حل مسئله تخصیص قابلیت اطمینان- مازاد با استفاده از الگوریتم تکاملی ترکیبی بر مبنای الگوریتم رقابت استعماری و الگوریتم ژنتیک

آزاده اندرخور،* حسین اقبالی**

تاریخ دریافت: ۱۳۹۹/۱۰/۱ **تاریخ پذیرش:** ۱۳۹۹/۱۱/۱۵ **نوع مقاله:** پژوهشی

چکیده

قابلیت اطمینان یک مشخصه مهم در سیستم‌های الکتریکی و مکانیکی است. یافتن سطح بهینه قابلیت اطمینان بر اساس محدودیت‌های موجود در سیستم، مسئله بهینه‌سازی قابلیت اطمینان نامیده می‌شود. یکی از روش‌های ارتقا قابلیت اطمینان سیستم، استفاده از اجزا مازاد در سیستم است. تخصیص قابلیت اطمینان مناسب به هر یک از اجزا و استفاده از اجزا مازاد به صورت موازی در کنار اجزای اصلی سیستم تحت عنوان تخصیص قابلیت اطمینان - مازاد شناخته می‌شود. اثبات شده است که مسئله تخصیص مازاد یک مسئله بهینه‌سازی چند جمله‌ای غیر قطعی است و با افزایش اندازه مسئله و محدودیت‌ها، محاسبات به صورت نمایی افزایش می‌یابد. از این رو، یافتن راه حل مناسب در این دسته مسائل اهمیت دارد. در این پژوهش، مسئله تخصیص افزونگی – مازاد، در ارتقای قابلیت اطمینان سیستم‌های سری، سری – موازی و پل مورد بررسی قرار گرفت و به منظور حل مسائل از ترکیب الگوریتم رقابت استعماری و ژنتیک استفاده شد. الگوریتم پیشنهادی در مقایسه با رویکردهای پیشین عملکرد بهتری را نشان داده است. در واقع، پژوهش حاضر الگوریتم و روش حل مناسبی را برای حل مسئله بهینه‌سازی قابلیت اطمینان ارائه کرده است.

واژگان کلیدی: بهبود قابلیت اطمینان، تخصیص افزونگی – مازاد، الگوریتم رقابت استعماری، الگوریتم ترکیبی.

---

*. دانش آموخته کارشناسی ارشد مهندسی صنایع، دانشگاه غیرانتفاعی ایوان‌کی

**. مربی، گروه مهندسی صنایع، دانشگاه غیرانتفاعی ایوان‌کی (مسئول مکاتبات) h.eghbali@eyc.ac.ir

مقدمه

در هر جامعه مدرن، مهندسان و مدیران فنی مسئول برنامه‌ریزی، طراحی، ساخت و بهره‌برداری از ساده‌ترین محصول تا پیچیده‌ترین سیستم‌ها هستند. از کار افتادن محصول‌ها و سیستم‌ها موجب وقوع اختلال در سطوح مختلفی می‌شود و می‌تواند حتی به عنوان تهدیدی شدید برای جامعه و محیط زیست نیز تلقی شود. به همین خاطر، مصرف‌کنندگان و به طور کلی جامعه انتظار دارند که محصول‌ها و سیستم‌ها، پایا، اطمینان‌بخش و ایمن باشند. باید به جای تمرکز بر کاهش هزینه‌ها بر افزایش پایایی متمرکز شد؛ زیرا با بهبود پایایی، به طور قطع، هزینه‌ها کاهش خواهد یافت (گارگ[1] و همکاران، ۲۰۱۴). تخصیص افزونگی[2] یک راه مستقیم برای افزایش قابلیت اطمینان سیستم است (ذوالفقاری و همکاران، ۲۰۱۵). افزونگی با اضافه کردن چند جز به جای یک جز و استفاده کردن از همه آنها سعی می‌کند احتمال درست کارکردن سیستم را بالا ببرد. بهینه‌سازی قابلیت اطمینان و تخصیص مازاد به طور همزمان یکی از راه‌های ارتقا قابلیت اطمینان سیستم است که تحت عنوان تخصیص قابلیت اطمینان - مازاد[3] شناخته می‌شود؛ چون ثابت کرد که مسائل بهینه‌سازی قابلیت اطمینان از نوع مسائل چند جمله‌ای غیر قطعی مشکل می‌باشند. یافتن جواب‌های بهینه عمومی در این نوع از مسائل، باتوجه به گستردگی و رشد نمایی فضای جواب‌ها و همچنین وجود بهینه‌های محلی متعدد بسیار مشکل بوده و با بالا رفتن تعداد زیرسیستم‌ها و افزایش حجم مسئله، حل این مسائل به کمک روش‌های غیر معمول عملاً غیر ممکن می‌شود. این روش‌ها نیازمند زمان‌های محاسباتی بسیار زیاد هستند و با بالا رفتن حجم مسئله عملاً در یافتن جواب‌های بهینه ناتوان می‌باشند (چرن[4] ، ۱۹۹۲).

در سالیان اخیر، عمده فعالیت‌های انجام‌شده در ارتباط با حل مسئله تخصیص اجزاء مازاد، شامل به کارگیری بسط و توسعه روش‌های ابتکاری و الگوریتم‌های فرا ابتکاری بوده است. در ابتدا به دلیل عدم دسترسی به روش‌های ابتکاری و فرا ابتکاری، بهینه‌سازی‌ها با استفاده از روش‌های دقیق ریاضی صورت می‌گرفت. یالاوی و همکاران از برنامه‌ریزی پویا برای حل مسئله تخصیص مازاد استفاده کرده‌اند. آنان یک سیستم سری - موازی را که به صورت برنامه‌ریزی خطی عدد صحیح مدل شده بود، به کمک رویکرد تجزیه به چندین زیرمسئله تبدیل کردند که هر زیرمسئله معادل با یک مسئله کوله‌پشتی تک بعدی بود؛ سپس هر کدام از این زیرمسئله‌ها را به کمک برنامه‌ریزی پویا حل کردند (یالائی و همکاران، ۲۰۰۵). یکی دیگر از روش‌های دقیق ، روش برنامه‌ریزی عدد صحیح

