نهمین کنفرانس سالانه انجمن کامپیوتر ایران
دانشگاه صنعتی شریف
۳۰-۲۸ بهمن ماه ۱۳۸۲

کاربرد تقسیم مجدد وظایف به صورت گسترده در سیستم‌های چندعاملهٔ غیرهمسان برای افزایش تحمل‌پذیری‌خطا

مریم سادات میریان حسین‌آبادی
mirian@ece.ut.ac.ir

مجید نیلی احمدآبادی
mnili@ut.ac.ir

قطب علمی کنترل و پردازش هوشمند و آزمایشگاه هوش مصنوعی و رباتیک،
گروه مهندسی برق و کامپیوتر. دانشکده فنی- دانشگاه تهران

چکیده

در این مقاله، برای افزایش تحمل‌پذیری خطا در سیستم‌های گسترده[1] (توزیع شده) وبه ویژه سیستم‌های چندعامله روشهایی پیشنهاد شده است. این روشها حتی با وجود شرایط نایقینی و در نظر گرفتن عاملهای غیرهمسان که برخی دارای قابلیتهای ویژه‌ای هستند، با مبتنی بودن بر یک الگوریتم نسبتا بهینه برای تقسیم مجدد وظایف، عمل می‌نمایند . روشهای پیشنهادی این تحقیق، در تکمیل پژوهشهای قبلی و به ویژه [۱۲]صورت گرفته اند. در این روش، عاملهای کمک‌رسان در صورت لزوم، بر اساس الگوریتمی گسترده و نسبتا بهینه[2] اقدام به انتخاب وظیفه می‌کنند و بدون برقراری ارتباط صریح، مناسب‌ترین وظیفه را از لحاظ سرعت و توان پردازشی(عملیاتی) و قابلیت اعتماد خود و نیز درجه اهمیت وظیفهٔ مورد نظر بر عهده می‌گیرند . بنابراین از این طریق همکاری ضمنی میان عاملهای کمک‌رسان برقرار می‌شود وتا زمانی که شرایط نیاز به کمک برطرف نشده، به این روش از کاهش کارایی سیستم جلوگیری می‌گردد. این مکانیزم جدید برای تقسیم وظیفه، عملکرد سیستم را در مقایسه با روشهای قبلی مطرح شده در پژوهشهای پیشین که راهکاری برای کنارآمدن با نایقینی و ناهمسانی عاملها، نداشتند به میزان قابل توجهی بهبود می بخشد

کلمات کلیدی : سیستم چندعامله، تحمل‌پذیری خطا، الگوریتم تقسیم وظیفه، تصمیم‌گیری گسترده[3].

۱. مقدمه

در سال‌های اخیر، بخش قابل‌توجهی از تحقیقاتی که در زمینهٔ هوش ماشین و رباتیک انجام شده، معطوف به سیستم‌های گسترده بوده است. دلایل زیر به صورت خاص برای توجیه گسترش و استفاده روزافزون سیستم‌های چندرباته یا چندعامله، عنوان شده است:[۱] گسترده بودن اغلب کاربردهای رباتیکی درمکان، زمان یاعملیات ، افزایش سرعتی که غالبا درصورت تقسیم‌شدن وظیفه بین مولفه‌های همکار که بصورت موازی عمل می‌کنند، به دست می آید، افزایش قابلیت اطمینانی که با فراهم آوردن افزونگی (در قابلیتها یا تعداد عامل‌ها) به دست می آید و بالاخره اینکه معمولاً ساخت تعدادی ربات یا عامل با توانایی اندک که با همکاری یکدیگر قادر به انجام یک مأموریت باشند، بسیار ارزان‌تر و عملی‌تر از ساخت یک ربات یا عامل است که به تنهایی و با قابلیت‌اطمینان مطلوب بتواند کل ماموریت را انجام دهد. البته با وجود پژوهش‌های فراوانی که دردانشگاهها و مراکزتحقیقاتی در مورد سیستم‌های چندعامله یا چندرباته انجام شده، هنوز در عمل و در بسیاری ازکاربردهای دنیای‌واقعی، این تکنولوژی جایگزین روشهای موجود نشده‌است. شاید دلیل اصلی این امر را بتوان در آسیب‌پذیری فراوان آنها نسبت به اشکال‌ها وخرابی‌های رایجی دانست که در مکانیزم‌های مختلف سازندهٔ رباتها رخ می‌دهد، یا اینکه کنارنیامدن عاملها را با تغییرات فراوانی که درمحیط عملیاتی پویای آنها رخ می‌دهد، علت این مساله برشمرد. این موارد یا دلایل مشابه اینها، لزوم افزودن‌قابلیت تحمل‌پذیری‌خطا به رباتها و عاملها و در نتیجه طراحی سیستم‌های چندعاملهٔ با قوام[4] بیشتر را تقویت می‌کند

این مقاله، به مسالهٔ ایجاد تحمل‌پذیری خطا به کمک روشهای ساده، عملی و در عین‌حال کم‌هزینه، در کاربردهایی می‌پردازد که قابلیت توزیع‌شدگی را دارند. چنین سیستم‌هایی با بهره‌گیری از راهکارهایی که در این تحقیق معرفی شده‌اند، می‌توانند طوری تغییر یابند که حتی در صورت بروز خطا برای تعداد زیادی از عناصر سیستم تا جایی که سیستم حداقل تعداد عناصر سالمی را شامل باشد، از کار باز نمانند و حداقل بحرانی‌ترین وظایفی که در زمان طراحی برایش تعیین شده را به درستی به انجام برساند. اگر توجه خود را در حالت کلی به سیستم‌های گسترده معطوف نماییم، هر

