بهینه‌سازی خودکار کارایی نرم‌افزار مبتنی بر مدل با استفاده از MOPSO

مریم آموزگار[۱] و مهدی افتخاری[۲]

۱- دانشکده مهندسی کامپیوتر- دانشگاه علم و صنعت- تهران- ایران

amoozegar@kgut.ac.ir

۲- استادیار، دانشکده مهندسی کامپیوتر- دانشگاه شهید باهنر - کرمان- ایران

m.eftekhari@uk.ac.ir

چکیده: مهندسی کارایی نرم‌افزار در فازهای اولیه تولید نرم‌افزار (مدل سازی)، کارایی و صرفه‌جویی در هزینه‌ها را به‌همراه دارد که هنوز به‌طور کامل خودکار نشده است. این مقاله یک روش بهینه‌سازی مبتنی بر الگوریتم چندهدفه پرندگان را برای جستجوی خودکار فضای طراحی ارائه می‌کند تا مقادیر بهینه تنظیمات سیستم را برای دستیابی به کارایی بیشتر، در اختیار قرار دهد. بدین منظور، مدل نرم‌افزار به مدل کارایی که مبتنی بر شبکه‌های صف لایه‌ای است، تبدیل می‌شود. سپس مدل کارایی بهینه و بازخورد لازم در مدل نرم‌افزار منعکس می‌شود. یک مورد مطالعه برای ارزیابی روش پیشنهادی آورده شده است که نتایج به دست آمده کارایی روش پیشنهادی را نشان می‌دهد.

واژه های کلیدی: بهینه‌سازی جمعیت پرندگان، بهینه‌سازی چند هدفه، مهندسی کارایی نرم‌افزار، MOPSO

۱- مقدمه

مهندسی کارایی نرم‌افزار[۱]، [۱] فرآیند ارزیابی کارایی نرم‌افزار در مرحله طراحی مدل نرم‌افزار است که علاوه بر کاهش هزینه‌های تولید، افزایش کیفیت و کارایی را به‌همراه دارد. تحقیقات انجام شده در حوزه نرم‌افزارهای مبتنی بر قطعه[۲] [۲] خودکارسازی این فرآیند را بسیار مهم می‌شمارد زیرا طراح نرم‌افزار می‌تواند براحتی و بدون نیاز به تسلط در حوزه کارایی، مدلی با کیفیت بالا طراحی کند. در این مقاله فرآیند ارزیابی به دو زیرفرآیند عمده تفکیک شده است: تبدیل مدل و بازخورد. تبدیل مدل مرحله‌ای است که در آن مدل نرم‌افزار به مدل کارایی تبدیل می‌شود، در حالی که بازخورد مربوط به انعکاس پارامترها و یا مشخصه‌های بهینه کارایی از مدل کارایی به مدل نرم‌افزار است. در سال‌های اخیر تحقیقات زیادی در حوزه تبدل مدل انجام شده است اما بهینه‌سازی خودکار و یا به عبارتی، ارائه مقادیر بهینه تنظیمات و مشخصه‌های کارایی در مدل نرم‌افزار به تحقیقات بیشتری نیاز دارد.

استفاده از الگوریتم‌های بهینه‌سازی در مهندسی نرم‌افزار در [۳] مرور شده است و بر مبنای همین الگوریتم‌ها برای مساله ارزیابی کارایی نرم‌افزار روش‌هایی در [۶-۴] رائه شده است.

در برخی تحقیقات [۴] تنها زمان پاسخ برای ارزیابی کارایی در نظر گرفته شده است، در حالی که پارامترهای مهمی همچون بهره‌وری و توان عملیاتی منابع نرم‌افزاری و سخت‌افزاری نیز به عنوان مشخصه‌های عمده کارایی محسوب می‌شوند. علاوه بر موارد ذکر شده، کنترل هزینه‌های توسعه و یا به عبارتی کاهش آن نیز از اهمیت زیادی برخوردار است. بنابراین، مساله مطرح شده، یک مساله بهینه‌سازی چندهدفه است. از این رو، این مقاله روشی را با استفاده از الگوریتم جمعیت پرندگان چند هدفه

[۱] تاریخ ارسال مقاله : ۱۳۹۰/۴/۱۴

تاریخ پذیرش مقاله : ۱۳۹۱/۲/۶

نام نویسنده مسؤول : مریم آموزگار

نشانی نویسنده مسؤول : ایران- کرمان- انتهای جاده هفت باغ- دانشگاه تحصیلات تکمیلی.

MOPSO [7] ارائه می‌کند که مقادیر بهینه مشخصه‌های کارایی توصیف شده در مدل نرم‌افزار را یافته، به طراح نرم‌افزار پیشنهاد می‌دهد.

ازجمله برتری‌های الگوریتم جمعیت پرندگان در مقایسه با الگوریتم‌های ژنتیک که به انتخاب آن برای این مساله منجر شد، این است که پاسخ بهینه، مستقل از مقادیر اولیه و پس از شرکت کردن کلیه ذرات در الگوریتم به دست می‌آید و علاوه بر آن، وابستگی و حساسیت کمتر الگوریتم به تابع هدف، پیاده‌سازی راحت و تنظیم ساده پارامترها از مزیت‌های دیگر این الگوریتم هستند[۸].

در ادامه، ضمن بررسی پیشینه تحقیقاتی موضوع، مراحل ارزیابی کارایی تشریح می‌شوند. در بخش سوم با معرفی مسائل بهینه‌سازی چند هدفه به طرح رسمی مساله پرداخته، الگوریتم جمعیت پرندگان و نسخه چند هدفه، آن معرفی می‌کنیم. در انتها روش پیشنهادی بر روی یک سیستم نرم‌افزاری نمونه اعمال و نتایج بررسی و تحلیل می‌شود.

