از منابع ديگر و همچنين به تجهيزات، نرم‌افزارها، جوازهای[[118]](#footnote-119) استفاده و خدمات خاص ديگر را فراهم کند. منابع اضافی می‌تواند در تعداد و يا در ظرفيت‌های اضافی فراهم شود.

برای مثال، اگر يک کاربر نياز داشته باشد که به منظور پياده‌سازی يک سيستم جستجوی داده‌کاوی، پهنای باند ارتباط خود با شبکه‌ی جهانی اينترنت را افزايش دهد، کار او می‌تواند بين چندين ماشين موجود در گريد که هرکدام ارتباط مستقلی با شبکه‌ی جهانی برقرار کرده است تقسيم شود. در اين صورت، مجموع توانايي‌های جستجو در اينترنت چند برابر می‌شود؛ زيرا هر ماشين ارتباط جداگانه‌ای با اينترنت دارد. اگر اين ماشين‌ها مجراهای ارتباطی خود را در گذشته به اشتراک گذاشته بودند، افزايش چندانی برای پهنای باند دسترسی به اينترنت بوجود نمی‌آمد. بعضی از ماشين‌ها ممکن است جواز استفاده از يک نرم‌افزار گران‌قيمتی را که بر روی آنها نصب شده است در اختيار داشته باشند. کارهای مورد نظر يک کاربر که به چنين نرم‌افزاری نياز دارد می‌تواند به ماشين مربوطه فرستاده شود تا از جواز استفاده از نرم‌افزار موجود بر روی آن ماشين بهره ببرد.

بعضی از ماشين‌ها در گريد ممکن است دارای وسايل جانبی خاصی باشند. بسياری از ما از چاپگرهای دور استفاده کرده‌ايم تا از توانمندي‌های چاپ تصادفی پيشرفته و يا سرعت بالای آنها استفاده کنيم. به طور مشابه، می‌توان از يک گريد برای استفاده از تجهيزات ويژه‌ی ديگر استفاده کرد. برای نمونه، يک ماشين ممکن است دارای يک دستگاه ذخيره‌سازی داده بر روی لوح دی وی دی[[119]](#footnote-120) باشد و ديگران می‌توانند از طريق گريد، برای انتشار سريع اطلاعات از آن استفاده کنند. همچنين بعضی از ماشين‌ها در گريد می‌توانند به يک ميکروسکوپ الکترونی متصل باشند که قابليت کار از راه دور را داراست و کاربران گريد در ساير نقاط می‌توانند از آن استفاده کنند.

### مديريت منابع در گريد

در ادامه، مدیریت منابع در سیستم های گرید توضیح داده خواهد شد.

#### تعاريف و نيازمندي‌ها

در گريدها، منبع[[120]](#footnote-121) عبارت است از يک نهاد[[121]](#footnote-122) با قابليت استفاده‌ی مجدد[[122]](#footnote-123) که برای برآورده کردن يک کار يا تقاضا به خدمت گرفته می‌شود. منبع مي‌تواند يک ماشين، شبکه يا هر سرويس ديگر (ترکيبي از ماشين‌ها، شبکه‌ها و نرم‌افزار) باشد.

فراهم‌کننده‌ی منبع [[123]](#footnote-124)، به عنوان يک عامل[[124]](#footnote-125) که منبع را کنترل مي‌کند، تعريف مي‌گردد. براي مثال، يک واسطه‌گر[[125]](#footnote-126) منبع می‌تواند به عنوان يک فراهم‌کننده‌ی منبع عمل کند. به همين صورت، مصرف‌کننده‌ی منبع[[126]](#footnote-127)، عاملي است که مصرف‌کننده را کنترل مي‌کند.

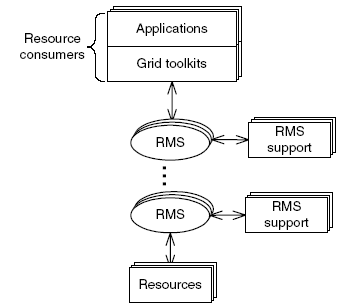
سيستم مديريت منبع[[127]](#footnote-128) (RMS)، سرويسي است که توسط يک شبکه‌ی محاسباتی[[128]](#footnote-129) فراهم مي‌گردد تا مجموعه‌ی وسيعي از منابع داراي نام که براي شبکه‌ی محاسباتی موجود مي‌باشند را مديريت کند، به طوری که شاخص کارايي (سيستم محور[[129]](#footnote-130) يا کار محور[[130]](#footnote-131)) بهينه گردد[77] .

به علت مواردي چون توسعه‌پذيري[[131]](#footnote-132)، سازگارپذيري[[132]](#footnote-133)، خودمختاري سايت[[133]](#footnote-134) و کيفيت سرويس، مديريت منبع در سيستم‌هاي گريد بسيار پرچالش‌تر از محيط‌هاي محاسباتی توزيع شده‌ی[[134]](#footnote-135) قديمی است.

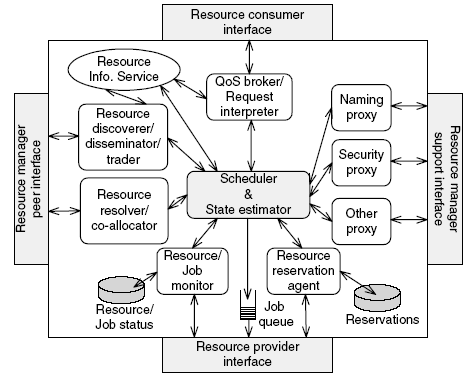
#### مدل انتزاعي سيستم مديريت منبع

تجربه‌ی كار با سيستم‌هاي محاسباتی شبكه‌اي نشان داده است كه برنامه‌ی كارا و كارايي سيستم، الزاماً يكي نيستند. به عبارت ديگر، يك زمانبند ممكن است نتواند هم برنامه را بهينه كند و هم كارايي سيستم را. يك راه‌حل استفاده از سيستم چندلايه است. علاوه بر اين، از آنجايي‌ كه مقياس مورد انتظار در سيستم‌هاي گريد، بزرگ است لذا يك سیستم مدیریت منابع معمولاً از اتصال سیستم مدیریت منابعهای متعددي تشكيل شده است كه با يكديگر همكاري مي كنند. شکل 2-25 يك نمودار بلوكي براي يك سيستم همراه با سیستم مدیریت منابعهاي متعدد به هم متصل را نشان مي‌دهد كه هر سیستم مدیریت منابع داراي چندين سطح مي‌باشد. برنامه‌های کاربردی از سرويس‌هاي جعبه‌ابزارهاي[[135]](#footnote-136) گريد استفاده مي‌كنند.

شکل 2-26 يک مدل انتزاعي از توابع اصلي که توسط يک سيستم مديريت گريد پشتيباني مي‌شود را نشان مي‌دهد. اين مدل داراي واحدهاي عملياتي متعدد و چهار رابط[[136]](#footnote-137) مي‌باشد: رابط مصرف‌كننده‌ی منبع، رابط فراهم‌کننده‌ی منبع، رابط پشتيباني مدير منبع[[137]](#footnote-138) و رابط همتاي مدير منبع[[138]](#footnote-139).



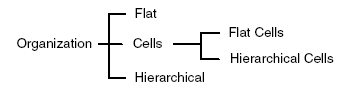
شکل ‏2‑25: زمينه‌ی سيستم سیستم مدیریت منابع



شکل ‏2‑26: ساختار انتزاعي سيستم مديريت منابع

#### سازماندهي ماشين

سازماندهي ماشين‌ها در گريد، الگوي ارتباطي سیستم مدیریت منابع را تحت تاثير قرار مي‌دهد و لذا نقش تعيين‌کننده‌اي در توسعه‌پذيري معماري حاصل دارد. شکل 2-27 طبقه‌بندي سازماندهي ماشين‌ها را نشان مي‌دهد.



شکل ‏2‑27: طبقه‌بندي سازماندهي ماشين‌ها در گريد

سازماندهي براي ماشين‌هاي درگير در مديريت منابع، در واقع، چگونگي اقدام به زمانبندي، ساختار ارتباطي بين ماشين‌ها و نقش‌هاي متفاوتي که ماشين‌ها در جهت زمانبندي ايفا مي‌کنند را روشن مي‌سازد.

در سازماندهي مسطح[[139]](#footnote-140)، همه‌ی ماشين‌ها مي‌توانند بدون نياز به يک رابط، مستقيماً با يکديگر ارتباط داشته باشند. در سازماندهي سلسله‌مراتبي[[140]](#footnote-141)، ماشين‌هاي واقع در يک سطح مي‌توانند با ماشين‌هايي که مستقيماً در بالا يا پايين آنها قرار دارند و همچنين با همتاهای خود در همان سطح، ارتباط مستقيم داشته باشند.

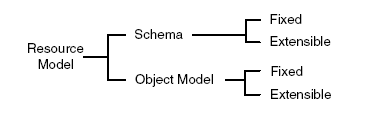
در ساختار سلولي، ماشين‌هاي واقع در يک سلول مي‌توانند با يکديگر از طريق سازماندهي مسطح، ارتباط داشته باشند. در هر سلول، ماشين‌هايي تعيين مي‌شوند که وظيفه‌ی ارتباط با ماشين‌هاي خارج از سلول را بر عهده دارند. ساختار دروني يک سلول از سايرين پنهان است. سلول‌ها خود مي‌توانند بصورت سلسله‌مراتبي يا مسطح سازماندهي شوند.

#### منابع

در اين بخش، جنبه‌هاي مختلف سيستم‌هاي مديريت منبع را بررسي خواهيم کرد که رابط بين منابعي که توسط سیستم مدیریت منابع اداره مي‌شود را فراهم مي‌سازند. برای توضيحات بيشتر می‌توان به [78,79]مراجعه کرد.

**مدل منبع**

مدل منبع، چگونگي توصيف منابع گريد توسط برنامه‌ها و سیستم مدیریت منابع را تعيين مي‌کند. شکل 2-28 طبقه‌بندي مدل منابع را نشان مي‌دهد.

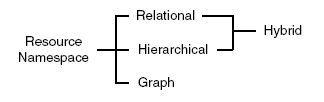


شکل ‏2‑28: طبقه‌بندي مدل منابع

در روش مبتني بر شِما[[141]](#footnote-142)، داده‌اي که يک منبع را شامل مي‌شود، توسط يک زبان، همراه با تعدادي محدوديت‌هاي جامعيت[[142]](#footnote-143) توصيف مي‌شود. در شِماي توسعه‌پذير، بر خلاف نوع ثابت، انواع جديد شِما براي توصيف منابع را مي‌توان اضافه نمود. در يک مدل شيء[[143]](#footnote-144)، عمليات روي منابع به عنوان بخشي از مدل منبع تعريف گرديده است. همانطور که در مورد شِماها گفته شد، مدل شيء نيز می‌تواند دو شکل داشته باشد: از پيش تعيين شده و تثبيت شده به عنوان بخشي از تعريف (مدل ثابت[[144]](#footnote-145)) و يا اينکه مکانيزم‌هايي براي توسعه در آن پيش‌بيني شده باشد (مدل توسعه‌پذير).

**سازماندهي فضاي نام منابع**

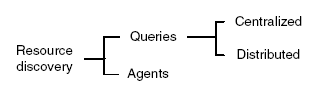
يک سيستم مديريت منبع گريد، بر روي انبوهي از منابع نامگذاری شده‌ی سراسري عمل مي‌کند؛ لذا سازماندهي فضاي نام[[145]](#footnote-146) تأثير عمده‌اي بر پروتکل‌ها و متدهاي يافتن منابع مي‌گذارد. شکل 2-29 طبقه‌بندي سازماندهي فضاي نام را نشان مي‌دهد. فضای نام رابطه‌ای، منابع را به شکل رابطه تقسيم می‌کند و از مفاهيم موجود در پايگاه داده‌ی رابطه‌ای برای اين کار استفاده می‌کند. فضای نام سلسله‌مراتبی، منابع گريد را در سلسله‌مراتب‌هايي پخش می‌کند؛ و در نهايت فضای نام مبتنی بر گراف از گره‌ها و اشاره‌گرها برای تقسيم منابع استفاده می‌کند.



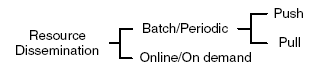
شکل ‏2‑29: طبقه‌بندي سازماندهي فضاي نام

**يافتن و انتشار منبع**

يافتن و انتشار منبع[[146]](#footnote-147) را مي‌توان به عنوان توابع تکميلي لحاظ كرد. يافتن، توسط يک برنامه‌ی كاربردي شبکه‌اي[[147]](#footnote-148)، براي پيدا کردن منابع مناسب در گريد صورت مي‌گيرد. انتشار، توسط يک منبع به منظور يافتن برنامه‌هاي مناسب که مي‌توانند از آن منبع استفاده کنند، انجام می‌شود. يافتن و انتشار منابع بايد بر اساس معيارهای مشخصی انجام گيرد (مانند کيفيت سرويس ). شکل 2-30 طبقه‌بندي يافتن منابع و شکل 2-31 طبقه‌بندي انتشار آنها را نشان مي‌دهد.



شکل ‏2‑30: طبقه‌بندي يافتن منبع



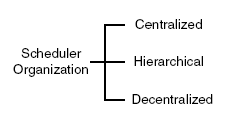
شکل ‏2‑31: طبقه‌بندي انتشار منبع

#### زمانبندي

در اين بخش طبقه‌بندي‌هايي را که وظايف زمانبندي و تخصيص منابع را در سیستم مدیریت منابع توصيف مي‌کنند، مورد بررسي قرار مي‌دهيم.

**سازماندهي زمانبند**

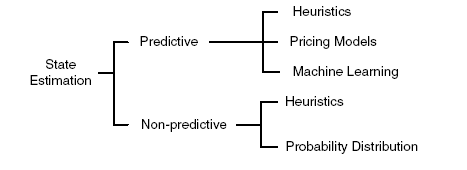
مؤلفه‌هاي زمانبندي در سیستم مدیریت منابع را مي‌توان به سه شکل، سازماندهی کرد (شکل 2-32). در سازماندهي متمرکز فقط يک کنترل‌کننده‌ی زمانبندي وجود دارد که در سطح سيستم تصميم‌گيری می‌کند. در سازماندهي سلسله‌مراتبي، کنترل‌کننده‌هاي زمانبندي در يک سلسله‌مراتب قرار دارند؛ به طوری که هر چه به سطوح بالاتر برويم، کنترل‌کننده‌ها می‌توانند بر مجموعه‌ی بزرگتری از منابع نظارت کنند. سازماندهي غيرمتمرکز، حالتي غير از دو حالت فوق‌الذكر است که در آن کنترل‌کننده‌ها در سطح گريد به صورت توزيع شده قرار می‌گيرند. هر کدام از اين سه حالت داراي مزايا و معايبي هستند. (براي اطلاعات بيشتر به[76,77] مراجعه کنيد)



شکل ‏2‑32: طبقه‌بندي سازماندهي زمانبند

**برآورد وضعيت[[148]](#footnote-149)**

به علت تأخير انتشار اطلاعات در سيستم‌هاي توزيعي مانند سيستم‌هاي گريد، برآورد وضعيت هميشه برحسب اطلاعات جزئي و کهنه صورت مي‌گيرد. تأکيد طبقه‌بندي ارائه شده در اين بخش روي مکانيزم‌هاي به کار گرفته شده براي برآورد وضعيت‌هايي‌ است که پياده‌سازي مدل انتزاعي را تحت تأثير خود قرار مي‌دهند. شکل 2-33 طبقه‌بندي برآورد وضعيت را نشان مي‌دهد.

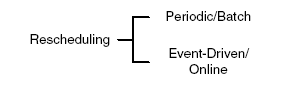


شکل ‏2‑33: طبقه‌بندي برآورد وضعيت

برآورد وضعيت غيرپيش‌گويانه[[149]](#footnote-150) فقط از اطلاعات وضعيت کار و منبع موجود استفاده مي‌کند؛ زيرا نيازي به در نظر گرفتن اطلاعات گذشته ندارد. روش‌هاي غيرپيش‌گويانه يا از مکاشفه[[150]](#footnote-151) بر اساس مشخصات کار و منبع استفاده می‌کنند و يا يک مدل توزيع احتمالي بر اساس تحليل آماري غيربرخط[[151]](#footnote-152) مشخصات کار مورد انتظار، را به کار می‌گيرند. يک روش پيش‌گويانه، اطلاعات کنوني و سوابق را براي برآورد وضعيت لحاظ مي‌کند. مدل‌هاي پيش‌گويانه از يکي از روش‌هاي مکاشفه‌ای، مدل‌هاي قيمت‌گذاري و يا يادگيري ماشين استفاده مي‌کنند. در روش مکاشفه‌ای، از قوانين از پيش تعيين شده براي هدايت برآورد وضعيت استفاده مي‌شود. اين قوانين، بر اساس رفتار مورد انتظار گريد براي برنامه‌ها استخراج مي‌شوند. در روش مدل قيمت‌گذاري، با استفاده از پويايي‌هاي بازار، منابع خريد و فروش مي‌شوند. در يادگيري ماشين، با استفاده از توزيع‌هاي ناشناخته‌ی بالقوه، از شِماي بر‌خط[[152]](#footnote-153) يا غيربرخط براي برآورد وضعيت استفاده مي‌شود.

**زمانبندي مجدد[[153]](#footnote-154)**

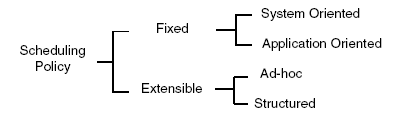
ويژگي‌هاي زمانبندي مجدد در يک سیستم مدیریت منابع مشخص مي‌کند كه چه هنگام، زمانبندي موجود، مورد بررسي مجدد قرار بگيرد و چه هنگام، اجراي کارها دوباره منظم شود. نظم‌بخشي مجدد به کارها برای اين منظور انجام می‌گيرد که بهره‌وري منابع، توان عملياتي، يا ساير معيارها بسته به سياست زمانبندي، ماکزيمم شود. روش به کار گرفته شده براي زمانبندي مجدد، مناسب بودن يک سيستم مديريت منبع را براي يک کاربرد خاص مشخص مي‌کند. شکل 2-34 طبقه‌بندي زمانبندي مجدد را نشان مي‌دهد.



شکل ‏2‑34: طبقه‌بندي زمانبندي مجدد

**سياست زمانبندي**

سيستم مديريت منبع از يک سياست زمانبندي[[154]](#footnote-155) استفاده مي‌کند تا ترتيب جزئي تقاضاها و کارها را هنگام زمانبندي تعيين کند. شکل 2-35 طبقه‌بندي سياست‌هاي زمانبندي را نشان مي‌دهد. در اين طبقه‌بندي، تأكيد بر ميزان متأثر بودن سياست زمانبندي از نهادهاي خارجي مي‌باشد. سيستم‌هاي گريد بزرگ با حوزه‌های مديريتی مختلف، معمولاً داراي سياست‌هاي به كارگيري منابع متعددي‌ مي‌باشند. در روش ثابت، سياست پياده‌سازي شده توسط مديريت منبع، از پيش تعيين شده است؛ در حالی که سياست زمانبندي توسعه‌پذير به نهادهاي خارجي اين امكان را مي‌دهد كه سياست زمانبندي را عوض كنند.



شکل ‏2‑35: طبقه‌بندي سياست‌هاي زمانبندي

### خلاصه و نتيجه‌گيری

محيط گريد يک زيربنای سخت­افزاری و نرم­افزاری را فراهم می‌کند تا در يک شبکه از ماشين‌های توزيع‌شده و ناهمگون، منابع محلی هر ماشين برای ديگران قابل استفاده باشد. در واقع، يک گريد با مجموعه­ای از منابع در مقياس وسيع در ارتباط است که به صورت مشترک مورد استفاده قرار می‌گيرند. درطبقه‌بندی سيستم‌های گريد می‌توان سه نوع گريد را در نظر گرفت: محاسباتی، داده‌ای و خدماتی. گريد‌های محاسباتی توانمندی‌های مختلفی دارند؛ از جمله استفاده از منابع بيکار، به کار گرفتن منابع اضافه و موازی‌سازی پردازنده‌ها برای افزايش کارايي. سيستم مديريت منبع به عنوان مهم‌ترين بخش يک گريد محاسباتی نقش مهمی در کنترل و نظارت بر استفاده از منابع دارد. مهم‌ترين وظيفه‌ی اين سيستم، زمانبندی کارها بر روی منابع می‌باشد. اين عمل می‌توان به صورت متمرکز، سلسله‌مراتبی و غيرمتمرکز انجام شود. با توجه به شرايط حاکم بر گريدهای محاسباتی، زمانبندی غيرمتمرکز و توزيع شده از بقيه موفق‌تر خواهد بود.

فصل سوم

**شبکه پتری تصادفی تطابقی (ASPN)**

# شبکه پتری تصادفی تطابقی

## مقدمه

شبکه­های پتری تصادفی[81] ابزاري مناسب براي مدل‌سازي رياضي و گرافيكي به حساب مي‌آيند. از اين ابزار مي‌توان براي مدل‌سازي، توصيف و تحليل سيستم‌هايي كه داراي ماهيتي همزمان، آسنكرون، توزيع شده، موازي، نامعين و يا اتفاقي هستند استفاده نمود [82]. در­واقع، شبكه‌هاي پتري جزء مدل‌هايي هستند كه قادرند به صورت همزمان حالت و عملكرد يك سيستم را نشان دهند [83]. شبكه پتري تصادفی و تصادفی عمومی یک الگوی گسسته و گسترده در مدلسازی ها بشمار می­آیند. این ابزارها توانایی مدل­کردن احتمالی یک سیستم را با توجه به شرایط مختلف دارا می­باشند.­ شبكه­هاي پتري تصادفی، همزماني­، همروندي و­ عملیات غیر همزمانی پدیده­ها را مورد پردازش قرار می­دهند.­ از رفتار پويا و ساختار سيستم مدل شده بوسيله آناليز شبكه پتري تصادفی، اطلاعات بسيار مفيدي اتخاذ مي‌گردد كه اين اطلاعات مي‌تواند جهت ارزشيابي و حدس­هايی براي ايجاد،­ بهبود و تغييرات در سيستم ها بکار برده شود.­ خاصيت گرافيكي شبكه­هاي پتري تصادفی[84] اجازة تجسم پيچيدگي سيستم را به مدل شبیه­سازی شده می­دهد. شبكه­هاي پتري تصادفی مي­توانند به راحتي رابطه­هاي اولويت­دار و اثرمتقابل ساختاري را در رويدادهاي همزمان و غيرهمزمان به نمايش بگذارند[4] .­ نرم­افزار­های متعددی برای شبیه­سازی و ارزیابی کارایی سیستم­ها توسط شبکه­های پتری و شبکه­های پتری تصادفی­ از سوی افراد متخصص و موسسات تحقیقاتی در دانشگاه­های مختلف ارائه شده است. از جمله این نرم­افزار­ها می­توان به PN Tool،PNK،,Petri net Simulator SPNP اشاره کرد [11,32,85]. شبکه­های پتری تصادفی برای آنالیز سیستم­هایی گسترده کاربرد بسزایی دارند. از جمله این سیستم­ها می­توان به نرم‌افزارهای توزيع شده، پروتكل ارتباطي، سيستم‌هاي توليد، VLSI، سیستم­های پایگاهی، سیستم­های فهرست و آمارهای منطقی، سيستم‌هاي شی­گرا­،­ سيستم‌هاي زمان واقعي و شبکه­های محاسباتی گرید اشاره کرد [86,87].

يكي از مشكلات شبکه پتری تصادفی عدم تطبيق­پذيري آنها مي­باشد و بهمين دليل­ در شبکه­های پتری تصادفی امکان دسترسی به ­اطلاعات قبلی وجود ندارد[81]، مسئله کنترل زمان در شبکه­های پتری تصادفی مورد بررسی ­قرار نمی­گيرد[88] و در سیستم­هایی که پیچیدگی (زیاد بودن حجم درخواست­ها و زیاد بودن عملیات) زیادی دارند شبکه­هاي پتری تصادفی با سرعت بسیار کمی کار تحلیل و آنالیز­ها را انجام می­دهند [89].

هدف اصلي از اين پروژه ارائه يك شبکه پتری تصادفی تطبیقی مبتني بر اتوماتاهاي يادگير مي­باشد.­ شبکه پتری تصادفی تطبیقی مبتني بر اتوماتاهاي يادگير از طریق اطلاعات بدست آمده از حالات قبلی سیستم و واکنش­های محیط پویا، حالت بهینه بعدی را پیشگویی نموده و وضعیت جاری سیستم را بر اساس آن بروز کرده و احتمال وقوع پیش­آمد­ها را در طول زمان تغییر می­دهد و باعث می­شود پیش­آمد­ها بر اساس احتمال وقوعشان فعال ­شوند (رخ ­دهند). این بروز شدن وضعیت­های سیستم بر اساس محیط پویا کمک شایانی در یادگیری و آموزش شبکه­های پتری می­کند­. در این پروژه از ابزار شبیه­سازی SPNP برای شبیه­سازی شبکه پتری تصادفی استفاده می­گردد.

در قسمت ديگري از اين پروژه­ كاربرد مدل تطبيقي پيشنهاد شده در قسمت اول، در شبکه هاي گرید مورد مطالعه و بررسي قرار مي­گيرد. براي اين منظور از مدل تطبيقي ارائه شده براي تخصیص بهينه منابع در گريد با توجه به معيار هزينه استفاده مي­گردد. اين مساله با توجه به فرضيات زير حل مي­گردد: 1. کارهای تخصیص یافته به منابع مستقل از هم هستند. 2. تخصیص منابع به صورت یکباره و در یک مرحله صورت می­گیرد[68,90,91]. الگوريتمهاي طراحي شده با الگوريتم‌های مکاشفه‌ای جديد (ABTO ,EBTO, AEBTO) و الگوريتم‌ يادگيری تقويتی(LATO) مقايسه مي­گردد[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 مصنوعی[[155]](#footnote-156)­، منطق فازی[[156]](#footnote-157) و سیستم های مبتنی بر دانش[[157]](#footnote-158) 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 بردانش­، 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 ها[[158]](#footnote-159) 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 Hybridو 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 ویژگی­های توابع توزیع آن مورد آزمایش قرار داده است. با این همه، بیشتر کارهای تحقیقی مستلزم هم کوشی شبکه هاي پتری وتکنیک های هوشمند می باشد که ایده های آن از مفاهیم منطق فازی و باز­شناسی الگوها در شبکه های عصبی مصنوعي با توجه به 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 می­شود. تغییراتی در ساختار اصلی شبکه پتری در این دسته مقالات اعمال شده است و باعث شده که با استفاده از تکنیک­های 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 تکنیک­های 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. مدل­های این دسته ساده­تر از مدل­های دسته اول می­باشند. در حالی که، در برخی اوقات زمان اجرای آن­ها بیشتر است. همچنین، میزان پیچیدگی در طراحی کمتر از روش اول می باشد. در این روش تکنولوژی­های مختلف در کنار هم قرار می گیرند و با تغییرات نسبی در هر کدام، مدل ارائه شده برای­ اجرا در محیط با بهره­وری و دقت بیشتری نسبت به حالت عادی اعمال می شود. این روش از خصوصیات اولیه شبکه پتری در کنار تکنیک­های هوشمندانه استفاده می کند که به کار­ها و درخواست­های داخلی هر بخشی از سیستم واقعی اعمال می شود. در اینجا، یک ترکیب واقعی و روشن از ویژگی­های 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 وبه برجسته سازی تلاش هایشان از دیدگاه متودولوژی تحلیلی بپردازیم.

### 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 شبکه­ها، قابلیت یاد­گیری برای تطبیق خود در محیط های متغیر را ندارند. از سوی دیگر، تکنیک های هوشمند ويژگی هایی را در اختیار می گذارند که اغلب دارای قابلیت تطبیق می باشند و قادر به ساخت استنتاج سطح بالا[[159]](#footnote-160)، رفتارهای فازی و توانایی­های یادگیری هستند. مسئله­ای 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 دانش[[160]](#footnote-161) 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.

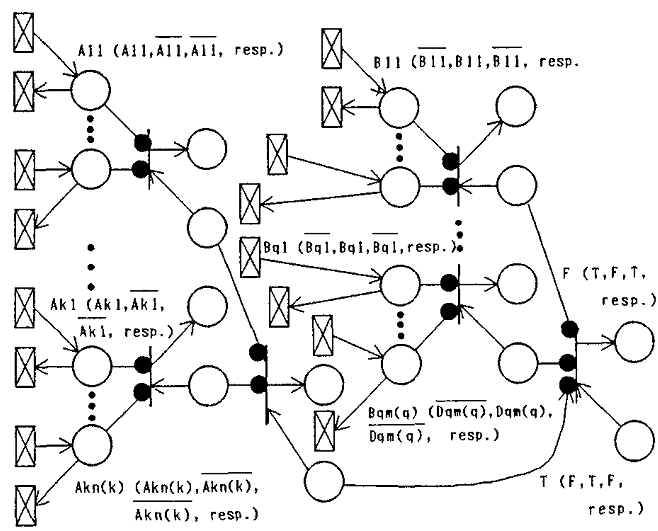
در واقع، مدلی طراحی شود تا بتوان تکنیک های هوشمند و شبکه های پتری را بصورت یک مدل پیوندی و ترکیبی در آن بیان کرد.

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 می­پردازد که بر شیوه­های متنوعی تاکید می­ورزند که در تلاش­اند تا خلاصه متفاوتی از سطوح آگاهی را به مدل­های شبکه ادغام کنند. ویژگی­های مختلفی نظیر معیارهای عضویت، وزن­ها، فاکتورهای قطعیت و قانون یادگیری که همه و همه اقتباسی هستند از تکنیک­های هوشمند که به روشهای مختلفی جهت ساخت مدلی قابل تطابقی از شبکه­های پتری معرفی شده­اند که در این بخش تحت عنوان Fusion Hybrid و Combination Hybrid طبقه­بندی شده ­اند.

### Fusion Hybrid

این رویکرد مستلزم مکتوب کردن ارائه یک متودولوژی به متودولوژی دیگر دلالت بر اینکه چنین گذری به طور طبیعی اتفاق نمی­افتد می­باشد. در اینجا، ایده­های محققین مختلف در مسائل مختلف مورد بررسی قرار گرفته است. سه تحقیق اول بر رویکردی تاکید می­کنند که به منظور بیان خصوصیات یادگیری تطابقی نوعی قیاس تمثیلی بین شبکه­های پتری و شبکه­های عصبی ارائه می کند.

تسوجی[104] در سال 1990 مدل شبکه پتری توسعه یافته­ای[[161]](#footnote-162) 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 فعال­شدن یک انتقال(گذر) می­باشد. ویژگی­های تحریک­پذیری[[162]](#footnote-163)، قابلیت دسترسی[[163]](#footnote-164)، حد و مرز پذیری[[164]](#footnote-165) و روش الگو­سازی موتور استنتاج فازی[[165]](#footnote-166) مورد تجزیه و تحلیل قرار گرفت. همانطور که در شکل­3-1 نشان داده شده، چهار قانون فازی تعریف شده در این مقاله به کمک شبکه پتری توسعه یافته مدل شده است. در جدول 3-1 میزان تغییرات نشانه گذاری­ها در مکان­ها و میزان تغییرات آن در هر گذر آورده شده است.