---

[1] Distributed
[2] Sub-Optimal
[3] Distributed Decision-Making
4 Robustness

سیستم گسترده را می‌توان به منزلهٔ تیمی از عامل‌های همکار[5] دانست که هر یک وظیفهٔ خاصی به عهده دارند و از تعامل آنها با یکدیگر یک عملکرد واحد حاصل می‌شود. این شیوهٔ نگرش موجب می‌شود که بتوان از ایدهٔ سیستم‌های چندعامله برای توسعهٔ یک سیستم گستردهٔ تحمل‌پذیر خطا استفاده نمود. ایدهٔ اساسی این تحقیق آن است که، به جای جایگزین نمودن یک عامل خراب با یک عامل دیگر از خارج، از قابلیت‌های رزرو دیگر عاملهای موجود استفاده کرده، نقش عامل خراب را به نحوی بر دوش همکارانش قرار دهیم تا شرایط بحرانی با کمترین تاثیر نامطلوب بر کارایی مرتفع گردد. به بیان دیگر، بهره‌گیری از خود عامل‌های درگیر در حل مساله به منظور رفع خطا در سیستم مورد تاکید است بدون اینکه از عامل واسطه‌ای[6] به منظور ایجاد هماهنگی در تنظیم وظایف جدید استفاده شود. در این مقاله پس از بررسی کارهای مشابه انجام شده در این زمینه و مرور دستاوردهای قبلی این تحقیق، به بیان نحوهٔ تحقق بستر آزمون[7] می‌پردازیم. سپس روشهای پیشنهادی این مقاله را تشریح می‌کنیم . در ادامه معیار ارزیابی کارایی شرح داده می‌شود و بعد از توضیحی در مورد طراحی آزمایشها و نتایج شبیه‌سازی، نتیجه‌گیری و کارهای آینده مطرح می‌گردد.

۲. مروری بر کارهای مشابه

Deen در[۲]محیطی توزیع شده برای سیستم‌های توزیع‌شده همکار پیشنهاد داده است که در این محیط تحمل‌پذیری خطا با جایگزینی عامل‌های خراب انجام می‌شود و پس از جایگزینی عامل در برنامهٔ گروه تجدیدنظر صورت می‌گیرد. در [۳] قابلیت سازمان‌دهی مجدد در مدل یادگیری سازمانی بررسی شده است. این عمل با هدف حفظ کارآیی مجموعه ربات‌ها در مواقع بروز خطا صورت می‌گیرد. هنگامی که در گروه ربات‌ها تغییری بوجود آید(بدین صورت که تعدادی ازآنها معیوب شوند ویا از کار بیفتند) ربات‌ها باید با تغییر سازمان سعی کنندکه وظیفهٔ گروه را به انجام برسانند. روش فوق برای سیستم‌های زمان‌حقیقی[8] روش مناسبی نیست. چون در این سیستم‌ها زمان اهمیت زیادی دارد و علاوه برآن با توجه به اهمیت میدان عملیاتی که خیلی از فرایندها برگشت‌پذیر نیستند جایی برای یادگیری وجود ندارد. در[۴]عامل‌های محافظ به عنوان روشی برای ایجاد تحمل‌پذیری خطا در سیستم‌های چند عامله پیشنهاد شده‌است. عامل‌های محافظ ، به منظور حفظ یک سری ویژگی‌ها و نیز جلوگیری از ورود به وضعیت‌های ناخواسته در سیستم قرار می‌گیرند. عامل‌های محافظ یک ساختار کنترلی برای سیستم‌های چندعامله پدید . در [۵] استفاده از کارتیمی برای ساختن معماری مقاومی که بتواند بر خطاهایی که برای واسطها به وجود می‌آید، فائق گردد پیشنهاد شده‌است. در[۶] ALLIANCE که یک معماری کنترلی توزیع شده تحمل‌پذیرخطا برای رباتهای همکارغیر متجانس است و با انتخاب اعمال به صورت تطبیقی کنترل همکاری بین رباتها را انجام می‌دهد، معرفی شده است. در این معماری، رباتها توانایی انجام تعدادی عملیات سطح بالا که در یک مأموریت، منجر به انجام وظیفه می‌شوند را دارا هستند و در هر زمان باید یک عمل مناسب را با توجه به نیازمندیهای مأموریت، فعالیت‌های سایر رباتها، حالت محیط و حالات درونی ربات انتخاب کنند. در[۷] که در ادامهٔ معماری ALLIANCE معرفی شده‌است، معماری تکمیل‌یافته‌ای با نام L-ALLIANCE معرفی شده‌است که در آن تقسیم‌وظیفهٔ اولیه به گونه‌ای انجام شده که مقداری همپوشانی در قابلیت انجام وظیفه میان رباتها موجود باشد. در این طرح دو عمل تقسیم مجدد وظیفه[9] و ترتیب‌دهی وظیفه[10] توسط رباتها صورت می‌گیرد. در فرایند کمک‌رسانی که در[۱] شرح داده شده است، دو نکته بسیار مهم در کمک رسانی وجود دارد که عبارتند از مکانیزم انتخاب عمل و هماهنگی بین ربات‌ها در عملیات کمک‌رسانی. در این تحقیق قابلیت پشتیبانی کمک به معماری ALLIANCE اضافه شده است. در حالی که این تحقیق منحصر به رباتهایی با وظیفه‌هایی مشخص است؛ مقاله حاضر مشابه همین ایده را به محدوده وسیعتری از سیستمها یعنی سیستمهای گسترده توسعه می دهد بدون اینکه معماری خاصی مانند ALLIANCE را مدنظر داشته باشد.