۲- کارهای انجام شده

مدل کارایی در روش‌ها و ابزارهای کلاسیک ارائه شده در ارزیابی کارایی [۲] عمدتا بر شبکه‌های صف لایه‌ای[۳] و شبکه‌های پتری مبتنی هستند که اکثر این روش‌ها پارامترهای کارایی را اندازه‌گیری کرده، اما امکان اعمال هیچ گونه بازخوردی به مدل نرم‌افزار را فراهم نمی‌کنند.

در [۲] روش نیمه‌خودکاری مبتنی بر قانون برای یافتن مقادیر بهینه مشخصه‌های کارایی ارائه شده است. دانش لازم در قالب قوانین در اختیار سیستم قرار گرفته، کاوش به طور خودکار آغاز می‌شود و تا زمان رفع گلوگاه‌ها و نیازهای کارایی و برقراری ثبات در سیستم ادامه پیدا می‌کند. این تحقیق به رفتار خاص نرم‌افزارهای مبتنی بر قطعه نمی‌پردازد، بلکه بر افزایش سرعت منابع سخت‌افزاری برای رفع گلوگاه‌ها تمرکز کرده است. همچنین، دامنه عملکرد آن وابسته و محدود به دانشی است که در اختیار سیستم قرار می‌گیرد.

در [۵] روش خودکاری برای بهبود کارایی مدل نرم‌افزارهای مبتنی بر قطعه ارائه شده است. فضای طراحی شامل گزینه‌های مختلفی، مثل انتخاب قطعات مناسب و تنظیمات مشخصه‌های کارایی است. کاوش فضای طراحی به وسیله الگوریتم ژنتیک انجام می‌شود. کاندیدهای اولیه ایجاد شده، پس از کاوش، نتایج در اختیار طراح نرم‌افزار قرار می‌گیرد. بدین ترتیب، طراح نرم‌افزار براحتی مدلی با کیفیت بالا تولید می‌کند. با این روش خلا بین استفاده از روش‌های کلاسیک و بهبود خودکار مدل نرم‌افزار پر می‌شود، اما نقطه ضعف این روش این است که مدلی را برای هزینه به عنوان یکی از توابع هدف درنظر نگرفته است.

در [۶] روش دیگری برای بهبود کارایی و قابلیت اعتماد نرم‌افزارهای سرویس‌گرا ارائه شده است. اساس کار پیدا کردن بهینه پرتو در الگوریتم‌های تکاملی چند هدفه است. فضای طراحی شامل اختصاص قطعه مناسب، انتخاب از بین قطعات همسان و سرعت منابع سخت‌افزاری است. زبان مدل‌سازی در این روش [۹] PCM[۴] است که یک متا-مدل اختصاصی برای ارزیابی کارایی و قابلیت اعتماد نرم‌افزارهای سرویس‌گراست. از این رو، طراح نرم‌افزار باید مدل نرم‌افزار را که با استانداردهایی، همچون [۱۰] UML مدل کرده به PCM تبدیل کند که این امر خود عدم یکپارچگی فرآیند ارزیابی کارایی در فرآیند تولید نرم‌افزار را به‌همراه دارد. نکته درخور توجه دیگر این است که به علت پرداختن به قابلیت اعتماد به عنوان تابع هدف برای کارایی تنها پارامتر زمان پاسخ در نظر گرفته شده است.

با توجه به مطالب فوق، این تحقیق با تمرکز بر تسهیل فرآیند ارزیابی کارایی و یکپارچگی آن در فرآیند تولید نرم‌افزار بهبودها و ملاحظات زیر را در نظر گرفته است:

۱. مدل نرم‌افزار بر مبنای UML است که استاندارد است.

۲. در این تحقیق مشخصه‌های کارایی بیشتری مورد توجه قرار گرفته‌اند. بهره‌وری منابع سخت‌افزاری و نرم‌افزاری که هدف، بیشینه شدن آنهاست، در کنار زمان پاسخ و هزینه که باید کمینه شوند، مدنظر گرفته شده‌اند و مساله را به یک مساله سه هدفه تبدیل کرده است که اهداف آن در تقابل با یکدیگر هستند.

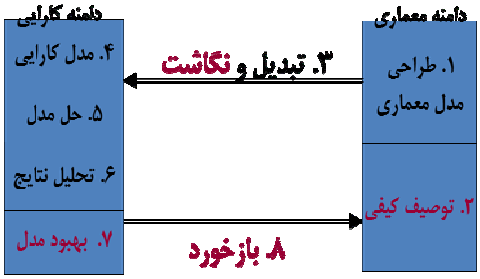
۳. رفتار و عملکرد MOPSO در مساله ارزیابی کارایی بررسی شده است.

۳- مهندسی کارایی نرم‌افزار

فرآیند ارزیابی کارایی مبتنی بر مدل که به آن مهندسی کارایی نرم‌افزار (SPE) گفته می‌شود و نخستین بار در [۱] ارائه شده، یک فرآیند سیستماتیک برای تولید نرم‌افزاری است که نیازهای کارایی آن تامین شده باشد.

با استناد به این فرآیند مرجع، در ادامه مراحل ارزیابی تا خودکارسازی و بهینه‌سازی تشریح می‌شوند. همان طور که شکل (۱) مراحل را نشان می‌دهد، در نخستین مرحله مدل نرم‌افزار با UML2 و با تکیه بر نمودارهای قطعه، فعالیت و استقرار که برای ارزیابی کارایی وجودشان ضروری است طراحی می‌شود. توصیف نیازهای کارایی و همچنین مشخصه‌های کارایی منابع سخت‌افزاری و نرم‌افزاری با استفاده از پروفایل کارایی MARTE [۱۱]، انجام می‌شود. این پروفایل دارای کلیشه‌ها[5] و برچسب‌هایی است که براحتی نمودارها را حاشیه‌نویسی می‌کند. با توجه به رفتار خاص نرم‌افزارهای مبتنی بر قطعه در [۱۲] برچسب‌هایی به منظور توصیف دقیقتر آنها اضافه شده است. در مراحل ۳ و ۴ تبدیل مدل نرم‌افزار به مدل کارایی انجام می‌شود. در این تحقیق، مدل کارایی [۱۳- ۱۴] CBML است که نسخه توسعه‌یافته‌ای از LQN برای مدل سازی کارایی نرم‌افزارهای مبتنی بر قطعه است و ابزار تبدیل طبق [۱۲] پیاده‌سازی و استفاده می‌شود. مدل کارایی در مرحله ۵ به وسیله ابزار LQNS[15] حل می‌شود. در مرحله ۶ و به منظور حل مدل به صورت پارمتریک مدل کارایی طبق قالب ابزار [۱۶] SPEX بازنویسی می‌شود.