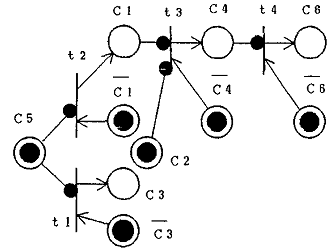


شکل ‏3‑1: فرم استاندارد انتقال توکن بین مکان­ها و انتقال­ها با قوانین فازی تعریف شده

جدول ‏3‑1: جدول تغییرات فازی درمقادیر نشانه­گذاری گذر­های شبکه پتری توسعه یافته

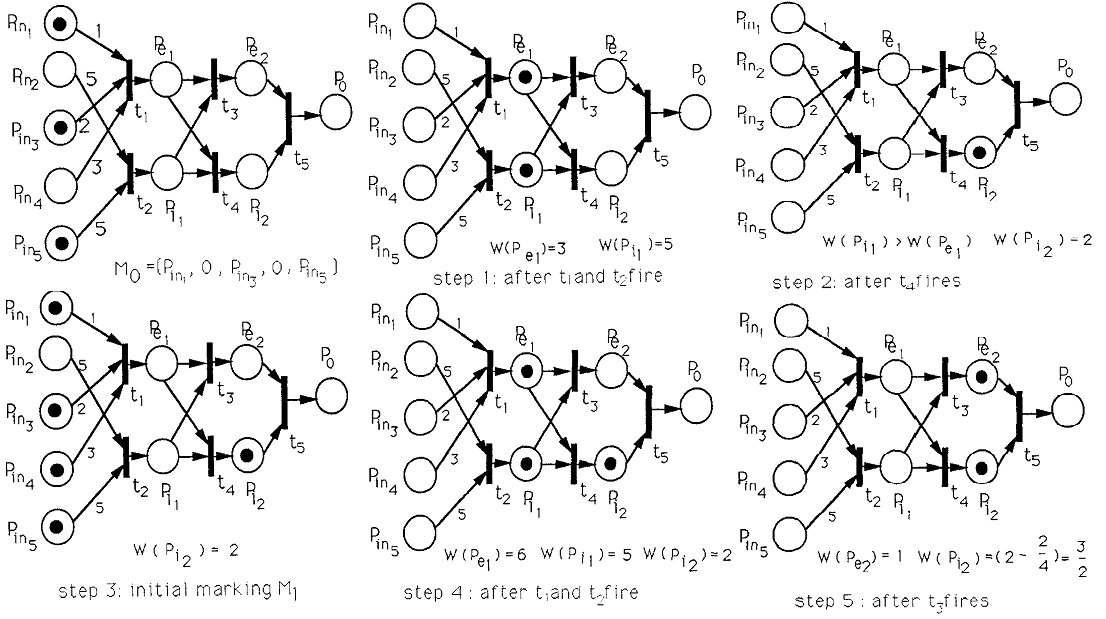
|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |  | **فعال شدن گذرها** |
| 1 | 1 | 1 | 1 | 0 | 0.5 | 0 | 0 | 0.8 | 0 | نشانه گذاری اولیه |
| 1 | 1 | 0.5 | 1 | 0 | 0.5 | 0 | 0.5 | 0.8 | 0 | بعد فعال شدن |
| 1 | 1 | 0.5 | 0.5 | 0 | 0.5 | 0 | 0.5 | 0.8 | 0.5 | بعد فعال شدن |
| 1 | 0.5 | 0.5 | 0.5 | 0 | 0.5 | 0.5 | 0.5 | 0.8 | 0.5 | بعد فعال شدن |
| 0.5 | 0.5 | 0.5 | 0.5 | 0.5 | 0.5 | 0.5 | 0.5 | 0.8 | 0.5 | بعد فعال شدن |
| 0.5 | 0.5 | 0.5 | 0.5 | 0.5 | 0.5 | 0.5 | 0.5 | 0.8 | 0.5 | نشانه گذاری انتهایی |

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

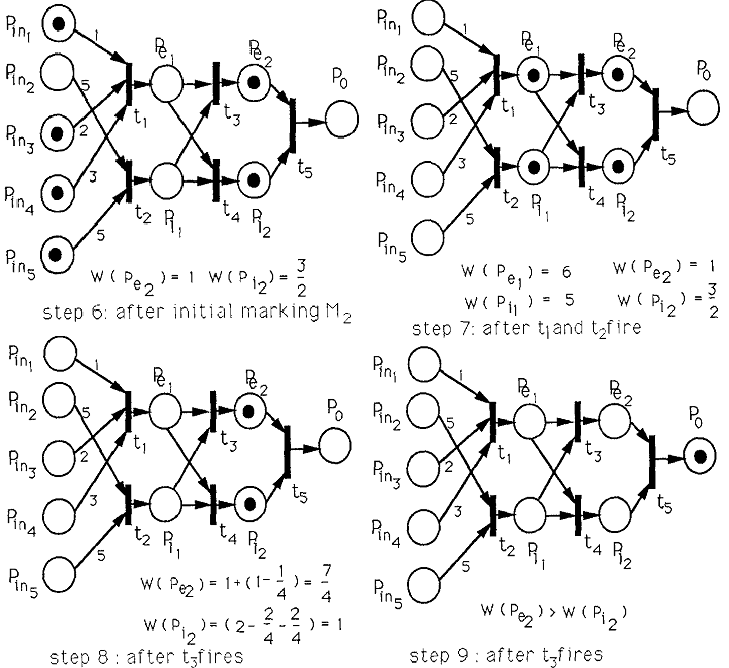


شکل ‏3‑2: موتور استنتاج فازی لحاظ شده در شبکه پتری توسعه یافته

هدجیئنیکولاو [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 ار فاکتور تضعیف کننده[[166]](#footnote-167) (که یک عدد حقیقی است)وزن در واحد زمان استفاده می­کند که تعداد توکن­ها را در برخی از واحدهای زمانی تغییر می دهد که، این فاکتور از طریق یک بیان­کننده کاهش مکرر توان قابلیت پیش سیناسپی است هدایت می­شود. این مدل از قابلیتی 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 یا تعداد بیشتری از تکنیک­های هوشمند مثل شبکه­های عصبی، منطق فازی و سیستم مبتنی بر دانش در مسیر­های موازی انجام گرفته است. شبکه پتری عصبی در شکل 3-3 و شکل 3-4 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

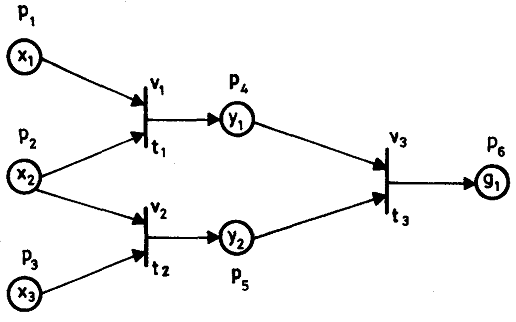
****

شکل ‏3‑4: شبکه پتری عصبی-قسمت دوم

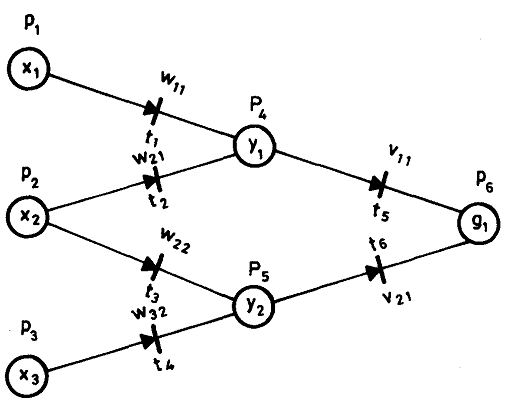
ونکاتش و همکارانش­[106]در سال 1993 یک رویکرد اساسی و بنیادی را برای مدلسازی رفتاری نرون­ها به کمک شبکه پتری سطح بالا[[167]](#footnote-168) ارائه داد. در این مقاله، یک المنت[[168]](#footnote-169) جدید به HRN اضافه شده تا تمام خصوصیات شبکه عصبی را در مدل اعمال کند. این المنت در مکان­ها،انتقال­ها، کمان­ها و وزن­ها اعمال شده اند. اگرچه، چنین رویکردی توسط ونکاتش جزئیات عملیاتی از الگو­سازی فعال آن ارائه نمی­دهد. اما، کمکی به مقایسه شبکه عصبی و نمونه­های شبکه پتری از نقطه نظر ویژگی­ها و شباهت­ها می­کند . این ایده، به شرح و توصیف بخش­های مغز و بخش قابل قیاس آن در شبکه­های ­پتری­ می­پردازد. به عنوان مثال، مکان­ها می­توانند شامل بدنه سلول­های مغز باشند و هم شامل شکاف­های پس سیناسپی[[169]](#footnote-170)(سیناپس پیوندگاه شکاف سلول می­باشد) باشند. اگرچه که هر دو عملکرد و وظایف مختلفی دارند. این شکاف­ها به عنوان نقاط ورودی و خروجی عمل می­کنند. این، در حالی است که، بدنه سلول­ها نقاط سیگنالی را جمع­آوری کرده و آستانه­ای برای آغاز تحرک می­باشند. کمان­ها بطور جداگانه نماینده دندریت­ها[[170]](#footnote-171)(شاخه های متعدد سلول عصبی)می­باشند و آکسون­ها[[171]](#footnote-172) بعنوان توابع تحریک[[172]](#footnote-173) و بازداشت[[173]](#footnote-174) می­باشند. 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 فازی[[174]](#footnote-175) پیشنهاد کرد که در این مدل خصوصیات شبکه عصبی و استنتاج فازی[[175]](#footnote-176) را در هم­آمیخت. در این مدل، شبکه پتری عصبی[[176]](#footnote-177) مکان­هایی را مورد استفاده قرار می­دهد که شامل سه زیر مجموعه می­باشد که عبارتند از: ورودی­ها، انتقال­ها وخروجی­ها. گذرها(انتقال­ها) شامل دو نوع معمولی و آغازی هستند. وزن­های گذر موجود در ورودی ثابت و برابر یک است در­حالی­که وزن­های لایه­های انتقال و خروجی قابل آموزش و تمرین بوده و تنها مقادیر بین 0و1و1- را می­پذیرند. پس، مدل شبکه پتری عصبی با استفاده از خصوصیات منطق فازی اصلاح شده است.

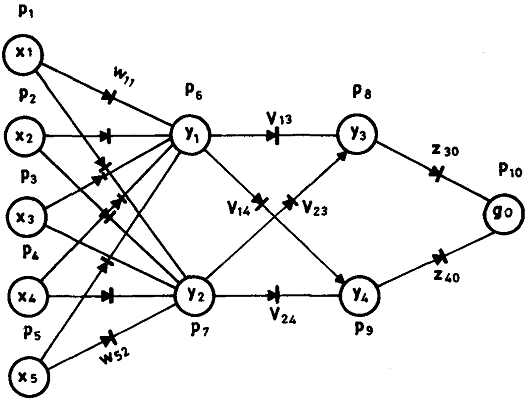
در ادامه در این مقاله، مدل شبکه پتری عصبی با استفاده از خصوصیات منطق فازی تغییر داده شد و سپس شبکه پتری عصبی فازی ایجاد شد. این مدل شامل هفت تاپل[[177]](#footnote-178)بود که به ویژگی های شبکه پتری عصبی اضافه می­شد. این تاپل­ها شامل مجموعه محدودی از گزاره­ها[[178]](#footnote-179) و مجموعه محدودی از مقادیر درستی فازی بر روی گذارها می­باشد.­ همچنین، شبکه پتری عصبی فازی از فاکتورهای اضافه شده­ای استفاده می­کند که از مکان ها به گذرها نگاشت می­شوند و نیز بعنوان توابع اتصال و پیوند که گذرها را به مقادیر واقعی و درست نگاشت می­کنند. این متدولوژی پیشنهاد شده در دسته­بندی الگو­های دودویی[[179]](#footnote-180) تست شد. در شکل 3-5 و 3-6 شبکه پتری فازی عصبی برای قوانینOR و AND نشان داده شده است. در شکل 3-7 مدل سه لایه­ای شبکه پتری فازی عصبی برای استنتاج فازی نشان داده شده است.



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

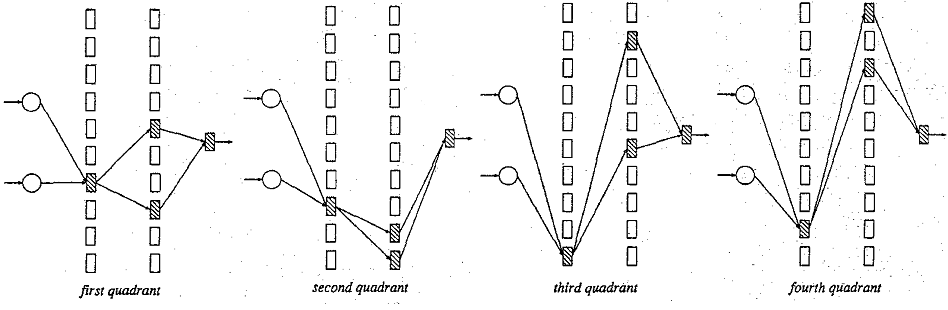
****

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

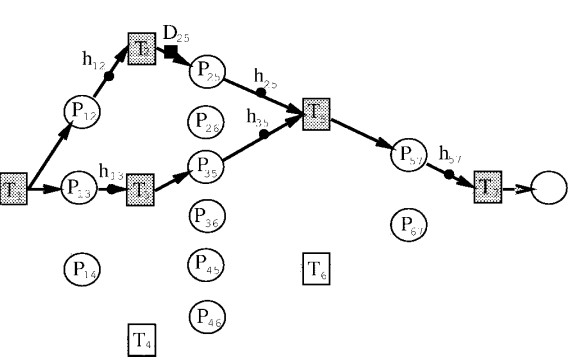
****

شکل ‏3‑7: مدل سه لایه­ای شبکه پتری فازی عصبی برای استنتاج فازی

هیراساوا و همکارانش [108,109] در سال­های 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 شده­ای از شبکه پتری پیشنهاد کرد که قابلیت یادگیری را همانند شبکه عصبی دارا می­باشد. در این مقاله مدلی بر اساس شبکه پتری بنام شبکه پتری یادگیر[[180]](#footnote-181) ارائه شده است که توانایی یادگیری L.P.N از طریق تنظیم و تعدیل کردن وزن­هایی که در مسیر اجرایی باشند کسب می­گردد. ساختار شبکه پتری یادگیر وزن­های در حال اجرا قرار­گرفتن را در کمان­های ورودی گذر­ها معرفی شد و علامت­های در حال اجرا قرار­گرفتن را که با تغییر مسیر توکن یا نشانه ها حرکت می­کند مورد استفاده قرار می­دهد. مطابق فرآیند یادگیری­، توکن یا نشانه­های منتقل شده به شبکه پتری یادگیر ­بسته به مقدار علامت­های در حال اجرا قرار گرفتن توکن یا نشانه­ها در مکان­های ورودی و مقدار وزن­های در حال اجرا قرار گرفتن بسته به کمان­های شبکه پتری یادگیر می­باشد. الگوریتم یادگیری که درشبکه پتری یادگیر استفاده شده است آمیخته­ای است از قانون یادگیری هب[[181]](#footnote-182) و انتشار به عقب[[182]](#footnote-183) که در هر دو وزن در حال اجرا قرار گرفتن و مقادیر آغازین طوری تنظیم شده­اند که علامت در حال اجرا قرار گرفتن یک گذر به مقدار مورد هدف خود نزدیک می­شود. تنظیم و تعدیل وزن نیز به گونه­ای تغییر داده شده است که عبور یک توکن یا نشانه زمانی که انتخاب می­شود تقویت شده و بنابراین تثبیت می­شود. شبیه­سازی به منظور تثبیت کنترل مسیر شبکه پتری یادگیر و همچنین مقایسه عملکرد شبکه پتری یادگیر با مدل آن شبکه عصبی اجرا می­شود. شبکه پتری یادگیر پيشنهادي به مسئله شناسایی سیستم­های غیرخطی ربط داده شد و نتایج شبکه پتری یادگیر ثابت کرد این مسئله برای سیستمی که توسط شبکه عصبی ساده­تری 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 نمونه­ای از چگونگی فعال­شدن و روشن شدن یک گذار در شبکه پتری یادگیر در هر برش زمانی را نشان می­دهد. این شبکه پتری یادگیر که بصورت لایه­ای فرض شده است از یک گذر ورودی، یک گذر خروجی و پنج گذر معمولی در دو لایه پنهان تشکیل شده است[109].

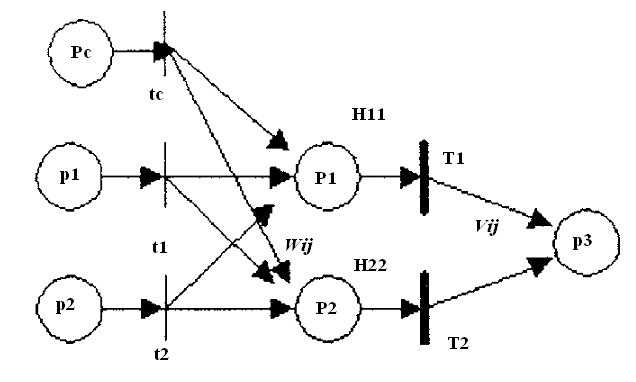


شکل ‏3‑8: یادگیری در شبکه پتری در چهار برش ورودی و خروجی شبکه عصبی

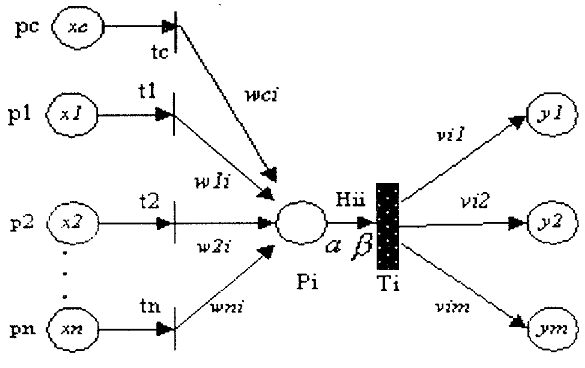
****

شکل ‏3‑9: چگونگی فعال­شدن و روشن­شدن یک گذار در شبکه پتری یادگیر در هر برش زمانی

ژا و همکارانش [110]در سال 1998 مدل شبکه پتری خبره عصبی فازی[[183]](#footnote-184) 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 سیستم­های غیر خطی و طراحی مجدد آن ها می باشد و کاربرد دیگر در کنترل پردازش های روبات های هوشمند می باشد. شکل 3-10 یک شبکه پتری عصبی دو لایه را نشان می دهد. در اینجا،pc بایاس کنترل جریان، p1و p2  لایه ورودی،P1 و P2 لایه پنهان، p2 لایه خروجی، t1و t2 گذرهای نرمال،T1 و T2 گذرهای انتقال دهنده،H11 و H22 مقادیر آستانه فعال­شدن و αوβ توابع انتقال کلی حلقوی یا مارپیچی[[184]](#footnote-185) می­باشند. در شکل 3-11 مدل شبکه پتری خبره فازی نشان داده شده است.



شکل ‏3‑10: شبکه پتری عصبی دو لایه



شکل ‏3‑11: مدل شبکه پتری خبره فازی

تسنگ و همکارانش [111] در سال 1999 ظرفیت یادگیری شبکه­های پتری فازی را از طریق مدلسازی تولید قوانین فازی[[185]](#footnote-186) و الگوریتم یادگیری شبکه عصبی مورد بررسی قرار دادند. الگوریتم یادگیری بکار رفته در مدل پیشنهادی در این مقاله شامل مقدار­دهی اولیه، ارائه نمونه­های آزمایشی، روش­های محاسبات Feed Forward و Feed Backward برای تنظیم ­کمان­ها می­باشد. این مقاله، یک مدل 14-Tuple از شبکه فازی پتری تعریف شده است که شامل مکان­ها ،گذرها، میزان آستانه آغاز­ها در دستورالعمل­های ورودی خروجی­، مجموعه­های فازی­، مجموعه وزن­های محلی یا میانی از قوانین تولید فازی، فاکتور قطعیت قوانین و مجموعه کمان­ها می­باشند. مجموعه کمان­ها ارائه کننده فاکتورهای قطعیت و انواع مختلف دستورالعمل­های ارتباط که به هر مکان یک مجموعه فازی و یک فاکتور قطعیت ، به هر کمان یک وزن مکانی و یک نقشه مطرح دقیق از مکان­ها به مقدار­های آغازی اختصاص می­دهند 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 می­باشد 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 می­برد اما در همان حال نیز همانند شبکه عصبی دارای قابلیت یادگیری بود. این بررسی برای طراحی یک نظام استنتاجی دانش پویا تاکید داشت که بر اساس تنوع دانش مثلا شناخت انسان[[186]](#footnote-187) و درک و تفکر انسان قابل توصیف بود. این مدل به معرفی مفهوم تولید قانون تنظیم شده فازی[[187]](#footnote-188) می­پردازد که این مفهوم به هریک از طرح­ها با توجه به سابقه کاربردی آن­ها و بر اساس یک قانون یک درجه اهمیت و ارزش در نظر می­گیرد. در این مقاله، الگوریتم­های یادگیری مجزا توسعه پیدا کردند و الگوریتم­های استنتاج فازی[[188]](#footnote-189) نامیده شدند و الگوریتم­های انتشار به عقب برای تائید وزن­های قابل تعدیل و یادگیری دانش تحت شرایط کلی استفاده شده است.

چن و همکارانش [114] در سال 2002 مدلی از شبکه پتری فازی برای طبقه­بندی الگوها[[189]](#footnote-190) مبتنی بر قاعده(قانون) ارائه دادند. در این مدل پیشنهادی قوانین فازی در حوزه دانش[[190]](#footnote-191) اعمال می­گردد بطوری که به کمک آن وزن­های محلی و فاکتور قطعیت[[191]](#footnote-192) توسط یک روال یادگیری بصورت آنلاین تنظیم می­گردد. در این جا، هر مکان ورودی به عنوان یک ویژگی از الگو در نظر گرفته می­شود و در هر انتقال(گذر) یک تولید قوانین فازی می­باشد. نشانه گذاری در هر مکان به عنوان مقدار عضویت کلاس[[192]](#footnote-193) می­باشد. شبکه از هفت لایه استفاده می­کند. این مدل برای دسته بندی مجموعه داده­های آیریس[[193]](#footnote-194) که به منظور آزمایش توانایی استنتاج شبکه­های پتری فازی در شناسایی نمونه­های مختلف مورد استفاده قرار می­گیرد.

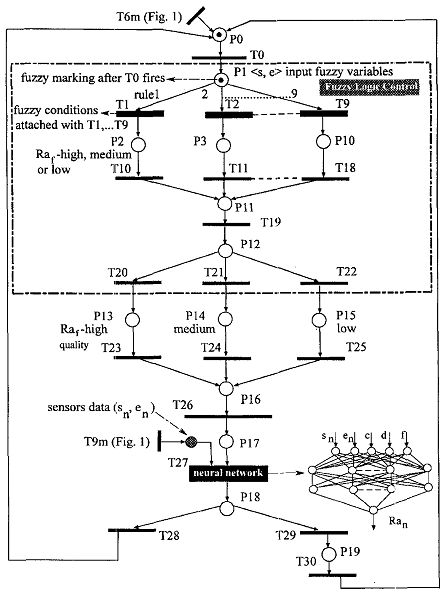
جونز و همکارانش [115] در سال 2003 مدلی از یک تست آزمایشگاهی از یک خط آهن و عبور قطار بر روی آن بر اساس زمان واقعی توسط چند پردازنده را ارائه کردند. در این مدل، خصوصیات شبکه عصبی، زمان­بندی و توکن­های تصادفی در شبکه پتری با یک دیگر ادغام شده است. ساخت و شکل مسیر قطار به طور ساختاری با بهره­گیری از مکان­ها، گذرها و کمان­های راست شده ساخته شده است. این در حالی است که، خود قطار توسط رنگ خاص و منحصر به فرد توکن یا نشانه­ها ارائه می­شود. سوزن­های ریل با استفاده از شبکه پتری عصبی که مجموعه­ای از گذرها را که از آرایش و نحوه فعال­سازی یک اثر در ساختار چندگانه و لایه­بندی شده ورودی و خروجی شاخته شده­اند مورد استفاده قرار می­دهد. ترکیبات مکان­های زمانی­، گذرها و کمان­های شبکه­های پتری ابزاری را در اختیار می­گذارد که میزان در حال اجرا قرار گرفتن سیناسپی را نمونه سازی کرده و سرعت اجرا را پردازش می­کند. یادگیری از طریق شبیه­سازی سوییچ­های شبکه پتری عصبی با بردارهای نشانه­گذاری شبکه و تنظیم وزن سیناپس­ها تا زمانی که رفتار مورد نظر کسب شود انجام می­شود.

گووا و همکارنش[116] در سال 2003 یک شبکه پتری استدلال فازی[[194]](#footnote-195) را برای سیستم تولید قوانین فازی ارائه کردند. در این مدل یک الگوریتم استدلالی استنتاج می­شود که فرمت ماتریکس تساوی را برمی­گزیند. این ماتریکس تصاویر بسیار شبیه به نوع خود در شبکه پتری معمولی می­باشد و توانایی استدلال موازی را به خوبی درجه صحت تمامی طرح­ها و مسائل از نخسیتن آنها حتی بدون استفاده از روش تحلیل قابلیت دسترس نشان می­دهد. آنها به چگونگی این امر از طریق نشان­دادن­، تامل کردن و استنتاج قوانینی که شامل اشتباهات چاپی منفی در الگوی شبکه پتری پیشنهادی خود هستند می­پردازند چنین مدلی از طریق یک سیستم تخصصی خطا یاب مورد اعتبار قرار می­گیرد.

### Combination Hybrid

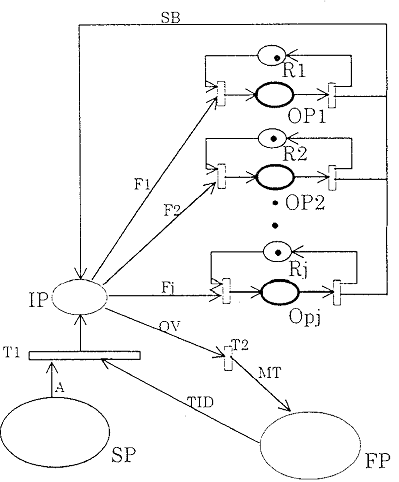
از نقطه نظر بهره­گیری از توانایی هر روش در ترکیب و ادغام به گونه­ای که هویت و ویژگی­های جداگانه آنها حفظ شود، این رویکرد شامل پیوند صریح و آشکار می­باشد.

هانا وهمکارانش[117] به منظور حفظ و نگهداری کیفیت ویژگیهای محصول در یک مرکز فرز[[195]](#footnote-196) دستگاه­های CNC­، مدلی مبتنی بر ترکیب و بکار­بردن دو روش ANN و شبکه پتری فازی را ارائه دادند. در اینجا، از شبکه پتری فازی بهمراه شبکه عصبی مصنوعی برای مدل­سازی و کنترل ناهموار سطوح فلزات[[196]](#footnote-197) در فرآیند فلزکاری CNC استفاده می­شود. این مقاله مثالی است از کار تحقیقاتی که در قالب مدل ترکیبی با استفاده از شبکه پتری و ANNانجام گرفته است. در این مدل ترکیبی علاوه بر گذرهای مرتبط با فرآیند فلزکاری دستگاه CNC، یک گذر هنگام فعال شدن ANN را راه اندازی میکند[[197]](#footnote-198) تا شبکه شروع بکار کند و گذر دیگر باعث فعال شدن خروجی درشبکه عصبی مصنوعی می­شود. اگر خروجی ANN مشابه خروجی شبکه پتری باشدآنگاه گذر دیگری برای تکرار این چرخه فعال می­شود، 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

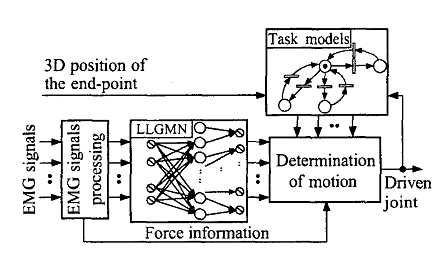
سانگ و همکارانش [118] یک تکنیک بهینه سازی برای سیستم­های تولید قابل انعطاف صنعتی (FMS)[[198]](#footnote-199) مطرح کردند، با کمک این تکنیک، مسئله زمانبندی تخصیص منابع بر اساس مدل شبکه عصبی هاپ فیلد[[199]](#footnote-200) و شبکه پتری تصادفی مدلسازی می­شود. در اینجا، تاخیر زمانی با مکان­ها و انتشار زمانی با توکن­ها همراه است، همچنین این مدل محدودیت ارتباط بین وظایف، دستگاه­ها و ساختار عینی را به یک ساختار انرژی نگاشت می­کند. با استفاده از این ساختار انرژی با مدل شبکه عصبی هپ فیلد، الگوریتم طراحی بهینه سازی برای FMS به دست می­آید. نمودار مدل سیستم­های تولید قابل انعطاف صنعتی در شکل 3-13 نشان داده شده است.



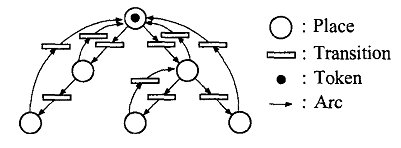
شکل ‏3‑13: نمودار شبکه پتری تصادفی مدل FMS

سر­انجام، آقای فاکودا و همکارانش[119] در سال 2002 برای بهبود صحت نمایش الگوی الکترومایگرام[[200]](#footnote-201) روش کنترل مایوالکتریک را با استفاده از شبکه عصبی سازمان­دهی شده و رخدادی که الگوی وظیفه را استخراج می­کند مطرح کردند. این مدل، باعث بهبود در کنترل دقت دستگاههای الکترومیوگرام با استفاده از هوشمندی شبکه­های عصبی و مدل وظیفه توسط شبکه­های 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 الکتروکاردیوگراف 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 الکترو­کاردیو­گراف[[201]](#footnote-202) بعنوان یک ورودی از شبکه عصبی می­باشد که می­خواهد روال آموزش در آن اجرا شود. در اینجا سنسورهای سیگنالهای الکتروکاردیوگراف بر روی دست و مفاصل قرار گرفته­اند 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 و ظیفه 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 می شود و در انتها آن را در محیط، پیاده سازی می کند. هر درخواست آمده به شبکه پتری بعنوان یک وظیفه محسوب می شود­. 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 می گیرد نشان داده می شود . مدل وظیفه در شکل 3- 15 نشان داده شده است.

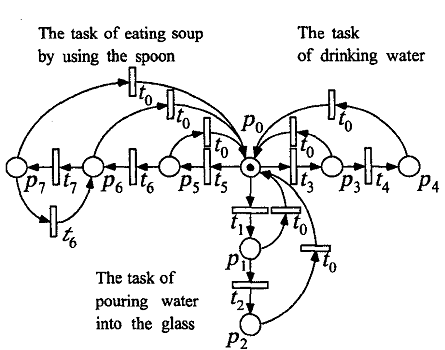


شکل ‏3‑14: ساختار کلی ایده پیشنهادی آقای فاکودا

****

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 خطی[[202]](#footnote-203) و یک شبکه مرکب گوسین[[203]](#footnote-204) 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 مستقلی می باشد که در توصیف خروجی شبکه عصبی استفاده می شود. مدل وظیفه با استفاده از سابقه­ای 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 پیشنهادی 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 3-2 مکان ها و گذر های یک وعده غذا خوردن ربات را نشان می دهد.



**شکل ‏3‑16: مدل وظیفه برای یک وعده غذایی**

جدول ‏3‑2: وضعیت مکان ها و گذر ها

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

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 اطالعات محیط بتواند در گام های دیگر طراحی به کمک وضعیت حالت فعلی حالت بعدی سیستم را پیش­بینی 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 تطبیقی[[204]](#footnote-205) صورت گرفت به این نتیجه می رسیم که یک شبکه پتری تصادفی را می­توان در سیستم­های هوشمند و سیستم­هایی که پویایی و تغییراتی زیادی دارند بکاربرد.

در فصل­های بعدی به کاربرد ASPN در گرید های محاسباتی و اقتصادی می­پردازیم.

فصل چهارم

**کاربرد شبکه پتری تصادفی تطابقی در گرید محاسباتی**

# 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 پیچیده[[205]](#footnote-206)، توزیع شده، ناهمگون[[206]](#footnote-207) ، پویا و خودمختار[[207]](#footnote-208) 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 ومساله 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 مراتبی[[208]](#footnote-209) لحاظ شده است. در اینجا این مدل را­HSPN ­ 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 خانگی[[209]](#footnote-210)، لایه محلی[[210]](#footnote-211) و لایه گرید[[211]](#footnote-212) تقسیم بندی شده اند. کارهای ما در این سه لایه قرار می گیرند. هر لایه دارای یک سری توابع خاص برای دریافت زیر کارها[[212]](#footnote-213) 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.