[۸] به عنوان اولین گام این تحقیقات، به بیان ساده‌ترین روش کمک‌رسانی می‌پردازد که تنها مبتنی بر درجه‌اهمیت وظیفهٔ عامل‌هاست.در ادامه برای افزودن بر پویایی مکانیزم تصمیم‌گیری،[۹] به بیان چند استراتژی سادهٔ کمک‌رسانی مبتنی بر ریسک می‌پردازد و در[۱۰] مفهوم بی‌صبری برای کمک‌رسانی با تعریف ساده‌ای معرفی شده‌است. مفهوم بی‌صبری در[۱۱]توسعه یافته و از دانش عاملها به نحو جامعتری استفاده می‌نماید. در[۱۲] استراتژی‌های ساده و ترکیبی برای تعیین ترتیب کمک‌رسانی معرفی شده‌اند، اما از تقسیم وظیفه گسترده میان عاملها در صورت بروز خطا و کنارآمدن با شرایط نایقینی و غیرهمسانی عاملها که در این مقاله بر آنها تاکید می‌گردد، بحثی به میان نیامده است. در[۱۳] ساده ترین روش غیرقطعی کمک‌رسانی مبتنی بر الگوریتم تقسیم وظیفهٔ نسبتا بهینه مطرح شده‌است.

5 Cooperative Agents
6 Middle Agents
7 Testbed
8 Real-Time Systems
9 Task Re-allocation
10 Task Ordering

۳. شرح سیستم

سیستمی که برای تحقق ایده‌های این پژوهش برگزیده شده‌است، شبیه یک سیستم کنترل گسترده[11] ولی بسیار ساده‌تر است. علت انتخاب این سیستم گسترده برای چنین منظوری آن است که این سیستم به علت داشتن عناصر مستقل و قابلیت تعریف انواع وظیفه‌ها برای هر عامل، نمونهٔ خوبی از یک سیستم توزیع‌شده در حالت کلی است. البته انتخاب این سیستم از کلیت طرح و ایده‌ها نمی‌کاهد، چرا که وابستگی مفهومی میان ایده‌های پیشنهادی در حالت کلی با این بستر آزمون وجود ندارد. در این سیستم هر عامل به منزله یک کنترلر است. داده‌های مورد نیاز هر عامل توسط تولید کنندهٔ بسته داده‌ای[12] از سنسورهای موجود در محیط جمع‌آوری می‌شود و در اختیار عاملها قرار می‌گیرد. عامل‌ها با دسترسی به باس مشترک ورودی دادهٔ مورد نظر خود را برمی‌دارند و پس از انجام وظیفهٔ موردنظر، نتیجهٔ خروجی خود را بر روی باس خروجی قرار می‌دهند. آنچه شرح داده شد، عملکرد سیستم در حالت عادی است. در صورتی که خطا در سیستم رخ بدهد، عامل نیازمند به کمک این مساله را به اطلاع بقیه می رساند. در صورتی که عاملی خود را واجد شرایط کمک‌رسانی بیابد (که در مورد مساله واجد شرایط بودن و نحوه همکاری ضمنی میان عاملها کمی جلوتر بحث خواهد شد) پس ازطی مراحل لازم، کمک‌رسانی به وی را آغاز می‌کند.

۳- ۱ شرح وظایف سیستم

در این سیستم، هر عامل در حالت عادی انجام یک وظیفه را بر عهده دارد. وظیفه ها در حالت قطعی به صورت پریودیک و در حالت غیرقطعی با پیروی از یک الگوی تصادفی گوسی، به عاملها محول می شوند. در صورتی که یک عامل وظیفه‌ای که به او محول شده‌است را پیش از به سر رسیدن مهلت زمانی‌اش به انجام برساند و نتیجهٔ خروجی را بر روی باس موردنظر قرار دهد، این وظیفه به طور موفقیت‌آمیزی به پایان رسیده‌است. ویژگی هایی که یک وظیفه به کمک آنها توصیف می گردند، عبارتند از: حجم متوسط پردازشی[13]، طول زمان اجرا[14]، مهلت زمانی مقرر[15] و درجهٔ اهمیت[16].

بستر آزمون در نظر گرفته شده برای تست، دارای مولفه‌های اصلی زیر است:

تولید کنندهٔ بستهٔ داده‌ای : به منظور تولید داده های مربوط به هر عامل، این مولفه بسته‌های داده‌ای را تولید می‌کند و از طریق باس مشترک ورودی در اختیار همهٔ عاملها قرار می‌دهد. از طریق این مولفه می‌توانیم با نرخ ثابت یا به صورت متغیر تصادفی با توزیع مشخص به تولید داده بپردازیم. در گونه‌ای که نرخ آمدن داده‌های ورودی یکسان و ثابت است، هر عامل بسته‌های داده‌ای با پریود ثابت و برابر به عامل منسوب می‌شوند و عاملها هم از این نرخ آگاهند و با قطعیت کامل تصمیم می‌گیرند. در مدل غیرقطعی، فرکانس تولید داده‌ها برای هر عامل از یک توزیع نرمال با میانگین و واریانس مشخص پیروی می‌نماید.

باس مشترک ورودی : این باس، باس مشترک ورودی سیستم است که توسط تولیدکنندهٔ بستهٔ داده، تغذیه می شود.