1. garg
2. redundancy allocation
3. reliability–redundancy allocation
4. chern

است. بیلیونیت برای حل یک مسئله قابلیت اطمینان که دارای مدلی با هدف افزایش قابلیت اطمینان است، از برنامه‌ریزی عدد صحیح استفاده کرد. این روش راه حل‌های تقریباً شدنی را به ما می‌دهد (بیلیونت[1]، ۲۰۰۸). ها و کائو روش شاخه و کران را ارائه دادند. در این مقاله، سیستم به صورت منسجم در نظر گرفته شده و مدل به صورت یک مسئله برنامه‌ریزی غیر خطی عدد صحیح است که برای حل آن از روش شاخه و کران استفاده شده است. نتایج عددی در این مقاله نشان داد که رویکرد شاخه و کران در بین روش‌های دقیق برای حل مسئله تخصیص مازاد برتری دارند (ها و کائو[2]، ۲۰۰۶). در تحقیقات یادشده مدل‌ها به گونه‌ای طراحی شدند که تنها استفاده از یک نوع از اجزاء مازاد در هر زیرسیستم مجاز در نظر گرفته شد که اصطلاحاً به این نوع مدل‌ها، مدل‌های تک‌انتخابه گفته می‌شود. در صورت مجاز دانستن بیش از یک نوع مازاد برای هر زیرسیستم و تبدیل مدل تک‌انتخابه به مدل چندانتخابه، فضای جواب‌های این مسئله غیر خطی بسیار وسیع شده است و دشواری حل آن نیز به میزان قابل توجهی افزایش می‌یابد. در این صورت، روش‌های دقیق ریاضی یادشده توانایی حل و بهینه‌سازی را نخواهند داشت. بنابراین، نقطه ضعف بزرگ روش‌های ریاضی در حل مسائل بزرگ با پارامترهای زیاد است که این معضل، توانایی بالای آنها را در دستیابی به جواب‌های دقیق تحت‌الشعاع قرار می‌دهد. با این نگرش، مارکز و همکاران راهکارهایی برای استفاده از روش‌های دقیق ریاضی ارائه کردند که استفاده از اجزاء متنوع در یک زیرسیستم را ممکن می‌ساختند. به عنوان مثال، رویکرد حداکثر - حداقل[3] را به عنوان یک جانشین مفید و کارامد برای مسائل بهینه‌سازی به حداکثر رساندن قابلیت اطمینان به کار بستند (مارکز[4] و همکاران، ۲۰۰۴). با این حال، استفاده از روش‌های ریاضی دقیق در عمل با مشکلاتی همراه است. به عنوان مثال، این روش‌ها در صورت بالارفتن تعداد زیرسیستم‌ها و با افزایش تنوع اجزاء و گسترش فضای جواب کارایی خود را ازدست می‌دهند و عملاً در حل مسائل و ساختارهای کوچک قابلیت دارند. از این رو، در سال‌های اخیر استفاده از روش‌های فرا ابتکاری نسبت به روش‌های دقیق ریاضی ارجحیت پیدا کرده‌اند. در همین راستا، لاین و دراگت با در نظر گرفتن سیستم‌هایی با خرابی ناکامل از الگوریتم ژنتیک برای حل آن استفاده کردند. در این مسئله، مدل به صورت چند هدفه است (لاین و دراگت[5]، ۲۰۱۱). همچنین ابویی اردکان و همدانی الگوریتم ژنتیک اصلاح‌شده را برای حل مسئله تخصیص قابلیت به کار بردند. در این مسئله، مدل تک هدفه است و به صورت برنامه‌ریزی عدد صحیح مختلط

1. Billionnet
2. Ha, C., & Kuo
3. Max-Min
4. Ramirez-Marquez
5. Lins, I., & Droguett, E

فرمول‌بندی شده است. در این مقاله، برای اولین بار از راهبرد مازاد آماده به کار سرد استفاده شده است (اردکان و همدانی، ۲۰۱۴). ابویی اردکان و همدانی، همچنین بار دیگر برای یک سیستم سری - موازی ، راهبردی فعال و آماده به کار سرد را با هم در نظر گرفتند که با عنوان راهبرد مختلط معرفی شد. بنابراین، در این نوع مسئله تعیین نوع اجزاء، سطوح مازاد، تعداد واحدهای فعال و آماده به کار به طور همزمان در هر زیرسیستم به منظور افزایش قابلیت اطمینان باید مشخص شود. این مدل نیز با الگوریتم ژنتیک حل شد.

الگوریتم بهینه‌سازی ذرات به وسیله گارک و شارما مورد استفاده قرار گرفت. در این مقاله، یک مسئله تخصیص قابلیت اطمینان - مازاد به صورت دو هدفه داریم که اهداف شامل کاهش هزینه‌ها و افزایش قابلیت اطمینان است. سیستم مورد بررسی یک سیستم با ساختار سری است (گارگ[1] و همکاران، ۲۰۱۴). با توجه به فاکتورهای غیر قطعی و نامشخص، کاهش هزینه‌های سیستم و بهبود قابلیت اطمینان به طور همزمان مشکل است. در چنین شرایطی تصمیم‌گیری دشوار است و وجود چند تابع هدف منجر به راه حل بهینه پاراتو به جای یک راه حل واحد بهینه می‌شود. با این حال، به منظور انعطاف‌پذیری و سازگاری بیشتر مدل برای تصمیم‌گیری بهتر، مسئله بهینه‌سازی به صورت یک مسئله برنامه‌ریزی غیر خطی فازی با اعداد فازی بیان می‌شود. بنابراین، در یک محیط فازی مسئله بهینه‌سازی چند هدفه فازی فرموله می‌شود و برای حل آن از الگوریتم بهینه‌سازی ذرات استفاده شده است.