## مرور اجمالی بر گرید محاسباتی

در سال های اخیر کاربرد گرید محاسباتی افزایش قابل توجهی داشته است. در واقع، گریدهای محاسباتی شبکه ای از کامپیوترها می باشند که بطور مجتمع[[213]](#footnote-214) قرار گرفته اند تا یک کامپیوتر مجازی با سرعت بسیار بالا را کارایی بالاتر از کارایی هر کامپیوتر را مهیا کنند. گریدهای 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 باشم حل کند. گروه های مختلفی در گرید محاسباتی در کنار هم مشارکت می کنند که به آن ها سازمان های مجازی[[214]](#footnote-215) گفته می شود. VOها برای حل یک مساله در کنار هم قرار می گیرند و با حل آن مساله همکاریشان 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 خودشان می پردازند). یکی از انتظاراتی که از گرید 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 گرید بصورت نا همگون می باشند. بعضی از این برنامه ها نیازمند دسترسی به منابع مدیریت شده توسط واحدهای خودمختار می باشند. واحد میریت منابع همان زمانبندی منابع می باشند. تکنولوژی گریدهای 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 گرید ارائه شده اند. از آنجایی که پویایی در خواست ها در گریدهای 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 ارزیابی ایده پیشنهادی با Min.Min و Max.Min می پردازیم و در انتها نتیجه­گیری و کارهای آینده ذکر خواهد شد.

## کارهای مرتبط در زمانبندی گرید محاسباتی به کمک شبکه پتری

یکی از مهمترین بخش ها در سیستم های گرید، 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 گرید تلاش های زیادی را در این زمینه انجام داده اند. کارهای زیاد و گوناگونی در زمینه محاسبات گرید انجام شده است. آقایان یو[[215]](#footnote-216)و بویا[[216]](#footnote-217) در سال های 2005 و 2006 تحقیقات زیادی را در زمینه گردش کار و وظایف در گرید انجام داده اند[121][120]. 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 گرید را تا حد قابل توجهی منظم تر،سریع تر و کاملتر کنترل کند. این مدل بر اساس الگوریتم های زمانبندی گراف جهت دار غیر حلقوی[[217]](#footnote-218) 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 گریدی 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 شبکه­های پتری همچون [[218]](#footnote-219)[122]CPN و SPN [34]بعنوان ابزاری کارآمد و کارا در زمینه زمانبندی و تخصیص منابع در گرید معرفی شدند. در [123] یک مدل سه لایه ای بر اساس شبکه پتری زمانی سلسله مراتبی[[219]](#footnote-220) در سال 2005 ارائه شدند. این مدل بر اساس مدل های لایه ای در شبکه پتری برای زمانبندی در گرید های محاسباتی ارائه شده است. دراین مقاله، سه لایه زمانبند محلی،خانگی و گرید معرفی شدند. این ایده براساس کارهای مستقل تعریف شده است و کارهای وابسته را مورد بررسی قرار نمی دهد. مدل کردن کارهای وابسته در [124] توسط یااوجین[[220]](#footnote-221) و همکارانش بوسیله شبکه پتری گسترش یافته در سال 2006 ارائه شد. مدل زمانبندی برنامه ای در گرید که بر اساس شبکه پتری بود در سال 2007 توسط ژاو[[221]](#footnote-222) و همکارانش در[125] ارائه شد. در این بررسی، تمامی کارهای انجام شده بر روی زمانبند گرید بر اساس خانواده شبکه پتری آورده شده است. وی در این مقاله، الگوریتم های زمانبندی را بصورت زمانبندی چهارسطحه 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 می­کنیم 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)

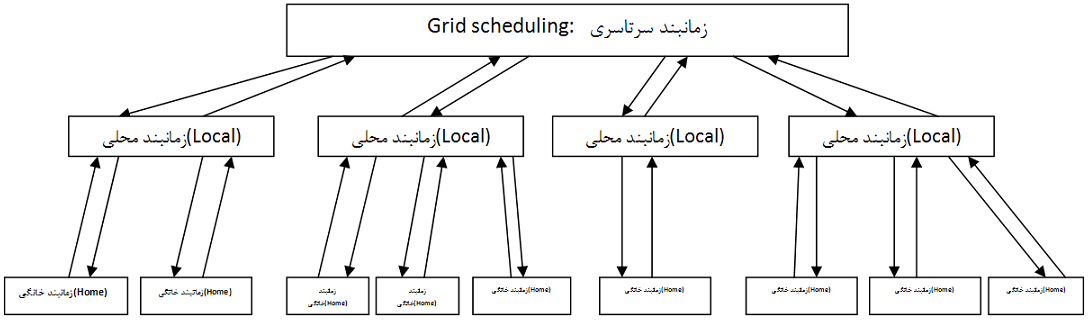
همانطور که می­دانیم، زمانبندی و تخصیص منبع در گرید محاسباتی مسئله مهمی است. دو روش برای تخصیص منبع به در خواست ها در سایت های مختلف وجود دارد.[126,127]

1. Distributed Scheme (طرح توزیع شده)
2. Hierarchical Schema (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 در خواست منابع را بصورت توزیع شده در نظر می گیرند و در خواست 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 در خواست ها با توجه به نوع درخواست ها در سایت­ها که بصورت سلسله مراتبی مرتب شده اند صورت می گیرند. در نتیجه منابع را با توجه به شرایط محیطی و 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 زمانبد 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 4-1 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 ‏4‑1: 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 در خواست 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:**

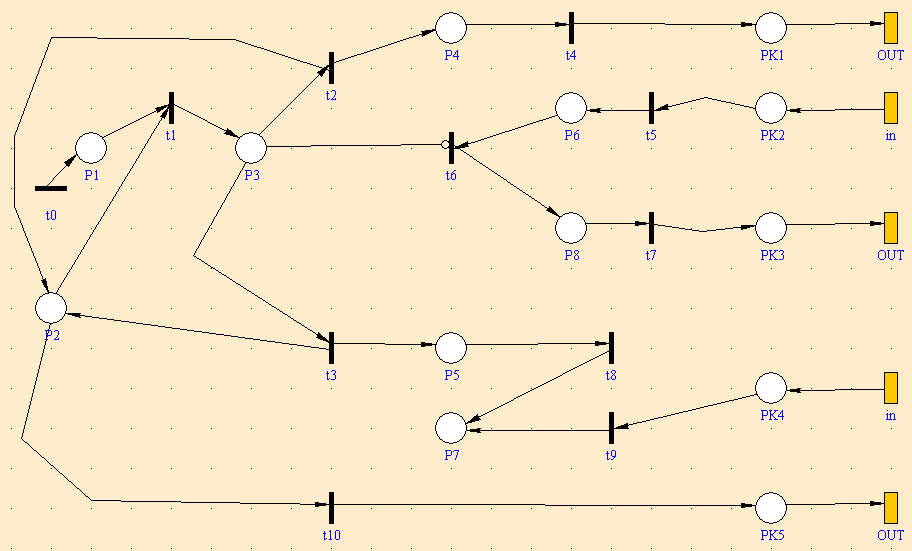
* دریافت اطلاعات از لایه دوم که توسط زمانبندهای محلی انجام شده است
* تخصیص کارها یا درخواستها به زمانبند محلی

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

|  |  |  |  |
| --- | --- | --- | --- |
| **توضیح** | **Transition** | **توضیح** | **Place** |
| تایید درخواست به کامپیوتر خانگی(ماشین خانگی) | t0 | درخواست های تایید شده توسط کاربر | P1 |
| دریافت درخواست و اطلاعات کامپیوتر خانگی(ماشین خانگی) | t1 | منابع | P2 |
| کار یا درخواست در زمان مقرر(Deadline) نتوانست به پایان برسد | t2 | آماده برای زمانبندی درخواست | P3 |
| کار یا درخواست در زمان مقرر(Deadline) توانست به پایان برسد | t3 | آماده برای تایید درخواست | P4 |
| ارسال کردن کار یا درخواست به زمانبند محلی | t4 | آماده برای اجرای درخواست | P5 |
| ارسال کردن کار یا درخواست به زمانبند خانگی | t5 | آماده برای اجرا در خواست از راه دور(Remote) | P6 |
| اجرا کردن کار یا درخواست بصورت  خانگی/راه دور(Remote) | t6یا t8 | درخواست به اتمام رسیده(Complete) یا کار تکمیل شده | P7 |
| تایید نهایی زیردرخواست های اجرا شده بصورت راه دور(Remote) | t7 | زیردرخواست هایی که بصورت remote تکمیل شده است | P8 |
| بازگرداندن درخواست پاسخ داده شده و به اتمام رسیده | t9 | واحد ها و بخش های خروجی تایید درخواست،تکمیل شدن کارهای کوچک یا زیردرخواست ها،حاوی اطلاعات ماشین خانگی(کامپیوتر خانگی) | Pk1,PK3,PK5 |
| مهیا کردن اطلاعات ماشین یا کامپیوتر خانگی | t10 | 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 | Pk2,PK4 |

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



شکل ‏4‑2: مدل ایده پیشنهادی HSPN

حال این مدل را به کمک GridSim مورد تست و آزمایش قرار می دهیم. در ایتدا 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 پروتکل‌های تعامل بين نهادهای[[222]](#footnote-223) GridSim، با استفاده از رخدادها پياده‌سازی شده است.. هر نهاد برای ارسال وضعیت خود به بروکر و زمانبند ها آن را بصورت یک رخداد منتقل می کند. در GridSim نهادها هم برای درخواست اطلاعات و هم برای تحويل اطلاعات از رخدادها استفاده می‌کنند. رخدادها می‌تواند توسط هر نهادی توليد گردد تا بلافاصله و يا با تأخير زمانی تعيين شده به نهادهای ديگر يا به خود او تحويل گردد. رخدادهايي که توسط همان نهاد توليد شده باشند رخدادهای داخلی[[223]](#footnote-224) و رخدادهايي که توسط نهادهای خارجی توليد شده باشند رخدادهای خارجی[[224]](#footnote-225) ناميده می‌شوند. نهادها می‌توانند اين رخدادها را با توجه به نشانی مبدأ رخداد که در کنار آن قرار دارد از هم تشخيص دهند.

از پروتکل‌های GridSim برای تعريف خدمات نهاد استفاده می‌شود. با توجه به پروتکل‌های سرويس، رخدادهای GridSim را می‌توان به رخدادهای همزمان و ناهمزمان[[225]](#footnote-226) نيز تقسيم‌بندی نمود. يک رخداد را همزمان می‌گوييم اگر نهاد مبدأ رخداد، صبر کند تا نهاد مقصد، تمام اعمالی که در ارتباط با رخداد است را انجام دهد (مانند تحويل يک سرويس کامل). يک رخداد را ناهمزمان می‌گوييم اگر نهاد مبدأ رخداد، آن را توليد کند و سپس بدون صبر کردن برای کامل شدن آن، به فعاليت‌های ديگر خود ادامه دهد. هنگامی که نهاد مقصد چنين رخدادها يا درخواست‌های سرويس را دريافت کرد، نتايج را به عنوان جواب با يک يا چندين رخداد به نهاد مبدأ برمی‌گرداند که خود می‌تواند باعث انجام اعمال مناسب بعدی گردد. بايد توجه کرد که رخدادهای خارجی می‌توانند به صورت هزمان يا ناهمزمان توليد شوند ولی رخدادهای داخلی بايد فقط به صورت ناهمزمان توليد شوند تا از به بن‌بست[[226]](#footnote-227) رسيدن نهاد جلوگيری شود. رخدادها در زمانبند گرید بصورت خارجی و در زمانبند های دیگر داخلی عمل می کند. در ادامه به ارزیابی روش پیشنهادی با الگوریتم های Min.Minو Max.Min می پردازیم.

## شبیه سازی ایده پیشنهادی

این بخش یکی از مهمترین بخش ها در این قسمت می باشد.در اینجا از پارامت های 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) بطور 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 کار[[227]](#footnote-228) 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 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 4-2 به 4 گروه تقسیم بندی شده اند. هر گروه شامل 25 کار می باشند.

جدول ‏4‑2: 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 محتواترین[[228]](#footnote-229) درخواست تا پرمحتواترین[[229]](#footnote-230) 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. پر محتواترین درخواست یعنی درخواستی که پاسخگویی به آن دشوار­تر(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 Max.min کافی است تغییراتی در کد بروکر Gridsim ایجاد کنیم. یک پکیج ایجاد می کنیم و آن را در Gridsim اضافه می کنیم. کلاس Main ، Task ، Job وNode 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 Taskمورد نظر به اتمام رسید. ان Task از لیست Taskها خارج می شود و این روال برای Taskهای باقی مانده ادامه می یابد تا تمام Taskها 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 گرید 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 درخواست[[230]](#footnote-231)(ART)**

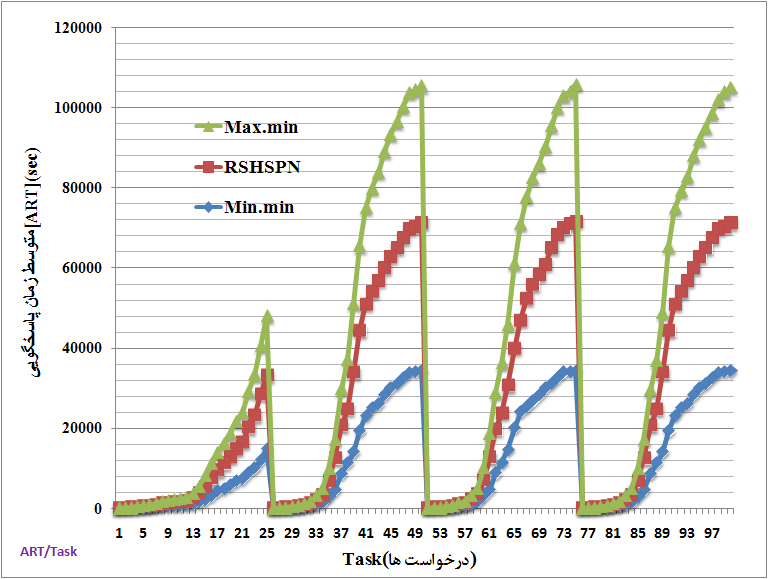
یکی از مهمترین پارامترها برای ارزیابی می باشد. این پارامتر نشاندهنده توالی بین دریافت پاسخ و ارسال پاسخ می باشد که زمان پاسخگویی نامیده می شود. زمان پاسخگویی در صفحات وب برای ارسال/دریافت پاسخ ها مهم می باشد. زمان متوسط پاسخگویی در HSPN همان مجموع زمان ها برای عبور یک در خواست از هر سه لایه می باشد. ممکن است Taskهایی 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 آنسنت 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 درخواست[[231]](#footnote-232)(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.

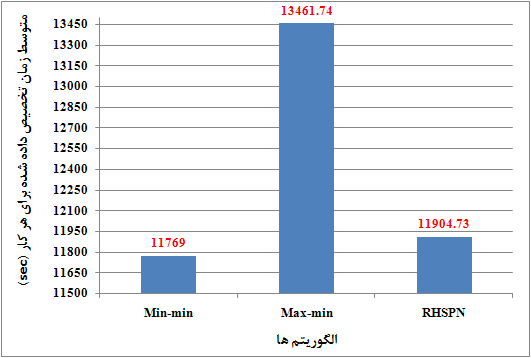
تمام الگوریتم های زمانبندی در طراحی و اجرای خود این پارامتر را مورد بررسی قرار می دهند. این الگوریتم ها منبع را در نزدیکترین حالت برای سایت Task در نظر می گیرند و هزینه آن ها از هزینه اجرا هر Task­کمتر می باشد. اما، ممکن است منابعی وجود داشته باشند که هزینه آن ها از هزینه مورد انتظار ما کمتر باشند ولی 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+((1/100)\*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 4-3 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 دسته بهتر عمل می کند و ولی در هر چهار حالت از الگوریتمMin.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 در خواست 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 ‏4‑3: 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.

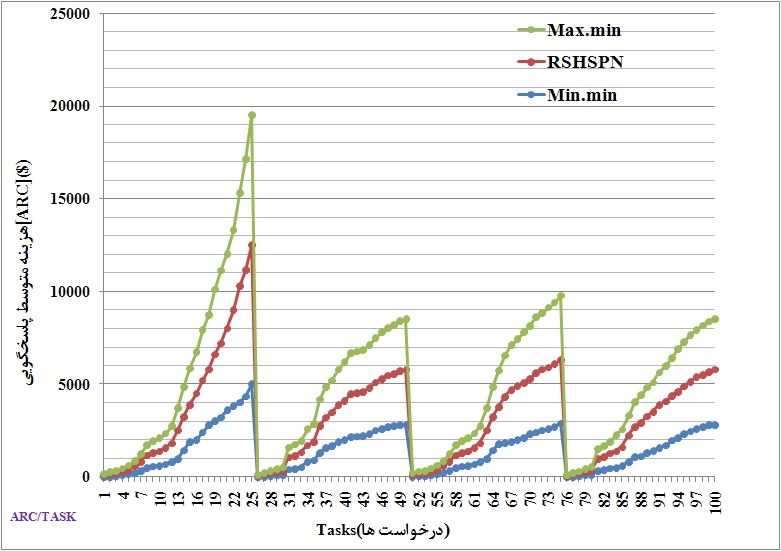
****

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

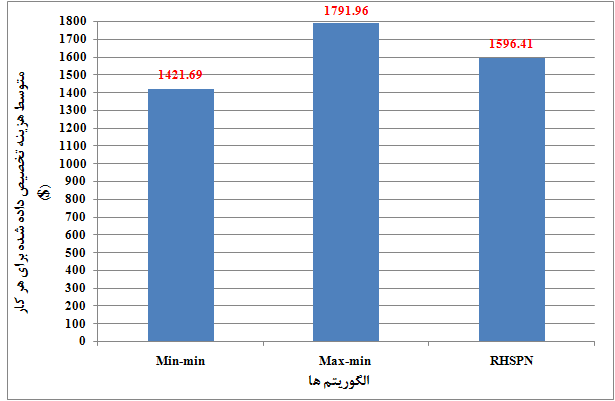
شکل 4-5 میزان هزینه متوسط پاسخگویی برای هر درخواست برای یک کاربر آورده شده است.. این نمودار نشان می دهد برای 25 Task اول (دسته اول) بخاطر عدم در نظر گرفتن هزینه و زمان هر سه الگوریتم مقدار هزینه بالایی را برای هر Task بدون در نظر گرفتن پر یا محتوی بودن آن در نظر می گیرند. اما در دسته دوم Task ها، ما 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 2-1 می باشد.

|  |  |
| --- | --- |
| **رابطه (4-1)** | **Cost(Task)+1%Cost(Task)** |

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 هز اوقات 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 100 Task 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 می­دهد. همچنین، زمانبند گرید که در ساخنار 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 الگوریتم­های هوشمند در بهبود ایده پیشنهادی در زمانبندی و تخصیص منابع می­باشد.

فصل پنجم

**کاربرد شبکه پتری تصادفی تطابقی در گرید اقتصادی**

# پیاده سازی شبکه پتری تصادفی تطابقی در گرید اقتصادی

## مروری بر الگوریتم های موجود و الگوریتم پیشنهادی

الگوريتمی که برای بهينه‌سازی زمان توسط بويا در[92-93]پيشنهاد شده و در [95] مورد ارزيابی قرار گرفته است، الگوريتمی مکاشفه‌ای است که برای کارهای ناهمگون جوابهای قابل قبولی را به کاربر ارائه نمی‌کند. بويا اين الگوريتم را برای زمانبندی برنامه‌های کاربردی پارامترروب طراحی کرده است. از آنجا که ويژگی اين برنامه‌ها اين است که از تعدادی کارهای همگون مستقل از هم تشکيل شده‌اند، الگوريتم گزارش شده فرض می‌کند که کارها دارای ناهمگونی بسيار کمی هستند و به همين دليل برای کارهای ناهمگون نمی‌تواند زمانبندی مناسبی را برای کاربر انجام دهد. بنابراين اين انگيزه بوجود می‌آيد که با استفاده از الگوريتم‌های مکاشفه‌ای جديد و الگوريتم‌های يادگيری بتوان نتايج موجود را بهبود داد. همچنين، بويا منابع را زمان‌مشترک فرض می‌کند و باعث می‌شود که پيچيدگی‌ها و نقص‌هايي در الگوريتم او وجود داشته باشد.

در ادامه، آقای مهدوی فر پس از بررسی و شبيه‌سازی روش‌های مختلف برای بهينه‌سازی زمان، الگوريتم‌هايي را برای اين کار پيشنهاد کرد. ما الگوریتمی تحت عنوان ALATO برای ایجاد توازن بار در تخصیص کارها به منابع گرید محاسباتی ارائه داده ایم. به اين ترتيب، الگوريتم‌های موجود برای بهينه‌سازی زمان عبارتند از:

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

در نتيجه می‌توان گفت که 6 الگوريتم برای بهينه‌سازی زمان در گريدهای محاسباتی ارائه شده است که الگوريتم BTO توسط بويا و الگوریتم های ABTO,EBTO,AEBTO,LATOتوسط آقای مهدوی فر ارائه شده اند و الگوريتمALATO توسط نگارنده پيشنهاد شده است. چهار الگوريتم اول روش‌های مکاشفه‌ای هستند و الگوريتم پنجم و ششم با استفاده از يادگيری و توسط مجموعه‌ای از اتوماتاهای يادگير، عمل زمانبندی را انجام می‌دهد. هدف همه‌ی اين الگوريتم‌ها اين است که بتوانند از بودجه‌ی تعيين شده توسط کاربر، استفاده‌ی حداکثر را کرده و زمان اجرای برنامه را تا جايي که امکان دارد کاهش دهند.الگوريتم ABTO با تغييراتی جزئی در الگوريتم BTO بدست آمده است و بدون دستکاری هسته‌ی اصلی الگوريتم، نتايج بهتری را ارائه می‌کند. الگوريتم EBTO از نظر مکانيزم زمانبندی با الگوريتم BTO متفاوت است ولی از ايده‌ی مطرح شده توسط بويا برای بهينه‌سازی زمان استفاده می‌کند. الگوريتم AEBTO تغييراتی را در الگوريتم EBTO ايجاد می‌کند که به توليد زمانبندی‌های بهتر منجر می‌شود. الگوريتم LATO با استفاده از اتوماتاهای يادگير سعی می‌کند زمان را با صرف بودجه‌ی تعيين شده به حداقل برساند .الگوریتم ALATO با تغییراتی در نحوه پیاده سازی و پرداخت پاداش و جریمه در الگوریتم LATO شکل گرفته است و بهترين نتيجه را در ميان الگوريتم‌های فوق بدست می آورد. در ادامه یک تضیح اجمالی در مورد خصوصیت مشترک الگوریتم ها می پردازیم.

### مراحل مشترک الگوريتم‌ها

الگوريتم‌های بهينه‌سازی، در ابتدای کار و قبل از انجام زمانبندی، اقداماتی را انجام می‌دهند که در همه‌ی آنها مشترک است. اين اعمال را می‌توان در سه مرحله به صورت زير بيان کرد:

1. **يافتن منابع:** شناسايي منابعی که می‌توانند در اجرای کارها مورد استفاده قرار بگيرند و همچنين بدست آوردن توانمندی‌ها و ويژگی‌هاي آنها. اين اعمال از طريق سرويس اطلاعاتی گريد انجام می‌گيرد.
2. **تجارت منابع:** شناسايي هريک از منابع بر حسب هزينه در واحد زمان (پول در ثانيه، G$/sec). سپس تشخيص ميزان توانمندی منبع در واحد زمان (نرخ اجرای ميليون دستورالعمل در ثانيه، MI/sec) و در نهايت بدست آوردن قيمت واقعی و مفيد منبع که همان هزينه‌ی اجرای هر ميليون دستورالعمل می‌باشد (پول به ازای هر ميليون دستوالعمل، G$/MI)
3. **مرتب‌سازی منابع:** يک الگوريتم بهينه‌سازی، حتماً به يک ترتيب خاص از منابع بر حسب قيمت آنها نياز دارد. در اين مرحله، منابع به ترتيب صعودی از نظر قيمت مفيد (که در مرحله‌ی قبل بدست آمده است) مرتب‌سازی می‌شوند. در واقع اولويت واگذاری کار با منابع ارزان‌تر می‌باشد. اگر دو منبع دارای قيمت مفيد يکسان باشند، منبعی که توانمندی بيشتری دارد در اولويت قرار می‌گيرد تا کارايي الگوريتم تا حد ممکن بيشتر باشد. در واقع برای بدست آوردن بهترين ترتيب، منابع با دو پارامتر مرتب می‌شوند: 1. قيمت مفيد 2. توانمندی

### الگوریتم BTO و الگوریتم ABTO

اين الگوريتم توسط بويا در [92,93] پيشنهاد شده و در [95] مورد ارزيابی قرار گرفته است. عمل بهينه‌سازی زمان نسبت به بهينه‌سازی هزينه دارای پيچيدگی بيشتری است و نمی‌توان با يک ايده‌ی ساده، زمانبندی را انجام داد. بويا برای زمانبندی کارها با هدف بهينه‌سازی زمان ايده‌ی زير را مطرح می‌کند:

* با در نظر گرفتن مقداری از بودجه برای هر کار، منبعی را برای يک کار انتخاب می‌کنيم که بتواند اجرای آن کار را زودتر از منابع ديگر به اتمام برساند و هزينه‌ی انجام کار بر روی آن از بودجه‌ی اختصاص يافته به کار کمتر باشد.

ايده‌ی اختصاص دادن بودجه به هر کار تنها عملی است که می‌توان با استفاده از آن بودجه‌ی تعيين شده توسط کاربر را به طور کامل مصرف کرده و بهينه‌سازی زمان را انجام داد. عمل اختصاص بودجه به کارها عمل ساده‌ای نيست و می‌توان اين عمل را با روش‌ها و معيارهای مختلفی انجام داد. در روش BTO اين عمل با استفاده از معيار طول کارها انجام می‌گيرد که يک معيار معقول به نظر می‌رسد. در اين روش هر کار به اندازه‌ی طول خود بودجه دريافت می‌کند. الگوريتم BTO به اين صورت عمل می‌کند که در ابتدا کارها را به طور صعودی از نظر طول مرتب می‌کند.

سپس از اولين 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 بين همه‌ی منابع انتخاب می‌شود. منابع ممکن منابعی هستند که هزينه‌ی اجرای کار بر روی آنها از بودجه‌ی اختصاص يافته به کار کمتر است. سپس از بين اين منابع، کار به منبعی نگاشت می‌شود که بتواند زودتر از منابع ديگر اجرای کار را تمام کند. البته اين منبع لزوماً توانمندترين منبع نيست زيرا قبلاً کارهايي به اين منابع واگذار شده است و بار کاری بيشترين تأثير را در زمان اتمام اجرای يک کار خواهد داشت. البته اگر همه‌ی منابع ممکن دارای بار کاری يکسانی باشند آنگاه مطمئناً منبع انتخابی منبعی است که از ديگران توانمندی بيشتري دارد (نرخ اجرای دستورالعمل در آن از بقيه بيشتر است) زيرا می‌تواند کار را زودتر تمام کند. پس از نگاشت کار می‌دانيم هزينه‌ی اجرای کار بر روی منبع انتخابی از بودجه‌ی اختصاص يافته کمتر است بنابراين هزينه‌ی اجرا از بودجه کسر شده و بقيه‌ی آن به بودجه‌ی کل برگردانده می‌شود. همچنين طول کار نيز از مجموع طول کارها کم می‌شود. سپس الگوريتم زمانبندی به سراغ دومين کار می‌رود و همين عمليات را تکرار می‌کند. اين عمل ادامه می‌يابد تا همه‌ی کارها به منابع نگاشت شوند. با انجام عمل‌های ذکر شده مرحله‌ی زمانبندی به پايان می‌رسد و بعد از آن عمل واگذاری کارها در مرحله‌ی توزيع انجام می‌شود (شکل 5-1). در اين مرحله به هر منبع حداکثر به تعداد پردازنده‌های بيکار آن، کار واگذار می‌شود. همانطور که پيش از اين نيز گفته شد، بويا از زمانبندی مرحله‌ای برای تخصيص منابع استفاده می‌کند. يعنی دو مرحله‌ی زمانبندی و توزيع به عنوان يک مرحله آنقدر تکرار می‌شود تا عمل واگذاری همه‌ی کارها انجام شود. البته بين هر دو تکرار بايد برای مدت زمانی، الگوريتم متوقف شود تا پردازنده‌هاي بيکار برای توزيع کارها بوجود بيايد. جزئيات بيشتر در توضيحات زمانبندی مرحله‌ای آمده است.

تا زمانی که کارهای پردازش نشده وجود دارد و از محدوديت‌های بودجه و مهلت تجاوز نکرده‌ايم تکرار کن:

1. **زمانبندی:** برای هر کار در صف کارهای پردازش نشده انجام بده:
2. با توجه به نسبت طول کار به مجموع طول کارهای باقيمانده، بودجه‌ای را به آن اختصاص بده و از بودجه‌ی کل کم کن.
3. گروهی از منابع را که هزينه‌ی اجرای کار بر روی آنها از بودجه‌ی اختصاص داده شده کمتر است از بقيه‌ی منابع جدا کن.
4. در گروه منابع انتخاب شده، منبعی را که بتواند زودتر از منابع ديگر گروه، اجرای کار را تمام کند انتخاب کرده و کار را به آن منتسب کن.
5. هزينه‌ی اجرای کار بر روی منبع انتخابی را از بودجه‌ی اختصاص داده شده کم کرده و مابقی را به بودجه‌ی کل برگردان.
6. کار را از صف کارهای پردازش نشده حذف کن.
7. **توزيع:** تعداد کارهايي را که می‌توان به يک منبع واگذار کرد، بدون اينکه با حالت overload مواجه شود، مشخص کرده و آنها را واگذار کن. سپس بقيه‌ی کارها را به صف کارهای پردازش نشده برگردان. سياست پيش‌فرض اين است که تعداد کارهای واگذار شده به يک منبع از تعداد پردازنده‌های آن بيشتر نباشد.
8. برای مدت زمان مشخصی الگوريتم را متوقف کن. در اين مدت زمان بايد احتمال بيکار شدن حداقل يکی از پردازنده‌های منابع وجود داشته باشد.

شکل ‏5‑1: الگوریتم BTO

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

|  |  |
| --- | --- |
| (قسمت اول) | #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  int \_tmain(int argc, \_TCHAR\* argv[])  { |
| (قسمت دوم) | for (int j=1;j<200;j++)  { insert(Q1,T1);Order\_ASC\_Q1;}//ascending 200 tasks |
| (قسمت سوم) | 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");  }//for-2  }//for-1  return 0;  }//end-prog |

شکل ‏5‑2: کد برنامه الگوریتم BTO

الگوريتم ABTO تغييری در روند الگوريتم BTO ايجاد نمی‌کند؛ بلکه تنها با توجه خاص به ترتيب بررسی و واگذاری کارها، کارايي الگوريتم را بهبود می‌دهد. در الگوريتم BTO توجه خاصی به ترتيب کارها نمی‌شود زيرا اين الگوريتم برای زمانبندی کارهای همگون طراحی شده است. به هر حال بويا ترتيب صعودی کارها از نظر طول را برای مرتب‌سازی انتخاب می‌کند. در اين الگوريتم صف کارهای پردازش نشده همواره بر حسب طول کارها به صورت صعودی مرتب می‌شود.

همانطور که در ايده‌ی اصلی بهينه‌سازی 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 کنيم و در پايان کارهای بزرگتر را بين منابع توزيع کنيم، اختلاف زمان اجرای کارها در منابع زياد می‌شود و در نتيجه اختلاف زمان اجرا در پرکارترين منبع با منابع ديگر زياد خواهد بود؛ زيرا به علت محدوديت بودجه‌ی باقيمانده نمی‌توان به راحتی شرايط انتقال يک کار بزرگ را از پرکارترين منبع به يک منبع سريع‌تر فراهم کرد و اين وضعيت برای بهينه‌سازی زمان مطلوب نيست. در حالی که اگر ابتدا کارهای بزرگتر را توزيع کنيم و سپس به کارهای کوچکتر بپردازيم، می‌توانيم در پايان زمانبندی، با استفاده از بودجه‌ی باقيمانده، اين کارهای کوچک را طوری بين منابع توزيع کنيم که زمان لازم برای اجرای کارهای واگذار شده در پرکارترين منبع نسبت به منابع ديگر اختلاف زيادی نداشته باشد. بنابراين برای کاهش زمان اجرای کارهای برنامه‌ی کاربردی متعلق به کاربر (زمان اجرای کارهای واگذار شده به پرکارترين منبع) که در واقع هدف يک الگوريتم بهينه‌سازی زمان است، بهتر است که ترتيب واگذاری کارها از نظر طول کارها به صورت نزولی باشد.