باسهای مشترک به منظور کمک‌رسانی : به منظور اعلام نیاز به کمک و نیز اعلام انجام کمک‌رسانی به دیگر کمک‌رسانها به منظور جلوگیری از ارسال کمک تکراری، دو باس مشترک، هر یک به صورت رجیسترهایی به تعداد عاملها در سیستم قرار گرفته است.

عاملها: در این سیستم تعدادی عامل غیرهمسان از لحاظ ویژگی‌های درونی وجود دارند که از تعامل آنها سیستم چندعامله موردنظر ساخته می‌شود. هرعامل دارای یک ID خاص است که همین ویژگی، موجب ایجاد تفاوتهایی در هر عامل در مقایسه با دیگر هم تیمی‌های وی می‌شود. این تفاوتهای درونی عبارتند از : سرعت ، قابلیت اعتماد(کمیت نسبی که میزان قابلیت اعتماد به عملکرد هر عامل را مشخص می‌کند) و نرخ یادگیری مجازی:(برای هر عامل اثری شبیه یادگیری فرض شده است که با افزایش این یادگیری، طول مدت زمان اجرای وظیفهٔ عامل کاهش یافته، زمان آزادی او افزایش می یابد. به عبارت دیگر هر چه تعداد وظیفه‌های تا به‌حال انجام شده توسط هر عامل افزایش یابد، تسلط نسبی عامل بر انجام وظیفه‌های مشابه بعدی اضافه می‌گردد. نرخ یادگیری مجازی هر عامل با دیگری متفاوت است.)

۳- ۲ فرضیات

در این سیستم به مسالهٔ تشخیص خطا پرداخته نشده‌است و فرض بر این است که هر عامل قادر است از رخداد خطای خود آگاهی یابد و این مساله را با ارسال درخواست کمک به بقیه نیز اطلاع دهد. علاوه بر این فرض شده است که یک عامل قادر است وظیفهٔ هم‌تیمی‌های خود را طبیعتا با سطحی مختلف از لحاظ معیارهای کارایی در مواقع لزوم انجام بدهد. البته چنانچه پیشتر هم اشاره شد به دلیل غیرهمسان بودن عاملها که به منظور افزودن بر جامعیت طرح در نظر گرفته شده است، تعدادی از عاملها با قابلیت ویژه فرض شده‌اند و انجام وظایف آنها توسط هر عامل دیگری لزوما امکان‌پذیر نیست و عاملها موظفند که در زمان تقسیم وظایف به این امر توجه نمایند..

---

[11] DCS: Distributed Control System
[12] Frame Generator
[13] Average Amount of Work
[14] Task Completion Time
[15] Deadline
[16] Criticality

٤. روشهای پیشنهادی

همانطور که قبلا نیز بیان شد، هدف این تحقیق ارائهٔ روشهایی است که در آنها خود عامل‌ها به صورت گسترده و با تغییر نقش، اقدام به تقبل نقش عاملهای خراب می‌کنند تا سیستم با تحمل‌پذیری خطا بیشتری قادر به ادامهٔ کار باشد. در این بخش روشهایی که برای حل این مساله در شرایط مختلف پیشنهاد شده‌است، شرح داده می‌شوند.

۴- ۱ روشهای مبتنی بر تصمیم‌گیری قطعی

طرح اول: ساده ترین تصمیم‌گیری قطعی در مورد امکان‌سنجی کمک(S.T.D.[17])

در این طراحی که ساده‌ترین طرح برای هدف این پروژه است و در[۱۲]تشریح شده است، به محض اینکه عامل کمک‌رسان وظیفهٔ خود را تکمیل نمود، شروع به پردازش تقاضاهای کمک می‌کند. اگر بیش از یک عامل نیازمند کمک باشند، عامل‌های کمک‌رسان بر اساس یکی از استراتژی های تعیین ترتیب (اعم از ساده یا ترکیبی) ترتیب کمک‌رسانی را تعیین می‌کنند. سپس عاملها از ابتدای لیست مرتب بررسی می‌کنند که آیا به عامل موردنظر کمک شده‌است یا خیر. اگر این عامل قبلا توسط عامل دیگری کمک مورد نظر را دریافت کرده‌باشد به سراغ عامل بعدی لیست می‌روند و به همین ترتیب ادامه می‌دهند تا اینکه عاملی بیابند که نیازمند کمک است و هنوز به وی کمک نشده باشد. در این لحظه پرچم موردنظر برای اعلام کمک‌رسانی به این عامل را بالا می‌برند و شروع به انجام کمک می‌نمایند. بنابراین تقسیم درخواست‌های کمک میان عاملهای کمک‌رسان به صورت ضمنی و بر اساس سرعت اجرای وظیفهٔ عامل‌ها صورت می‌گیرد و تنها به کمک مکانیزم شرح داده شده، از ارسال کمک‌های تکراری جلوگیری می‌شود. در طراحی روشهای بعدی سعی شده است که نقاط ضعف موجود در این روش که در بخش نتایج شبیه سازی مطرح شده است، برطرف گردد.

طرح دوم : اِعمال ملاحظات بیشتر به طرح اول برای انتخاب کمک‌رسان بهتر[18](H.H.C.C.)