در ادامه، روش‌های نوظهوری مانند الگوریتم جست و جوی هارمونی جهانی مؤثر از سوی زو و همکاران ارائه شد. در این مقاله، الگوریتم جست و جوی هارمونی جهانی مؤثر برای حل دو مسئله قابلیت اطمینان به کار می‌رود. در مسئله اول، این مدل برای حل یک مسئله بهینه‌سازی با ساختار شبکه پل به کار می‌رود و مسئله دوم، یک سیستم حفاظت از سرعت غیر مجاز توربین گازاست (زو[2] و همکاران، ۲۰۱۱). هر دو مسئله به صورت برنامه‌ریزی عدد صحیح مختلط فرمول‌بندی می‌شوند. الگوریتم جست و جوی هارمونی جهانی مؤثر با الگوریتم هارمونی با مفهوم هوش جمعی در الگوریتم بهینه‌سازی ذرات ترکیب می‌شود. این الگوریتم توسعه‌یافته، همگرایی و ظرفیت جست و جوی فضای بیشتری را نسبت به الگوریتم جست و جوی هارمونی دارد.

در پژوهشی مشابه برای حل یک مسئله تخصیص قابلیت اطمینان - مازاد که هدف تعیین قابلیت اطمینان اجزا و تعداد آنها در هر زیر سیستم می‌باشد از الگوریتم جست و جوی فاخته با یک

1. Garg
2. Zou

تابع جریمه استفاده شده است. نتایج عددی نشان می‌دهد که الگوریتم فاخته با تابع جریمه، عملکرد بهتری نسبت به الگورتیم فاخته بدون تابع جریمه دارد (گارگ، ۲۰۱۵).

الگوریتم رقابت استعماری یکی از الگوریتم‌های فرا ابتکاری نوظهور است که در مسائل بسیاری کاربرد دارد. شریفی و ممبینی برای حل مسئله تخصیص مازاد که یکی از مهم‌ترین مسائل در زمینه قابلیت اطمینان است، از الگوریتم رقابلت استعماری استفاده کردند. مسئله مورد بررسی شامل: سطوح مازاد با هدف افزایش قابلیت اطمینان تحت یکسری محدودیت است (شریفی و ممبینی، ۲۰۱۵). روش تابع تولید جهانی و الگوریتم بازگشتی برای ارزیابی قابلیت اطمینان در پژوهشی که از سوی خورشیدی و همکاران ارائه شده به کار رفته است. این الگوریتم‌ها برای سیستم وزنی چند حالته با ساختار k از n مورد استفاده قرار گرفته است (خورشیدی و همکاران، ۲۰۱۵).

در پژوهش لویتین انتخاب راهبرد مازاد مورد بررسی قرار گرفته است. راهبرد مازاد آماده به کار گرم سریع‌تر حالت خرابی را تشخیص می‌دهد و هزینه‌های نگهداری در حالت مازاد آماده به کار گرم بیشتر از مازاد آماده به کار سرد است. در حالت فعال جزء مازاد به صورت سری به زیرسیستم مورد نظر متصل می‌شود و هر دو همزمان با یکدیگر شروع به کار می‌کنند و تا زمانی که یکی از این دو (جزء اصلی یا جزء اضافی) کار کنند، زیرسیستم کار می‌کند اما در حالت راهبردی آماده به کار جزء مازاد به صورت موازی به زیرسیستم مورد نظر متصل می‌شود. زمانی که جزء اصلی از کار بیفتد جزء مازاد شروع به کار می‌کند و جانشین جزء اصلی در سیستم می‌شود و تا زمانی که این جزء اضافی از کار نیفتاده باشد، سیستم به کار خود ادامه خواهد داد. در این پژوهش، راهبرد مازاد آماده به کار مورد استفاده قرار گرفته و برای حل مدل از الگوریتم ژنتیک استفاده شده است (لوتئین و زینگ[1]، ۲۱۰۴). یک الگوریتم ترکیبی جدید بر مبنای الگوریتم بهینه‌سازی ذرات و جست و جوی فاخته نیز از سوی وانگ و چای ارائه شد (وانگ، ۲۰۰۹). برای بررسی و آشنایی بیشتر با مسائل تخصیص مازاد، بهینه‌سازی قابلیت اطمینان و بهینه‌سازی مازاد اشاره‌ای می‌کنیم به مقاله هیلینگ و وب که در آن خلاصه‌ای از تمام مقالات منتشرشده از سال‌های ۱۹۹۶تا۲۰۱۳ در این زمینه است (هیلینگ و وب[2]، ۲۰۱۴).

در این مقاله، مدل برنامه‌ریزی عدد صحیح مختلط برای سه سیستم سری، سری – موازی و پل ارائه می‌شود و یک الگوریتم ترکیبی بر مبنای الگوریتم رقابت استعماری و ژنتیک ارائه خواهد شد. ساختار این مقاله به شرح زیر است: در بخش ۲ به معرفی سه نوع از مسائل تخصیص قابلیت اطمینان – مازاد می پردازیم. در بخش ۳ علاوه بر توضیح مختصری در مورد الگوریتم رقابت

1. Levitin, G., Xing, L
2. Heiling, L., & Vob, S

استعماری و ژنتیک به شرح الگوریتم ترکیبی خواهیم پرداخت . در بخش ۴ کارایی الگوریتم ارائه‌شده را بررسی می‌کنیم و در نهایت، بخش ۵ شامل نتیجه‌گیری است.

تعریف مسئله

از آنجایی که مقادیر عدد صحیح سطوح افزونگی منجر به بازه پیوسته صفر و یک برای قابلیت اطمینان مؤلفه‌هاست، مسئله تخصیص قابلیت اطمینان - مازاد در دسته مسائل برنامه‌ریزی عدد صحیح مختلط قرار می‌گیرد (شیخعلی شاهی و ابراهیم پور، ۲۰۱۳) مدل ریاضی مسئله تخصیص قابلیت اطمینان - مازاد به صورت زیر است:

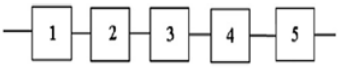
$$\max R_S = f(r,n) \qquad (1)$$
$$g(r,n) \le b$$
$$1 \le i \le m \quad , \quad 0 \le r_i \le 1 \qquad n_i \in Z^+$$

R_S	قابلیت اطمینان سیستم
n	بردار تخصیص افزونگی (تعداد اجزا مازاد)
m	تعداد زیر سیستم‌ها
b	حد بالای منبع
r	قابلیت اطمینان اجزا
r_i	قابلیت اطمینان جز در زیرسیستم i ام
g_i	i امین تابع محدودیت