برای رسيدن به کارايي بالاتر، الگوريتم ABTO نسبت به الگوريتم BTO دارای يک عمل اضافه با تأکيد خاص خواهد بود:

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

با افزودن عمل فوق به الگوريتم BTO، به الگوريتم ABTO دست پيدا می‌کنيم که دارای کارايي بالاتری خواهد بود. اين الگوريتم نيز از روش زمانبندی مرحله‌ای استفاده می‌کند و تنها تفاوت آن با الگوريتم پيشنهادی بويا در ترتيب 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

### الگوریتم EBTO

الگوريتم EBTO از تغيير الگوريتم BTO و استفاده از روش زمانبندی يکباره حاصل می‌شود؛ البته همچنان از ايده‌ی اصلی اين الگوريتم استفاده می‌کند و روش اختصاص بودجه به کارها را نيز تغيير نمی‌دهد. همانطور که گفته شد، يکی از نقاط ضعف الگوريتم BTO اين است که عمل زمانبندی و نگاشت کارهای پردازش نشده را در هنگام اجرای کارهای واگذار شده به منابع انجام می‌دهد و در واقع از روش زمانبندی مرحله‌ای استفاده می‌کند. ولی الگوريتم EBTO از روش زمانبندی يکباره استفاده می‌کند که در آن فرض می‌شود همه‌ی منابع به صورت فضامشترک رفتار می‌کنند. همانطور که قبلاً گفته شد اين روش تنها راهکار قابل استفاده در گريدهای اقتصادی برای زمانبندی کارهای متعلق به چندين کاربر مختلف به صورت همزمان می‌باشد. الگوريتم EBTO سعی می‌کند با استفاده از ايده‌ی الگوريتم BTO، پس از نگاشت هر کار به يک منبع، بلافاصله عمل واگذاری آن را به منبع انتخابی انجام دهد. به اين ترتيب با زمانبندی يکباره، امکان سرويس‌دهی همزمان به چندين کاربر را فراهم می‌کند. همچنين از آنجا که اين الگوريتم، کارها را به صورت نزولی مرتب می‌کند، از نظر کارايي نيز الگوريتم BTO را بهبود می‌دهد.

الگوريتم EBTO در ابتدا کارها را از نظر طول به صورت نزولی مرتب می‌کند. سپس با استفاده از روشی که بويا برای اختصاص بودجه به کارها مطرح کرده بود، اولين کار را از مجموعه‌ی کارها جدا کرده و بودجه‌ای را به آن اختصاص می‌دهد. سپس از بين منابعی که می‌توانند اين کار را با هزينه‌ای کمتر از بودجه‌ی اختصاص يافته به آن اجرا کنند، منبعی را انتخاب می‌کند که بتواند اجرای کار را زودتر از بقيه تمام کند و کار را به اين منبع واگذار می‌کند. اين عمل برای همه‌ی کارها به ترتيب انجام می‌شود تا واگذاری همه‌ی آنها به اتمام برسد.

1. کارها را به صورت نزولی از نظر طول مرتب کن.
2. برای هر کار به ترتيب انجام بده:
3. با توجه به نسبت طول کار به مجموع طول کارهای باقيمانده، بودجه‌ای را به آن اختصاص بده و از بودجه‌ی کل کم کن.
4. گروهی از منابع را که هزينه‌ی اجرای کار بر روی آنها از بودجه‌ی اختصاص داده شده، کمتر است، از بقيه‌ی منابع جدا کن.
5. در گروه منابع انتخاب شده، منبعی را که بتواند زودتر از منابع ديگر گروه، اجرای کار را تمام کند انتخاب کرده و کار را به آن نگاشت کن.
6. هزينه‌ی اجرای کار بر روی منبع انتخابی را از بودجه‌ی اختصاص داده شده کم کرده و مابقی را به بودجه‌ی کل برگردان.
7. کار را از صف کارها خارج کن.

شکل ‏5‑4: الگوريتم EBTO

همانطور که در شکل 5-4 مشاهده می‌شود، الگوريتم EBTO با توجه به طول کارها به آنها بودجه تخصيص می‌دهد. در واقع نسبت طول کار به مجموع طول کارهايي که در صف قرار دارند برابر است با نسبت بودجه‌ی اختصاص يافته برای کار به کل بودجه‌ی باقيمانده. اين، همان روشی است که بويا در الگوريتم BTO برای تخصيص بودجه ارائه کرده بود. در اين روش کارهای بزرگتر به نسبت طولشان بودجه‌ی بيشتری دريافت می‌کنند. می‌دانيم اگر کارهای بزرگتر در ابتدا و کارهای کوچکتر در انتهای صف کارهای واگذار شده در منابع قرار بگيرند می‌توان با جابجايي کارهای کوچک، تعادل کاری بين منابع برقرار کرده و زمان را در پرکارترين منبع کاهش داد

به همين دليل بهتر است که به کارهای بزرگ، بيشتر و به کارهای کوچک کمتر از نسبت طولشان بودجه تخصيص بدهيم. الگوريتم AEBTO که در بخش بعد خواهد آمد يک روش جديد برای تخصيص بودجه ارائه می‌کند که به افزايش کارايي الگوريتم منجر می شود.

### الگوریتم AEBTO

نحوه‌ی تخصيص بودجه به کارها در کارايي يک الگوريتم زمانبندی با هدف بهينه‌سازی زمان تأثير زيادی دارد. مجموع هزينه‌های اجرای کارها نبايد از بودجه‌ی تعيين شده توسط کاربر فراتر رود زيرا در اين صورت زمانبندی بدست آمده ارزشی نخواهد داشت. از طرف ديگر نحوه‌ی تخصيص بودجه بايد طوری باشد که مجموع هزينه‌های صرف شده برای کارها تا حد ممکن به بودجه‌ی تعيين شده توسط کاربر نزديک باشد تا بتوان حداکثر استفاده را از آن داشت. روشی که بويا برای اين کار ارائه کرده است دارای اين ويژگی‌ها می‌باشد. اولاً در اين روش به هر کار با توجه به نسبت طول آن به مجموع طول کارها بودجه اختصاص می‌يابد بنابراين از آنجا که هيچ کاری نمی‌تواند بيش از بودجه‌ای که دارد هزينه صرف کند، تضمين می‌شود که مجموع هزينه‌های صرف شده از بودجه‌ی تعيين شده کمتر باشد. ثانياً می‌دانيم هر کار، منبعی را انتخاب می‌کند که هزينه‌ی اجرایش در آن منبع از بودجه‌ی اختصاص يافته کمتر باشد. بنابراين مقداری از بودجه برای اجرای کار بر روی منبع انتخابی اضافه است. در روش بويا، کارها اين مقدار اضافه را به بودجه‌ی باقيمانده برمی‌گردانند تا در هنگام تخصيص بودجه به کارهای ديگر استفاده شود. به اين ترتيب سعی می‌شود مجموع هزينه‌های اجرای کارها تا حد ممکن به مقدار بودجه‌ی تعيين شده نزديک باشد و در پايان عمليات زمانبندی همه‌ی کارها، مقدار کمی از بودجه باقی بماند.

الگوريتم AEBTO با تغيير دادن روش پيشنهادی بويا برای تخصيص بودجه باعث می‌شود کارايي نسبت به الگوريتم EBTO که از روش تخصيص بودجه‌ی بويا استفاده می‌کند افزايش يابد. می‌دانيم برای اينکه بتوان زمان اجرا در پرکارترين منبع را کاهش داد بايد نوعی تعادل بين زمان اجراهای منابع استفاده شده ايجاد کرد. برای اين عمل بايد هنگامی که در حال زمانبندی کارهای پايانی هستيم طوری آنها را بين منابع توزيع کنيم که اين تعادل برقرار شود. زمانی می‌توان اين اقدام را به خوبی انجام داد که قبلاً نوعی تعادل در کارهايي که بين منابع توزيع شده است وجود داشته باشد. يعنی کارهای بزرگ به يک منبع و کارهای کوچک به يک منبع ديگر واگذار نشده باشد. از طرف ديگر می‌دانيم کارهای بزرگ برای اجرا بر روی منابع گران نسبت به کارهای کوچک به بودجه‌ی بيشتری نياز دارند. بنابراين اگر بتوانيم به کارهای بزرگتر بودجه‌ی بيشتری نسبت به طولشان بدهيم می‌توان اميدوار بود که کارهای بزرگ و کوچک به صورت تقريباً يکسان بر روی منابع ارزان و گران توزيع می‌شوند. در حالی که در روش پيشنهادی بويا برای تخصيص بودجه، کارهای بزرگ تنها می‌توانستند به منابع ارزانتر واگذار شوند و صف کارها در منابع گران پر از کارهای کوچک می‌شد.

روش تخصيص بودجه‌ی پيشنهادی ما برای الگوريتم AEBTO سعی می‌کند از مقداری که از بودجه‌ی اختصاص يافته به هر کار زياد می‌آيد برای تخصيص بودجه به کارهای بزرگ استفاده کند. اين الگوريتم در ابتدا صف کارها را از نظر طول به صورت نزولی مرتب می‌کند. سپس هنگامی که همانند الگوريتم EBTO در حال زمانبندی کارها به ترتيب می باشد، به جای اينکه مقدار اضافی بودجه‌ی اختصاص يافته به هر کار را به بودجه‌ی کل برگرداند، اين مقدار را به بودجه‌ی کار بعدی اضافه می‌کند. از آنجا که در ابتدا کارهای بزرگتر زمانبندی می‌شوند، اين عمل باعث می‌شود کارهای بزرگ بودجه‌ی بيشتری نسبت به طولشان دريافت کنند و بتوانند بر روی منابع گرانتر نيز اجرا شوند. جزئيات الگوريتم AEBTO در شکل 5-5 آمده است.

1. کارها را به صورت نزولی از نظر طول مرتب کن.
2. برای هر کار به ترتيب انجام بده
3. با توجه به نسبت طول کار به مجموع طول کارهای باقيمانده، بودجه‌ای را به آن اختصاص بده و از بودجه‌ی کل کم کن.
4. مقداری را که از بودجه‌ی کار قبل باقيمانده است، به بودجه‌ی اختصاص يافته به اين کار، اضافه کن.
5. گروهی از منابع را که هزينه‌ی اجرای کار بر روی آنها از بودجه‌ی اختصاص داده شده، کمتر است، از بقيه‌ی منابع جدا کن.
6. در گروه منابع انتخاب شده، منبعی را که بتواند زودتر از منابع ديگر گروه، اجرای کار را تمام کند، انتخاب کرده و کار را به آن نگاشت کن.
7. هزينه‌ی اجرای کار بر روی منبع انتخابی را از بودجه‌ی اختصاص داده شده، کم کرده و مابقی را برای کار بعدی ذخيره کن.
8. کار را از صف کارها خارج کن.

شکل ‏5‑5: الگوريتم AEBTO

بايد توجه کرد که روش تخصيص بودجه‌ای که در اينجا پيشنهاد شده است هر دو ويژگی يک روش تخصيص خوب را داراست. اولاً با توجه به بودجه‌های اختصاص يافته به کارها هيچ گاه هزينه‌ی انجام کارها از مقدار بودجه‌ی تعيين شده توسط کاربر فراتر نمی‌رود. زيرا مقدار بودجه‌ی اختصاصی هر کار از بودجه‌ی کل کارها کم می‌شود و هيچ مقداری نيز به بودجه‌ی کل اضافه نمی‌شود. ثانياً مقدار اضافه‌ی بودجه‌ی اختصاص يافته به هر کار، به کار بعدی داده می‌شود و به اين ترتيب سعی می‌شود از بودجه‌ی تعيين شده به طور کامل استفاده شود.

### الگوریتم LATO

الگوريتم LATO سعی می‌کند با استفاده از مجموعه‌ای از اتوماتاهای يادگير عمل پيچيده‌ی بهينه‌سازی زمان را با کارايي بهتری نسبت به الگوريتم‌های مکاشفه‌ای انجام دهد. اين الگوريتم برای زمانبندی کارهای برنامه‌ی کاربردی متعلق به کاربر، هر کدام از کارها را به يک اتوماتای يادگير مجهز می‌کند. اين اتوماتاهای يادگير دارای ساختار متغير بوده و اعمال آنها منابع گريد می‌باشد. در واقع اتوماتای يادگير متعلق به يک کار، منبعی را که کار بايد به آن واگذار شود، انتخاب می‌کند. هنگامی که همه‌ی کارها به ترتيب، منابع مورد نظر خود را انتخاب کردند، از طرف محيط پاداش گرفته يا جريمه می‌شوند. سپس برای بار ديگر، همه‌ی کارها عمل انتخاب منبع را انجام می‌دهند تا پاسخ محيط را دريافت کنند. اين عمل تا تعداد معينی تکرار می‌شود تا همه‌ی کارها منبع مناسب خود را بيابند و به آن واگذار شوند. الگوريتم LATO همانند الگوريتم‌های پيشنهادی ديگر، از روش زمانبندی يکباره استفاده می‌کند و صف کارها را از نظر طول به صورت نزولی مرتب می‌کند.

در الگوريتم LATO نيز در هر تکرار، زمانبندی ايجاد شده بايد به طور موقت به منابع اِعمال شود تا بتوان آن را ارزيابی کرد. برای اين کار، هر منبع دارای يک صف موقت به نام صف انتساب است که در ابتدای هر تکرار خالی می‌شود و سپس هنگامی که کارها به ترتيب، در حال انتخاب منبع هستند، بلافاصله در صف انتساب منبع انتخاب شده قرار می‌گيرند.

پاسخ محيط به اتوماتاهای يادگير متعلق به کارها، با توجه به شرايط ايده‌آل در يک بهينه‌سازی زمان انجام می‌گيرد. برای اين منظور، محيط از سه نکته‌ی زير برای دادن پاداش يا جريمه به کارها استفاده می‌کند:

1. مجموع هزينه‌ی اجرای کارها نبايد از بودجه‌ی تعيين شده توسط کاربر بيشتر باشد.
2. زمان اجرای کارهای واگذار شده به پرکارترين منبع نبايد با زمان اجرا در منابع ديگر اختلاف زيادی داشته باشد.
3. هر کار بايد به منبعی واگذار شود که بتواند اجرای آن کار را زودتر از بقيه‌ی منابع به اتمام برساند.

به اين ترتيب محيط با استفاده از دو نکته‌ی اول، کارها را جريمه می‌کند و با توجه به نکته‌ی سوم به کارها پاداش می‌دهد. با توجه به نکته‌ی اول، اگر مجموع هزينه‌های اجرای کارها بر روی منابع انتخاب شده توسط آنها از مقدار بودجه‌ی تعيين شده فراتر رود محيط کارهايي را که هزينه‌ی زيادی برای اجرا لازم دارند جريمه می‌کند. در واقع عمل جريمه کردن از انتهای صف کارها در گران‌ترين منبع استفاده شده (گران‌ترين منبعی که حداقل يک کار به آن منتسب شده است) شروع می‌شود. با جريمه شدن هر کار و حذف آن از صف انتساب منبع انتخابی، اختلاف هزينه‌ی اجرای آن بر روی اين منبع با هزينه‌ی اجرای آن بر روی ارزان‌ترين منبع موجود، از مقدار کل هزينه‌های اجرا کسر می‌شود. عمل جريمه دادن تا زمانی ادامه می‌يابد که مقدار کل هزينه‌ها از مقدار بودجه‌ی تعيين شده بيشتر باشد. در واقع با اين جريمه، کارها به سمت استفاده از منابع ارزانتر هدايت می‌شوند.

با توجه به نکته‌ی دوم، محيط جريمه کردن آخرين کار در صف انتساب پرکارترين منبع را در نظر می‌گيرد. اگر يکی از منابع استفاده شده در زمانبندی (منابعی که حداقل يک کار منتسب شده دارند) به جز آخرين (گران‌ترين) منبع استفاده شده، بتواند اين کار را زودتر از پرکارترين منبع پردازش کند، محيط اين کار را جريمه می‌کند تا در تکرارهای بعدی منبع مناسب‌تری را برای خود انتخاب کند. اين عمل باعث می‌شود کارها به صورت متعادل بين منابع توزيع شوند.

در نهايت محيط با توجه به نکته‌ی سوم، به کارها پاداش می‌دهد. کارهايي که توانسته‌اند زمان اتمام کمتر يا برابر نسبت به تکرار قبل برای خود بدست بياورند پاداش می‌گيرند؛ البته به شرطی که در همين تکرار جريمه نشده باشند. به اين ترتيب هر کار تشويق می‌شود منبعی را برای واگذاری خود انتخاب کند که در کمترين زمان، اجرایش را به اتمام برساند.

1. **مرتب‌سازی:** کارها را از نظر طول به صورت نزولی مرتب کن.
2. **يادگيری:** به تعداد 10000 بار تکرار کن:
3. صف‌های انتساب منابع را خالی کن.
4. **انتخاب:** برای هر کار به ترتيب انجام بده:
5. با استفاده از اتوماتای يادگير مربوطه، منبعی را برای واگذاری، انتخاب کن.
6. کار را در صف انتساب منبع انتخابی قرار بده.
7. **جريمه (1):** تا هنگامی که مجموع هزينه‌ی اجرای کارها بر روی منابع انتخاب شده از بودجه‌ی تعيين شده بيشتر است تکرار کن:
8. از انتهای صف انتساب در گران‌ترين منبع استفاده شده، يک کار را جدا کن.
9. اختلاف هزينه‌ی اجرای کار بر روی اين منبع با هزينه‌ی اجرای کار بر روی ارزان‌ترين منبع موجود را از مجموع هزينه‌ی اجرای کارها کم کن.
10. کار را جريمه کن (مقدار جريمه: 0.01).
11. **جريمه (2):** آخرين کار منتسب شده به پرکارترين منبع را در نظر بگير. اگر يکی از منابع استفاده شده به جز منبع آخر (گران‌ترين منبع استفاده شده) می‌تواند اجرای اين کار را زودتر تمام کند اين کار را جريمه کن (مقدار جريمه: 0.01).
12. **پاداش:** برای هر کار انجام بده:

اگر جريمه نشده است: اگر زمان اتمام اجرای آن بر روی منبع انتخابی از زمان اتمام در تکرار قبل کمتر يا با آن برابر است به آن پاداش بده (مقدار پاداش: 0.1).

1. **زمانبندی:** برای هر کار به ترتيب، با استفاده از اتوماتای يادگير متعلق به آن، منبعی را انتخاب کرده و کار را به آن منبع، نگاشت کن.

شکل ‏5‑6: الگوريتم LATO

همانطور که در شکل 5-6 آمده است، الگوريتم LATO در ابتدا کارها را بر حسب طولشان به صورت نزولی مرتب می‌کند. سپس در تکرارهای زياد، منبع مناسب برای اجرای هر کار را پيدا می‌کند. اين الگوريتم در نهايت کارها را به آن منابع واگذار کرده و از روش زمانبندی يکباره استفاده می‌کند. محيط در هنگام يادگيری الگوريتم از دو پاسخ جريمه و يک پاسخ پاداش استفاده می‌کند. از آنجا که اگر يک کار در هيچ کدام از دو حالت جريمه نشده باشد و همچنين زمان اتمام کوچکتر يا برابر نسبت به تکرار قبل بدست آورده باشد، احتمالاً منبع مناسبی را برای خود پيدا کرده است، مقدار پاداش در اين حالت (0.1) از مقادير جريمه‌ها در دو حالت اول (0.01) بيشتر است.

### الگوریتم مینیمم هزینه محاسبات

اجرای يک برنامه‌ی کاربردی بر روی گريد اقتصادی با صرف بودجه‌ای که توسط کاربر فراهم شده است، دارای يک کران پايين يا مينيمم برای زمان محاسبات می‌باشد. مينيمم نشان می‌دهد کاربر هنگامی که حاضر است بودجه‌ی معينی را برای اجرای برنامه‌ی کاربردی خود پرداخت کند، حداقل زمانی که بايد برای دريافت نتيجه صبر کند چقدر است. البته در بيشتر موارد، هيچ زمانبندی وجود ندارد که بتواند کارها را در مدت زمان مينيمم به کاربر تحويل دهد و الگوريتم‌های زمانبندی سعی می‌کنند تا حد ممکن زمان محاسبات را به زمان مينيمم نزديک کنند.

برای بدست آوردن مينيمم زمان محاسبات می‌توان از الگوريتم‌های ارائه شده برای بهينه‌سازی زمان استفاده کرد‌ که به جای اينکه کارها را زمانبندی کنيم عمل زمانبندی را برای دستوالعمل‌ها به عنوان کوچکترين واحد اجرايي انجام می‌دهيم. يعنی به هر ميليون دستورالعمل، بودجه تخصيص می‌دهيم و آن را برای اجرا به منبع مناسب واگذار می‌کنيم. روش ديگر برای بدست آوردن مينيمم زمان محاسبات اين است که از روش ارائه شده برای بدست آوردن مينيمم هزينه استفاده کنيم. برای اين کار روش مينيمم هزينه را با در نظر گرفتن مهلت اجرا می‌کنيم و هزينه‌ی بدست آمده را بررسی می‌کنيم. سپس از مهلت مقداری کم می‌کنيم و دوباره روش مينيمم هزينه را اجرا می‌کنيم. اين عمل را آنقدر تکرار می‌کنيم تا به مهلتی برسيم که مينيمم هزينه‌ی انجام کارها در آن مهلت با بودجه‌ی تعيين شده تقريباً برابر شود. در اين حالت، مهلت استفاده شده در روش مينيمم هزينه، همان مينيمم زمان محاسبات خواهد بود. ما از اين روش برای بدست آوردن مينيمم زمان اجرای کارها با صرف بودجه‌ی تعيين شده توسط کاربر، استفاده می‌کنيم.

### الگوریتم پیشنهادی ALATO

الگوريتم پيشنهادی در اين پايان‌نامه، از روش زمانبندی يکباره استفاده می‌کند و در هنگام ورود کاربر به سيستم، در صورتی که انجام برنامه‌ی کاربردی او در محدوده‌ی مهلت و بودجه‌ی تعيين شده، ممکن باشد، کاربر را می‌پذيرد و اتمام برنامه را با رعايت اين محدوده‌ها تضمين می‌کند. برای اين منظور، در پايان الگوريتم پيشنهادی و پس از مشخص شدن زمانبندی (نگاشت کارها به منابع)، يک مرحله‌ی کنترل پذيرش[[232]](#footnote-233) ساده وجود دارد که مجموع زمان انجام کارها و هزینه اجرای برنامه که به طور تقريبی در زمانبندی، بدست آمده‌اند را بررسی می‌کند. در صورت پذيرش کاربر، مرحله‌ی توزيع انجام می‌گيرد که در آن، کارها به منابع مشخص شده در زمانبندی، واگذار می‌شوند.

بنابر این، الگوريتم ALATO سعی می‌کند با استفاده از مجموعه‌ای از اتوماتاهای يادگير عمل پيچيده‌ی بهينه‌سازی زمان را با کارايي بهتری نسبت به الگوريتم‌های مکاشفه‌ای انجام دهد. اين الگوريتم از مجموعه‌ای از اتوماتاهای يادگير استفاده می‌کند که به کارها نسبت داده می‌شوند. اين اتوماتاهای يادگير دارای ساختار متغير بوده و اعمال آنها منابع گريد می‌باشد. در واقع اتوماتای يادگير متعلق به يک کار، منبعی را که کار بايد به آن واگذار شود، انتخاب می‌کند. هنگامی که همه‌ی کارها به ترتيب، منابع مورد نظر خود را انتخاب کردند، از طرف محيط پاداش گرفته يا جريمه می‌شوند. سپس برای بار ديگر، همه‌ی کارها عمل انتخاب منبع را انجام می‌دهند تا پاسخ محيط را دريافت کنند. اين عمل تا تعداد معينی تکرار می‌شود تا همه‌ی کارها منبع مناسب خود را بيابند و به آن واگذار شوند. الگوريتم ALATO همانند الگوريتم‌های پيشنهادی ديگر، از روش زمانبندی يکباره استفاده می‌کند و صف کارها را از نظر طول به صورت نزولی مرتب می‌کند.

الگوريتم ALATO نيز همانند الگوريتم‌های ديگری که در بخش‌ قبلی ارائه شد فرض می‌کند منابع به صورت فضامشترک رفتار می‌کنند و عمل زمانبندی را به صورت يکباره انجام می‌دهد. همچنين، کارها را به صورت نزولی مرتب می‌کند تا زمان بيکاری منابع گران‌تر را کاهش دهد. الگوريتم ALATO نمی‌تواند تنها با يکبار بررسی کارها و منابع به زمانبندی نهايي برسد؛ بلکه با تکرارهای زيادی اين عمل را انجام می‌دهد و در هر تکرار يک زمانبندی جديد را ايجاد می‌کند. محيط به کارها با توجه به منابعی که انتخاب کرده‌اند، پاسخ (پاداش يا جريمه) می‌دهد. کارها از اين پاسخ محيط، در تکرار بعدی استفاده می‌کنند تا زمانبندی جديدی که ايجاد می‌شود به زمانبندی بهينه نزديکتر باشد. اين تکرارها به تعداد زيادی انجام می‌شود تا همه‌ی کارها منبع مناسب خود را پيدا کنند و تغيير ديگری در تصميم خود ندهند.

در واقع، تا جايي تکرارها ادامه می‌يابد که همه‌ی اتوماتاهای يادگير همگرا شده و زمانبندی يکسانی در تکرارها ايجاد شود. اين زمانبندی يکسان، همان زمانبندی نهايي است که بر اساس آن، کارها به منابع واگذار می‌گردد.

در الگوريتم ALATO نيز در هر تکرار، زمانبندی ايجاد شده بايد به طور موقت به منابع اِعمال شود تا بتوان آن را ارزيابی کرد. برای اين کار، هر منبع دارای يک صف موقت به نام صف انتساب است که در ابتدای هر تکرار خالی می‌شود و سپس هنگامی که کارها به ترتيب، در حال انتخاب منبع هستند، بلافاصله در صف انتساب منبع انتخاب شده قرار می‌گيرند.

پاسخ محيط به اتوماتاهای يادگير متعلق به کارها، با توجه به شرايط ايده‌آل در يک بهينه‌سازی زمان انجام می‌گيرد. برای اين منظور، محيط از سه نکته‌ی زير برای دادن پاداش يا جريمه به کارها استفاده می‌کند:

1. مجموع هزينه‌ی اجرای کارها نبايد از بودجه‌ی تعيين شده توسط کاربر بيشتر باشد.
2. زمان اجرای کارهای واگذار شده به پرکارترين منبع نبايد با زمان اجرا در منابع ديگر اختلاف زيادی داشته باشد.
3. هر کار بايد به منبعی واگذار شود که بتواند اجرای آن کار را زودتر از بقيه‌ی منابع به اتمام برساند.

به اين ترتيب محيط با استفاده از دو نکته‌ی اول، کارها را جريمه می‌کند و با توجه به نکته‌ی سوم به کارها پاداش می‌دهد. با توجه به نکته‌ی اول، اگر مجموع هزينه‌های اجرای کارها بر روی منابع انتخاب شده توسط آنها از مقدار بودجه‌ی تعيين شده فراتر رود محيط کارهايي را که هزينه‌ی زيادی برای اجرا لازم دارند جريمه می‌کند. در واقع عمل جريمه کردن از انتهای صف کارها در گران‌ترين منبع استفاده شده (گران‌ترين منبعی که حداقل يک کار به آن منتسب شده است) شروع می‌شود. با جريمه شدن هر کار و حذف آن از صف انتساب منبع انتخابی، اختلاف هزينه‌ی اجرای آن بر روی اين منبع با هزينه‌ی اجرای آن بر روی ارزان‌ترين منبع موجود، از مقدار کل هزينه‌های اجرا کسر می‌شود.

عمل جريمه دادن تا زمانی ادامه می‌يابد که مقدار کل هزينه‌ها از مقدار بودجه‌ی تعيين شده بيشتر باشد. در واقع با اين جريمه، کارها به سمت استفاده از منابع ارزانتر هدايت می‌شوند. هدف در این مرحله، پایین آوردن سریع هزینه و رساندن آن به میزانمقدار کمتر از مهلت در نظر گرفته شده با انتقال کارها از گرانترین منبع به ارزانترین منبع می باشد.

با توجه به نکته‌ی دوم، محيط جريمه کردن کارها در صف انتساب پرکارترين منبع را در نظر می‌گيرد. این نکته را توجه داشته باشید که لزوما پرکارترین منبع گرانترین منبع نیست. از انتهای صف پرکارترین منبع شروع می کنیم. به تعدادY کار از انتهای صف پرکارترین منبع منابع را انتخاب می کنیم. اگر يکی از منابع استفاده شده در زمانبندی (منابعی که حداقل يک کار منتسب شده دارند) به جز آخرين (گران‌ترين) منبع استفاده شده، بتواند اين کار را زودتر از پرکارترين منبع پردازش کند، محيط اين کار را جريمه می‌کند تا در تکرارهای بعدی منبع مناسب‌تری را برای خود انتخاب کند، در غیر این صورت این کار هیچ جریمه ای نمی شود. این عمل برای همهY کار از پرکارترین منبع اعمال می شود. اين عمل باعث می‌شود کارها باتعادل و توازن بسیار بیشتری بین منابع توزیع گردند. مقدار­Y با توجه به جدول 5-1 از رابطه­های (5-1)و (5-2) محاسبه می شود:

جدول ‏5‑1: 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 |

|  |  |
| --- | --- |
| **رابطه (5-1)** |  |
| که X میزان کارهای متوسط اختصاص داده شده بازای R-1 منبع می باشد. در واقع تعداد کارهایی که باید بر روی همه منابع بغیر از گرانترین منبع پخش شود.(اختصاص دو­باره کارها بر روی منابه بغیر از گرانترین منبع، چونکه، گرانترین منبع هزینه اجرا را افزایش می دهد.) | |
| **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 برروی منابع می باشد و تعداد کارهای صف پرکارترین منبع بیشتراز کارهای سایر منابع یا میزان متوسط کارها می باشد.  مقدار  میزان متوسط منابع قابل اجرا بغیر از گرانترین منبع و پرکارترین منبع به منابع موجود غیر از گرانترین منبع می باشد. | |

در نهايت محيط با توجه به نکته‌ی سوم، به کارها پاداش می‌دهد. کارهايي که توانسته‌اند زمان اتمام کمتر يا برابر نسبت به تکرار قبل برای خود بدست بياورند پاداش می‌گيرند؛ البته به شرطی که در همين تکرار جريمه نشده باشند. به اين ترتيب هر کار تشويق می‌شود منبعی را برای واگذاری خود انتخاب کند که در کمترين زمان، اجرایش را به اتمام برساند.

1. **مرتب‌سازی:** کارها را از نظر طول به صورت نزولی مرتب کن.
2. **يادگيری:** به تعداد 10000 بار تکرار کن:
3. صف‌های انتساب منابع را خالی کن.
4. **انتخاب:** برای هر کار به ترتيب انجام بده:
5. با استفاده از اتوماتای يادگير مربوطه، منبعی را برای واگذاری، انتخاب کن.
6. کار را در صف انتساب منبع انتخابی قرار بده.
7. **جريمه (1):** تا هنگامی که مجموع هزينه‌ی اجرای کارها بر روی منابع انتخاب شده از بودجه‌ی تعيين شده بيشتر است تکرار کن:
8. از انتهای صف انتساب در گران‌ترين منبع استفاده شده، يک کار را جدا کن.
9. اختلاف هزينه‌ی اجرای کار بر روی اين منبع با هزينه‌ی اجرای کار بر روی ارزان‌ترين منبع موجود را از مجموع هزينه‌ی اجرای کارها کم کن.
10. کار را جريمه کن (مقدار جريمه: 0.01).
11. **جريمه (2):** به تعداد Yبار تکرار کن:
12. کار آخر منتسب شده به پرکارترين منبع را در نظر بگير.
13. اگر يکی از منابع استفاده شده به جز منبع آخر (گران‌ترين منبع استفاده شده) می‌تواند اجرای اين کار را زودتر تمام کند اين کار را جريمه کن (مقدار جريمه: 0.02).
14. **پاداش:** برای هر کار انجام بده:

اگر جريمه نشده است: اگر زمان اتمام اجرای آن بر روی منبع انتخابی از زمان اتمام در تکرار قبل کمتر يا با آن برابر است به آن پاداش بده (مقدار پاداش: 0.1).