روش‌های زیادی [۲] ارائه شده است که مشابه مراحل فوق را انجام می‌دهند، اما آنچه کمتر مورد توجه و بررسی قرار گرفته و عامل اساسی در فرآیند ارزیابی است، خودکارسازی آن است؛ بدین معنی که مقادیر مختلف مشخصه‌های کارایی در مدل قرار گرفته، ارزیابی می‌شوند تا زمانی که بهترین مقادیر به دست آید و براحتی در اختیار طراح قرار گیرند. آنچه در این تحقیق انجام شده، در قالب مراحل ۷ و ۸ و استفاده از الگوریتم MOPSO است که در بخش‌های بعدی به تفصیل به آن پرداخته شده است.



شکل (۱): فرآیند مهندسی کارایی نرم‌افزار

۴- بهینه‌سازی چندهدفه

مسائل بهینه‌سازی چندهدفه [7] بیش از یک تابع هدف دارند. اگر $\vec{x} = [x_1, x_2, \dots, x_n]$ آرایه متغیرهای ورودی یا به عبارتی بردار تصمیم باشد و $f_i: R^n \to R, i = 1, \dots, k$، در این صورت k هدف و تابع شایستگی تعریف می‌شود و هدف کمینه کردن $\vec{f}(\vec{x}) := [f_1(\vec{x}), f_2(\vec{x}), \dots, f_k(\vec{x})]$ است. البته، شرایط و قیدهای زیر را نیز می‌توان در نظر گرفت:

$$g_i, h_j : R^n \to R, i = 1,\dots, m, j = 1,\dots, p$$
$$g_i(\vec{x}) \le 0, h_j(\vec{x}) = 0$$

همان طور که مشخص است، مسائل چندهدفه مجموعه‌ای از جواب‌ها را تولید می‌کنند که نسبت به هم برتری ندارند و این مجموعه را بهینه پرتو می‌نامند. تعاریف به‌طور کامل در [۱۷] آمده و در اینجا تنها به مفهوم مغلوب شدن اشاره می‌شود (در اینجا هدف کمینه‌سازی اهداف است).

تعریف ۱: برای دو بردار $\vec{x}, \vec{y} \epsilon R^k$ گفته می‌شود $\vec{x}$ بردار $\vec{y}$ را مغلوب می‌کند($\vec{x} \prec \vec{y}$)، در صورتی که اگر رابطه $x_i \le y_i$ برای $i = 1, \dots, k$ برقرار باشد، آنگاه $f(\vec{x}) \le f(\vec{y})$. به عبارت دیگر، در هیچ بعدی $\vec{x}$ بدتر از $\vec{y}$ نباشد. بدین ترتیب، بردار تصمیم x متعلق به مجموعه

مجاز یک پاسخ بهینه پرتو است، اگر بردار تصمیم دیگری مانند y متعلق به مجموعه مجاز F وجود نداشته باشد که بر آن غلبه کند.

۵- طرح مساله به صورت رسمی

نتایج حاصل از حل مدل کارایی، توان عملیاتی و بهره‌وری هر یک از منابع نرم‌افزاری و سخت‌افزاری به‌همراه زمان پاسخ رویدادهاست که با انتساب به برچسب‌های پروفایل کارایی به عنوان مقادیر صفات کیفی به مدل نرم‌افزار اعمال می‌شوند. رسیدن به وضعیتی که در آن کارایی سیستم مطلوب باشد، به تکرار مداوم فرآیند ارزیابی نیاز دارد. اگر تغییرات ساختاری در نظر گرفته نشود، سؤال اساسی این است که اعمال تغییر بر کدام صفت کیفی، کدام عنصر از مدل نرم‌افزار و به چه میزان، کارایی بهتری را به‌همراه دارد. بدین ترتیب، اجرای این فرآیند بر اساس سعی و خطا، خصوصاً برای نرم‌افزارهای بزرگ بسیار زمانبر خواهد بود. لذا می‌توان این مساله را در قالب یک مساله بهینه‌سازی مطرح کرد و از آنجا که کارایی تنها شامل یک هدف (زمان پاسخ کمینه) نمی‌شود و بهره‌وری و توان عملیاتی منابع نیز از اهمیت بسزایی برخوردارند، مساله از نوع مسائل بهینه‌سازی چندهدفه است.

لازم است ابتدا مساله به طور کاملا شفاف و رسمی مطرح شود. همان طور که قبلا گفته شد، مشخصه‌های کارایی با استفاده از پروفایل کارایی در مدل نرم‌افزار حاشیه‌نویسی شده‌اند. این مشخصه‌ها معادل متغیرهای ورودی مساله و به قرار زیر هستند:

۱. سرعت پردازنده‌های سرور که با برچسب PArate از کلیشه PAhost در نمودار استقرار مقداردهی می‌شود.

۲. تعداد ریسمان‌های هر شئ فعال که درمدل نرم‌افزار با برچسبPAmultiplicity از کلیشه PAstep مشخص می‌شود.