در این طرح نیز همانند طرح اول، تنها مساله‌ای که موجب رقابت میان عامل‌های کمک‌رسان می‌شود، سرعت رسیدن عامل‌ها به مرحلهٔ کمک‌رسانی است. امّا آنچه این طرح را با طرح اول متفاوت می‌سازد، ملاحظات بیشتری است که عامل‌های کمک‌رسان پیش از اقدام به کمک در نظر می‌گیرند. عاملهای کمک‌رسان در هنگام مواجهه با یک درخواست کمک (که به ترتیب از ابتدای لیست مرتب برمی‌دارند)، پارامترهای قابلیت اطمینان خود، درجه اهمیت وظیفهٔ عامل نیازمند کمک و توجه به داشتن قابلیت ویژهٔ خود و اینکه آیا کمک‌رسانی نیازمند استفاده از قابلیت ویژه است یا خیر را برای تشخیص مناسب بودن عملیات کمک رسانی خود مورد بررسی قرار می‌دهند. در صورتی که درجهٔ اهمیت کار عامل نیازمند کمک زیاد باشد و قابلیت اعتماد عامل کمک‌رسان چندان قابل‌توجه نباشد، عملیات کمک‌رسانی نیاز به تامل بیشتر دارد. در این شرایط عامل کمک‌رسان بررسی می‌کند که آیا کمک‌رسان مناسب‌تری از لحاظ قابلیت اعتماد به پردازنده‌اش وجود دارد یا خیر.

طرح سوم : تقسیم وظیفه مبتنی بر الگوریتم صریح نسبتا بهینه(S.O.T.D.[19])

در روش سوم مبنای کار این است که: انتخاب بهترین و مناسب‌ترین کمک‌خواه‌ها توسط کمک‌رسان‌های واجد شرایط صورت گیرد و به عبارت دقیق‌تر، وظیفه‌های بر زمین‌مانده میان عاملهای کمک‌رسان صریحا تقسیم گردد. در راستای نیل به این هدف، بررسی الگوریتمهایی برای تقسیم وظیفه و زمانبندی در حوزهٔ سیستمهای چندپردازنده‌ای نشان می‌دهند که این روشها غالبا ذات متمرکز دارند و از پیچیدگی زمانی NP-Complete برخوردارند و در ضمن غالب این روشها ابتکاری[20] بوده، محدودیت های فیزیکی یا زمانی برای پردازنده ها یا وظیفه ها قائل نیستند.[۱۴] ، [۱۵]و [۱۶] . در حالی که الگوریتمی که در این طرح برای تقسیم‌وظیفه میان عامل‌ها ارائه شده‌است ، دارای مزایایی از قبیل سادگی در توزیع‌شدگی، پیچیدگی زمانی کم، کم بودن بار ناشی از ارتباطات / هماهنگ‌سازی تصمیم میان عاملها، منطقی بودن تصمیم‌گیری در مقایسه با الگوریتمهای جستجوی حریصانه و بالاخره در نظر گرفتن محدودیتهای فیزیکی / زمانی در طرح است. الگوریتم نسبتا بهینه تقسیم وظیفه در فلوچارت شکل ۱ نشان داده شده است. این الگوریتم مبنای عمل در روشهای بعدی نیز می باشد.

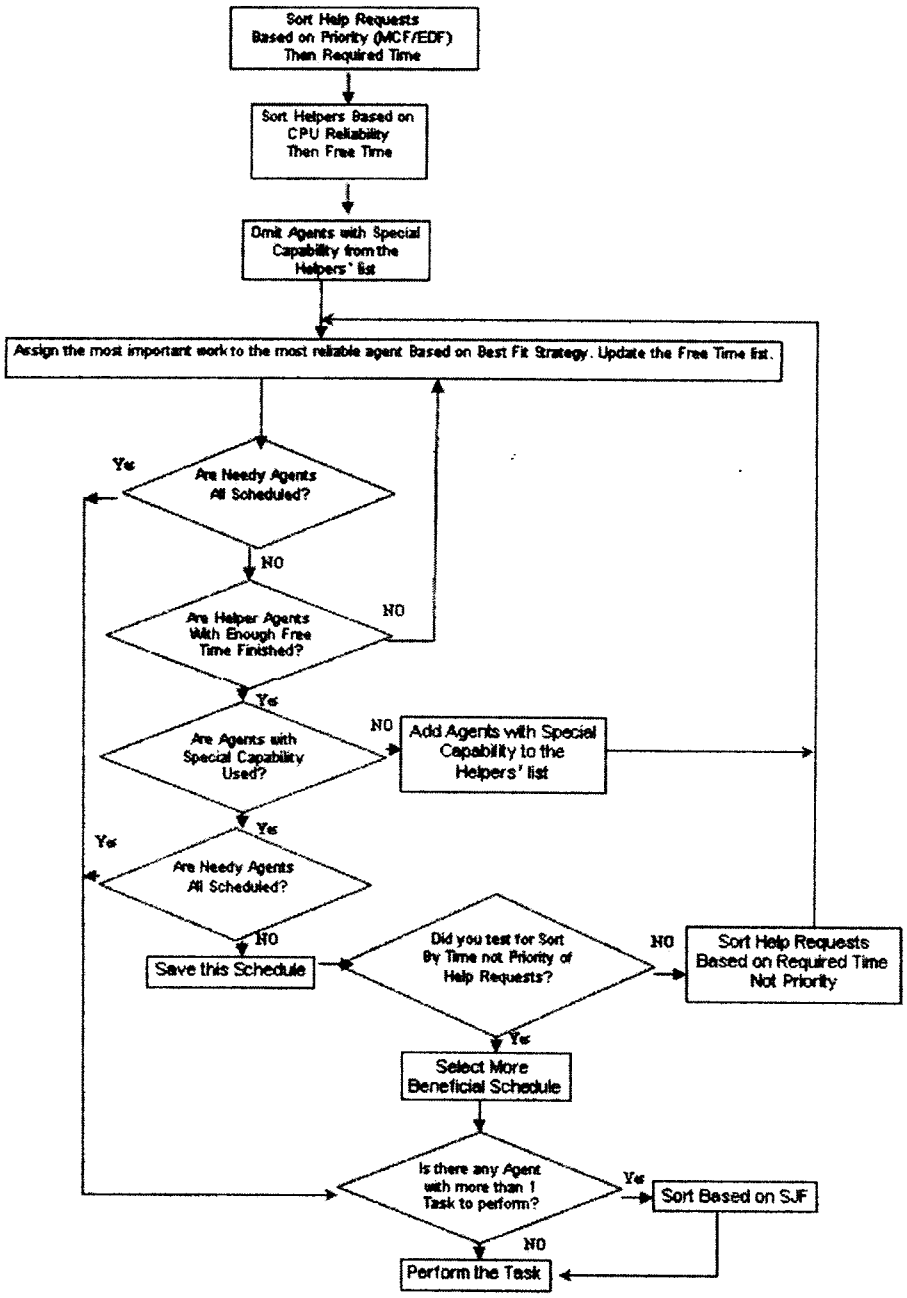
۴- ۲ روش‌های مبتنی بر تصمیم‌گیری غیرقطعی