1. **زمانبندی:** برای هر کار به ترتيب، با استفاده از اتوماتای يادگير متعلق به آن، منبعی را انتخاب کرده و کار را به آن منبع، نگاشت کن.

شکل ‏5‑7: الگوريتم ALATO

همانطور که در شکل 5-7 آمده است، الگوريتم ALATO در ابتدا کارها را بر حسب طولشان به صورت نزولی مرتب می‌کند. سپس در تکرارهای زياد، منبع مناسب برای اجرای هر کار را پيدا می‌کند. اين الگوريتم در نهايت کارها را به آن منابع واگذار کرده و از روش زمانبندی يکباره 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، محيط در هنگام يادگيری الگوريتم از دو پاسخ جريمه و يک پاسخ پاداش استفاده می‌کند. از آنجا که اگر يک کار در هيچ کدام از دو حالت جريمه نشده باشد و همچنين زمان اتمام کوچکتر يا برابر نسبت به تکرار قبل بدست آورده باشد، احتمالاً منبع مناسبی را برای خود پيدا کرده است، مقدار پاداش در اين حالت (0.1) از مقادير جريمه‌ها در حالت اول (0.01) و در حالت دوم (0.02) بيشتر است. علت بیشتر بودن مقدار جریمه در حالت دوم بعلت قرار گیری کارهای کوچکتر در پرکارترین منبع و افزایش تراکم در آن منبع می باشد.در واقع، با افزایش جریمه کارها به سمت قرار گیری در منابع کم کارتر و افزایش تعادل بار در هر تکرار گام بر می دارند و سرعت همگرا شدن اتوماتا بصورت چشمگیری افزایش می یابد. کد برنامه در ضمیمه آورده شده است.

اتوماتای یادگیر مدل S را با مقادیر مختلفی برای نرخ پاداش و جریمه تست کرده­ایم، در روش­های مدل S، اگر در تکرار *n*اُم عمل انتخاب شده باشد، پاسخ (پاسخ نامطلوب) محيط به آن برابر  می­باشد و پاسخ (پاسخ مطلوب) محيط به آن عمل برابر  می­باشد، (یعنی به نسبت انتخاب منبع ارزان­تر پاسخ بهتری از محیط دریافت می­کند). که طبق نتایج که در فصل شبیه­سازی 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.

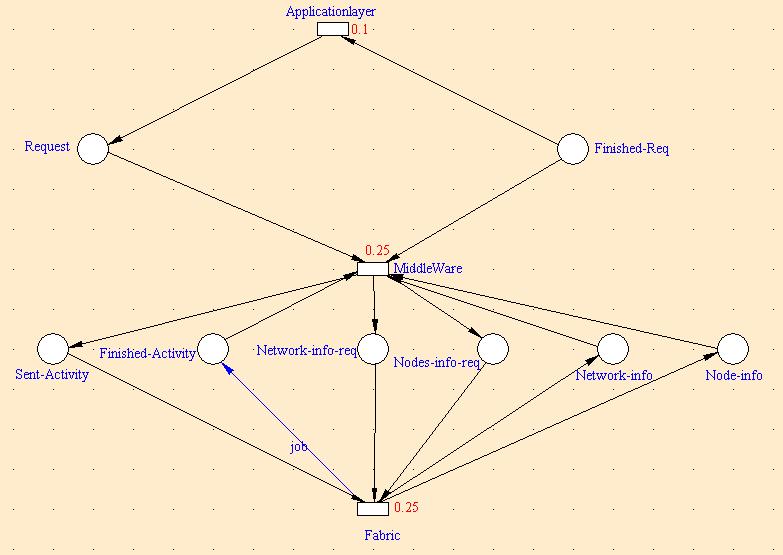
## شبیه سازی الگوریتم ها

### مقدمه

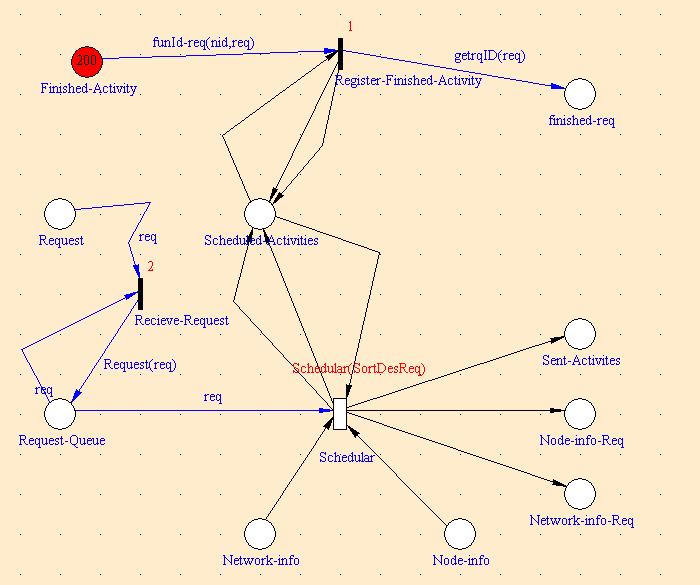
در اينجا ابتدا به شرح کاربرد مدل شبکه پتری تصادفی نطابقی پیشنهاد در محیط گرید و بررسی و آنالیز آن می­پردازیم، سپس کارايي الگوريتم­های ارائه شده در بخش‌های قبلی را که زمانبندی کارها را با اتخاذ استراتژی بهينه‌سازی زمان در يک گريد اقتصادی انجام می‌دادند ارزيابی می‌کنيم. اين الگوريتم‌ها سعی می‌کنند در يک محيط گريد اقتصادی، عمل زمانبندی تعدادی کار مستقل از هم را انجام دهند. اين کارها متعلق به يک برنامه‌ی کاربردی است که کاربر آن را برای اجرا به گريد تحويل داده است. کاربر به همراه اين برنامه، پارامترهای کيفيت سرويس مورد نظر خود يعنی مهلت زمانی، بودجه و استراتژی بهينه‌سازی را برای سيستم مشخص می‌کند. الگوريتم‌های شبيه‌سازی شده، استراتژی بهينه‌سازی را به کار می‌برند. کاربر با انتخاب استراتژی بهينه‌سازی زمان برای زمانبندی، انتظار دارد که سيستم گريد، اجرای برنامه‌ی کاربردی او را قبل از مهلت تعيين شده، به پايان برساند و تا حد ممکن زمان اجرا را کاهش دهد. با انجام شبيه‌سازی می‌توان مشخص کرد که کدام يک از الگوريتم‌های پيشنهادی برای بهينه‌سازی زمان می‌توانند زمان کمتری را برای اجرای برنامه بدست آورند.

### مدل شبکه پتری تصادفی تطابقی در محیط گرید

معماری گرید شامل 3 لایه است، در بالای لایه­ی برنامه کاربردی[[233]](#footnote-234)، محیط گردش کار[[234]](#footnote-235) است که در آن کاربران به شرح برنامه­های خود می­پردازند، در پایین، لایه­ی فابریک[[235]](#footnote-236)، شامل گره­های گرید متصل شده در شبکه است، در بین آن­ها، لایه­ی میان­افزار[[236]](#footnote-237) 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.

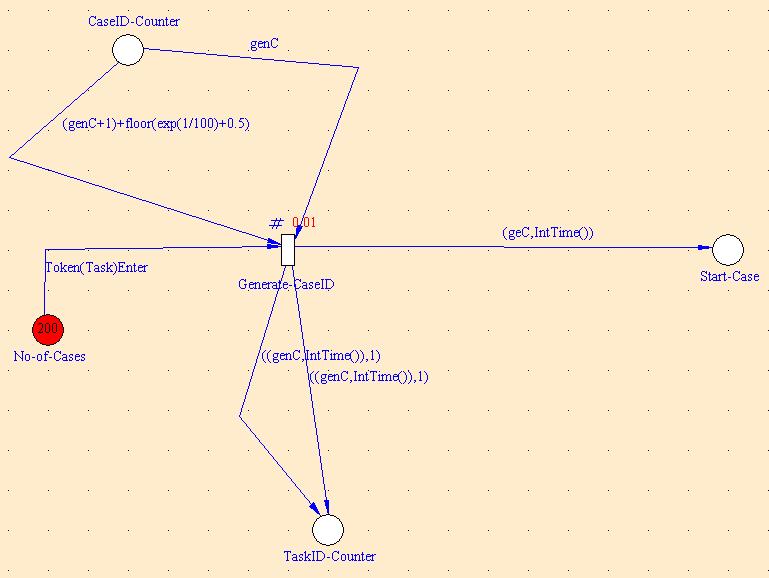


شکل ‏5‑8: سه لایه شبکه گرید در شبکه پتری تصادفی



شکل ‏5‑9: تخصیص کارها در لایه میان افزار

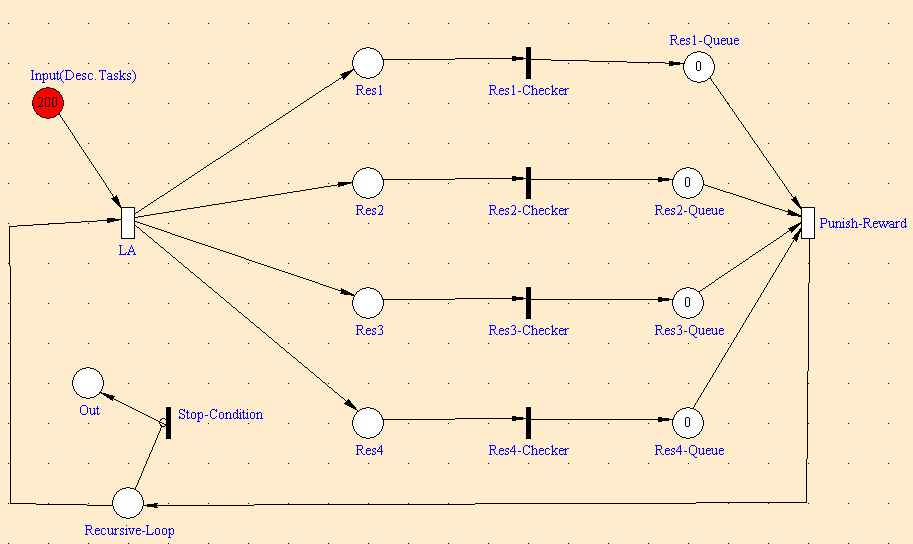
در لایه میان­افزار، برنامه­ی کاربر از طریق لایه­ی برنامه کاربردی دریافت می­شود و پس از گرفتن اطلاعات پویای منابع، درخواست زمان­بندی کار کاربر به ماژول زمان­بند داده می­شود، کارها بر اساس الگوریتم پیشنهادی ابتدا به صورت نزولی مرتب می­شوند که در شکل 5-10 نشان داده شده است. در ابتدا مدل کلی سیستم گرید را با 200 کار در نظر گرفته ایم. همانطور که در شکل 5-10 ملاحظه می شود 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 درخواست ها در نظر گرفته شده است. در نتیجه، کارهای مرتب شده بر حسب زمان تقاضا­ها بصورت نزولی مرتب می شود.



شکل ‏5‑10: نزولی کردن کارها

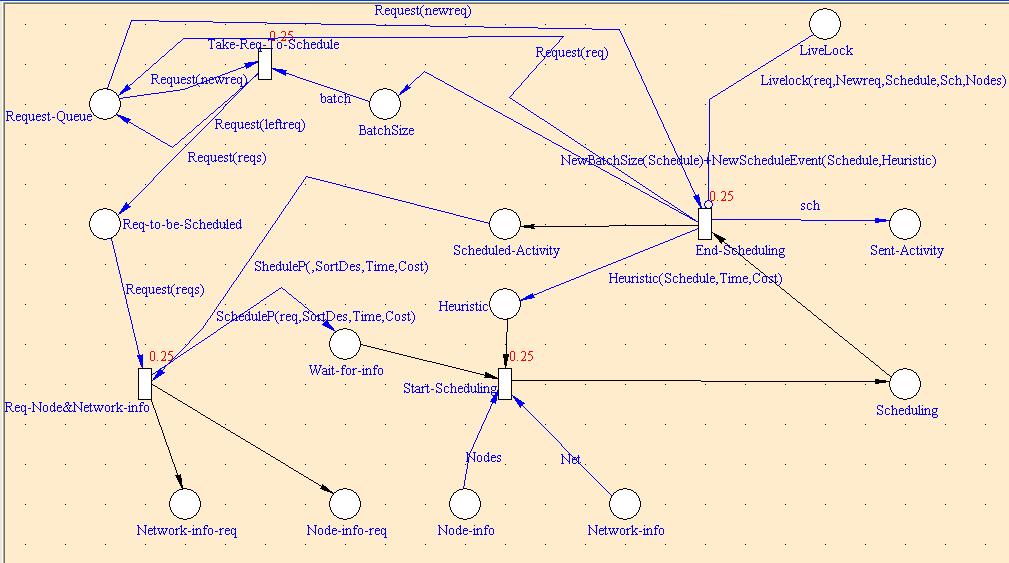
هر توکن نشان­دهنده­ی یک کار از کار­های کاربر است، هر توکن شامل اطلاعاتی از جمله طول کار، شماره اولویت بر اساس طولش و اتوماتای یادگیر است. شبکه پتری تصادفی تطابقی ارائه شده، جزء دسته Fusion Hybrid می­باشد، همانطور که در فصل های پبیشین بیان شد، معایبی هم این دسته دارند. به ­علت ادغام شبکه­های پتری با تکنیک­های 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 نظير شبکه­هاي عصبي مصنوعي، منطق فازي و سيستم­هاي مبتني بر دانش بهره برده­اند، ولی شبکه پتری تصادفی تطابقی که ما ارائه کرده­ایم مبتنی بر اتوماتای یادگیر و هچنین 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 الگوی­های از قبل مشخص­شده نیست، در سیستم­هایی که محیط کاملا 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 کاملا 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 محاسبانی 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 ان کار تعیین کرده است و الگوریتمی که در فصل قبل پیشنهاد داده شده، جریمه یا پاداش به آن عمل تعلق می­گیرد. منابعRes1,Res2,Res3و Res4 منابع چهار گانه ما برای انجام عمل تخصیص می باشند. که هر کدام دارای یک صف مجزا می باشند که به ترتیب در مکان­های Res1\_Queue,Res2\_Queue,Res3\_Queue و Res4\_Queue قرار گرفته اند. در هر بار تخصیص کار­ها به هر منبع شماره کار­ها به ترتیب تخصیص در این صف­ها قرار می گیرند و گذار Punish\_Reward عمل پاداش ی جریمه را به هر منبع اختصاص می دهد. این روال بصورت بازگشتی تا 10000 بار بر روی منابع انجام می شود و نتایج بدست آمده هر بار اجرا بعد از ذخیره شدن در مکان Recursive\_Loop به اجرا مرحله بعدی می رود. کد ها 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

### پیاده سازی الگوریتم ها

در اين بخش کارايي الگوريتم‌ ارائه شده در بخش‌های قبلی را که زمانبندی کارها را با اتخاذ استراتژی‌های مختلف بهينه‌سازی در يک گريد اقتصادی انجام می‌دادند ارزيابی می‌کنيم. اين الگوريتم‌ها سعی می‌کنند در يک محيط گريد اقتصادی، عمل زمانبندی تعدادی کار مستقل از هم را انجام دهند. اين کارها متعلق به يک برنامه‌ی کاربردی است که کاربر آن را برای اجرا به گريد تحويل داده است. کاربر به همراه اين برنامه، پارامترهای کيفيت سرويس مورد نظر خود يعنی مهلت زمانی، بودجه و استراتژی بهينه‌سازی را برای سيستم مشخص می‌کند. الگوريتم‌های شبيه‌سازی شده، استراتژی بهينه‌سازی هزينه يا زمان يا هر دو را به کار می‌برند. کاربر با انتخاب استراتژی بهينه‌سازی زمان برای زمانبندی، انتظار دارد که سيستم گريد، اجرای برنامه‌ی کاربردی او را کمتر از بودجه تعيين شده، به پايان برساند و تا حد ممکن زمان را کاهش دهد. با انجام شبيه‌سازی می‌توان مشخص کرد که کدام يک از الگوريتم‌ها برای بهينه‌سازی زمان می‌توانند زمان کمتری را برای اجرای برنامه بدست آورند. کاربر همچنين می‌تواند با انتخاب استراتژی بهينه‌سازی زمان از سيستم گريد بخواهد تا برنامه‌ی کاربردی او را با صرف بودجه‌ی مشخص شده و در کمترين زمان ممکن اجرا کند. با شبيه‌سازی می‌توان نشان داد که کارايي الگوريتم‌های بهينه‌سازی زمان در مقايسه با يکديگر چگونه است.

### محیط گرید

محيط شبيه‌سازی شده برای گريد شامل يک يا چندين کاربر و تعدادی منبع است. کاربران در هر زمانی ممکن است وارد گريد شوند و برنامه‌ی کاربردی خود را برای اجرا عرضه کنند. اين برنامه شامل تعدادی کار مستقل از هم است که هر کدام می‌تواند بر روی منبع دلخواه اجرا شود. نهاد کاربر پس از ورود به سيستم، دارای يک 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، نتايج را به کاربر برمی‌گرداند.

### مدل کاربر

اغلب اوقات در آزمايش‌های[[237]](#footnote-238) شبيه‌سازی، محيط به صورت تک‌کاربره مدل می‌شود؛ مگر اينکه بخواهيم يک محيط چندکاربره را مدل کرده و الگوريتم‌ها را در چنين وضعيتی با هم مقايسه کنيم. منظور از مدل تک‌کاربره اين نيست که از آغاز تا پايان شبيه‌سازی تنها يک کاربر وارد سيستم می‌شود؛ بلکه بدين معنی است که زمانبندی هيچ دو کاربری با هم تداخل پيدا نمی‌کند. برای نمونه برای مقايسه‌ی الگوريتم‌های LATO و ELATO بايد دو کاربر را با برنامه‌های کاربردی يکسان طوری وارد سيستم کنيم که بر هم تأثير نگذارند (مثلاً کاربر اول با الگوريتم LACO در زمان 0 وارد و در زمان 2000 خارج شود و سپس کاربر دوم با الگوريتم ALATO در زمان 3000 وارد شود و همان برنامه‌ی کاربردی کاربر قبلی را به سيستم عرضه کند). البته نمی‌توانيم از دو آزمايش شبيه‌سازی جداگانه استفاده کنيم؛ زيرا بايد برنامه‌های کاربردی از نظر ناهمگونی کارها يکسان باشند تا بتوان الگوريتم‌ها را با هم مقايسه کرد.

کاربر در هنگام ورود، مقادير بودجه و مهلت زمانی را مشخص می‌کند که در آزمايش‌های گوناگون، مقادير مختلفی را برای آنها قرار می‌دهيم. فرض می‌کنيم برنامه‌ی کاربردی ارائه شده، همواره در مهلت و بودجه‌ی تعيين شده قابل اجرا است. اگر کاربر خواستار بهينه‌سازی هزينه باشد آنگاه پس از مشخص شدن امکان اجرا، سيستم زمانبندی، بايد برنامه‌ی کاربر را در قبل از مهلت تعيين شده و با حداقل هزينه، تمام کند. برای نمونه، اگر کاربر مقدار 200هزار را برای بودجه و 10هزار را برای مهلت تعيين کند و در زمان 5000 وارد سيستم شود آنگاه سيستم بايد بتواند برنامه‌ی کاربر را قبل از زمان 15000 و با حداقل هزينه‌ی ممکن (که البته کمتر از مقدار بودجه خواهد بود) به کاربر تحويل دهد.

### مدل برنامه

برنامه‌ی کاربردی، از تعدادی کار مستقل از هم تشکيل شده است. در آزمايش‌ها، در اغلب مواقع، تعداد کارها را 200 عدد در نظر می‌گيريم. هر کار دارای طول مخصوص به خود بر حسب ميليون دستورالعمل (MI) می‌باشد. طول کارها را متغير در نظر می‌گيريم؛ به اين صورت که يک محدوده‌ی (حداکثر...حداقل) برای طول کارها انتخاب می‌شود؛ سپس برای طول هر کار، يک مقدار به صورت تصادفی از اين محدوده انتخاب می‌شود، به طوري که توزيع طول کارها يکنواخت باشد. برای همگون‌ترين حالت، محدوده‌ی توزيع طول کارها را (110000...100000) و برای ناهمگون‌ترين حالت محدوده‌ی (200000...10000) را در نظر می‌گيريم. ميانگين محدوده را در هر حالتی مقدار 105000 قرار می‌دهيم تا بتوان يک الگوريتم را در ناهمگونی‌های مختلف کارها با هم مقايسه کرد. برای راحتی، واحد طول را در بعضی اوقات MI1000 قرار می‌دهيم و برای نمونه، محدوده‌ی ناهمگون‌ترين حالت را به صورت (200...10) بيان می‌کنيم.

در ابزار GridSim هر کار به صورت يک Gridlet مدل می‌شود که دارای پارامترهای طول و اندازه‌ی فايل‌های ورودی و خروجی می‌باشد. از آنجا که در الگوريتم‌های زمانبندی، فقط زمان اجرای کارها بر روی منابع برای ما مهم است و تأخير شبکه برای جابجايي فايل‌ها را در نظر نمی‌گيريم مقدار 0 را برای پارامتر اندازه‌ی فايل‌ها قرار می‌دهيم. البته در واقعيت نيز اندازه‌ی فايل‌ها آنقدری نيست که تأخير شبکه در مقابل زمان اجرای کارها قابل ملاحظه باشد.

همچنين کارهای مورد مطالعه کاملاً محاسباتی هستند و فقط در ابتدا مقادير مورد نياز را از فايل ورودی دريافت می‌کنند و در پايان اجرا نتايج بدست آمده را در فايل خروجی قرار می‌دهند. از اين تعامل داده‌ای اندک کارها نيز صرف نظر می‌کنيم و تنها به جنبه‌ی محاسباتی آنها می‌پردازيم.

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

### مدل منابع

ما از يک پيکربندی نمونه با نام GR1 برای منابع گريد در آزمايش‌ها استفاده می‌کنيم که البته در جهان خارج وجود ندارد. يک نمونه از پيکربندی واقعی با نام WWG در [133] آمده است که ما از آن نيز در آزمايش‌ها استفاده کرده‌ايم؛ ولی فقط نتايج حاصل با پيکربندی GR2 و GR1 را آورده‌ايم. خصوصيات منابع در GR2 و GR1 طوری است که بتوان زمانبندی‌های مختلف را به خوبی با هم مقايسه کرد. همه‌ی اين منابع دارای يک پردازنده هستند و به صورت فضامشترک رفتار می‌کنند. اين پيکربندی ها را می‌توان در جدول 5-2 و 5-3 مشاهده کرد. در اين جداول، منابع بر حسب قيمت به صورت صعودی مرتب شده‌اند و به همين ترتيب نيز نام‌گذاری شده‌اند تا در هنگام شبيه‌سازی بتوان بار کاری هر کدام از منابع و کارهای واگذار شده را راحت‌تر بررسی کرد (البته برای مرتب‌سازی منابع، از قيمت مفيد (G$/MI) آنها استفاده می‌شود که در اينجا با ترتيب قيمت‌های اعلام شده (G$/sec) يکسان است). اين همان ترتيبی است که الگوريتم‌های بهينه‌سازی هزينه به آن علاقه دارند. اين الگوريتم‌ها برای واگذاری کارها از ارزان‌ترين منبع يعنی R1 شروع می‌کنند.

همراه با افزايش قيمت اعلام شده برای منابع، توانمندی آنها (نرخ اجرای دستورالعمل‌ها در واحد زمان) نيز زياد می‌شود. در ستون آخر، قيمت مفيد منابع را می‌توان ديد که ترکيبی از قيمت و توانمندی هر منبع است. در واقع اين قيمت از تقسيم قيمت اعلام شده بر توانمندی منبع بدست می‌آيد که سپس برای اينکه مقادير خيلی کوچک نباشد در عدد 1000 ضرب می‌شود. لازم به ذکر است که هر جا از قيمت (ارزانی يا گرانی) صحبت می‌شود منظور قيمت مفيد (G$/MI) است و قيمت اعلام شده (G$/sec) ارزش خاصی ندارد زيرا نمی‌توان از آن نتيجه‌ای گرفت.

جدول ‏5‑2: پيکربندی GR1 برای منابع

|  |  |  |  |
| --- | --- | --- | --- |
| **نام منبع** | **نرخ اجرا (MI/sec)** | **قيمت (G$/sec)** | **قيمت مفيد (G$/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 ‏5‑3: پيکربندی GR2 برای منابع

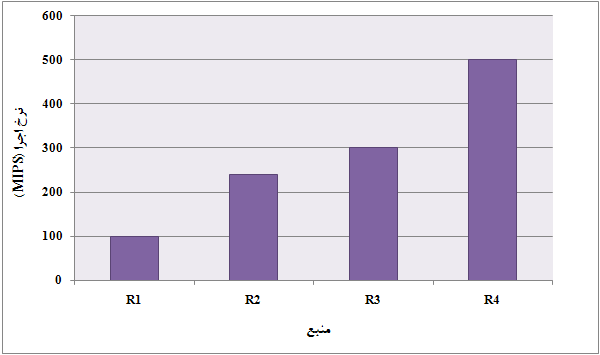
|  |  |  |  |
| --- | --- | --- | --- |
| **نام منبع** | **نرخ اجرا (MI/sec)** | **قيمت (G$/sec)** | **قيمت مفيد (G$/1000MI)** |
| R1 | 100 | 0.5 | 5 |
| R2 | 180 | 1.0 | 5.55 |
| R3 | 240 | 1.5 | 6.25 |
| R4 | 280 | 2 | 7.14 |
| R5 | 300 | 2.5 | 8.33 |
| R6 | 400 | 4 | 10 |
| R7 | 500 | 6 | 12 |
| R8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

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

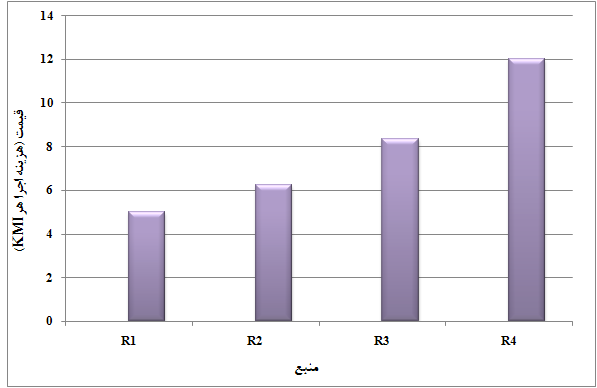
|  |
| --- |
| void Res(GR1 R[4]) //policy baraye manabeh(FIX)  { R[1].benf\_cost=5; R[1].cost=0.5; R[1].Run\_Rate=100; R[1].no=1;  R[2].benf\_cost=6.25; R[2].cost=1.5; R[2].Run\_Rate=240; R[2].no=2;  R[3].benf\_cost=8.33; R[3].cost=2.5; R[3].Run\_Rate=300; R[3].no=3;  R[4].benf\_cost=12; R[4].cost=6; 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 منبع­GR1 کد منبع GR2

شايد اگر کسی تنها به قيمت اعلام شده‌ی منابع توجه کند تصور کند که منابعی که توانمندی زيادی دارند بسيار گران‌تر از منابع دارای توانمندی پايين هستند و استفاده از آنها برای کاربر بسيار پرهزينه است. مشاهده می‌شود که به نظر می‌آيد استفاده از منبع R4 در مقابل منبع R1 اصلاً مقرون به صرفه نيست. نکته اينجاست که ما بايد پارامتر توانمندی منبع را همزمان با قيمت آن در نظر بگيريم. برای مثال، اگر قيمت اعلام شده برای يک منبع، دو برابر قيمت منبع ديگر باشد، ولی توانمندی آن نيز دو برابر باشد، آنگاه اين دو منبع از نظر يک الگوريتم بهينه‌سازی هزينه، دارای ارزش يکسانی هستند. توانمندی‌های منابع GR1 در شکل 5-15 نشان داده شده است. مشاهده می‌شود که منابع گرانتر دارای توانمندی بالاتری هستند؛ ولی اين افزايش توانمندی تقريباً به صورت خطی است و رشد توانمندی از رشد قيمت اعلام شده کمتر است. بنابراين انتظار داريم قيمت مفيد منابع نيز به همين ترتيب افزايش يابد ولی رشد آن از رشد قيمت اعلام شده کمتر باشد. اين مسأله در شکل 5-16 تأييد می‌شود. در اين شکل می‌توان ارزش واقعی منابع و هزينه‌ی اجرای کارها بر روی آنها را با هم مقايسه کرد. می‌بينيم بر خلاف آنچه قبلاً ديده بوديم، اختلاف قيمت منبع R4 و R1 خيلی زياد نيست و در بعضی مواقع در صورت موجود بودن بودجه و کوتاه بودن مهلت کاربر، استفاده از R4 منطقی به نظر می‌آيد.



شکل ‏5‑15: توانمندی منابع در پيکربندی GR1



شکل ‏5‑16: قيمت مفيد منابع در پيکربندی GR1

### شبیه سازی زمان

الگوريتم‌های زمانبندی که دارای هدف بهينه‌سازی زمان هستند بايد بتوانند برنامه‌ی کاربردی کاربر مراجعه کننده را با صرف بودجه‌ی تعيين شده، بر روی گريدی که از مدل اقتصادی بازار کالا استفاده می‌کند، اجرا کرده و نتايج را در کمترين زمان ممکن به کاربر ارائه کنند. از نظر صاحبان و مشتريان منابع، الگوريتمی از بقيه کاراتر است که بتواند کارهای مستقل از هم که برنامه را تشکيل داده‌اند را طوری به منابع موجود در گريد واگذار کند که زمان لازم برای اجرای کارها در پرکارترين منبع نسبت به ساير الگوريتم‌ها کمتر باشد. يک الگوريتم خوب بايد بتواند نتايج قابل قابولی را در شرايط گوناگون مانند مهلت‌های کم و زياد، حضور کاربران مختلف به صورت همزمان در گريد، ناهمگونی کارها و منابع به کاربر ارائه کند. در ادامه الگوریتم های BTO ,AEBTO,LATO,Minimum و الگوریتم پیشنهادی ALATO در حالت های مختلف مورد بررسی قرار گرفته است. حالت اول حالتی است مقدار 200 کار با طول ثابت که مقدار هزینه و زمان توسط کاربر معین می باشد. حات دوم مقدار 200 کار با طول ثابت در یک زمان تعیین شده توسط کاربر در ناهمگونی های 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 ناهمگونی های مختلف بودجه برای 4 منبع مورد بررسی قرار گرفته است. حالت چهارم، مقدار 400 کار در حالت های سه گانه بالا ذگر شده اند. هر چهار حالت در اتوماتای یادگیر استاندارد و حالت های سه گانه مدل S مورد آزمایش قرار گرفته اند و نتایج آورده شده اند. توجه داشته باشید که الگوریتم Minimum بهترین حالت اجرا و پیاده سازی می باشد که طریقه بکاربردن آن در فصل قبل شرح داده شد و مقدار آن در حالتهای مختلف اتوماتا تغییری نمی کند.