۱. سیستم سری

در این سیستم‌ها، عملکرد صحیح سیستم، در گرو عملکرد صحیح تمامی اجزا آن است و در صورت از کار افتادگی یک جزء یا هر جزء، تمامی سیستم از کار خواهد افتاد. در واقع، در سیستم‌های سری تعدادی زیرسیستم به صورت متوالی به هم متصل شده‌اند. یک سیستم سری تا زمانی به کارکرد صحیح خود ادامه خواهد داد که همه زیرسیستم‌های متوالی به کار رفته در آن به درستی کار کنند. به عبارت دیگر، به محض اینکه یکی از زیرسیستم‌ها از کار بیفتد کل سیستم از کار کردن باز

می‌ایستد. این بدان معنی است که در یک سیستم سری طول عمر سیستم برابر با مینیمم طول عمر زیرسیستم‌های آن است. شکل و مدل سیستم سری به صورت زیر است:



شکل ۱: سیستم سری

$$\max f\ (r,n) = \prod_{i=1}^{n} R_i\,(n_i) \qquad (2)$$

$$g_1(r,n)=\sum_{i=1}^{m} w_i v_i^{\,2} n_i^{\,2} \le v \qquad (3)$$

$$g_2(r,n)=\sum_{i=1}^{m} a_i \left(-\frac{1000}{\ln r_i}\right)^{\beta_i} [n_i + e^{0.25 n_i}] \le c \qquad (4)$$

$$g_3(r,n)=\sum_{i=1}^{m} w_i n_i e^{0.25 n_i} \le w \qquad (5)$$

$$0 \le r_i \le 1 \quad , \quad 1 \le i \le m \quad , \quad n_i \in Z^+ \quad r_i \in R$$

معادله (۲) قابلیت اطمینان سیستم را نشان می‌دهد. محدودیت (۳) ترکیبی از وزن، تخصیص مازاد و حجم سیستم است. V حد بالای حجم است. محدودیت (۴) مربوط به هزینه سیستم است و C نشان‌دهنده حد بالای هزینه است. محدودیت (۵) مربوط به وزن سیستم و W حدأکثر میزان وزن موجود است.

v_i حجم مؤلفه در زیرسیستم i ام

w_i وزن مؤلفه در زیرسیستم i ام

n_i تعداد اجزا در زیرسیستم i ام

β_i ویژگی‌های فیزیکی

a_i ویژگی‌های فیزیکی

m تعداد زیرسیستم ها

r_i قابلیت اطمینان اجزا

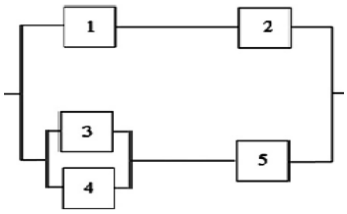
جدول زیر داده‌های ورودی سیستم سری است:

جدول ۱: داده‌های ورودی سیستم سری

W	C	V	w_i	$w_i.v_i^2$	β_i	$10^5 a_i$	stage
۲۵۰	۱۷۵	۱۱۰	۷	۱	۱	۲,۳۳۰	۱
			۸	۲	۲	۱,۴۵۰	۲
			۸	۳	۳	۰,۵۴۱	۳
			۹	۴	۴	۸,۰۵۰	۴
			۶	۲	۲	۱,۹۵۰	۵

۲. سیستم سری – موازی

در اغلب موارد، سیستم‌ها ترکیبی از زیرمجموعه‌های سری و موازی هستند. در چنین حالتی، شبکه مورد نظر را می‌توان به زیرسیستم‌های موازی و سری تقسیم کرد. سیستم زیر یک سیستم سری – موازی با پنج زیر سیستم است. محدودیت‌ها مانند سیستم سری است اما با توجه به ساختار مسئله تابع هدف به صورت زیر است:



شکل ۲: سیستم سری – موازی

$$\max f\ (r,n) = 1-(1-R_1R_2)(1-(1-R_3)(1-R_4)R_5) \qquad (6)$$

$$g_1(r,n)=\sum_{i=1}^{m} w_i v_i^2 n_i^2 \le v$$

$$g_2(r,n)=\sum_{i=1}^{m} a_i \left(-\frac{1000}{\ln r_i}\right)^{\beta_i} [n_i + e^{0.25 n_i}] \le c$$

$$g_3(r,n)=\sum_{i=1}^{m} w_i n_i e^{0.25 n_i} \le w$$

$$0 \le r_i \le 1 \quad , \ 1 \le i \le m \quad , \quad n_i \in Z^+ \quad r_i \in R$$

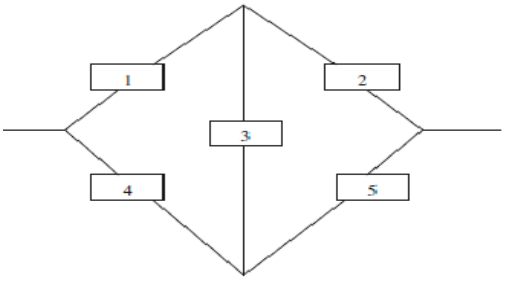
داده‌های ورودی این سیستم در جدول زیر است:

جدول ۲: داده‌های ورودی سیستم سری – موازی

stage	$10^5.\alpha_i$	β_i	$w_i.v_i^2$	w_i	V	C	W
۱	۲,۵۰۰	۱,۵	۲	۳,۵	۱۸۰	۱۷۵	۱۶۵
۲	۱,۴۵۰	۱,۵	۴	۴,۰			
۳	۰,۵۴۱	۱,۵	۵	۴,۰			
۴	۰,۵۴۱	۱,۵	۸	۳,۵			
۵	۲,۱۰۰	۱,۵	۴	۴,۵			

۳. سیستم پل

در این سیستم‌ها برخی از اجزا به صورت اشتراکی بین زیرسیستم‌ها عمل می‌کنند. این شبکه‌ها دارای حالت بینابین سیستم‌های موازی و سری هستند. در این سیستم نیز همان محدودیت‌های سیستم سری را خواهیم داشت اما تابع هدف متفاوت است:



شکل ۳: سیستم پل

$$\max f\ (r,n) = R_1R_2 + R_3R_4 + R_1R_4R_5 - R_2R_3R_5 - R_1R_2R_3R_4 - R_1R_2R_3R_5 - R_1R_3R_4R_5 - R_2R_3R_4R_5 + 2R_1R_2R_3R_4R_5 \qquad (6)$$

$$g_1(r,n)=\sum_{i=1}^{m} w_i v_i^2 n_i^2 \le v$$

$$g_2(r,n)=\sum_{i=1}^{m} a_i \left(-\frac{1000}{\ln r_i}\right)^{\beta_i} [n_i + e^{0.25n_i}] \le c$$

$$g_3(r,n)=\sum_{i=1}^{m} w_i n_i e^{0.25n_i} \le w$$

$$0 \le r_i \le 1 \quad , \ 1 \le i \le m \quad , \quad n_i \in Z^+ \quad r_i \in R$$

در جدول زیر داده‌های ورودی این مسئله را خواهیم داشت:

جدول ۳: داده‌های ورودی سیستم پل

stage	$10^5.\alpha_i$	β_i	$w_i.v_i^2$	w_i	V	C	W
۱	۲,۳۳۰	۱,۵	۱	۷	۱۱۰	۱۷۵	۲۵۰
۲	۱,۴۵۰	۱,۵	۲	۸			
۳	۰,۵۴۱	۱,۵	۳	۸			
۴	۸,۰۵۰	۱,۵	۴	۶			
۵	۱,۹۵۰	۱,۵	۲	۹			

پیاده‌سازی الگوریتم ترکیبی رقابت استعماری و ژنتیک

۱. الگوریتم ژنتیک

الگوریتم ژنتیک یک روش آماری است که در حل مسائل بهینه‌سازی به‌کار می‌رود. ایده اولیه این روش از نظریه تکاملی داروین الهام گرفته شده است و کارکرد آن براساس ژنتیک طبیعی استوار است. اصول اولیه الگوریتم ژنتیک از سوی هلند و همکارانش در دانشگاه میشیگان ارائه شد. سپس در سال۱۹۷۵، مبانی ریاضی آن در کتابی از سوی هلند با نام"تطابق در سیستم‌های طبیعی و مصنوعی"منتشر شد (مقدم و همکاران، ۱۳۸۴). آنان در تحقیقات خود به فرایند سازگاری در سیستم‌های طبیعی توجه و برای مدل‌سازی آن در سیستم‌های مصنوعی که باید دارای توانایی‌های سیستم‌های طبیعی باشند، تلاش کردند. الگوریتم ژنتیک یکی از مهم‌ترین الگوریتم‌های فرا ابتکاری است که از آن برای بهینه‌سازی برای توابع تعریف‌شده روی دامنه محدود استفاده می‌شود. در این الگوریتم، اطلاعات گذشته با توجه به موروثی بودن الگوریتم استخراج شده و در روند جستت و جو مورد استفاده قرار می‌گیرد. مفاهیم الگوریتم ژنتیک در سال۱۹۸۹ از سوی گلبرگ توسعه داده شد.

- **ساختار الگوریتم ژنتیک**

۱. کروموزوم

رشته یا دنباله‌ای از بیت‌ها که به عنوان شکل کد شده یک جواب ممکن (مناسب یا نامناسب) از مسئله مورد نظر است، چنانچه از کدگذاری دودویی استفاده شود، هر بیت، یکی از مقادیر صفر و یک را می‌پذیرد.

۲. تابع هدف و برازندگی

تابعی است که مقدار متغیر مسئله در آن قرار داده شده است، بدین طریق، مطلوبیت هر جواب مشخص می‌شود. در مسائل بهینه‌سازی[1]، تابع هدف به عنوان تابع برازندگی به کار می‌رود.

تابع هدف برای تعیین اینکه افراد چگونه در محدوده مسئله ایفای نقش می‌کنند، استفاده می‌شود و تابع برازندگی معمولاً برای تبدیل مقدار تابع هدف به یک مقدار برازندگی وابسته به آن استفاده می‌شود. به عبارت دیگر، داریم:

$$F(n) = g(f(x)) \qquad (7)$$

به طوری که f تابع هدف بوده و تابع g مقدار تابع هدف را به یک عدد غیر منفی تبدیل می‌کند و F برازندگی مربوط به آن است. مناسب بودن یا نبودن جواب با مقداری که از تابع برازندگی به دست می‌آید، سنجیده می‌شود. چون مسئله از نوع بهینه‌سازی است، تابع برازش با تابع هدف مسئله یکسان است.

۳. اندازه جمعیت و تعداد تولید

تعداد کروموزوم‌ها را اندازه جمعیت می‌گویند. یکی از مزیت‌های الگوریتم‌های ژنی نسبت به روش‌های جست و جوی سنتی این است که از جست و جوی موازی استفاده می‌شود. با تعریف فوق، اندازه جمعیت، اندازه جست و جوهای موازی است. یکی از ویژگی‌های الگوریتم ژنتیک این است که به جای تمرکز بر روی یک نقطه از فضای جست و جو یا یک کروموزوم بر روی جمعیتی از کروموزوم کار می‌کند.

- **عملگرهای ژنتیک**

برای پیدا کردن یک نقطه در فضای جست و جو باید از عملگرهای ژنتیک استفاده کرد. دو مورد از این عملگرها عبارت‌اند از:

۱. عملگر تقاطعی

عملگر اصلی جهت تولید کروموزوم‌های جدید در الگوریتم ژنتیک، عملگر تقاطع است. این عملگر مشابه همتای خودش در طبیعت، افراد جدیدی تولید می‌کند که اجزای (ژن های) آن از والدینش تشکیل می‌شود.

۲. عملگر جهش

1. optimization

جهش، یک فرایند تصادفی است که در آن محتوای یک ژن با ژن دیگر برای تولید یک ساختار ژنتیک جدید جایگزین می‌شود. عملگر جهش، نقش بسیار مهمی در فرار از بهینه‌های محلی و افزایش سرعت همگرایی الگوریتم ژنتیک ایفا می‌کند (والنت و گانکالوز[1]، ۲۰۰۹).