طرح چهارم : تصمیم‌گیری غیرقطعی در مورد امکانسنجی کمک(N.D.D.M.[21])

این طرح به عنوان تعمیمی برای روش سوم توسعه داده شده است. در این طرح، عامل‌های کمک‌رسان پس از تقسیم وظایف نیازمند کمک‌رسانی، با تصمیم‌گیری غیرقطعی به بررسی امکان‌سنجی انجام‌پذیر بودن وظیفهٔ مورد تقاضا برای کمک‌رسانی از طریق محاسبهٔ احتمالاتی زمان رسیدن وظیفهٔ بعدی خود می‌کنند و در صورتی که با احتمال مناسبی این کار را امکان‌پذیر بیابند، اقدام به انجام کمک می‌کنند و در غیر اینصورت از انجام آن منصرف

---

[17] Speed-Based Task Distribution
[18] Task Distribution with: Help-Helper Compatibility Checking
[19] Sub-Optimal Task Distribution
20 Heuristic
[21] Task Distribution with: Non-Deterministic Decision-Making



شکل ۱ – فلوچارت الگوریتم نسبتا بهینه تقسیم وظیفه

می‌شوند. در این طرح، مولفهٔ تولیدکنندهٔ بسته وظیفه‌هایی با پریودی که از توزیع احتمالاتی برخوردار هستند، برای عاملها تولید می‌کند. البته عاملها از میانگین و انحراف معیار توزیع مطلعند و به همین وسیله قادرند که با عدم قطعیت اقدام به تصمیم‌گیری نمایند.

ایجاد عدم قطعیت در سیستم

مهم‌ترین پارامتری که در این سیستم باید به صورت غیرقطعی تبدیل گردد تا بر جامعیت ایده ها افزوده گردد، پریود رسیدن وظیفه برای عاملهاست. به عبارت دیگر، بهتر است که به جای یک باس سنکرون یک باس آسنکرون را معیار عمل قرار دهیم. بنابراین یک توزیع گوسی ، معیار تولید متغیرهای تصادفی شده است. در اینصورت هر عامل باید محاسبه نماید که با چه احتمالی کار کمک‌رسانی به یک عامل نیازمند، پیش از رسیدن کار بعدی خودش به پایان خواهد رسید. محاسبهٔ احتمال بالا، به این شکل خواهد بود:

$$X = NextTaskAssignment\ ,\quad T = Now + CompletionTime_{HelpTask} + WorstCase_CompletionTime_{NextMyOwnTask}$$

$$P(x > T) = 1 - P(x < T) = 1 - \frac{1}{2}[1 + erf(\frac{x-\mu}{\sigma\sqrt{2}})]$$

در صورتی که این احتمال از حد قابل قبولی که آن را *احتمال ریسک* می‌نامیم بیشتر باشد، عامل می‌تواند با دلگرمی نسبی به اینکه در صورت اقدام به کمک، خللی در انجام وظیفهٔ بعدی وی رخ نمی‌دهد، شروع به کمک کند. احتمال ریسک می‌تواند در چند سطح مختلف تغییر نماید تا بهترین وضعیت از لحاظ سیستم تعیین شود. این سطوح مختلف عبارتند از: احتمال ریسک زیاد، احتمال ریسک متوسط، احتمال ریسک کم و احتمال ریسک تطبیقی. البته بدیهی است که به دلیل احتمالاتی بودن شرایط و عدم قطعیت در تصمیم‌گیری، در مواقعی ممکن است که عامل وظیفهٔ خود را از دست بدهد. برای رفع این مشکل ۲ راه حل وجود دارد : ۱- عامل کمک‌رسان در صورت مشغول بودن به کمک برای وظیفهٔ خود درخواست کمک صادر کند که این روش در همین طرح به وسیلهٔ عامل‌ها به اجرا درمی‌آید. ۲- در صورت رسیدن وظیفهٔ خود عامل، در صورت مشغول بودن وی به کمک‌رسانی، به عامل کمک‌رسان وقفه داده شود تا او پس از طی یک فاز اضافهٔ تصمیم‌گیری، به این نتیجه برسد که آیا اتمام وظیفهٔ کمک‌رسانی مهم‌تر است یا پرداختن به وظیفهٔ خودش. این راه حل نیز در طرح پیشنهادی آخر توسط عاملها به اجرا درمی‌آید.