#### 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 داده­ایم، برای بررسی مناسب طول کارها در تست تمامی مدل­ها برابر در نظر گرفته شده است. پارامتر محدوده‌ی طول کارها نشان‌دهنده‌ی محدوده‌ی توزيع يکنواخت طول کارها می‌باشد. پارامتر تعداد آزمايش‌ها نشان می‌دهد که برای بدست آوردن زمان اجرای برنامه با استفاده از الگوريتم‌های موجود در يک مهلت معين، تعداد 20 آزمايش انجام شده است و سپس ميانگين مقادير برای بررسی انتخاب می‌شود.

جدول ‏5‑4 : شرايط آزمايش بهينه‌سازی زمان با بودجه‌ ثابت

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| پارامتر | الگوريتم | تعداد کاربر | تعداد کارها | مهلت | بودجه | محدوده‌ی طول کارها | پيکربندی منابع | تعداد آزمايش‌ها |
| مقدار | متغير | 1 | 200 | 50000 | 150000 | [200...10] | GR1 | 20 |

حال شرایط زیر را در حالت های مختلف اتوماتای استاندارد(مدل P-LRP)، اتوماتای مدل S(S-LRP، S-LRI و S-LReP)مورد ارزیابی قرار می دهیم.

##### **مدل P-LRP**

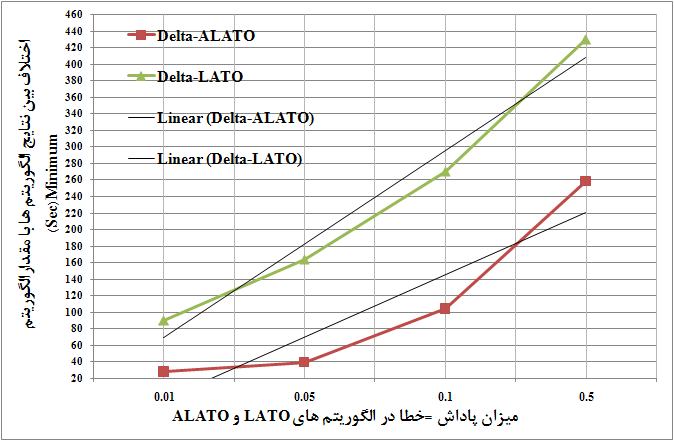
برای این مدل نرخ پاداش و جریمه را برابر با مقادیر مختلفی قرار داده­ایم و نتایج را با هم تست کرده­ایم، در این مدل (P) ، پاسخ  با احتمال ci برابر با 1 (پاسخ مطلوب) و با احتمال -ci1 برابر با صفر است (پاسخ نامطلوب). از آنجایی که تنها دو الگوریتم LATO و ALATO الگویتم های پیاده سازی شده بر اساس اتوماتای یادگیر می باشند، بنابر این، میزان تغییرات پاداش و جریمه در بودجه 150000 برای این الگوریتم ها مد نظر قرار می دهیم. این نتایج در اتوماتای استاندارد در جدول 5-5 نشان داده شده است.

از آنجاکه نرخ پاداش و جريمه برابر هستند، باید مقادير کمی داشته باشند تا الگوريتم بتواند به تدريج و به آهستگی، زمانبندی مناسب را بدست آورد. در بودجه ثابت150000 میزان نرخ پاداش و جریمه یکسان و 0.01 بهترین حالت برای الگوریتم های بر پایه اتوماتای یادگیر می باشند.

جدول ‏5‑5: تغییرات میزان پاداش و جریمه در مدل P-LRP و در الگوريتم‌های بهينه‌سازی زمان با بودجه‌ ثابت 150000

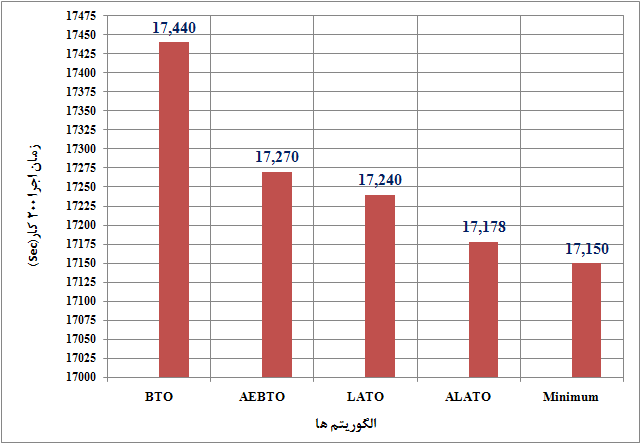
|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| الگوريتم‌ها | | | | | نرخ پاداش | نرخ جریمه |
| Minimum | ALATO | LATO | AEBTO | BTO |
| 17,150 | 17,175 | 17,240 | 17,240 | 17,440 | a=0.01 | b=0.01 |
| 17,150 | 17,275 | 17,314 | 17,440 | 17,900 | a=0.05 | b=0.05 |
| 17,150 | 17,354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

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 پاداش­ها­ی (جریمه ها) متفاوت می باشد و همچنین Delta-LATO میزان اختلاف زمانی الگوریتم ALATO با مقدار الگوریتم Minimum در پاداش ها(جریمه ها)ی متفاوت می باشد.میزان شیب Linear(Delta-ALATO) از میزان شیب Linear(Delta-LATO) کمتر می باشد.این 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**

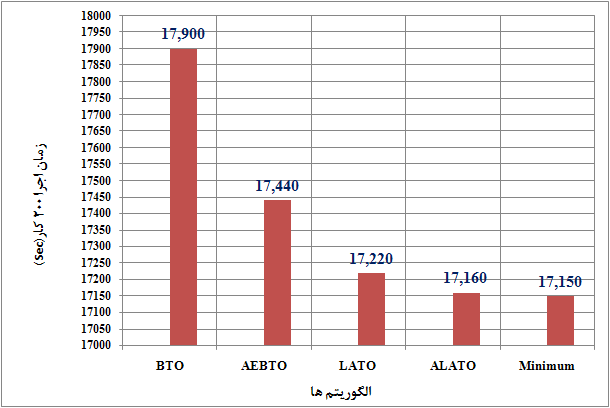
در روش­های مدل S، اگر در تکرار *n*اُم عمل انتخاب شده باشد، پاسخ (پاسخ نامطلوب) محيط به آن برابر  می­باشد و پاسخ (پاسخ مطلوب) محيط به آن عمل برابر  می­باشد، (یعنی به نسبت انتخاب منبع ارزان­تر پاسخ بهتری از محیط دریافت می­کند). که طبق نتایج که در فصل شبیه­سازی مشاهده خواهیم نمود که روش­ S-LReP با نرخ پاداش برابر با 0.1 و نرخ جریمه برابر با 0.05 بهترین نتیجه را خواهد داشت. این نتایج در جدول 5-6 نشان داده شده است. از آنجاکه نرخ پاداش و جريمه برابر هستند، باید مقادير کمی داشته باشند تا الگوريتم بتواند به تدريج و به آهستگی، زمانبندی مناسب را بدست آورد. در بودجه ثابت150000 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 جريمه باهم برابر هستند، پس باید مقادير کمی داشته باشند تا الگوريتم بتواند به تدريج 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 شده­است.

جدول ‏5‑6: تغییرات میزان پاداش و جریمه در مدل S-LRP و الگوريتم‌های بهينه‌سازی زمان با بودجه‌ ثابت 150000

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

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 الگوریتمALATO 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 الگوریتم ها در اتوماتای S-LRP با بودجه ثابت 150000

##### **مدل S-LRI**

در این مدل نرخ جریمه برابر با صفر (b=0) است و 4 مقدار مختلف را برای نرخ پاداش آزمایش کرده­ایم. . این نتایج در جدول 5-7 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 ‏5‑7: تغییرات میزان پاداش و جریمه در مدل S-LRI و الگوريتم‌های بهينه‌سازی زمان با بودجه‌ ثابت 150000

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

##### **مدل S-LReP**

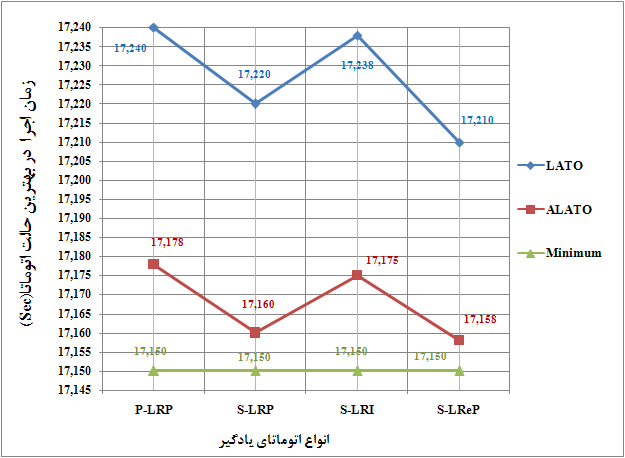
در این مدل نرخ جریمه بسیار کوچک­تر از نرخ پاداش است و 4 مقدار مختلف را برای نرخ پاداش و جریمه آزمایش کرده­ایم. این نتایج در جدول 5-8 نشان داده شده است.

جدول ‏5‑8: تغییرات میزان پاداش و جریمه در مدل S-LReP و الگوريتم‌های بهينه‌سازی زمان با بودجه‌ ثابت 150000

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

با توجه به جدول 5-8، 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 nام به عمل انتخابی() 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 عمراه الگوریتم Mnimun صورت گرفته است. در اینجا بهترین زمان اجراهای هر اتوماتا برای بودجه 150000 در نظر گرفته شده است. حد پایین و بالای نموار 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  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

#### بهینه سازی زمان در زمان معین با بودجه‌های مختلف (حالت دوم)

در اين بخش، الگوريتم‌های زمانبندی که هدفشان بهينه‌سازی زمان است را با فراهم کردن بودجه‌های مختلف با هم مقايسه می‌کنيم. همچنين نشان داده می‌شود که هر کدام از الگوريتم‌ها در شرايط يکسان (کاربر، برنامه‌ی کاربردی و منابع)، اجرای برنامه‌ی کاربر را طی چه زمانی به پايان می‌رسانند. مطمئناً الگوريتمی برای محيط گريد مناسب‌تر است که عمل زمانبندی کارها را طوری روی منابع انجام دهد که در کمترين زمان، نتايج اجرای برنامه را به کاربر ارائه کند.

الگوریتم پیشنهادی را با انواع مدل­های اتوماتای یادگیر با توجه به شرایط جدول 5-9 مورد آزمایش قرار داده­ایم، برای بررسی مناسب طول کارها در تست تمامی مدل­ها برابر در نظر گرفته شده است. پارامتر محدوده‌ی طول کارها نشان‌دهنده‌ی محدوده‌ی توزيع يکنواخت طول کارها می‌باشد. پارامتر تعداد آزمايش‌ها نشان می‌دهد که برای بدست آوردن زمان اجرای برنامه با استفاده از الگوريتم‌های موجود در يک مهلت معين، تعداد 20 آزمايش انجام شده است و سپس ميانگين مقادير برای بررسی انتخاب می‌شود.

جدول ‏5‑9: شرايط آزمايش بهينه‌سازی زمان با بودجه‌ متغییر

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| پارامتر | الگوريتم | تعداد کاربر | تعداد کارها | مهلت | بودجه | محدوده‌ی طول کارها | پيکربندی منابع | تعداد آزمايش‌ها |
| مقدار | متغير | 1 | 200 | 50000 | متغير | [200...10] | GR1 | 20 |

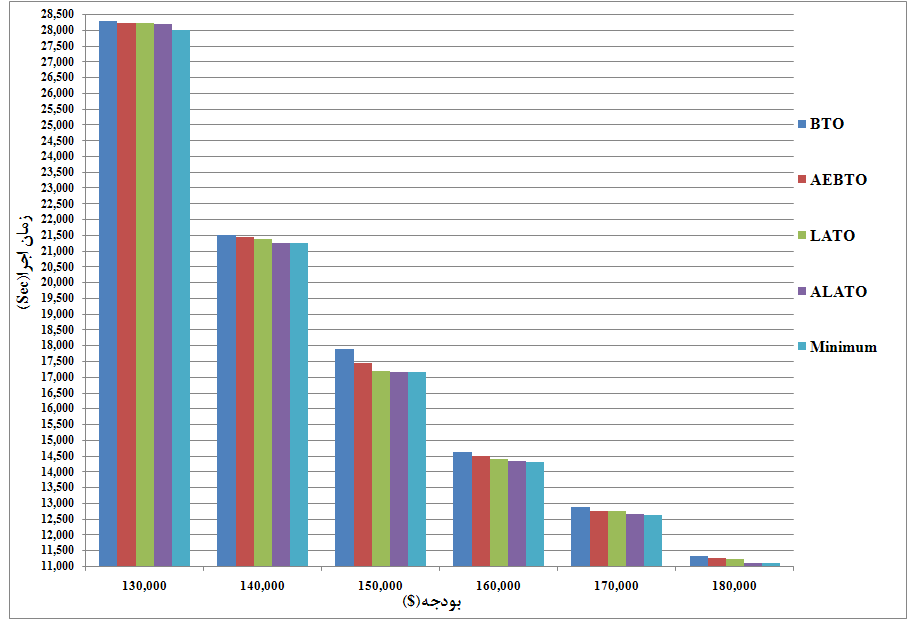
از آنجایی که اتوماتای مدل S زمان بهتری را نسبت به اتوماتای استاندارد ارائه می کند. این اتوماتا برای بررسی انتخاب می شود. در انواع مدل S اتوماتای S-LReP نتیجه بهتری را ارائه کرده است. بنابر این، تنها نتایج را در این اتوماتا مورد ارزیابی قرار می دهیم.

نتايج آزمايش‌های انجام شده برای مقايسه‌ی الگوريتم‌های بهينه‌سازی زمان در جدول 5-10 آورده شده است. این نتایج در اتوماتای مدل S-LReP که بهترین زمان را ارائه کرده است لحاظ شده است. ستون اول، بودجه‌ی تعيين شده توسط کاربر را نشان می‌دهد. مقدار بودجه از 130000 تا 180000 تغيير می‌کند تا بتوانيم کارايي الگوريتم‌ها را با بودجه‌های مختلف، ارزيابی کرده و با يکديگر مقايسه کنيم. هر کدام از ستون‌های بعدی نمايانگر يکی از الگوريتم‌ها می‌باشند و ستون آخر، مقدار مينيمم زمان محاسبات را با صرف بودجه‌های مختلف نشان می‌دهد که اين مينيمم در بيشتر موارد برای هيچ الگوريتمی دست يافتنی نيست. هر کدام از خانه‌های جدول مقدار زمان اجرای برنامه‌ی کاربر را برای الگوريتم زمانبندی مربوطه و با صرف بودجه‌ی تعيين شده مشخص می‌کند. برای مثال، با در اختيار داشتن بودجه‌ی 150000، الگوريتم زمانبندی BTO اجرای کارهای برنامه‌ی کاربردی متعلق به کاربر را بعد از مدت زمان 17900 واحد به اتمام می‌رساند و الگوريتم LATO و ALATO برای اجرای همين برنامه و با صرف بودجه‌ی فراهم شده، به ترتیب به مدت زمانهای 17210و 17158 واحد نياز دارد. برای هر کدام از بودجه‌ها تعداد 20 آزمايش انجام داده‌ايم و ميانگين مقادير بدست آمده را برای هر کدام از الگوريتم‌ها به عنوان زمان متوسط اجرای کارها با بودجه‌ی تعيين شده در نظر گرفتيم. اگر تنها از مقادير بدست آمده از يک آزمايش استفاده کنيم، آنگاه اگرچه ممکن است بتوانيم الگوريتم‌ها را با بودجه‌ی 140000 با هم مقايسه کنيم ولی نمی‌توان مقادير را در کنار مقادير بدست آمده با يک بودجه‌ی ديگر مانند 150000 قرار داد و تغييرات را بررسی کرد. مقادير مختلفی برای زمان اجرای کارها با استفاده از هر کدام از الگوريتم‌ها بدست آمده است که مجبور می‌شويم در نهايت پس از انجام آزمايشات متعدد، عمل ميانگين‌گيری زمان اجرا را برای هر کدام از الگوريتم‌ها انجام دهيم. اين اختلافات به علت توزيع تصادفی طول کارها در محدوده‌ی تعيين شده بوجود می‌آيد و هر چه کارها ناهمگون‌تر باشند (محدوده‌ی توزيع طول کارها بزرگتر باشد) اين اختلافات نيز بيشتر خواهد بود.

جدول ‏5‑10: زمان اجرای برنامه در الگوريتم‌های بهينه‌سازی زمان با بودجه‌های مختلف

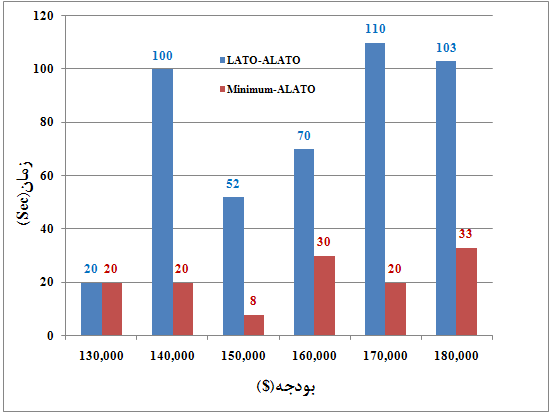
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **بودجه** | **الگوريتم‌ها** | | | | |
| **BTO** | **AEBTO** | **LATO** | **ALATO** | **Minimum** |
| 130,000 | 28,300 | 28,220 | 28,220 | 28,200 | 28,020 |
| 140,000 | 21,500 | 21,450 | 21,370 | 21,305 | 21,250 |
| 150,000 | 17,900 | 17,440 | 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 الگوریتم های بهینه سازی با بودجه های مختلف را نشان می دهد. حد پایین و بالای قسمت زمان در نموار 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 می شود. در همه حالات الگوریتم ALATO از سایر الگوریتم ها بهتر عمل کرده و بسیار به الگوریتم ایده آل Minimum نزدیک شده است. الگوریتم AEBTO از BTO بهتر عمل کرده است ولی از الگوریتم های پیاده سازی شده با اتوماتا زمان بیشتری را صرف می کند. همانطور که در نمودار مشاهده می‌شود، مدت زمانی که همه‌ی الگوريتم‌ها برای اجرای کارها بدست آورده‌اند تقريباً نزديک به هم است. با افزایش بودجه همواره مقدار زمان اجرا کاهش پیدا می کند و همواره به مینیمم نزدیکتر می شوند در واقع، هر چه قدر که بودجه‌ی تعيين شده توسط کاربر بيشتر باشد، الگوريتم‌ها به زمان کمتری برای اجرای کارهای برنامه‌ی کاربردی نياز دارند؛ زيرا هنگامی که کاربر بودجه‌ی فراوانی را برای اجرای برنامه‌ی کاربردی خود بر روی گريد در نظر می‌گيرد، الگوريتم‌های زمانبندی می‌توانند گستره‌ی وسيع‌تری از منابع را به کار گرفته و با اجرای کارها بر روی منابع گران و توانمند زمان اجرا را کاهش دهند.



شکل ‏5‑21: نتايج الگوريتم‌های بهينه‌سازی زمان با بودجه‌های مختلف

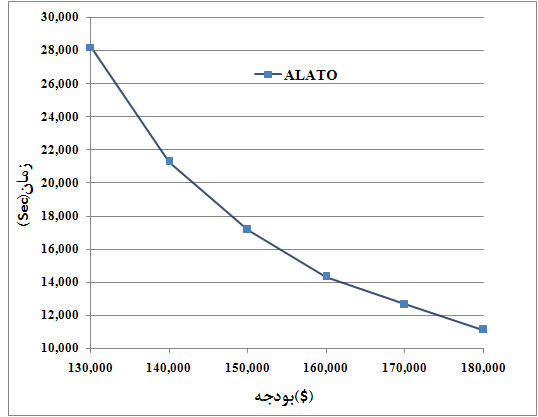
از آنجا که هدف ما بهبود بهترین الگوريتم گزارش شده برای بهينه‌سازی زمان تا قبل از نگارش اين پايان‌نامه الگوريتم LATO بود، در اينجا الگوريتم ALATOرا که از اتوماتاهای يادگير استفاده می‌کند، با اين الگوريتم مقايسه می‌کنيم. شکل 5-22 اختلاف نتايج اين دو الگوريتم را در بودجه‌های مختلف نشان می‌دهد. مشاهده می‌شود که زمان اجرای بدست آمده با استفاده از الگوريتم ALATOاز زمان اجرا در الگوريتم LATO کمتر است. می‌بينيم که الگوريتم ALATO در بودجه‌های کم توانسته است تا 100 واحد در زمان صرفه‌جويي کند. البته اين ميزان صرفه‌جويي در بودجه‌های زياد به حدود 110 واحد می‌رسد.



شکل ‏5‑22: مقايسه‌ی الگوريتم‌های ALATO وLATO و Minimum با بودجه‌های مختلف

برای اينکه ببينيم چقدر امکان بهبود الگوريتم ALATO که بهترين الگوريتم پيشنهاد شده می‌باشد وجود دارد بايد اين الگوريتم را با مينيمم زمان اجرای کارها مقايسه کنيم. اختلاف نتايج بدست آمده از الگوريتم ALATO با مينيمم زمان محاسبات در بودجه‌های مختلف در شکل 5-22 نشان داده شده است. مشاهده می‌شود که اين اختلاف بسيار ناچيز و در همه‌ی موارد کمتر از 8 واحد است. اين مقدار در حالی که زمان اجرای کارها در همه‌ی بودجه‌ها بيش از 10هزار واحد می‌باشد اصلاً قابل ملاحظه نيست. بنابراين می‌توان گفت که الگوريتم ALATO تا حد ممکن به زمان مينيمم نزديک شده است و از آنجا که می‌دانيم اين مينيمم در اغلب موارد توسط هيچ الگوريتمی قابل دستيابی نيست، ارائه‌ی يک الگوريتم جديد حداکثر تا کمتر از 8 واحد می‌تواند بهبود حاصل کند که اصلاً نمی‌توان آن را در نظر گرفت.

همانطور که قبلاً ديده شد، همه‌ی الگوريتم‌های پيشنهادی روند يکسانی را در تغييرات زمان بر حسب بودجه دارند. برای نمونه، اين روند برای الگوريتم LATO در شکل 5-23 نشان داده شده است. مهمترين نکته در اين تغييرات، کاهش زمان اجرای کارها در اثر افزايش بودجه است. علت اين امر اين است که با افزايش بودجه‌ی فراهم شده توسط کاربر، الگوريتم‌ها می‌توانند با به کار بردن منابع گرانتر، از منابع بيشتری برای اجرای کارها استفاده کرده و زمان اجرا را کاهش دهند. اين روند کاهش زمان در همه‌ی الگوريتم‌ها ديده می‌شود. البته اين کاهش در ابتدا سرعت بيشتری دارد و هر چه بودجه بيشتر شود اين کاهش نيز کمتر می‌شود. در واقع تغييرات زمان بر حسب بودجه همانطور که ديده می‌شود به صورت غيرخطی است. در ابتدا که بودجه کم است الگوريتم‌ها مجبورند برای اينکه هزينه‌ی اجرای کارها از بودجه فراتر نرود، از تعداد اندکی از منابع ارزان استفاده کنند؛ ولی با زياد شدن بودجه، الگوريتم‌ها می‌توانند از منابع گران‌تر نيز برای اجرای کارها استفاده کرده و زمان را کاهش دهند. البته به تدريج منابع اضافه شده گرانتر می‌شوند و به کار بردن آنها به هزينه‌ی بيشتری نياز دارد و بنابراين تأثير افزايش بودجه در کاهش زمان محاسبات، کمتر می‌شود. در نهايت، هنگامی که با صرف بودجه‌ی تعيين شده بتوان از همه‌ی منابع برای زمانبندی کارها استفاده کرد ، آنگاه ديگر افزايش بودجه، تغييری در زمان محاسبات ايجاد نمی‌کند و نمودار به صورت يک خط افقی در می‌آيد. از اين طرف هم نقطه‌ی شروع نمودار کمترين بودجه‌ای است که با صرف آن می‌توان کارها را قبل از مهلت تحويل داد که در واقع مينيمم هزينه‌ی محاسبات می‌باشد. برای مهلت 100هزار که در اين آزمايش‌ها استفاده می‌شد، مقدار مينيمم هزينه حدود 115هزار محاسبه شده بود و ما آزمايش‌ها را از بودجه‌ای بيشتر از آن يعنی مقدار 120000 شروع کرده‌ايم.

ُ

شکل ‏5‑23: روند تغييرات زمان اجرای کارها در الگوريتم ALATO با بودجه‌های مختلف

#### بهينه‌سازی زمان در ناهمگونی‌های مختلف(حالت سوم)

در اين بخش، الگوريتم‌های زمانبندی که هدفشان بهينه‌سازی زمان است برای زمانبندی کارهايي با ناهمگونی‌های مختلف با هم مقايسه می‌شوند. همچنين بررسی می‌کنيم که هر الگوريتم در ناهمگونی‌های مختلف کارها، چه زمانی برای اجرای آنها بدست می‌آورد. همانطور که قبلاً گفته شد، برنامه‌ی کاربردی عرضه شده توسط کاربر از تعدادی کار مستقل از هم تشکيل شده است. هر کار، دارای طول مخصوص به خود است که بر حسب ميليون دستورالعمل (MI) بيان می‌شود. هميشه طول کارها را متغير در نظر می‌گيريم؛ به اين صورت که يک محدوده‌ی (حداکثر...حداقل) برای طول کارها انتخاب می‌شود. سپس برای طول هر کار، يک مقدار به صورت تصادفی از اين محدوده انتخاب می‌شود به طوريکه توزيع طول کارها يکنواخت باشد. برای همگون‌ترين حالت، محدوده‌ی توزيع طول کارها را (110000...100000) و برای ناهمگون‌ترين حالت محدوده‌ی (200000...10000) را در نظر می‌گيريم. ميانگين محدوده را در هر حالتی مقدار 105000 قرار می‌دهيم تا بتوان نتايج يک الگوريتم معين را در ناهمگونی‌های مختلف کارها بررسی کرد. برای راحتی واحد طول را MI1000 قرار می‌دهيم و برای مثال، محدوده‌ی ناهمگون‌ترين حالت را به صورت (200...10) بيان می‌کنيم.

شرايط انجام آزمايش شبيه‌سازی در جدول 5-11 آمده است. پارامتر محدوده‌ی طول کارها، متغير است و آزمايش‌ها در محدوده‌های مختلف (يعنی ناهمگونی‌های مختلف) انجام می‌شود. پارامتر تعداد آزمايش‌ها مشخص می‌کند که برای بدست آوردن مقادير زمان اجرای الگوريتم‌ها در يک ناهمگونی معين تعداد 20 آزمايش انجام شده و سپس ميانگين مقادير برای بررسی، انتخاب شده است.

جدول ‏5‑11: شرايط آزمايش بهينه‌سازی زمان در ناهمگونی‌های مختلف و بودجه 150000

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| پارامتر | الگوريتم | تعداد کاربر | تعداد کارها | مهلت | بودجه | محدوده‌ی طول کارها | پيکربندی منابع | تعداد آزمايش‌ها |
| مقدار | متغير | 1 | 200 | 20000 | 150000 | متغير | GR1 | 20 |

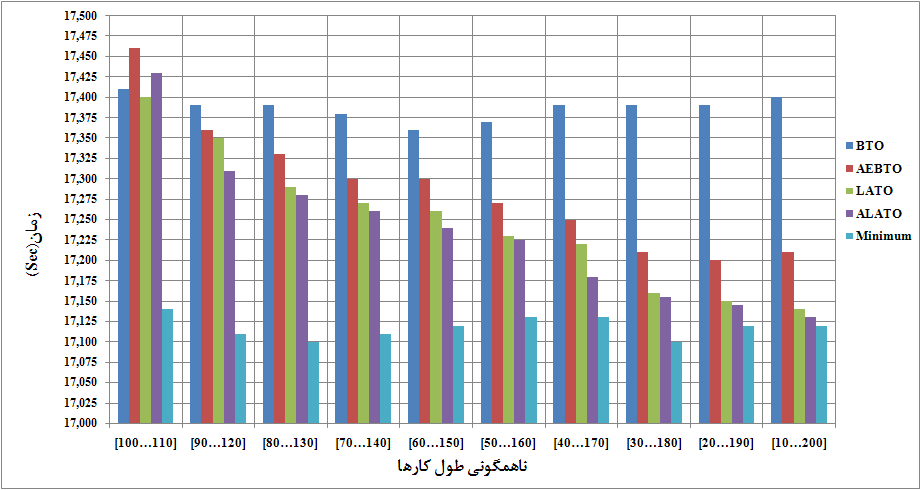
از آنجایی که اتوماتای مدل S زمان بهتری را نسبت به اتوماتای استاندارد ارائه می کند. این اتوماتا برای بررسی انتخاب می شود. در انواع مدل S اتوماتای S-LReP نتیجه بهتری را ارائه کرده است. بنابر این، تنها نتایج را در این اتوماتا مورد ارزیابی قرار می دهیم. برای هر ناهمگونی، تعداد 20 آزمايش انجام داديم و ميانگين مقادير بدست آمده را برای هر کدام از الگوريتم‌ها، به عنوان زمان متوسط اجرای کارها در نظر گرفتيم. اين عمل، همانند آنچه در آزمايش‌های بودجه‌ی متغير در بخش قبل آمد، انجام می‌شود.

نتايج آزمايش‌های انجام شده برای مقايسه‌ی الگوريتم‌های بهينه‌سازی زمان، در جدول 5-12 آورده شده است. ستون اول، محدوده‌ی توزيع طول کارها را نشان می‌دهد. اين محدوده از همگون‌ترين حالت يعنی (110...100) شروع می‌شود و با افزايش 10هزار واحد از هر دو طرف در هر مرحله، به ناهمگون‌ترين حالت يعنی محدوده‌ی (200...10) می‌رسد. ستون‌های بعدی هر کدام نمايانگر يکی از الگوريتم‌ها می‌باشد و ستون آخر، مقدار مينيمم را برای زمان محاسبات در آن ناهمگونی نشان می‌دهد. اين مينيمم در بيشتر موارد برای هيچ الگوريتمی دست يافتنی نيست. هر کدام از خانه‌های جدول، مقدار زمان اجرای کارها را برای الگوريتم زمانبندی مربوطه و در ناهمگونی مورد نظر مشخص می‌کند. برای مثال، الگوريتم زمانبندی AEBTO، کارهايي را که دارای ناهمگونی (170...40) می‌باشند (تعداد 200 کار)، در مدت زمان 17250 واحد انجام می‌دهد و الگوريتم ALATO در همين شرايط، برای اجرای کارها به زمان 17220 واحد نياز دارد.

جدول ‏5‑12: زمان اجرای برنامه در الگوريتم‌های بهينه‌سازی زمان با ناهمگونی‌های مختلف

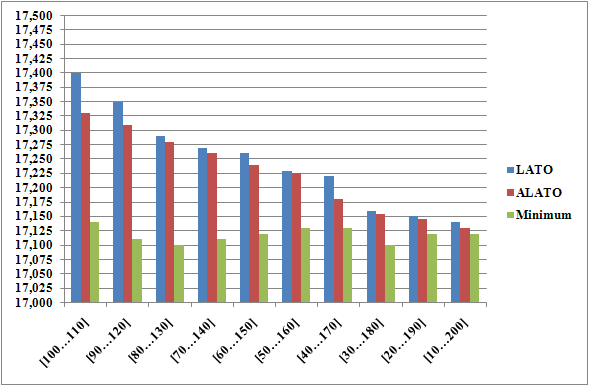
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **ناهمگونی** | **الگوريتم‌ها** | | | | |
| **BTO** | **AEBTO** | **LATO** | **ALATO** | **Minimum** |
| [100…110] | 17,410 | 17,460 | 17,400 | 17,430 | 17,140 |
| [90…120] | 17,390 | 17,360 | 17,350 | 17,310 | 17,110 |
| [80…130] | 17,390 | 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 |

نتايج بدست آمده از الگوريتم‌های زمانبندی با استراتژی بهينه‌سازی زمان، در شکل 5-24 نشان داده شده است. همانطور که در نمودار مشاهده می‌شود، تغييرات زمان در اثر افزايش ناهمگونی کارها، در هر کدام از الگوريتم‌ها روند متفاوتی دارد. زمان اجرای بدست آمده در الگوريتم BTO، بسيار کم تحت تأثير تغييرات ناهمگونی قرار گرفته است. در واقع پارامتر ناهمگونی، تأثيری بر کارايي اين الگوريتم ندارد که اين اصلاً مطلوب نيست. ما از يک الگوريتم خوب انتظار داريم که بتواند از ناهمگونی کارها استفاده کند و زمان اجرای کارها را کاهش دهد الگوريتم AEBTO در ناهمگونی کم از همه‌ی الگوريتم‌ها بدتر عمل کرده است ولی با افزايش ناهمگونی توانسته است کارايي خود را افزايش دهد و به عنوان بهترين الگوريتم مکاشفه‌ای زمان اجرای خوبی را بدست آورد. الگوريتم LATO و ALATOکه از اتوماتاهای يادگير استفاده می‌کنند توانسته اند با افزايش ناهمگونی در کارها، کاهش قابل قبولی را در زمان محاسبات بدست آورند. زمان مينيمم، همانطور که مشاهده می‌شود، در ناهمگونی‌های مختلف تغييرات کمی دارد؛ زيرا ميانگين محدوده‌هايي که در هر ناهمگونی برای توزيع تصادفی و يکنواخت طول کارها مورد استفاده قرار می‌گيرند يکسان و برابر با 105 kMI (هزار ميليون دستورالعمل) می‌باشد. بنابراين با توجه به اينکه تعداد کارها 200 عدد می‌باشد، مجموع طول کارها در همه‌ی حالات برابر با 105\*200 kMI می‌باشد و به اين ترتيب تغيير خاصی نبايد در زمان مينيمم ايجاد 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.