۳. تعداد پردازنده‌های هر سرور

بنابراین، بردار تصمیم به ترتیب شامل موارد فوق و به صورت (۱) خواهد بود. n و m به تعداد پردازنده و تعداد اشیا اشاره می‌کند.

$$X=[cpu\_spped_i, thread\_multiplicity_j, cpu\_multiplicity_i] \quad (۱)$$
$$i=1,..n, j=1,...,m$$

هر متغیر در بردار X به ترتیب به بندهای ۱ تا ۳ اشاره دارد. اهداف مساله به صورت زیر هستند:

۱. زمان پاسخ باید کمینه شود و خروجی ابزار LQNS است.

۲. هزینه برای کنترل و اعمال محدودیت در استفاده از منابع در نظر گرفته شده است. از آنجا که مسلم است، هر چه تعداد پردازنده‌ها بیشتر باشد، زمان پاسخ کمتر خواهد شد، اما برای کنترل هزینه اعمال قید ضروری است. تابع هزینه که در واقع با سرعت و تعداد پردازنده‌ها متناسب است از رابطه (۲) به دست می‌آید و باید کمینه شود.

$$TCost=\sum_{k=1}^{s} H * cpu\_speed_k * cpu\_multiplicity_k \quad (۲)$$

cpu_speed و *cpu_multiplicity* به ترتیب سرعت و تعداد پردازنده‌ها روی سرور kام، s تعداد سرورها و H ضریب تاثیر سرعت روی هزینه را نشان می‌دهند.

۳ . بهره‌وری منابع که باید بیشینه باشد.

تحلیل دقیق موارد فوق تقابل و تضاد اهداف را نشان می‌دهد، زیرا برای دو هدف اول مقدار کمینه و برای هدف آخر مقدار بیشینه مدنظر است.

۶- الگوریتم جمعیت پرندگان

بهینه‌سازی جمعیت پرندگان یک روش جستجوی ابتکاری است (البته، طبق نظر مؤلف[۱۸]) که پرواز پرندگان برای یافتن غذا را شبیه‌سازی می‌کند و خصوصا در سال‌های اخیر بسیار استفاده شده است. این روش در مقایسه با روش‌های دیگر مزیت‌هایی دارد، از جمله اینکه پاسخ بهینه مستقل از مقادیر اولیه و پس از شرکت کردن کلیه ذرات در الگوریتم به دست می‌آید و علاوه بر آن وابستگی و حساسیت کمتر الگوریتم به تابع هدف، پیاده‌سازی راحت و تنظیم ساده آن مزیت‌های مهمی هستند[۸].

در PSO هر جواب پیشنهادی که یک ذره نامیده می‌شود، یک نقطه در فضای جستجوست. هر ذره، در فضای جستجوی چند بعدی پرواز می‌کند و موقعیت خود را بر حسب تجربه خود و همسایگانش تغییر می‌دهد. کارایی هر ذره (میزان نزدیکی هر ذره به بهینه سراسری) بر اساس تابع شایستگی تعیین شده اندازه‌گیری می‌شود. این تابع متناسب با مساله تعریف می‌شود[۱۹].

فرض کنید فضای جستجو D بعد داشته باشد و m ذره در جمعیت وجود داشته باشند. هر ذره در موقعیت $X_i=[x_{i1},x_{i2},\ldots,x_{iD}]$ قرار گرفته، با سرعت $V_i=[v_{i1}, v_{i2}, \ldots, v_{iD}]$ حرکت می‌کند. هر ذره به سمت بهترین نقطه‌ای که تا آن لحظه تجربه کرده؛ یعنی $Pbest_i=[pbest_{i1}, pbest_{i2},\ldots, pbest_{in}]$ حرکت می‌کند. بهترین موقعیت در کل جمعیت نیز با $Gbest=[gbest_1, gbest_2, \ldots, gbest_n]$ نمایش داده می‌شود. هر ذره موقعیت خود را برحسب سرعت مشخصی در هر تکرار تغییر می‌دهد که این سرعت به طور تصادفی و بر مبنای تمایل حرکت ذره به سمت pbest و gbest تنظیم می‌شود. برای هر ذره r در بعد s سرعت جدید؛ یعنی v_{rs} و موقعیت جدید x_{rs} بر اساس رابطه (۳) و (۴) به دست می‌آیند.

$$v_{rs}^{t} = wv_{rs}^{t-1} + c_1 r_1\left(pbest_{rs}^{t-1} - x_{rs}^{t-1}\right) + c_2 r_2\left(gbest_{rs}^{t-1} - x_{rs}^{t-1}\right) \quad (۳)$$

$$x_{rs}^{t} = x_{rs}^{t-1} + v_{rs}^{t-1} \quad (۴)$$

در روابط فوق t شماره تکرار و w وزن اینرسی است که برای کنترل سرعت و برقراری تعادل بین کاوش و بهره‌گیری تعریف شده است. مقدار بزرگ برای w سرعت ذرات را زیاد و از افتادن آنها در بهینه محلی جلوگیری می‌کند. مقدار کوچک برای آن باعث کاهش سرعت و تشویق ذره به بهره‌گیری از همان ناحیه‌ای که در آن واقع است، می‌شود. ثابت‌های c_1 و c_2 ضریب شتاب هستند که میزان نزدیکی ذره به pbest و gbest را تعیین می‌کنند. r_1 و r_2 اعداد تصادفی مستقلی هستند که مقدار بین صفر و یک دارند. اتمام PSO می‌تواند بر اساس تعداد مشخصی تکرار یا رسیدن به مقدار تعیین شده‌ای از pbest باشد و یا زمانی که هیچ بهبود و تغییری در آن حاصل نشد، به پایان برسد.

مراحل الگوریتم PSO استاندارد به صورت زیر است[۲۰]:

۱. مقداردهی اولیه ذرات در موقعیت‌های تصادفی با سرعت اولیه، روی D بعد فضای جستجو؛

۲. بروزرسانی سرعت و موقعیت هر ذره طبق روابط (۳) و (۴)؛

۴. محاسبه تابع شایستگی و بروزرسانی همزمان pbest و gbest در صورت نیاز؛

۵. برگشت به مرحله ۲ تا برقراری شرط اتمام.