طرح پنجم: تکمیل تصمیم‌گیری بر اساس محاسبهٔ اضطراب (N.T.S.[22])

در این طرح که به عنوان طرح تکمیلی روش چهارم مطرح است، باز هم عاملها با تصمیم‌گیری غیر قطعی مواجه هستند. اما در اینجا، عامل کمک‌رسان در صورت دریافت وقفه‌ای از سوی سیستم به دلیل وجود وظیفه‌ای منتظر برای خودش، ابتدا وارد فاز بررسی میزان اضطراب خود در انجام وظیفهٔ خود می‌شود. بدین معنا که میزان تمایل خود را بر ادامهٔ کار کمک‌رسانی یا نگرانی پرداختن به وظیفهٔ جدیدی که برای خودش رسیده است، محاسبه می‌نماید. در صورتی که به ادامهٔ کار تمایل بیشتری داشته باشد، برای انجام وظیفهٔ خود درخواست کمک صادر می‌کند و به ادامهٔ کارقبلی خود می‌پردازد (مانند روش چهارم) و اگر نگران پرداختن به وظیفهٔ خودش باشد، کار کمک‌رسانی را با وجود اینکه تا حدی انجام داده است، رها می‌نماید و پرچم مربوط به اعلام وضعیت کمک به آن عامل را که خود بالا برده بود، مجددا Reset می‌نماید تا امکان کمک را به دیگران منتقل نماید.

نحوهٔ محاسبهٔ میزان اضطراب در عامل‌ها در لحظهٔ دریافت وقفه

در محاسبهٔ میزان اضطراب عامل کمک‌رسان در لحظهٔ دریافت وقفه،عوامل زیر دخیل هستند: میزان باقیمانده از کار فعلی، زمان باقیمانده به از دست رفتن مهلت زمانی کار فعلی، زمان باقیمانده به از دست رفتن مهلت زمانی کار جدید، درجهٔ اهمیت کار فعلی، درجهٔ اهمیت کار جدید، میزان این احتمال که وظیفهٔ خود عامل پس از تکمیل کار فعلی برسد، میزان این احتمال که وظیفهٔ خود عامل پس از تکمیل کار جدید بیاید و بالاخره مسالهٔ نیاز یا عدم نیاز کار فعلی و کار جدید به قابلیت ویژهٔ عامل. پس از محاسبه اضطراب، عامل، میزان اضطراب خود را با آستانه اضطراب در نظر گرفته شده مقایسه می نماید و در صورتی بیش از حد آستانه مضطرب باشد، به سراغ کار جدید رفته، وظیفه قبلی را رها می نماید. بدین ترتیب بر خلاف روش قبلی عامل تصمیم گیری را در لحظه درخواست کمک بر اساس میزان اضطرابش انجام می دهد و این روند منطقی‌تری است که برتریهای عملی خود را در نتایج آزمایشها نشان می دهد. نحوهٔ تعیین بهترین آستانه‌ها در[۱۷] تشریح شده است.

۵. شرح آزمایشها

آزمایشهای انجام شده شامل ۳۰ الگوی خطا است که ۲۰ الگوی اول به صورت تصادفی تولید شده‌اند ولی ۱۰ الگوی بعدی، موارد خاص و نسبتا دشوارتری هستند که به صورت دستی و با گنجاندن شرایط ویژه طراحی شده‌اند. الگوهای تصادفی از لحاظ رخداد یا عدم رخداد خطا برای عاملها و نیز طول خطای آنها، با توجه به میزان قابلیت اعتماد عاملها طراحی شده‌اند. بنابراین خطاهای عادی دارای احتمال وقوعی به مراتب بیشتر از خطاهای ویژه می باشند. هر الگوی خطا مستقل از نحوه طراحی آن، اطلاعاتی شامل اینکه کدام عاملها در چه زمانی دچار خطا شده و تا چه هنگامی در وضعیت خطا می مانند را نشان می دهد. در مورد هر یک از الگوهای خطا ۴ وضعیت برای باس ورودی سیستم که توسط تولیدکنندهٔ بستهٔ داده‌ای تغذیه می‌شود و دادهٔ مورد نیاز عاملها را برای انجام وظیفه‌شان در اختیارشان قرار می‌دهد، در نظر گرفته‌شده است: حالت قطعی[23]، حالت غیر قطعی با درجهٔ عدم قطعیت پایین[24]، حالت غیرقطعی با درجه اهمیت متوسط و حالت غیر قطعی با درجهٔ عدم قطعیت بالا. لازم به ذکر است که شرح بیشتر آزمایشها و جزئیات الگوهای خرابی در [۱۷]آورده شده است.

[22] Non-Deterministic Decision-Making with: Nervousness-based Task Switching
23 Deterministic Situation
24 Low Non-Deterministic Situation

۶. معیارهای ارزیابی

کارایی: به عنوان رایج‌ترین معیار ارزیابی، تعداد وظیفه‌هایی که با موفقیت به انجام رسیده‌اند با احتساب درجه اهمیت آنها در مقایسه با کل وظیفه‌هایی که به عاملها محول شده‌اند، در نظر گرفته می شود.