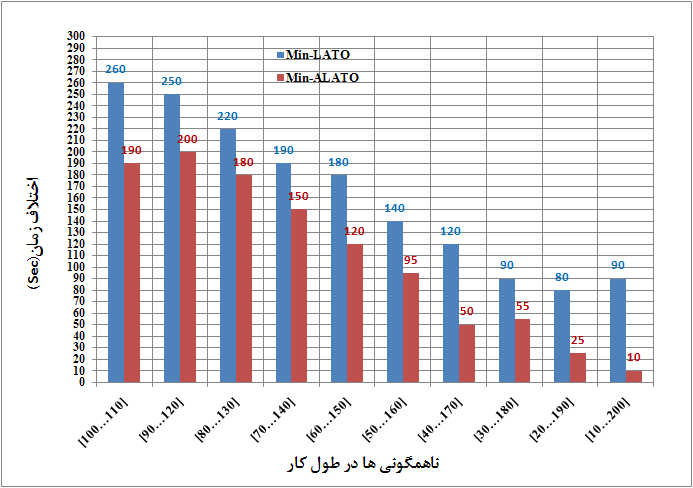
شکل ‏5‑24: نتايج الگوريتم‌های بهينه‌سازی زمان در ناهمگونی‌های مختلف

الگوريتم های LATO و ALATO که از اتوماتاهای يادگير استفاده می‌کنند و ALATO به عنوان بهترين الگوريتم پيشنهادی، قصد دارد تا الگوريتم LATO را بهبود دهد. در اينجا تأثير ناهمگونی کارها بر ميزان بهبود حاصل از الگوريتم LATO و ALATO را بررسی می‌کنيم. شکل 5-25 نتايج اين دو الگوريتم به همراه الگوریتم Minimum را در کنار هم و در ناهمگونی‌های مختلف نشان می‌دهد. مشاهده می‌شود که بهبود حاصل از زمانبندی الگوريتمهای ALATO و LATO با افزايش ناهمگونی کارها بيشتر می‌شوند. روند کاهش زمان در الگوریتم ALATO از الگوریتم LATO بیشتر است. درواقع، سرعت به مینیمم رسیدن در ALATO از LATO بیشتر است.



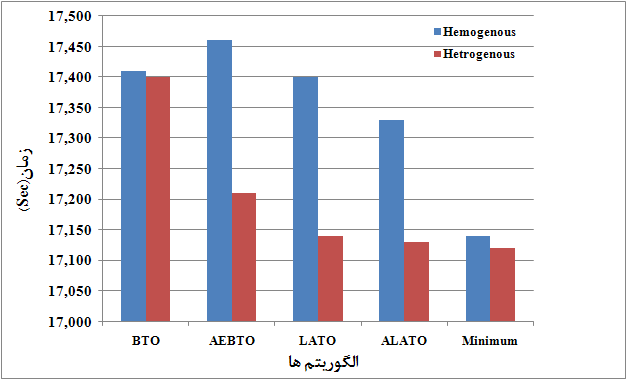
شکل ‏5‑25: مقايسه‌ی الگوريتم‌های LATO و ALATOو Minimum در ناهمگونی‌های مختلف

برای مقايسه‌ی دقيق‌تر و مشاهده‌ی ميزان بهبود حاصل شده، به نمودار اختلاف زمان اجرا هرکدام از این دو الگوريتم با الگوریتم مینیمم نياز داريم. این نمودار در شکل 5-26 نشان داده شده است. مشاهده می‌کنيم که هنگامی که کارها همگون هستند الگوريتم ALATOتقريباً زمان زیادی مصرف می کند و فاصله‌ی نتيجه‌ی اين الگوريتم با زمان مينيمم زياد است. علت اين امر اين است که چون طول کارها تقريباً به هم نزديک است اختلاف زمان اجرای کارها بر روی پرکارترين منبع با منابع ديگر زياد است و نمی‌توان آن را کاهش داد. با افزايش ناهمگونی کارها، الگوريتم ALATO به سرعت خود را به مينيمم نزديک می‌کند و در ناهمگون‌ترين حالت، اختلاف خود را به کمترين ميزان ممکن می‌رساند. در اين روند به تدريج با افزايش فاصله‌ی طولی بين کوچکترين و بزرگترين کارها، الگوريتم می‌تواند با تهيه‌ی يک زمانبندی مناسب و توزيع متعادل کارها بين منابع، زمان اجرای کارها بر روی منابع را به هم نزديک کرده و زمان اجرا بر روی پرکارترين منبع (زمان اجرای برنامه‌ی کاربردی از نظر کاربر) را کاهش دهد ولی به تدريج با ناهمگون شدن کارها توانسته است افزايش کارايي بيشتری را داشته باشد به طوری که در ناهمگونی (200...10)، میزان نزدیک شدن ALATO به الگوریتم مینیمم که بهترین حالت اجرای کارهاست درحدود 10 واحد می بباشد. در حالی که در این حات LATO به میزان 90 واحد تا حالت بهینه اش فاصله دارد.



شکل ‏5‑26: اختلاف زمان اجرای کارها در الگوريتم‌های LATO و ALATOبا الگوریتم Minimum

در شکل 5-27 نتيجه‌ی بدست آمده از الگوريتم‌های بهينه‌سازی زمان در دو حالت همگون و ناهمگون آورده شده است. در حالت همگون، محدوده‌ی توزيع تصادفی و يکنواخت طول کارها محدوده‌ی (110...100) و در حالت ناهمگون اين محدوده برابر با (200...10) می‌باشد. به جز الگوريتم BTO که در هر دو حالت تقريباً يکسان عمل می‌کند، در بقيه‌ی الگوريتم‌ها مقداری بهبود در حالت ناهمگون نسبت به حالت همگون بوجود آمده است. اين بهبود در الگوريتم‌های LATO و ALATO نسبت به بقيه قابل ملاحظه‌تر می‌باشد. همچنين می‌توان مشاهده کرد که در حالت همگون، نتيجه‌ی بدست آمده از الگوريتم‌ها به هم نزديک است؛ در حالی که در وضعيت ناهمگون، زمان اجرای کارها در الگوريتم‌ها با هم اختلاف زيادی پيدا می‌کند.



شکل ‏5‑27: نتايج الگوريتم‌های بهينه‌سازی زمان برای کارهای همگون و ناهمگون

## خلاصه و نتیجه گیری

در اين فصل، عمل شبيه‌سازی الگوريتم‌ پيشنهادی و الگوریتم های قبلی ارائه شده در فصل قبل را انجام داديم. اين کار با استفاده از جعبه‌ابزار GridSim و نرم افزار SPNP که امکانات لازم برای شبيه‌سازی گريدهای محاسباتی اقتصادی و سيستم‌های مديريت منابع در آنها را فراهم می‌کند، انجام گرفت. همچنين از بسته‌ی نرم‌افزاری gridbroker که توسط بويا برای مدل‌سازی وظايف واسطه‌گر منبع عرضه شده است، به صورت تغيير يافته و با افزودن امکانات لازم استفاده کرديم.

با استفاده از شبيه‌سازی الگوريتم‌های بهينه‌سازی زمان مشخص شد که الگوريتم يادگير ALATO در ناهمگونی زياد کارها، نسبت به الگوريتم‌های مکاشفه‌ای نتايج بهتری را بدست می‌آورد و می‌تواند برنامه‌ی کاربردی متعلق به کاربر را با صرف بودجه‌ی تعيين شده، در مدت زمان کمتری نسبت به الگوريتم‌های ديگر، بر روی گريد اجرا کند. همچنين با توجه به اختلاف اندک زمان بدست آمده از اين الگوريتم با مينيمم زمان محاسبات، امکان بهبود قابل ملاحظه در الگوريتم ALATO وجود ندارد.

فصل ششم

**نتیجه گیری**

# نتيجه‌گيری و پيشنهادها

هدف اصلي از اين پروژه ارائه يک شبکه پتري تصادفی تطبيقي مبتني بر اتوماتاي يادگيري مي­باشد، شبکه پتري تصادفی تطبيقي از طريق اطلاعات بدست آمده از حالات قبلي سيستم و واکنش­هاي محيط پويا، حالت بهينه بعدي را پيشگويي نموده و وضعيت جاري سيستم را بروز و احتمال وقوع رويدادها را در طول زمان تغيير مي­دهد و باعث مي­شود رويداد­ها بر اساس احتمال وقوعشان فعال ­شوند. بروز شدن وضعيت­هاي سيستم بر اساس واکنش محيط پويا کمک شاياني در يادگيري و آموزش شبکه­هاي پتري مي­کند. در قسمت ديگري از اين پروژه­ كاربرد مدل تطبيقي پيشنهاد شده، در گريد محاسباتی و گرید اقتصادی مورد مطالعه و بررسي قرار مي­گيرد

کارهای انجام شده در زمینه­ی شبکه پتری تطابقی به دو دسته­ی Fusion Hybrid و - Combination Hybrid تقسیم شده­اند. شبکه پتری تصادفی تطابقی ارائه شده برای بررسی در گرید محاسباتی، جزء دسته Fusion Hybrid می­باشد، همانطور که بیان شد. این دسته معایبی دارند. به­علت ادغام شبکه­های پتری با تکنیک­های هوشمند، اغلب اوقات یادگیری در شبکه های پتری از این دسته باعث افزایش پیچیدگی عملیات و افزایش سربار محاسباتی می­شود. در شبکه­های پتری تطابقی ارائه شده تاکنون، از تکنیک­های هوشمند نظير شبکه­هاي عصبي مصنوعي، منطق فازي و سيستم­هاي مبتني بر دانش بهره برده­اند، ولی شبکه پتری تصادفی تطابقی که ما ارائه کرده­ایم مبتنی بر اتوماتای یادگیر و هچنین 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 الگوی­های از قبل مشخص­شده نیست، در سیستم­هایی که محیط کاملا 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 کاملا 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 محاسبانی 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. شبکه پتری تصادفی فازی تطابقی ارائه شده از تکنیک هوشمند اتوماتای یادگیر بعلاوه سيستم­هاي مبتني بر دانش در شبکه پتری تصادفی استفاده شده است، که جزء هر دو دسته Fusion Hybridو 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، كاربرد مدل­های تطبيقي پيشنهاد شده، در گريد محاسباتی و گرید اقتصادی مورد مطالعه و بررسي قرار گرفت. در گرید اقتصادی از مدل تطبيقي ارائه شده براي تخصيص بهينه منابع در گريد با توجه به معيار زمان استفاده شد، الگوريتم پیشنهادی (ALATO) با الگوريتم‌های مکاشفه‌اي(BTO)(AEBTO) و الگوريتم‌ يادگيري تقويتي(LATO) مقايسه گردید، که در مقايسه با الگوريتم LATO تنها مراحل جريمه و پاداش تغيير کرد. با تغيير مهمی که در مرحله‌ی جريمه دادن به کارها، ايجاد شد، کارايي الگوریتم پیشنهادی نسبت به الگوريتم LATO بهبود قابل ملاحظه‌ای پيدا کرد. شبيه‌سازی‌های انجام شده نشان داد که نتايج الگوريتم يادگير در استراتژی بهينه‌سازی زمان اختلاف کمی با مينيمم زمان محاسبات دارد. اين اختلاف اندک برای الگوريتم يادگير ارائه شده برای بهينه‌سازی زمان نيز نسبت به مينيمم زمان محاسبات وجود دارد.

از کارهايي که می‌توان در ادامه‌ی کار اين پايان‌نامه در آينده انجام داد، ارائه‌ی استراتژی‌های جديد برای زمانبندی اقتصادی می‌باشد. بايد توجه کرد که يک استراتژی جديد بايد برای کاربر گريد قابل فهم بوده و بتواند در صرفه‌جويي زمان و هزينه، نقش داشته باشد.

يکی از پيشنهادهايي که می‌توان در مورد الگوريتم‌های زمانبندی اقتصادی مطرح کرد، بررسی و ارائه‌ی روش‌های زمانبندی جديد در مدل‌های اقتصادی غير از مدل بازار کالا (که در اين پايان‌نامه، مد نظر بود) می‌باشد. از رايج‌ترين آنها می‌توان به مدل‌های اقتصادی حراجی و چانه‌زنی اشاره کرد.

به عنوان آخرين پيشنهاد می‌توان به مبحث کاملاً جديد بيمه اشاره کرد. در اين پیشنهاد بيمه در برابر تأخير زمانی مورد بررسی قرار خواهد گرفت که در آن کاربر در برابر تأخير در تحويل نتيجه‌ی برنامه‌ی خود از سيستم خسارت دريافت کند. تأخير در انجام درخواست مشتريان می‌تواند به دلايل مختلفی مانند خرابی شبکه، قطع برق، از کار افتادن بخشی از يک منبع و حوادث احتمالی ديگر بوجود بيايد که سيستم از قبل از رخ دادن آنها اطلاع ندارد. بنابراين گريد می‌تواند شبکه و اجزای مختلف منابع و تجهيزات ديگر خود را توسط شرکت‌های بيمه در مقابل اين حوادث بيمه کرده و در مقابل، خود، مشتريان را با دريافت مبلغی اضافه در برابر تأخيرهای احتمالی بيمه کند. برای اين کار بايد تحليل جامعی بر روی اتفاقاتی که ممکن است بر روند اجرای برنامه‌ها در گريد تأثير بگذارد انجام شود و با توجه به اطلاعات گذشته و هزينه‌های بيمه‌ای و خسارت‌های احتمالی وارد شده، ميزان افزايش هزينه‌ی اجرای برنامه‌ها محاسبه گردد و از کاربرانی که خواستار بيمه در برابر همه يا بخشی از اين حوادث هستند، دريافت گردد.

همچنین، می توان از بکاربردن سایر الگوریتم های هوشمند همچون شبکه عصبی، الگوریتم های ژنتیک در شبکه پتری تصادفی تطابقی در سایر کاربردها نام برد.

# 

فصل هفتم

**پیوست ها**

# پیوست ها

## واژه نامه ها

|  |  |
| --- | --- |
| عمل | Action |
| احتمالات عمل‌ها | Action Probability |
| سازگارپذيري | Adaptability |
| کنترل قابل تطبيق | Adaptive Control |
| شبکه پتری تصادفی تطبیقی | Adaptive Stochastic Petri Net(ASPN) |
| حوزه‌ی مديريتی | Administrative Domain |
| کنترل پذيرش | Admission Control |
| عامل | Agent |
| شبکه های عصبی مصنوعی | Artificial Neural Network (ANN) |
| ناهمزمان | Asynchronous |
| خودمختار | Autonomous |
| هزینه متوسط پاسخگویی بازای هر درخواست | Average Response Cost per Request (ART) |
| زمان متوسط پاسخگویی بازای هر درخواست | Average Response Time per Request (APT) |
| آکسون ها | Axons |
| انتشار به عقب | Back propagation |
| برنامه‌ی دسته­ای | Batch Application |
| شاخص اندازه گیری- شاخص سنجیدن | Benchmarks |
| دسته بندی الگو های دودویی | Binary Pattern Classification |
| حد و مرز پذیری | Boundness |
| واسطه‌گر-دلال | Broker |
| بویا | Buyya |
| ارتباط اتفاقي | Casual Connection |
| فاکتور قطعیت | Certainty Factor |
| چاندراسِکار | Chandrasekhar |
| مقدار عضویت کلاس | Class Membership Value |
| محاسبات کلاستري | Cluster Computing (CC) |
| همکاری کننده | Collaborative |
| شبکه پتری رنگی | Colored Petri Net(CPN) |
| زمان اتمام- زمان کامل شدن | Completion Time |
| پیچیده | Complex |
| گريدهای محاسباتی | Computational Grids |
| شبکه‌ی محاسباتی | Computational Network |
| همروندی | Concurrency |
| زنجیره مارکو پیوسته زمانی | Continues-Time Markov Chain(CTMC) |
| حلقه ربایی | Cycle-Stealing |
| گريدهاي داده‌اي | Data Grids |
| داده‌کاوی | Data Mining |
| انباره‌ی داده | Data Warehouse |
| بن بست | Deadlock |
| اشکال زدائی | Debugging |
| فاکتور تضعیف کننده | Decay Factor |
| دندریت (شاخه های متعدد سلول عصبی) | Dendrite |
| قطعی | Deterministic |
| اتوماتاي قطعي | Deterministic Automata |
| شبكه‌هاي پتري تصادفی معين | Deterministic Stochastic Petri Net(DSPN) |
| گراف جهت دار غیر حلقوی | Directed Acyclic Graph(DAG) |
| محيط‌هاي محاسباتی توزيع شده‌ | Distributed Computing Environments (DCE) |
| شبكه‌هاي پتري تصادفی توزیع شده | Distributed SPN(DSPN) |
| حوزه دانش | Domain Knowledge |
| دی وی دی | DVD |
| الکتروکاردیوگراف | Electrocardiograph |
| الکترومایگرام | Electromyogram |
| المنت- عنصر | Element |
| نهاد | Entity |
| گرید داده ای اروپایی | European Data Grid (EDG) |
| توابع تحریک | Excitatory |
| نمایش آشكار | Explicit Representation |
| آزمايش‌ | Experiment |
| شبکه پتری توسعه یافته ای | Extended Petri Net(EPN) |
| شبكه‌هاي پتري تصادفی گسترش یافته | Extended SPN(ESPN) |
| توسعه‌پذيري | Extensibility |
| رخدادهای خارجی | External Events |
| شكست‌ها | Failures |
| مطلوب | Favorable |
| ماشين حالت متناهي | Finite State Machine (FSM) |
| فعال شدن | Fire |
| ثابت | Fixed |
| اتوماتاي با ساختار ثابت | Fixed Structure |
| مسطح | Flat |
| سیستم های تولید قابل انعطاف صنعتی | Flexible Manufacturing System(FMS) |
| فو | Fu |
| استنتاج فازی | Fuzzy inference |
| موتور استنتاج فازی | Fuzzy inference engine |
| منطق فازی | Fuzzy Logic (FL) |
| شبکه پتری عصبی فازی | Fuzzy Neural Petri Net(FNPN) |
| تولید قوانین فازی | Fuzzy production Rules |
| الگوریتم های استنتاج فازی | Fuzzy Reasoning Algorithm |
| شبکه پتری استدلال فازی | Fuzzy Reasoning Petri Net(FRPN) |
| تئوري بازي | Game Theory |
| شبکه مرکب گوسین | Gaussian Mixed Network |
| شبکه پتری تصادفی عمومی | General Stochastic Petri Net(GSPN) |
| توابع انتقال کلی حلقوی یا مارپیچی | Generic Sigmoid Transfer Function |
| محاسبات سراسري | Global Computing (GC) |
| گلوباس | Globus |
| جداسازی گراف | Graph partitioning |
| گريد محاسباتی | Grid Computing |
| لایه گرید | Grid Layer |
| گریدلت | Gridlet |
| هاس | Hass |
| هب | HEBB |
| ناهمگون | Heterogeneous |
| مکاشفه ای | Heuristic |
| سلسله‌مراتبي | Hierarchical |
| شبکه پتری سطح بالا | Hierarchical Petri Net(HPN) |
| شبکه پتری تصادفی که بصورت سلسله مراتبی | Hierarchical Stochastic Petri Net(HSPN) |
| شبکه پتری زماندار سلسله مراتبی | Hierarchical Timed Petri Net(HTPN) |
| استنتاج سطح بالا | High level Reasoning |
| لایه خانگی | Home Layer |
| هاپ فیلد | Hopfield |
| شناخت انسان | Human Cognition |
| ژنراتور بسيار ريز | Infinitesimal Generator |
| توابع بازداشت | Inhibitory |
| بطور مجتمع | Integrated |
| محدوديت‌هاي جامعيت | Integrity Constraints |
| رابط - واسط | Interface |
| رخدادهای داخلی | Internal Events |
| عه داده های آیریس | Iris Data set |
| کار محور | Job Centric |
| سیستم های مبتنی بر دانش | Knowledge Base System (KBS) |
| بازنمایی دانش | Knowledge Representation |
| اتوماتاي يادگير | Learning Automata (LA) |
| اتوماتای یادگیر: تئوری و کاربردها | Learning Automata: Theory and Application |
| شبکه پتری یادگیر | Learning Petri Network(L.P.N) |
| جوازها - مجوزها | License |
| روش عدم پاداش خطي | Linear Reward Inaction (LRI ) Scheme |
| روش جريمه‌ی پاداش خطي | Linear Reward Penalty (LRP ) Scheme |
| روش جريمه‌ی جزئي پاداش خطي | Linear Reward Penalty Epsilon Penalty (LReP ) Scheme |
| تحریک پذیری | Liveness |
| لایه محلی | Local Layer |
| مدل ثبت وقایع بصورت خطی | Log Linear Model |
| نشانه گذاری- علامت گذاری | Marking |
| مک لارن | McLaren |
| اندازه گیری | Measurement |
| فرامحاسبات | Meta Computing |
| ناهموار سطوح فلزات | Metal Surface Roughness |
| متودولوژی | Methodology |
| فرز- ذوب | Milling |
| مولی | Molloy |
| چند رسانه¬ای | Multimedia |
| اجراي دوجانبه | Mutual Exclusion |
| نَجیم | Najim |
| فضاي نام | Namespace |
| نارندرا | Narendra |
| ناتکین | Natkin |
| برنامه‌ی كاربردي شبکه | Network Application |
| شبکه پتری خبره عصبی فازی | Neural Fuzzy Expert Petri Net(NFEPN) |
| شبکه پتری عصبی | Neural Petri Net(NPN) |
| غيرپيش‌گويانه | Non-predictive |
| محيط‌هاي ناپايدار | non-stationary |
| خالی- صفر | Null |
| اتوماتاي مهاجرت اشياء | Object Migration Automata (OMA) |
| مدل شيء | Object Model |
| سيستم‌هاي شی گرا | Object Oriented Systems |
| تقسيم‌بندي اشياء | Object Partitioning |
| غيربرخط | Off-line |
| بر خط | On-line |
| سربار- سرایند | Overhead |
| تخمين پارامترها | Parameter Estimation |
| پارامترروب | Parameter Sweep |
| برنامه‌ريزي مسير | Path planning |
| طبقه بندی الگوها | Pattern Classification |
| تشخيص الگو | Pattern Recognition |
| جریمه | Penalty |
| ارزیابی کارایی | Performance Evaluation |
| شکاف های پس سیناسپی | Pre-Synaptic Clefts |
| تابع شدت احتمال | probability density function (pdf) |
| تابع توزیع احتمال | Probability Distribution Function(PDF) |
| گزاره ها | Propositions |
| پیش نوع ها | Prototype |
| پوژنیاک | Pznyak |
| تئوری صف | Queuing Theory |
| قابلیت دسترسی | Reachability |
| آنی – بلا درنگ | Real Time |
| واقع گرایانه | Realistic |
| سيستم‌هاي زمان واقعي | Real-Time Systems |
| راه دور | Remote |
| زمانبندی مجدد | Rescheduling |
| منبع | Resource |
| مصرف‌کننده‌ی منبع | Resource Consumer |
| يافتن و انتشار منبع | Resource Discovery and Dissemination (RDD) |
| سيستم مديريت منبع | Resource Management System (RMS) |
| رابط همتاي مدير منبع | Resource Manager Peer Interface (RMPI) |
| رابط پشتيباني مدير منبع | Resource Manager Support Interface (RMSI) |
| فراهم‌کننده‌ی منبع | Resource Provider (RP) |
| استفاده‌ی مجدد | Reusability |
| پاداش | Reward |
| بی خطری – امن بودن | Safeness |
| مقياس¬پذيری | Scalability |
| سياست زمانبندي | Scheduling policy |
| شِما- ااگوی خاص | Schema |
| محسوس | Sensitive |
| گريد خدماتي | Service Grid |
| شِن | Shen |
| شیلدر | Shielder |
| سرور واحد | Single Server |
| خودمختاري سايت | Site Autonomy |
| برآورد وضعیت | State Estimation |
| حالات- وضعیت | States |
| محيط‌هاي پايدار | Stationary |
| پایدار | Steady-State |
| اتوماتاهاي يادگير تصادفي | Stochastic Learning Automata (SLA) |
| شبكه‌هاي پتري تصادفی | Stochastic Petri Net (SPN) |
| زیر کارها | Subtasks |
| موفقيت‌ها | Successes |
| ابرمحاسبات | Supercomputing |
| همزمانی | Synchronization |
| سيستم محور | System Centric |
| علامتگذاري‌هاي محسوس | Tangible Marking |
| کار- درخواست | Task |
| موقتی | Temporal |
| تاتاچار | Thathachar |
| درخواست پر محتوا | Thick Request |
| درخواست کم محتوا | Thin Request |
| توان عملياتی | Throughput |
| حالت زمانی- گذار زمانی | Time Transition |
| زمان سنج | Timer |
| علائم – نشانه ها | Tokens |
| جعبه‌ابزارها | Toolkits |
| ماتريس انتقال | Transition matrix |
| گذارها- حالت ها | Transitions |
| شفاف | Transparent |
| راه اندازی کردن | Triggers |
| تستلين | Tsetlin |
| تاپل- ستون جدول | Tuple |
| نامطلوب | Unfavorable |
| غير فعال | Unfired |
| اتوماتای با ساختار متغير | Variable Structure |
| اتوماتاي با ساختار متغير | Variable structure learning Automata |
| وارشاوسکی | Varshavski |
| بنگاه‌های مجازی | Virtual Enterprises (VE) |
| سازمان های مجازی | Virtual Organizations(VOs) |
| مجازي‌سازي | Virtualization |
| ویواناتان | Viswanatan |
| وورونتسودا | Vorontsova |
| قانون تنظیم شده فازی | Weighted Fuzzy Production Rule |
| پردازش کلمه –پردازش لغت | Word Processing |
| بار کاری | Workload |
| ايستگاه‌هاي کاري | Workstations |
| سيپکين | Y.Z. Tsypkin |

## مروری بر نرم افزار SPNP

نرم افزار Stochastic Petri Net Package(SPNP) یک ابزار مدلسازی برای مدل های شبکه پتری می باشند. مدل های شبکه پتری تصادفی با زبان CSPL یا زبان C در SPN توصیف می شوند. این زبان یک زبان گسترش یافته بر اساس زبان برنامه نویسی C با اضافه کردن ساختارها و توانمدیها­ی قابل استفاده در مدل SPN می باشد. در ادامه به توصیف کلی برنامه نویسی زبان CSPL و نحوه اجرا و راه اندازی این نرم افزار می پردازیم.

### راه اندازی در ویندوز XP

يك کاربر باید یک دیسک فشرده اي كه شامل كتاب راهنما شارپ رابط کاربری گرافیکی و یا فایل شارپ Gui.zip مي باشد را دريافت كند. 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 بروید و 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 متغییر Path اضافه کنید. حال در ادامه با زدن دگمه OK به ادرس C:\Spnp-Gui\spnp بروید و فایل Interface SPNP.exe را اجرا کنید.

### فایل­های خروجی در SPNP

فايلهاي مياني توسط پكيج ( بسته ) توليد ميشوند و نتايج نهايي در همان مسيري كه filename.c هست ، خواهد بود ( جايي كه شما دستور را از آنجا صادر كرديد ). فايلها بر اساس نوع اطلاعاتي كه آنها منتقل ميكنند مصاديق مختلفي دارند. اگر فايل CSPL شما test.c نامگذاري شده است ، بنابراين فايلهاي زير توليد خواهند شد :

* test.o: هنگام تركيب test.c بدست ميآيد.
* test.spn: توسط فايلهاي موضوعي پكيج (بسته) يكديگر با test.o فايلهاي اجرايي بدست مي­آيد.
* test.rg: شامل اطلاعات نمودارقابل لمس ميشود: تركيب هر نشانه گذاري ، توصيف انتقال ميان آنهاست و غيره.
* test.mc: شامل (عددي) CTMC/DTMC متناظر با SRN.
* test-parmname.mc: شامل (عددي) مشتق از CTMC با در نظر گرفتن پارامتر parmname (يك فايل كه توليد شده براي هر پارامتر توليد شده در test.c )
* test.prb: شامل (عددي) نتايج تجزيه CTMC پايين: احتمال شرايط ثابت يا گذرا براي هر نشانه ملموس ، تجميع زمان موقت در وضعيت انتقال به زمان حل و مشتقات (با در نظر گرفتن پاترامترها) از مقياس مذكور.
* test.prbdtmc: شامل (عددي) نتايج محاط كردن DTMC.
* test.out: شامل درخواست خروج (براساس چيزي كه در test.c مشخص شده توابع مهيا شده استفاده ميشوند).
* test.log: شامل همه پيغامهاي خروجي توليد شده توسط پكيج (بسته) مدل حل.
* test.dot: شامل توصيف شبكه در زبان نموداري dot.

### زبان CSPL

زبان CSPL ميتواند بعنوان يك سيستم براي توصيف تعريف شده باشد، پردازش و تحليل تصادفي شبكه ها (SRNs). نحو و معاني CSPL بر اساس زبان ANSI C است. بعلاوه يك فايل CSPL صحيح همچنين يك فايل ANSI C صحيح است.چيزي كه CSPL را از ANSI C تميز ميدهد مجموعه اي از توابع از پيش تعريف شده براي تعيين و تحليل SRNsهاست.آشنايي با زبان ANSI C مزيت بسيار بزرگي براي يك كاربر براي بيشترين كار برجسته از تركيب زبان CSPL خواهد بود.بنابراين آن بايد براي كاربر مشكل نباشد توانايي تعريف و حل SRNs خودش بموجب مطالعه قسمت كمك كتاب مباني، مخصوصا" با مثالهاي درون آن. هر ساختار قانوني زبان ANSI C براساس اينكه نياز شده در زبان CSPL منظور شده است. توابع و متغير هاي تعريف شده كاربر مي تواند در فايل CSPL استفاده شده باشد.خصوصا" همه توابع استاندارد C، از قبيل fbrintf(), fscanf(), log(), exp() و غيره ، در صورت نياز ميتواند در CSPL نامگذاري شود. يك فايل CSPL بايد شامل توابع پايه زير باشد:

* Options()
* Net()
* Assert()
* Ac\_init()
* Ac\_reach()
* Ac\_final()

هر تابع فهرست شده براي اجراي يك (يا چند) وظيفه (وظائف) توسط بعضي از زيرروالها ي (توابع) مربوطه طراحي شده است.اين وظایف مربوط به تعاريف و تحليل پردازش بجاي شبكه ها هستند.

**تابع Option()**

**Void options (void);**

يك فايل معتبر CSPL بايد شامل تابع options() باشد.تابع options با زيرروالهاي زير:

* Iopt**(**)
* Fopt()
* Input()
* Finput()

براي تنظيم اختياراتي كه از طريق توصيف و تحليل SRN تأثير خواهد داشت ناميده مي­شوند. لیست های زیر در Option می آید. جدول های7-1 تا 7-4 در زیر آمده است.