۷- الگوریتم جمعیت پرندگان چندهدفه

نخستین نسخه چندهدفه الگوریتم جمعیت پرندگان در سال ۱۹۹۹ در [۲۱] ارائه شد. الگوریتم چندهدفه مورد استفاده در این تحقیق MOPSO[7] است که مراحل آن در شکل (۲) نشان داده شده است. این الگوریتم از عملگری به نام عملگر جهش واحد [۲۲] استفاده می‌کند که یک عضو از جمعیت را انتخاب کرده، مقدار یک بعد آن را به عددی در محدوده مقادیر معتبر تغییر می‌دهد. همچنین، یک سیاست نخبه‌گرایی به منظور نگه داشتن نتایج برتر و غالب در تکرارهای الگوریتم تعریف شده است. پاسخ‌های غالب در آرشیو خارجی که ساختار گرید [۲۳] دارد، ذخیره می‌شوند. دستاورد مدیریت صحیح آرشیو، تنوع جمعیت و توزیع مناسب پاسخ‌هاست.

انتخاب pbest و gbest طبق مکانیزم خاصی انجام می‌شوند. الگوریتم چندهدفه نمی‌تواند از رابطه (۳) برای شناسایی آنها استفاده کند، زیرا همه پاسخ‌های غالب به یک میزان برتری دارند. تنها زمانی pbest بروز می‌شود که ذره جدیدی مقدار قبلی آن را مغلوب کند. gbest نیز در هر تکرار از بین پاسخ‌های غالب موجود در آرشیو انتخاب می‌شود.

```
1.  SMOPSO{
2.  Init Pop();
3.  Init Velocity();
4.  EvaluatePop();
5.  Update Fbest();
6.  Update Pbest();
7.  Insert nodom();
8.  Gbestpos = rnd(0,nodomfileSize)
9.  for(i=1 to MAXCYCLES){
10.     for(j=0to MAXPARTICLES){
11.        Update Velocity();
12.        Update Particle();  }
13.     Keeping();
14.     Evaluate Pop();
15.     Update Fbest();
16.     Update Pbest();
17.     Insert nodom();
18.     Gbestpos = rnd(0,nodomfileSize)  {
19.  Print Statistics ();
```

شکل (۲): الگوریتم MOPSO [7]

۸- نمونه مورد مطالعه

برای بررسی راهکار بهینه‌سازی ارائه شده سیستم نرم‌افزاری فروشگاه الکترونیکی که امکان مدیریت محصولات، خرید و پرداخت الکترونیکی را فراهم می‌کند، در نظر گرفته شد.

ابتدا سیستم نرم‌افزاری با نرم‌افزار Enterprise Architect و بر مبنای UML2 مدل شد. نمودارهای قطعه، فعالیت و استقرار این نرم‌افزار در شکل‌های ۳ تا ۵ آمده است. سپس نمودارهای مذکور با استفاده از پروفایل MARTE برای توصیف کارایی مدل حاشیه‌نویسی شده‌اند که به منظور وضوح بیشتر این مشخصه‌ها در خارج از نمودارها و در جداول (۱) تا (۳)آمده‌اند.

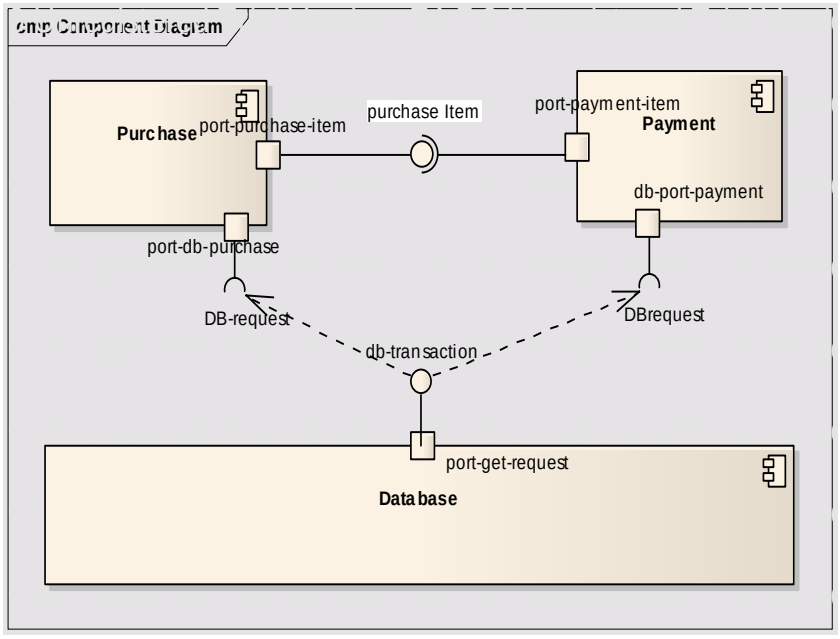
پس از اینکه مدل کارایی به وسیله ابزار تولید شده به دست آمد، به منظور مشاهده نتایج اولیه با ابزار LQNS حل می‌شود.