متوسط زمان پاسخ وزندار[25] : با احتساب اهمیت وظیفهٔ تکمیل‌شده برای سیستم و نیز تاخیر میان زمان انتساب وظیفه تا تکمیل آن، شناختی از نحوهٔ عملکرد طراحی‌های مختلف در مقایسه با یکدیگر به‌دست می آوریم:

$$AWRT = \frac{1}{AgentsNo}\sum_{j=1}^{AgentsNo}\frac{1}{n}\sum_{i=1}^{n}C_iT_i$$

که در این فرمول $AgentsNo$ تعداد عاملهای موجود در سیستم، n تعداد وظیفه‌های انجام شده، C_i ضریب بحران وظیفهٔ تکمیل شده و T_i نشاندهندهٔ میزان تاخیر(زمان سپری شده) از لحظهٔ محول شدن وظیفه به یک عامل تا لحظهٔ تکمیل آن است.

قابلیت کنارآمدن با خطاهای آینده[26] : در نظرگرفتن تعداد عاملهایی که کاری به آنها تخصیص نیافته است و در ضمن از مدت زمان کافی برخوردار هستند به همراه ضریب قابلیت اطمینان آنها، می‌تواند معیاری برای ارزیابی کیفیت تقسیم وظیفهٔ سیستم از لحاظ کمک‌رسانی در آینده باشد. با توجه به اینکه همهٔ عاملهای ما در سیستم، از توانایی‌های یکسانی برخوردار نیستند و برای تعدادی از آنها ویژگی برخورداری از قابلیت ویژه در نظر گرفته شده است،عاملهای کمک‌رسان معمولی[27] و عاملهای خاص[28] در ارزیابی این مساله از هم جدا شده اند.

$$\text{IFFC-Ordinary} = \frac{1}{AgentsNo}\sum_{i=1}^{AgentsNo}Fo_i.R_i \qquad \text{IFFC-Special} = \frac{1}{AgentsNo}\sum_{i=1}^{AgentsNo}Fs_i.R_i$$

که در این دو فرمول Fo_i و Fs_i به ترتیب تعداد دفعاتی است که عاملهای معمولی یا خاص با ضریب قابلیت اطمینان R_i با داشتن وقت کافی آزاد مانده‌اند. در [۱۷]چند معیار ارزیابی کارایی دیگر هم معرفی شده اند.

۷. نتایج شبیه‌سازی

در این پروژه به دلیل اهمیتی که برای ایجاد تحمل پذیری خطا به عنوان تاکید اصلی وجود داشته است، پس از کارایی به مسالهٔ تحمل پذیری خطا توجه داریم. بدیهی است که برای انتقال ایده های این تحقیق به سیستمهایی که نیازمند پاسخ زمانی کوتاه هستند، قطعا یا ترتیب دیگری از روشها حاصل می گردد یا در روشهای فعلی باید تغییراتی صورت بگیرد . خلاصه نتایج شبیه سازی به صورت کیفی درجدول۱ نشان داده شده است. برای اینکه معنای برچسبهای زبانی به کار رفته در جدول۱ مشخص‌تر گردد، لازم به ذکر است که برچسبهای هر ستون صرفا بیانگر یک مقایسه نسبی بوده و معنای مطلق آنها با محاسبات دقیق عددی در[۱۷]آورده شده است و به دلیل حجم زیاد آنها از ارائهٔ مقایسهٔ کمی روشها در اینجا خودداری شده است. این نتایج حاکی از آن است که در صورت قطعی بودن شرایط مساله نیازی به تحمل پیچیدگی های زیاد الگوریتمهای تقسیم وظیفه نداریم و هر یک از روشهای ساده تر قطعی پاسخگوی شرایط مساله خواهند بود. چنانچه لازم باشد که با شرایط غیرقطعی موجود در سیستم کنار بیاییم و خطاهای احتمالی هم دارای الگوی عادی و رایجی نباشند، بهترین روش از لحاظ کارایی و زمان پاسخ روش پنجم است، چرا که با برسی اضطراب عامل در زمان رسیدن درخواست کمک، منطقی ترین تصمیم را اتخاذ می کند و در عین حال به تحمل پذیری خطا نیز توجه کافی نموده‌است.

جدول ۱. مقایسهٔ کیفی روشهای پیشنهادی

		پیچیدگی	خطاهای عادی		خطاهای ویژه و خاص		
			کارایی	AWRT	کارایی	AWRT	IFFC
شرایط قطعی	S.T.D.	کم	قابل قبول	مناسب	نامناسب	زیاد	نامناسب
	H.H.C.C.	متوسط	قابل قبول	مناسب	نامناسب	زیاد	متوسط
	S.O.T.D.	زیاد	خوب	مناسب	خوب	مناسب	خوب
شرایط غیرقطعی با کمی نایقینی	S.O.T.D.	زیاد	خوب	مناسب	خوب	مناسب	خوب
	N.D.D.M.	زیاد	خوب	مناسب	خوب	مناسب	خوب
	N.T.S.	زیاد	خوب	مناسب	خوب	مناسب	خوب
شرایط غیرقطعی با نایقینی زیاد	S.O.T.D.	زیاد	نامناسب	نامناسب	نامناسب	نامناسب	ناکافی
	N.D.D.M.	زیاد	قابل قبول	مناسب	قابل قبول	نامناسب	خوب
	N.T.S.	زیاد	خوب	مناسب	خوب	مناسب	خوب

25 Average Weighted Response Time
26 Immediate Future Fault Compensation(IFFC)
27 Ordinary Helper Agents
28 Special Helper Agents