جدول ‏7‑1: گزینه های موجود برای فایل­های میانی

|  |  |  |
| --- | --- | --- |
| پیش فرض | ارزش ها | نام |
| VAL NO  VAL NO  VAL CANONIC  VAL YES  VAL NO  VAL NO  VAL NO  VAL NO  VAL FROMTO  VAL NO  VAL NO  VAL NO | VAL YES, VAL NO, VAL TAN  VAL YES, VAL NO  VAL CANONIC VAL LEXICAL VAL MATRIX  VAL YES, VAL NO  VAL YES, VAL NO  VAL YES, VAL NO  VAL YES, VAL NO  VAL YES, VAL NO  VAL FROMTO, VAL TOFROM  VAL YES, VAL NO  VAL YES, VAL NO  VAL YES, VAL NO | IOP PR RSET  IOP PR RGRAPH  IOP PR MARK ORDER  IOP PR MERG MARK  IOP PR FULL MARK  IOP USENAME  IOP PR MC  IOP PR DERMC  IOP PR MC ORDER  IOP PR PROB  IOP PR PROBDTMC  IOP PR DOT |

جدول ‏7‑2: راه حل تحلیلی و عددی option های موجود

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

جدول ‏7‑3: Option های شبیه­ساز موجود

|  |  |
| --- | --- |
| ارزش ها | نام |
| VAL YES, VAL NO  non-negative int  VAL REPL, VAL BATCH, VAL RESTART VAL SPLIT, VAL IS, VA  non-negative int  VAL YES, VAL NO  VAL YES, VAL NO  non-negative **double**  VAL YES, VAL NO  non-negative **double**  VAL YES, VAL NO  non-negative **double**  non-negative **double**  non-negative **double**  non-negative **double** | IOP SIMULATION  IOP SIM RUNS  IOP SIM RUNMETHOD  IOP SIM SEED  IOP SIM CUMULATIVE  IOP SIM STD REPORT  IOP SPLIT LEVEL DOWN  IOP SPLIT PRESIM  IOP SPLIT NUMBER  IOP SPLIT RESTART FINISH  IOP SPLIT PRESIM RUNS  FOP SIM LENGTH  FOP SIM CONFIDENCE  FOP SIM ERROR |

جدول ‏7‑4: Option های متفرقه

|  |  |  |
| --- | --- | --- |
| پیش فرض | ارزش ها | نام |
| VAL REDONTHEFLY  VAL NO  VAL NO  VAL YES  VAL YES  0.0  VAL NO  0.000001  0.000001 | VAL REDONTHEFLY, VAL REDAFTERRG, VAL REDNEVER  VAL YES, VAL NO  VAL YES, VAL NO  VAL YES, VAL NO  VAL YES, VAL NO  non-negative double  VAL YES, VAL NO  non-negative double  non-negative double | IOP ELIMINATION  IOP OK ABSMARK  IOP OK VANLOOP  IOP OK TRANS M0  IOP OK VAN M0  FOP ABS RET M0  IOP DEBUG  FOP FLUID EPSILON  FOP TIME EPSILON |

**تابع Net()**

**Void net (void);**

يك فايل CSPL معتبر بايد شامل تابع net() باشد. تابع net() مجموعه اي از توابع تعريف شده در SRN ناميده مي­شود.

توابع پايين كافي هستند براي تعريف يك SRN ساده بدون تركيب توسعه يافته از نشانه هاي وابسته فعال، نرخ/وزن و اعداد شمارشي و غيره .

**تابع Place() و init()**

Void place (char\*p);

يك مكان با نام p تعريف ميشود.

يك اسم قانوني است اگر: (1) طول رشته آن بين 1 و MAX\_NAME\_LENGTH بر اساس تعريف شده در فايل const.h باشد ( مثلا" 20)، (2) آن فقط از كاراكترهاي {0... 9 ، a … z ، A … Z } حروفچيني شده است، (3) اولين كاراكتر در مجموعه { A…Z ، a…z } است.همه اسامي بايد متمايز باشد ، كه هست .داشتن دو مكان اشتباه است ، دو ارتباط ، يا يك مكان و ارتباط با نام يكسان.

**تـــابع ِ Void init (char\*p, int n)**

شماره آغازين اعتبار در مكان p به n بودن را نشان ميدهد.بر اساس پيش فرض مكانها از جاتي ديگر در ابتدا خالي هستند.

**تابع Imm()**

تحولات و تغييرات بصورت اتوماتيك تعريف ميشود وقتي كه مقدار دهي آنها (يا توزيع پذيري آنها) تعريف شد (پايين را ببينيد) . بنابراين تغييرات بي واسطه بايد تعريف شوند توسط :

**Void imm (char\*t);**

**تــوابع ِRateval() and probval()**

**; \*t, doubl Val) Void rateval (cha**

**Void probval (char\*t, doubl Val);**

تعريف ميزان انتقال زمان سوخت شده t و وزن تغييرات بي واسطه سوخت شده (احتمالا" غير معمول) t ، بر اساس يك ارزش ثابت val. تابع rateval ( يا بيشتر در نسخه هاي عمومي آن در بخش زيرين ) ضمنا" تعريف ميشودانتقالات زماني بايد تعريف شده باشد براي هر تغييرات .تابع probval احتياج به نامگذاري دارد فقط اگروزن سوخت شده متفاوت است با ارزش پيش فرض 0.1 كه با تعريف تغييرات بي واسطه توسط imm مي آيد.

**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 زمانیکه 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 ac\_reach. SPNP پیش از شروع ساختار نمودار دسترسی، ac\_init() را فرا می خواند. تابعvoid 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، تابعvoid pr\_parms(void) می تواند در اینجا برای ارائه نامِ گذارهای پارامتری در امتداد نام و مقدار کنونی فایل “.out” فراخوانی شود.

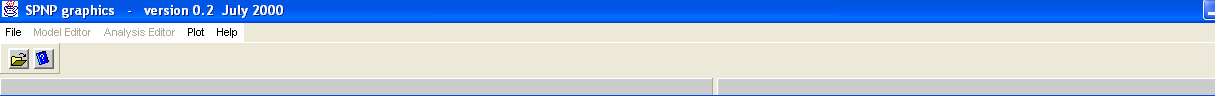
**تابع ac\_final()**

**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.

### طراحی در نرم افزار SPNP

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

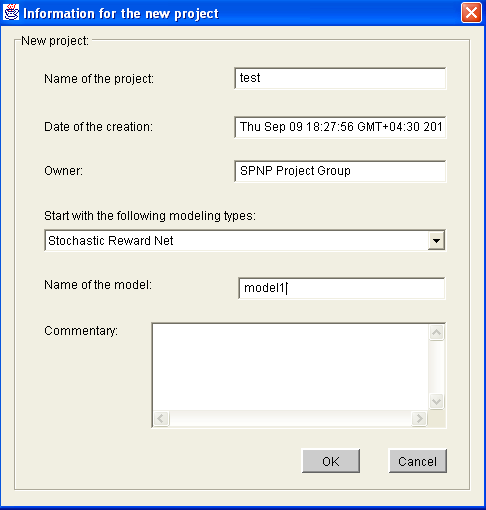


شکل ‏7‑1: قاب اصلي

کاربر باید با کلیک بر روی "فایل" و سپس منوی "جدید" برای ایجاد پروژه جدید و یا ستفاده 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 هنگامي كه جعبه گفتگو ظاهر شد ،فيلدهاي مختلف را پر و بروي Ok كليك كنيد.فيلدهاي 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 :

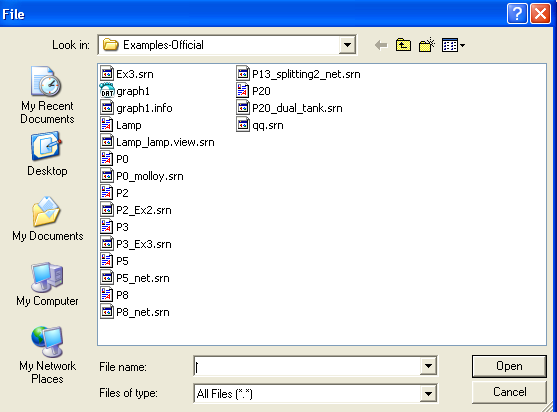
* 1. قسمت سمت چپ که شامل لیست اسامی مدل های موجود در این پروژه است.
  2. سمت راست پانل مورد استفاده برای طراحی مدل فعلی (انتخاب شده از لیست سمت چپ).

برای ذخیره پروژه كاربر مي تواند پروژه فعلی را با تمام مدل های اجراشده در داخل آن ذخیره کند. همچنین می تواند یک مدل تک را ذخیره کند (نگاه کنید به "ذخیره مدل".) پروژه ها مي توانند با فرمت rgl و مدلها مي تواند با فرمت SPNP ذخيره شوند(شکل7-2) . 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 بايد "معتبر" را فشار دهيد. 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 كليك كنيد. شما همچنین می توانید توضیحات مالک و ویرایش را تفسیر كنيد. برای تغییر نام یک مدل از طرح در حال حاضر با كليك بروي "مدل كردن ويرايش گر" وسپس بر روي" اطلاعات " كليك كنيد وبعد بروي نوار مدلها كليك كنيد.



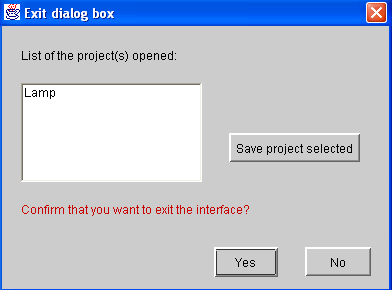
شکل ‏7‑2: پروژه­های جدید

شما می توانید رابطه بین تمام مدل های ایجاد شده در پروژه کنونی را تجسم كنيد. هنگامي كه كادر محاوره اي زير ظاهر مي شود فايل Rgl را انتخاب نموده و بروي Ok كليك كنيد.پسوند پروژه در فايل هاي رابط هميشه Rgl است مانند شکل 7-3.



شکل ‏7‑3

برای بستن پروژه هنگامی که کاربرمي خواهد پروژه را ببندد كادرمحاوره اي ظاهر مي شود كه از شما سوال مي كند كه قبل از بسته شدن مي خواهيد فایل یا فایل های بازشده ای که در لیست پروژه های باز مانده قرار گرفته اند را ذخيره كنيد(شکل 7-4).



شکل ‏7‑4

## کدها

### کد الگوریتم اول

#### 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 findMinMin() {//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() ;    }  }  } قسمت اصلی کدMax.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

### 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(selcetion 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)  {  fpritf("this is req status ",req);  fpritf("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[])  {  fpritf("Please Enter newreq");  scanf(newreq);  fpritf("Please Enter batch group");  scanf(batch[]);  end  func getreqID(req)  void getreqID(int req)  {  if reqestISDel(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  \* Inhibtor Arcs  end  echo \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  echo \*\*\*\*\*\*\*\*\* Outputs asked for the model: General-Cases \*\*\*\*\*\*\*\*\*\*\*\*\*\*  end |

### 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; // ari 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 ariff 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; //(MI/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 barmigardanad  {  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 barmigardanad  {  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(“curren\_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(“curren\_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  {    } |

### 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[])  {  fpritf("Please Enter newreq");  scanf(newreq);  fpritf("Please Enter batch group");  scanf(batch[]);  end  func getreqID(req)  void getreqID(int req)  {  if reqestISDel(req)  empty  else  printf("request have been Finished");  end  func newSchedulingEvent(Schedule, Time)  double newSchedulingEvent(int \*schedule,double Time)  {  return(SchedulingEvent);  end  srn Schedular(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  \* Inhibtor 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()f 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 |

# مراجع

**مراجع**

M. K. Molloy, *On the Integration of Delay and Throughput Measures in Distributed Processing Models*, PhD thesis, UCLA, Los Angeles, CA, 1981.

S. Natkin, *Les Reseaux de Petri Stochastiques et leur Application a l’Evaluation des Systemes Informatiques*. PhD thesis, CNAM, Paris, France, 1980.

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.

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>

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>

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.

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.)

J. D. C. Little, *A Proof of the Queueing Formula L = w*, Operations Research, Vol. 9, 1961, pp. 383-387. DOI: <http://dx.doi.org/10.2307/167570>

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>

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.

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.

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

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.

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 inAdvances on Petri Nets '87 G. Rozenberg editor, LNCS 266, Springer Verlag, 1987.

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, LCNS 266, Springer Verlag, 1987, pp.132-145.

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>

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>

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>

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.

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>

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, EEE Computer Society, Torino, Italy, ISBN: 0-8186-0674-6, July 1985, pp.96-103.

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.

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>

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>

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>

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>

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>

M. K. Molloy, *"Structurally Bounded Stochastic Petri Nets"*, Proceedings of the International Workshop on Petri Nets and Performance Models,published in IEEE Computer Scoiety, Madison, WI, USA, ISBN: 0-8186-0796-3, 1987, pp. 156-163. DOI: <http://portal.acm.org/citation.cfm?id=645810.670635>

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>

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>

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>

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 Mdorca, Spain, September 1989.

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>

G. Balbo, *Introducation 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/0dg47hmt2ynjyk2m/fulltext.pdf>

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>

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.

Ya. Z. Tsypkin, *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>

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>

K. S. Narendra, R. Viswanathan, *"Learning models using stochastic automata"*, Proceedings of International Conference of Cybernetics and Society, Washington DC, October 1972.

V. I. Varshavsky, I. P. Vorontsova, *On the behavior of stochastic automata with variable structure*, Avtomatika & Telemechanika, Vol. 24, 1963, pp. 353-360.

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.

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.

K. S. Fu, T. J. Li, *On stochastic automata and languages*, Journal of Information Science, Vol. 1, Issue 4, 1969, pp. 403-419.

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.

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.

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.

K. S. Narendra, M. A. L. Thathachar, *Learning automata: An introduction*, Prentice Hall, 1989.

K. A. Najim, S. Poznyak, *Learning automata: theory and application*, Tarrytown, New York: Elsevier Science Publishing Ltd., 1994.

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.

S. Lakshmivarahan, *Learning algorithms: theory and applications*, New York: Springer-Verlag, 1981.

K. S. Narendra, M. A. L. Thathachar, *Learning automata a survey*, IEEE Transactions on Systems, Man and Cybernetics, vol. 4, Issue 4, July 1974.

B. J. Oommen, D. C. Y. Ma, *Deterministic learning automata solutions to equi-partioning problem*, IEEE Transactions on Computers, Vol. 37, January 1988,pp. 2-14.

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.

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.

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.

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.

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.

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.

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.

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.

K. S. Narendra, M. A. L. Thathachar, *Learning automata: An introduction*, Prentice Hall, 1989.

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.

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.

L.G. Mason, *An Optimal Learning Algorithm for S-model Environments,* IEEE Transactions on Automatic Control, October 1973, pp. 493-496.

I. Foster, C. Kesselman, *the Grid 2: Blueprint for a New Computing Infrastructure*, Morgan Kaufmann, San Francisco, 2003.

I. Foster, C. Kesselman, S. Tuecke, *The anatomy of the Grid: Enabling scalable virtual organizations*, International Journal of Supercomputer Applications, 2001.

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.

V. Berstis, *Fundamentals of Grid Computing*, IBM Redbook series, November 2002, <http://ibm.com/redbooks>

W. T. Sullivan, D. Werthimer, S. Bowyer, J. Cobb, D. Gedye, 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/>

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.

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/>

S. Smallen, W. Cirne, 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.

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.

I. Foster, *"An Open Grid Services Architecture"*, IBM OMG Web Services Workshop, March 2002.

I. Ekmecic, I. Tartalja, V. Milutinovic, *"A taxonomy of heterogeneous computing"*, IEEE computer, Vol. 28, No.12, 1995, pp. 68-70.

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.

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.

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.

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.

I. Ekmecic, I. Tartalja, V. Milutinovic, *"A taxonomy of heterogeneous computing"*, IEEE computer, Vol. 28, No.12, 1995, pp. 68-70.

J. Peterson, *Petri Net Theory and the Modeling of Systems*, Perantic Hall, Englewood Cliffs, NJ, Vol. 1, 1981.

W. Reisig, *Petri Nets: An Introduction*, EATCS Monographs on Theoretical Computer Science, Springer Verlag, Vol. 1, 1985.

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.

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.

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.

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.

J. Moore, L. Hahn, *"Petri Net Modeling of High-Order Genetic Systems Using Grammatical Evolution"*, Bio Systems 72, 2003, pp. 177–186.

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.

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.

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.

M. LaLena, *"Travelling Salesman Problem Using Genetic Algorithms"*, <http://www.lalena.com/ai/tsp/> , 2007.

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.

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.

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.

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.

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.

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.

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>

M. C. Zhou, M. D. Jeng, *Modeling, analysis, simulation, scheduling, and control of semiconductor manufacturing systems: A Petri netapproach*, 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>

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>

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>

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>

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>

K. Tsuji, T. Matsumoto, *“Extended Petri net Models fur 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>

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>

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>

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>

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>

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>

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](http://dx.doi.org/10.1109/ROBOT.1998.677304%20)

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>

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>

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>

X. Chen, D. Jin, L. I. Zhijan, *"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](http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=1179002%20)

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](http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=1244222%20%20)

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>

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>

A. H. Song, J. T. Ootsuki, W. K. Yoo, Y. Fujii, T. Aekigchu, *"FMS’s Scheduling by colored Petri Net Model and Hopefield 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>

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>

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>

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>

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.

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>

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>

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>

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.

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.

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>

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.

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.

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/>.

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.

R. Buyya, "*The World-Wide Grid (WWG)*", <http://www.buyya.com/ecogrid/wwg/>, 2002.

****

**دانشگاه آزاد اسلامي - واحد قزوين**

**دانشکدهء مهندسي برق، کامپیوتر و فناوري اطلاعات**

**گروه کامپیوتر**

**پايان نامه کارشناسي ارشد مهندسی کامپيوتر**

**گرايش نرم‌افزار**

**يك مدل شبکه پتری تصادفی تطبیقی مبتنی بر اتوماتاهای یادگیر­ و­ کاربردهای آن درتخصیص منابع در شبکه­های گرید**

**نگارش: محمد شجاع‌فر**

**استاد راهنما: دكتر محمد‌رضا ميبدي**

**تابستان 1389**

1. Molloy [↑](#footnote-ref-2)
2. Natkin [↑](#footnote-ref-3)
3. Transitions [↑](#footnote-ref-4)
4. Fire [↑](#footnote-ref-5)
5. Stochastic Petri Net (SPN) [↑](#footnote-ref-6)
6. Marking [↑](#footnote-ref-7)
7. probability density function (pdf) [↑](#footnote-ref-8)
8. Timer [↑](#footnote-ref-9)
9. Null [↑](#footnote-ref-10)
10. Continues-Time Markov Chain [↑](#footnote-ref-11)
11. Performance Evaluation [↑](#footnote-ref-12)
12. Concurrency [↑](#footnote-ref-13)
13. Synchronization [↑](#footnote-ref-14)
14. Temporal [↑](#footnote-ref-15)
15. Time Transition [↑](#footnote-ref-16)
16. Deterministic [↑](#footnote-ref-17)
17. Tokens [↑](#footnote-ref-18)
18. Probability Distribution Function(PDF) [↑](#footnote-ref-19)
19. Queuing Theory [↑](#footnote-ref-20)
20. Safeness [↑](#footnote-ref-21)
21. Measurement [↑](#footnote-ref-22)
22. Benchmarks [↑](#footnote-ref-23)
23. Prototype [↑](#footnote-ref-24)
24. Workload [↑](#footnote-ref-25)
25. General Stochastic Petri Net(GSPN) [↑](#footnote-ref-26)
26. Real Time [↑](#footnote-ref-27)
27. Sensitive [↑](#footnote-ref-28)
28. Tangible Marking [↑](#footnote-ref-29)
29. Infinitesimal Generator [↑](#footnote-ref-30)
30. Explicit Representation [↑](#footnote-ref-31)
31. Realistic [↑](#footnote-ref-32)
32. Distributed SPN(DSPN) [↑](#footnote-ref-33)
33. Extended SPN(ESPN) [↑](#footnote-ref-34)
34. Deterministic Stochastic Petri Net(DSPN) [↑](#footnote-ref-35)
35. Steady-State [↑](#footnote-ref-36)
36. Single Server [↑](#footnote-ref-37)
37. Unfired [↑](#footnote-ref-38)
38. Object Oriented Systems [↑](#footnote-ref-39)
39. Real-Time Systems [↑](#footnote-ref-40)
40. Hass [↑](#footnote-ref-41)
41. Shielder [↑](#footnote-ref-42)
42. Debugging [↑](#footnote-ref-43)
43. Casual Connection [↑](#footnote-ref-44)
44. Mutual Exclusion [↑](#footnote-ref-45)
45. Stochastic Learning Automata (SLA) [↑](#footnote-ref-46)
46. Tsetlin [↑](#footnote-ref-47)
47. Object Partitioning [↑](#footnote-ref-48)
48. Adaptive Control [↑](#footnote-ref-49)
49. Learning Automata (LA) [↑](#footnote-ref-50)
50. Action [↑](#footnote-ref-51)
51. Y.Z. Tsypkin [↑](#footnote-ref-52)
52. Narendra [↑](#footnote-ref-53)
53. Viswanatan [↑](#footnote-ref-54)
54. Varshavski [↑](#footnote-ref-55)
55. Vorontsova [↑](#footnote-ref-56)
56. Action Probability [↑](#footnote-ref-57)
57. States [↑](#footnote-ref-58)
58. Fu [↑](#footnote-ref-59)
59. Parameter Estimation [↑](#footnote-ref-60)
60. Pattern Recognition [↑](#footnote-ref-61)
61. Game Theory [↑](#footnote-ref-62)
62. McLaren [↑](#footnote-ref-63)
63. Chandrasekar [↑](#footnote-ref-64)
64. Shen [↑](#footnote-ref-65)
65. Thathachar [↑](#footnote-ref-66)
66. Najim [↑](#footnote-ref-67)
67. Pznyak [↑](#footnote-ref-68)
68. Learning Automata: Theory and Application [↑](#footnote-ref-69)
69. Graph Partitioning [↑](#footnote-ref-70)
70. Path Planning [↑](#footnote-ref-71)
71. Variable Structure Learning Automata [↑](#footnote-ref-72)
72. Finite State Machine (FSM) [↑](#footnote-ref-73)
73. Deterministic Automata [↑](#footnote-ref-74)
74. Fixed Structure [↑](#footnote-ref-75)
75. Variable Structure [↑](#footnote-ref-76)
76. Reward [↑](#footnote-ref-77)
77. Penalty [↑](#footnote-ref-78)
78. Stationary [↑](#footnote-ref-79)
79. Non-Stationary [↑](#footnote-ref-80)
80. Favorable [↑](#footnote-ref-81)
81. Unfavorable [↑](#footnote-ref-82)
82. Linear Reward Penalty (LRP) Scheme [↑](#footnote-ref-83)
83. Linear Reward Penalty Epsilon Penalty () Scheme [↑](#footnote-ref-84)
84. Linear Reward Inaction () Scheme [↑](#footnote-ref-85)
85. Transition matrix [↑](#footnote-ref-86)
86. Successes [↑](#footnote-ref-87)
87. Failures [↑](#footnote-ref-88)
88. Object Migration Automata (OMA) [↑](#footnote-ref-89)
89. Computational Grids [↑](#footnote-ref-90)
90. Virtual Enterprises (VE) [↑](#footnote-ref-91)
91. Workstations [↑](#footnote-ref-92)
92. Grid Computing [↑](#footnote-ref-93)
93. Supercomputing [↑](#footnote-ref-94)
94. Transparent [↑](#footnote-ref-95)
95. Virtualization [↑](#footnote-ref-96)
96. Global Computing [↑](#footnote-ref-97)
97. Meta Computing [↑](#footnote-ref-98)
98. cluster Computing [↑](#footnote-ref-99)
99. Administrative Domain [↑](#footnote-ref-100)
100. Workstations [↑](#footnote-ref-101)
101. Cycle-Stealing [↑](#footnote-ref-102)
102. Throughput [↑](#footnote-ref-103)
103. Completion Time [↑](#footnote-ref-104)
104. Parameter Sweep [↑](#footnote-ref-105)
105. Data Grids [↑](#footnote-ref-106)
106. Data Warehouse [↑](#footnote-ref-107)
107. Data Mining [↑](#footnote-ref-108)
108. European DataGrid [↑](#footnote-ref-109)
109. Globus [↑](#footnote-ref-110)
110. Service grid [↑](#footnote-ref-111)
111. Collaborative [↑](#footnote-ref-112)
112. Multimedia [↑](#footnote-ref-113)
113. Overhead [↑](#footnote-ref-114)
114. Batch Application [↑](#footnote-ref-115)
115. Word Processing [↑](#footnote-ref-116)
116. Remote [↑](#footnote-ref-117)
117. Scalability [↑](#footnote-ref-118)
118. License [↑](#footnote-ref-119)
119. DVD [↑](#footnote-ref-120)
120. Resource [↑](#footnote-ref-121)
121. Entity [↑](#footnote-ref-122)
122. Reusability [↑](#footnote-ref-123)
123. Resource Provider [↑](#footnote-ref-124)
124. Agent [↑](#footnote-ref-125)
125. Broker [↑](#footnote-ref-126)
126. Resource Consumer [↑](#footnote-ref-127)
127. Resource Management System (RMS) [↑](#footnote-ref-128)
128. Computational Network [↑](#footnote-ref-129)
129. System Centric [↑](#footnote-ref-130)
130. Job Centric [↑](#footnote-ref-131)
131. Extensibility [↑](#footnote-ref-132)
132. Adaptability [↑](#footnote-ref-133)
133. Site Autonomy [↑](#footnote-ref-134)
134. Distributed Computing Environments (DCE) [↑](#footnote-ref-135)
135. Toolkits [↑](#footnote-ref-136)
136. Interface [↑](#footnote-ref-137)
137. Resource Manager Support Interface (RMSI) [↑](#footnote-ref-138)
138. Resource Manager Peer Interface (RMPI) [↑](#footnote-ref-139)
139. Flat [↑](#footnote-ref-140)
140. Hierarchical [↑](#footnote-ref-141)
141. Schema [↑](#footnote-ref-142)
142. Integrity Constraints [↑](#footnote-ref-143)
143. Object Model [↑](#footnote-ref-144)
144. Fixed [↑](#footnote-ref-145)
145. NameSpace [↑](#footnote-ref-146)
146. Resource Discovery and Dissemination [↑](#footnote-ref-147)
147. Network Application [↑](#footnote-ref-148)
148. State Estimation [↑](#footnote-ref-149)
149. Non-predictive [↑](#footnote-ref-150)
150. Heuristic [↑](#footnote-ref-151)
151. Off-line [↑](#footnote-ref-152)
152. On-line [↑](#footnote-ref-153)
153. Rescheduling [↑](#footnote-ref-154)
154. Scheduling Policy [↑](#footnote-ref-155)
155. Artificial Neural Network(ANN) [↑](#footnote-ref-156)
156. Fuzzy Logic(FL) [↑](#footnote-ref-157)
157. Knowledge Base System(KBS) [↑](#footnote-ref-158)
158. Methodology [↑](#footnote-ref-159)
159. High level Reasoning [↑](#footnote-ref-160)
160. Knowledge Representation [↑](#footnote-ref-161)
161. Extended Petri Net(EPN) [↑](#footnote-ref-162)
162. Liveness [↑](#footnote-ref-163)
163. Reachability [↑](#footnote-ref-164)
164. Boundness [↑](#footnote-ref-165)
165. Fuzzy inference engine [↑](#footnote-ref-166)
166. Decay Factor [↑](#footnote-ref-167)
167. Hierarchical Petri Net(HPN) [↑](#footnote-ref-168)
168. Element [↑](#footnote-ref-169)
169. Pre-Synaptic Clefts [↑](#footnote-ref-170)
170. Dendrite [↑](#footnote-ref-171)
171. Axons [↑](#footnote-ref-172)
172. Excitatory [↑](#footnote-ref-173)
173. Inhibitory [↑](#footnote-ref-174)
174. Fuzzy Neural Petri Net(FNPN) [↑](#footnote-ref-175)
175. Fuzzy inference [↑](#footnote-ref-176)
176. Neural Petri Net(NPN) [↑](#footnote-ref-177)
177. Tuple(ستون جدول) [↑](#footnote-ref-178)
178. Propositions [↑](#footnote-ref-179)
179. Binary Pattern Classification [↑](#footnote-ref-180)
180. Learning Petri Network(L.P.N) [↑](#footnote-ref-181)
181. HEBB [↑](#footnote-ref-182)
182. Back propagation [↑](#footnote-ref-183)
183. Neural Fuzzy Expert Petri Net(NFEPN) [↑](#footnote-ref-184)
184. Generic Sigmoid Transfer Function [↑](#footnote-ref-185)
185. Fuzzy production Rules [↑](#footnote-ref-186)
186. Human Cognition [↑](#footnote-ref-187)
187. Weighted Fuzzy Production Rule [↑](#footnote-ref-188)
188. Fuzzy Reasoning Algorithm [↑](#footnote-ref-189)
189. Pattern Classification [↑](#footnote-ref-190)
190. Domain Knowledge [↑](#footnote-ref-191)
191. Certainty Factor [↑](#footnote-ref-192)
192. Class Membership Value [↑](#footnote-ref-193)
193. Iris Data set [↑](#footnote-ref-194)
194. Fuzzy Reasoning Petri Net(FRPN) [↑](#footnote-ref-195)
195. Milling [↑](#footnote-ref-196)
196. Metal Surface Roughness [↑](#footnote-ref-197)
197. Triggers [↑](#footnote-ref-198)
198. Flexible Manufacturing System(FMS) [↑](#footnote-ref-199)
199. Hopfield [↑](#footnote-ref-200)
200. Electromyogram [↑](#footnote-ref-201)
201. Electrocardiograph [↑](#footnote-ref-202)
202. Log Linear Model [↑](#footnote-ref-203)
203. Gaussian Mixed Network [↑](#footnote-ref-204)
204. Adaptive Stochastic Petri Net(ASPN) [↑](#footnote-ref-205)
205. Complex [↑](#footnote-ref-206)
206. Heterogeneous [↑](#footnote-ref-207)
207. Autonomous [↑](#footnote-ref-208)
208. Hierarchical Stochastic Petri Net(HSPN) [↑](#footnote-ref-209)
209. Home Layer [↑](#footnote-ref-210)
210. Local Layer [↑](#footnote-ref-211)
211. Grid Layer [↑](#footnote-ref-212)
212. Subtasks [↑](#footnote-ref-213)
213. Integrated [↑](#footnote-ref-214)
214. Virtual Organizations(VOs) [↑](#footnote-ref-215)
215. Yu [↑](#footnote-ref-216)
216. Buyya [↑](#footnote-ref-217)
217. Directed Acyclic Graph(DAG) [↑](#footnote-ref-218)
218. Colored Petri Net(CPN) [↑](#footnote-ref-219)
219. Hierarchical Timed Petri Net(HTPN) [↑](#footnote-ref-220)
220. Yaojin [↑](#footnote-ref-221)
221. Zhao [↑](#footnote-ref-222)
222. Gridlet [↑](#footnote-ref-223)
223. Internal Events [↑](#footnote-ref-224)
224. External Events [↑](#footnote-ref-225)
225. Asynchronous [↑](#footnote-ref-226)
226. Deadlock [↑](#footnote-ref-227)
227. Task [↑](#footnote-ref-228)
228. Thin Request [↑](#footnote-ref-229)
229. Thick Request [↑](#footnote-ref-230)
230. Average Response Time per Request [↑](#footnote-ref-231)
231. Average Response Cost per Request [↑](#footnote-ref-232)
232. Admission Control [↑](#footnote-ref-233)
233. Application Layer [↑](#footnote-ref-234)
234. Workflow [↑](#footnote-ref-235)
235. Fabric Layer [↑](#footnote-ref-236)
236. Middleware Layer [↑](#footnote-ref-237)
237. Experiment [↑](#footnote-ref-238)