جدول(۱): مشخصه‌های کارایی نمودار قطعه

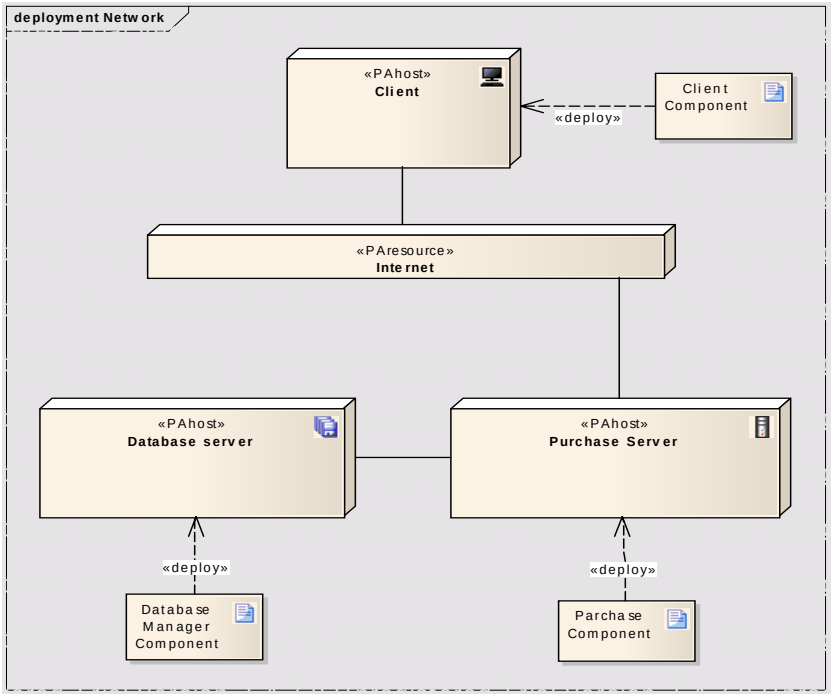
عنوان برچسب	عنصر	مقدار برچسب در قطعه خرید		مقدار برچسب در قطعه پایگاه داده
PAinstances	Component	1		1
PAcontainersched	Component	Fcfs		Fcfs
		port-db-purchase	port-purchase-item	port-get-request
PAmapping	Port	0/5	0/5	0/5

جدول (۲): مشخصه‌های کارایی نمودار استقرار

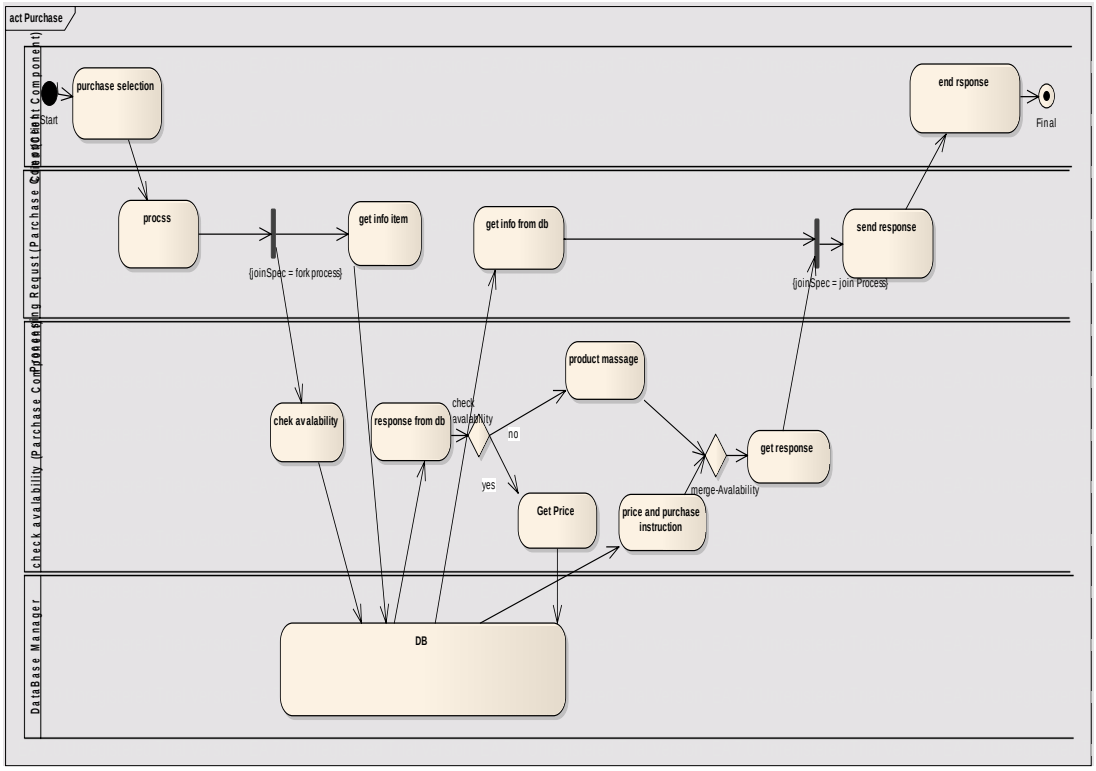
عنوان برچسب	Client	Purchase server	Database server
PAmultiplicity	1	1	1
PAschdPolicy	Ref	Fcfs	Fcfs
PArate	1	1	1



شکل (۲): نمودار قطعه



شکل (۳): نمودار استقرار



شکل (۴): نمودار فعالیت

جدول (۳) : حاشیه‌نویسی نمودار فعالیت

	قطعه خرید										قطعه پایگاه داده				
عنوان برچسب	Processing Request				check availability						Database Manager				
PAmultiplicity	1				1						1				
PApriority	0				0						0				
	Send response	Get info from db	Get info item	Process	get response	price and purchase instruction	Get Price	product massage	response from db	Check availability	Send Response	Update Query	Select Query	Insert Query	Process Query
PAdemand	1	1	1	1	1	1	1	1	1	1	1	3	2	1	1

۹- شبیه‌سازی و تحلیل نتایج

در نخستین مرحله از فاز بهینه‌سازی مدل کارایی طبق قالب ابزار SPEX بازنویسی می‌شود. سپس الگوریتم MOPSO پیاده‌سازی شده، این ابزار در پس زمینه برنامه بهینه‌سازی قرار می‌گیرد تا بر مبنای مقادیر بردار تصمیم، مقادیر توابع هدف را مشخص کند.