- **نسل**

هر تکرار الگوریتم را که منجر به ایجاد یک جمعیت جدید می‌شود، یک نسل می‌گویند . جهش یکی از پدیده‌های علم ژنتیک که به ندرت در برخی از کروموزوم‌ها رخ می‌دهد. و در طی آن فرزندان خصوصیاتی پیدا می‌کنند که متعلق به هیچ یک از والدین نیست. نقش جهش در الگوریتم ژنتیک بازگرداندن مواد ژنتیکی گم شده و یا پیدا نشده داخل جمعیت است تا از همگرایی زودرس الگوریتم به جواب‌های بهینه محلی جلوگیری شود. در جهش، یکسری از ژن‌ها را به طور تصادفی برمی‌گزینیم و صفرها را به یک و یک‌ها را به صفر تبدیل می‌کنیم. یکی از روش‌های جهش بدین صورت است که ابتدا با توجه به یک عدد کوچک‌تر از یک به نام احتمال جهش برای هر ژن از جمعیت یک عدد تصادفی فراخوانی می‌شود. اگر این عدد تصادفی از احتمال جهش کوچک‌تر بود، در آن ژن جهش رخ می‌دهد.

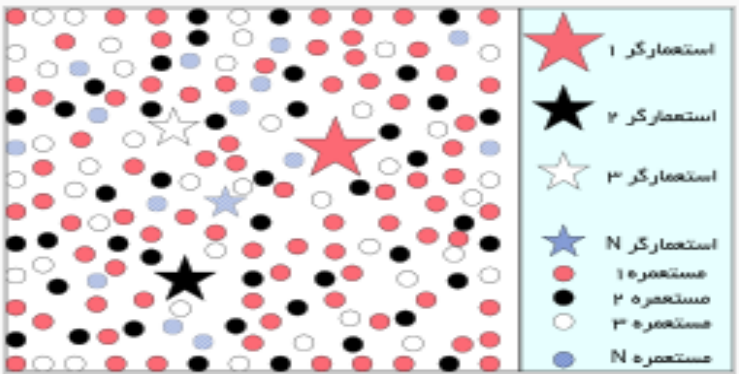
۲. الگوریتم رقابت استعماری

الگوریتم رقابت استعماری[2] روشی در حوزه محاسبات تکاملی است که به یافتن پاسخ بهینه مسائل مختلف بهینه‌سازی می‌پردازد. این الگوریتم با مدلسازی ریاضی فرایند تکامل اجتماعی - سیاسی، الگوریتمی برای حل مسائل ریاضی بهینه‌سازی ارائه می‌دهد. از لحاظ کاربرد، این الگوریتم در دسته الگوریتم های بهینه‌سازی تکاملی همچون: الگوریتم ژنتیک، الگوریتم بهینه‌سازی ذرات، الگوریتم تبرید شبیه‌سازی شده[3]، الگوریتم کلونی مورچگان و ... قرار می‌گیرد. همانند همه الگوریتم‌های قرار گرفته در این دسته، الگوریتم رقابت استعماری نیز مجموعه اولیه‌ای از جواب‌های احتمالی را تشکیل می‌دهد. این جواب‌های اولیه در الگوریتم ژنتیک با عنوان "کروموزوم"، در الگوریتم ازدحام ذرات با عنوان "ذره" و در الگوریتم رقابت استعماری نیز با عنوان "کشور" شناخته می‌شوند. الگوریتم رقابت استعماری با روند خاصی که در ادامه می‌آید، این جواب‌های اولیه (کشور ها) را به تدریج بهبود می‌دهد و در نهایت، جواب مناسب مسئله بهینه‌سازی (کشور مطلوب) را در اختیار می‌گذارد.

پایه‌های اصلی این الگوریتم را سیاست همسان‌سازی، رقابت استعماری و انقلاب تشکیل می‌دهند. این الگوریتم با تقلید از روند تکامل اجتماعی، اقتصادی و سیاسی کشورها و با مدل‌سازی

1. Valente, J., & Goncalves
2. Imperialist Competitive Algorithm
3. Simulated Annealing

ریاضی بخش‌هایی از این فرایند، عملگرهایی را در قالب منظم به صورت الگوریتم ارائه می‌دهد که می‌توانند به حل مسائل پیچیده بهینه‌سازی کمک کنند. در واقع، این الگوریتم جواب‌های مسئله بهینه‌سازی را در قالب کشورها نگریسته و سعی می‌کند، در طی فرایندی تکرارشونده این جواب‌ها را رفته رفته بهبود دهد و در نهایت به جواب بهینه مسئله برساند .همانند دیگر الگوریتم‌های تکاملی، این الگوریتم، نیز با تعدادی جمعیت اولیه تصادفی که هر کدام از آنها یک «کشور» نامیده می‌شوند، شروع می‌شود. تعدادی از بهترین عناصر جمعیت (معادل نخبه‌ها در الگوریتم ژنتیک) به عنوان استعمارگر[1] انتخاب می‌شوند. باقی‌مانده جمعیت نیز به عنوان مستعمره[2]، در نظر گرفته می‌شوند. استعمارگران بسته به قدرتشان، این مستعمرات را با یک روند خاص که در ادامه می‌آید، به سمت خود می‌کشند. قدرت کل هر امپراطوری، به هر دو بخش تشکیل‌دهنده آن یعنی کشور استعمارگر (به عنوان هسته مرکزی) و مستعمرات آن بستگی دارد. در حالت ریاضی، این وابستگی با تعریف قدرت امپراطوری به صورت مجوع قدرت کشور استعمارگر به اضافه درصدی از میانگین قدرت مستعمرات آن، مدل شده است. با شکل‌گیری امپراطوری‌های اولیه، رقابت استعماری میان آنها شروع می‌شود. هر امپراطوری‌ای که نتواند در رقابت استعماری، موفق عمل کند و بر قدرت خود بیفزاید (و یا حداقل از کاهش نفوذش جلوگیری کند)، از صحنه رقابت استعماری حذف خواهد شد. بنابراین، بقای یک امپراطوری، وابسته به قدرت آن در جذب مستعمرات امپراطوری‌های رقیب و به سیطره در آوردن آنها خواهد بود. در نتیجه، در جریان رقابت‌های استعماری، به تدریج بر قدرت امپراطوری‌های بزرگ‌تر افزوده می‌شود و امپراطوری‌های ضعیف‌تر حذف خواهند شد. امپراطوری‌ها برای افزایش قدرت خود مجبور خواهند شد تا مستعمرات خود را نیز پیشرفت دهند.

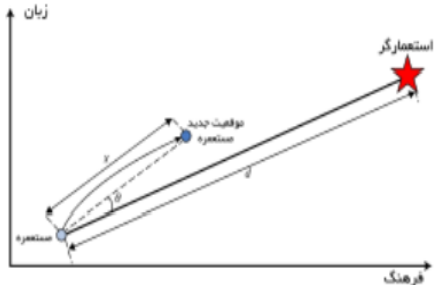


شکل ۴: نحوه تقسیم مستعمرات میان کشورهای استعمارگر

1. Imperialist
2. Colony

• سیاست جذب[1]

یعنی حرکت مستعمره به سمت استعمارگر. امپراطوری‌ها با پی‌گیری سیاست جذب، تلاش بسیاری را برای جذب مستعمرات در خود داشتند.



شکل ۵ : اعمال سیاست جذب در الگوریتم رقابت استعماری

خطی که مستعمره را به استعمارگر وصل می‌کند، یک بردار است با مختصات : $d=t-x$.

T موقعیت استعمارگر و x موقعیت مستعمره می‌باشد. اگر یک مستعمره وسط این خط باشد مختصات آن ضریبی از مختصات مستعمره اول می‌شود که این ضریب در الگوریتم رقابت استعماری با بتا شناخته می‌شود که داریم:

$$0 \le \beta \le 2 \qquad (8)$$

وقتی نتایجی که ارئه می‌دهیم، معتبر است که تقریباً مطمن باشیم تمام پاسخ‌های ممکن را بررسی کرده‌ایم. به همین دلیل، در سیاست جذب از حرکت مستقیم نمی‌توان نتیجه خوبی گرفت؛ زیرا جواب‌ها تنوع لازم را ندارند. به همین دلیل، از پارامتر دیگری به نام θ استفاده می‌کنیم. در واقع، به اندازه زاویه کوچکی انحراف ایجاد می‌کنیم تا سایر فضاها را هم جست و جو کنیم. همان‌طور که گفته شد، با ایجاد مقداری کمی انحراف (به اندازه زویه θ) می‌توانیم فضاهای دیگر را هم جست و جو کنیم و محدود به یک خط راست نباشیم اما پیاده‌سازی این کار کمی مشکل است؛ زیرا در فضای n بعدی به تعداد $n-1$ زاویه نیاز داریم. بنابراینریا، با افزایش بعد فضا به تعداد بیشتری θ نیاز است که پیاده‌سازی همه آنها مشکل است. برای رفع این مشکل برای هر کدام از مؤلفه‌های بردار تفاضل یک عدد تصادفی مجزا تعریف می‌کنیم. در واقع، با این کار محدود به فضای خاصی نمی‌شویم. (دقیقا همان نتیجه‌ای را به ما می‌دهد که با پارامتر θ انجام می‌شد).

1. Assimilation

$$x' = \beta r(t - x) \qquad (9)$$

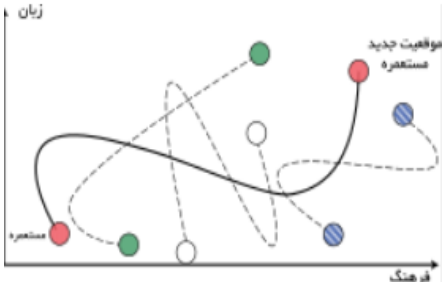
x' = موقعیت مستعمره جدید

β =assimilation ضریب

r = یک عدد تصادفی بین (۱،۰)

- **انقلاب**[1]

تغییرات ناگهانی در موقعیت یک کشور. در واقع، اگر کشوری نخواهد مانند استعمارگرخود عمل کند، انقلاب اتفاق می‌افتد. بروز انقلاب تغییرات ناگهانی را در ویژگی‌های اجتماعی سیاسی یک کشور ایجاد می‌کند. در الگوریتم رقابت استعماری، انقلاب با جا به جایی تصادفی یک کشور مستعمره به یک موقعیت تصادفی جدید مدل‌سازی می‌شود. انقلاب از دیدگاه الگوریتمی باعث می‌شود، کلیت حرکت تکاملی از گیر کردن در نقاط بهینه محلی نجات یابد که در بعضی موارد باعث بهبود موقعیت یک کشور می‌شود و آن را به یک محدوده بهینگی بهتری می‌برد.

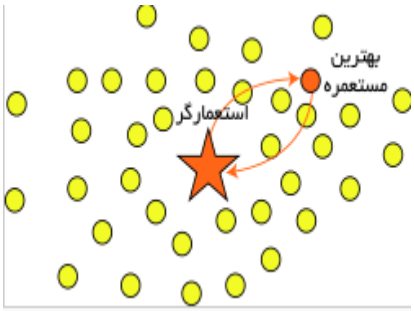


شکل۶: اعمال سیاست انقلاب در الگوریتم رقابت استعماری

- **تعویض مستعمره و استعمارگر**

گاهی درراهبرد جذب یکی از مستعمرات مقدار برازش بهتری نسبت به استعمارگر دارد. در این حالت، الگوریتم موقعیت استعمارگر و مستعمره را تغییر می‌دهد؛ سپس این رویه با استعمارگر جدید و در موقعیت جدید ادامه پیدا می‌کند و مستعمرات به سمت موقعیت استعمارگر جدید حرکت می‌کنند.