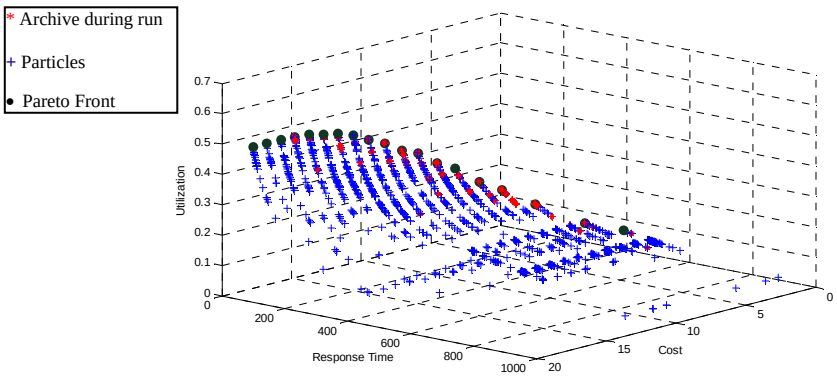
هر ذره یک بردار ۵ بعدی به صورت [تعداد پردازنده‌های سرور Purchase ، تعداد پردازنده‌های سرور database، تعداد ریسمان‌های وظیفه Processing-Request، تعداد ریسمان‌های وظیفه check-availability تعداد ریسمان‌های وظیفه Database-manager] است که مقدار هر بعد عددی بین ۱ تا ۱۱ است و توابع هدف نیز طبق بخش ۵ تعریف می‌شوند.

پارامترهای الگوریتم، طبق جدول (۴) تنظیم می‌شوند.

در آزمایش اول اندازه آرشیو ۱۹ انتخاب شده است که بهینه پرتو بزرگتر و در نتیجه انتخاب‌های بیشتری را در اختیار طراح نرم‌افزار قرار می‌دهد. نمودار مربوطه در شکل (۶) نشان داده شده است. در آزمایش دوم اندازه آرشیو به ۱۰ کاهش داده شده است. بهینه پرتو تولید شده در جدول ۵ و بردارهای تصمیم معادل و یا مجموعه پرتو در جدول(۶) نمایش داده شده‌اند. در واقع، در این مرحله هدف بررسی و مشاهده رفتار و عملکرد الگوریتم در هنگام کنترل اندازه آرشیو است.

جدول۴: پارامترهای MOPSO

پارامتر	مقدار
تعداد تکرار	۱۰۰
اندازه جمعیت	۱۵
اندازه آرشیو	۱۹
اینرسی w	۰/۵
نرخ جهش	۰/۵
c_1	۱/۵
c_2	۲



شکل (۶): نمودار حاصل از اجرای MOPSO با ۳ تابع هدف

یکی از معیارهای ارزیابی الگوریتم‌های چندهدفه، توزیع پاسخ‌ها در طول آرشیو است؛ به این معنی که اگر پاسخ‌ها به جای ازدحام در یک ناحیه در سراسر طول آرشیو توزیع شوند، تنوع و حق انتخاب بیشتری را فراهم کرده، کارایی الگوریتم را بالاتر می‌برند. همان طور که بهینه پرتو جدول (۵)نشان می‌دهد، الگوریتم MOPSO توانسته است با توزیع مناسب، انتخاب‌های متنوعی را به طراح پیشنهاد دهد. بدین ترتیب، طراح نرم‌افزار دیگر دغدغه تنظیم و کارکردن همراه با سعی و خطا را بر روی پارامترها و مشخصه‌های کارایی ندارد و می‌تواند مقادیری را که کارایی بیشتر و مطلوبتری را به همراه داشته باشد، در مدل نرم‌افزار اعمال کند.

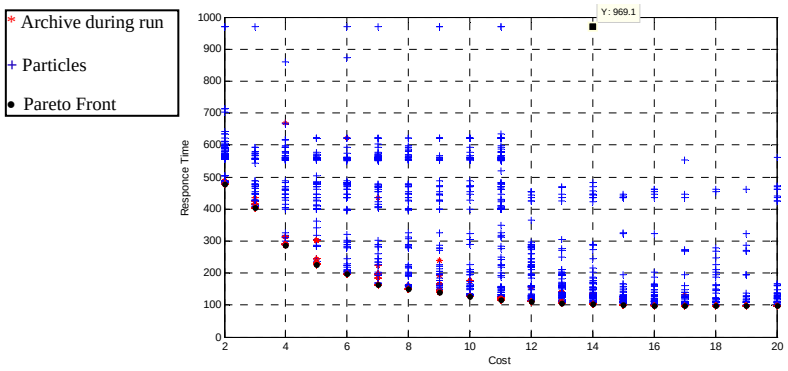
جدول (۵) : بهینه پرتو

$F_1(x)$	$F_2(x)$	$F_3(x)$
2	476/932	0/1049
3	399/475	0/1252
4	290/761	0/172
5	225/514	0/2217
6	194/115	0/2576
7	165/409	0/3023
10	129/296	0/3867
14	118/206	0/423
16	100/171	0/4991
19	97/8859	0/5108
20	97/5106	0/5128

جدول (۶) : مجموعه پرتو

x_1	x_2	x_3	x_4	x_5
1	1	9	9	1
1	2	10	6	5
2	2	10	9	6
2	3	10	8	7
3	3	10	10	10
3	4	10	8	8
4	6	10	8	10
5	9	9	8	9
7	9	10	9	9
9	10	10	10	9
10	10	10	10	10

در آزمایش سوم طول آرشیو ۱۹ و بـه منظـور مشـاهده ساده تر رفتار الگوریتم، توابع هدف به دو تابع زمان پاسخ و هزینه کاهش داده شد. شکل (۷) نمودار بـه دسـت آمـده را نشان می‌دهد.



شکل (۷): نمودار حاصل از اجرای MOPSO با دو تابع هدف

۱۰- نتیجه‌گیری

خودکارسازی فرآیند ارزیابی کارایی از اهمیـت ویـژه‌ای برخوردار است، زیرا طراح نرم‌افزار مـی‌توانـد بـه راحتـی و بدون نیاز به تسلط در حوزه کارایی مدلی با کیفیت و کارایی بالا تولید کند. در این مقاله، روشی برای بهینه‌سازی خودکار کارایی مدل نـرم‌افـزار ارائـه شـده اسـت کـه مقـادیر بهینـه تنظیمات و مشخصه‌های کارایی حاشیه‌نویسی شده در آن را پیشنهاد می‌کند. بهینه‌سازی مبتنی بر روش MOPSO است. به این ترتیب، ضمن اعمال این الگوریتم قدرتمند در حـوزه ارزیـابی کـارایی، رفتـار و عملکـرد آن تحلیـل و بررسـی و پاسخ‌های رضایت‌بخشی حاصل شد. از طرف دیگر، در این مقاله به جای استفاده از متا–مـدل‌هـای اختصاصـی کـارایی، همان مدل استاندارد مبتنـی بـر UML بـا پروفایـل کـارایی حاشیه‌نویسی و به مدل کارایی CBML تبدیل شد تا طـراح نرم‌افزار را از تسلط به حوزه کارایی بی‌نیاز کند.

مراجع

[1] Smith, C.U., “Performance Engineering of Software Systems”, Addison-Wesley, 1990.

[2] Koziolek, H. ‘’Performance evaluation of component-based software systems: A survey”, Performance Evaluation, Vol. 67 Issue. 8, August. 2010, pp. 634-658doi:10.1016/j.peva.2009.07.007.

[3] Harman, M., “The Current State and Future of Search Based Software Engineering”, Future of Software Engineering, 2007. FOSE ’07, pages 342-357, May 23-25 2007.

[4] Jing Xu., “Rule-based automatic software performance diagnosis and improvement”, In WOSP ’08: Proceedings of the 7th international work- shop on Software and performance, pages 1–12, New York, NY, USA, 2008.

[5] Martens, A. and Koziolek, H., “Automatic, model-based software performance improvement for component-based software designs”, In 6th International Workshop on Formal Engineering approaches to Software Components and Architectures (FESCA). Elsevier, 2009.

[6] Martens, A. and Koziolek, H., "Optimising Multiple Quality Criteria of Service-Oriented Software Architectures", 1st international workshop on Quality of service-oriented software systems,2009.

[7] Cagnina, L., Esquivel, S., Coello Coello, C.A., “A Particle Swarm Optimizer for Multi-Objective Optimization”, JCS&T, 4 (5), pp. 204-210, 2005.

[8] Marinakis, Y. and Marinaki, M. and Dounias, G., “A hybrid particle swarm optimization algorithm for the vehicle routing

problem", Engineering Applications of Artificial Intelligence, vol. 23 Issue 4, June. 2010, doi:10.1016/j.engappai.2010.02.002.

[9] Becker, S., Koziolek, H. and Reussner, R., "The palladio component model for model-driven performance prediction", J. of Systems and Software, 82:3{22, 2009.

[10]Object Management Group. Unified Modeling Language: Superstructure, Version 2.0, ptc/03-07-06, July 2003, http://www.omg.org/cgi-bin/doc?ptc/2003-08-02.

[11]Object Management Group (OMG). UML Profile for MARTE, Beta 1. http://www.omg.org/cgi- bin/doc?ptc/2007- 08-04 ,

[12]Amoozegar, M., "A Solution for Performance Evaluation of Component-Based Software Architecture", A Thesis Submitted in Partial Fulfilment of The Requirement for degree of Master of Science in Software Engineering.

[13]Wu, X., "An approach to predicting peformance for component based systems", Master's thesis, Department of Systems and Computer Engineering, Carleton University, Ottawa, Ontario, Canada, August 2003.

[14]Wu, X. and Woodside, C.M., "Performance Modeling from Software Components", Proceedings of the Fourth International Workshop on Software and Performance,WOSP 2004, Redwood Shores, California, USA, January 14-16, 2004, New York, NY, ACM Press (2004) 290–301.

[15]Greg, F., Maly, P., Woodside, M., Petriu D.C. and Hubbard, A. "Layered Queueing Network Solver and Simulator User Manual", Department of Systems and Computer Engineering Carleton University, 2005.

[16]Hubbard, A., "SPEX: Software Performance Experiment Driver", http://www.sce.carleton.ca/rads/lqn/lqn-documentation/spex.txt , August 1997.

[17]Coello Coello, C. and Lechuga, M.," Mopso: A proposal for multiple objective particle swarm optimization", In Congress on Evolutionary Computation, IEEE Service Center. pages 1051{ 1056, Piscataway, NJ., 2002.

[18]Russell C. Eberhart and Yuhui Shi., "Comparison between genetic algorithms and particle swarm optimization", Proceedings of the Seventh Annual Conference on Evolutionary Programming, pages 611– 619. Springer-Verlag, March 1998.

[19]Andries P.Engelbrecht, "computational intelligence," 2 ed, Wily,2007, pp. 9.

[20]Sha, D. Y. and Hsing Hung Lin ,"A particle swarm optimization for multi-objective flow-shop scheduling," The International Journal of Advanced Manufacturing Technology, vol. 45, Numbers 7-8, 2008, P. 749-758, doi:10.1007/s00170-009-1970-6

[21]Moore, J. and Chapman, R. "Application of particle swarm to multiobjective optimization", Department of Computer Science and Software Engineering, Auburn University, 1999.

[22]Back, T., Fogel, D. and Michalewicz, Z. , "Handandbook of Evolutionary Computation", IOP Publishing Ltd and Oxford University Press, 1997.

[23]Deb, K., "Multi-Objective Optimization using Evolutionary Algorithms", John Wiley & Sons, Ltd., England, 2001.

زیرنویس‌ها

1 SPE
2 Component
3 Layered Queueing Networks
4 Palladio Component model
5 Stereotype