1. Revolution



شکل ۷: جا به جایی موقعیت مستعمره و استعمارگر

- **رقابت بین امپراطوری‌ها**

برای ارزیابی امپراطوری‌ها از یک شاخص ارزیابی استفاده می‌کنیم:

Mean (f (col)) f (imp) + zeta = مقدار تابع هدف برای امپراطوری

f(imp) مقدار تابع هدف برای استعمارگر

Zeta ضریب ثابت با مقدار ۰,۱

f(col) مقدار تابع هدف برای مستعمره

۳. الگوریتم ترکیبی

با ترکیب هوشمندانه از الگوریتم‌های تکاملی رقابت استعماری و ژنتیک، این الگوریتم‌ها می‌توانند نقاط ضعف یکدیگر را بپوشانند. الگوریتم ژنتیک توانایی تشخیص مهم‌ترین مناطق از مناطق بهینه‌سازی را ندارد. این الگوریتم همیشه کل منطقه بهینه‌سازی را برای یافتن جواب بهینه جست و جو می‌کند و تمرکزی روی منطقه خاص که امکان وجود جواب در آن بیشتر است، ندارد. این ضعف باعث همگرایی زودرس می‌شود (لیو، ۲۰۰۲). بنابراین، الگوریتم رقابت استعماری در تکرار اول بهینه‌سازی به جای الگوریتم ژنتیک روی مناطق بهینه‌سازی که شانس بیشتری برای بهینه جهانی شدن دارند، تمرکز می‌کند. بر همین اساس، در تکرارهای اول الگوریتم رقابت استعماری سریع‌تر عمل می‌کند و از لحظه‌ای که به یک نقطه خاص می رسد تا زمانی که مقدار بهینه مورد نظر را پیدا کند، بسیار زمان صرف می‌کند اما الگوریتم ژنتیک اگرچه دیرتر به نقطه خاص می‌رسد اما برای رسیدن به نقطه دوم که پاسخ است، زمان کمتری صرف می‌کند و این نشان می‌دهد که الگوریتم ژنتیک در گام دوم بهینه‌سازی سریع‌تر عمل می‌کند. از طرفی، در الگوریتم رقابت استعماری یک کروموزوم (کشور) وقتی می‌خواهد از نقطه‌ای به نقطه دیگر انتقال یابد، باید یک مسیر مشخصی را طی کند تا به مکان مورد نظر برسد اما در الگوریتم ژنتیک کروموزوم‌ها به ناگهان از نقطه‌ای از

صفحه دو بعدی به نقطه دیگر انتقال می‌یابند. فرض کنید با استفاده از الگوریتم رقابت استعماری یک جمعیت اولیه به صورت تصادفی ایجاد شده که این جمعیت به سمت امپراطوری در حرکت است. اگر الگوریتم رقابت استعماری به همین شکل ادامه یابد، ممکن است این جمعیتی که حتی با زاویه های تصادفی به سمت امپراطوری خود در حرکتند در نزدیکی نقاط بهینه مطلق از نقاطی رد شده و صرف نظر کرده باشند. در واقع، وقتی از نقطه‌ای رد می‌شوند، امکان برگشت به آن نقطه بسیار پایین است و فقط می‌توان تمهیداتی را در الگوریتم رقابت استعماری اعمال کرد تا جمعیت با آن نقطه برگردد. از خاصیت آشفتگی الگوریتم ژنتیک استفاده خواهیم کرد. با اعمال الگوریتم ژنتیک جمعیتی که جمع شده است، آشفته می‌شود و این کار باعث می‌شود جمعیتی که طبق یک روال عادی و تکراری در اطراف امپراطوری‌های‌شان جمع شدند، به عقب و نقاطی که در نظر نگرفته‌اند، نگاهی بیندازند و در نهایت، یکبار دیگر این جمعیت با الگوریتم رقابت استعماری منظم می‌شود (خوزانی و همکاران، ۲۰۱۰). در این مقاله، در مرحله اول الگوریتم رقابت استعماری اجرا می‌شود. ابتدا جمعیت اولیه تشکیل می‌شود که شامل ترکیبی از کروموزوم‌هایی(کشورهایی) است که به صورت تصادفی تولید شده‌اند که در وقع، همان قابلیت اطمینان هر جز (r) و تعداد اجزا مازاد (n) است. سپس جمعیت اولیه که شامل ترکیبی از جواب‌های نهایی حاصل از اجرای فرایند رقابت استعماری در مرحله اول و جواب‌هایی است که به طور تصادفی تولید شده‌اند، به الگوریتم ژنتیک منتقل می‌شوند و سپس الگوریتم ژنتیک به کمک این مجموعه جواب اولیه اجرا شده و این فرایند تا رسیدن به معیار توقف ادامه می‌یابد. اگر شرایط راضی‌کننده نباشد، بار دیگر الگوریتم رقابت استعماری اجرا خواهد شد. در نهایت، آن قدر بین الگوریتم ژنتیک و رقابت استعماری تکرار انجام خواهد شد تا به پاسخ بهینه برسیم.

تحلیل نتایج

با توجه به اینکه نرم‌افزار متلب یک نرم‌افزار بهینه‌سازی بوده و قادر به یافتن مقدار بهینه برای مسائل بهینه‌سازی است، مسائل محک را با این نرم‌افزار حل کردیم و بعد از حل مسائل تعداد بهینه اجزا مازاد در هر زیرسیستم، قابلیت اطمینان هر یک از از اجزا و قابلیت اطمینان کلی سیستم به دست آمد. این نتایج در جدول زیر نشان داده شده است:

جدول ۴: نتایج قابلیت اطمینان در سیستم‌های سری، سری – موازی و پل

parameter	Series system	Series_ parallel system	Bridge system
F(r,n)	0.952417	0.999828	0.999974
n_1	3	6	3
n_2	3	4	4

parameter	Series system	Series_ parallel system	Bridge system
n_3	3	1	3
n_4	3	1	3
n_5	3	1	1
r_1	0.7994	0.8134	0.8223
r_2	0.8070	0.5871	0.8404
r_3	0.8648	0.8886	0.9083
r_4	0.7206	0.5750	0.6511
r_5	0.7931	0.7065	0.7875

همان طور که از نتایج مشخص است، الگوریتم ترکیبی ارائه‌شده کارایی لازم برای حل مسئله بهینه‌سازی قابلیت اطمینان را داراست. اگر چه در تمامی سیستم‌ها قابلیت اطمینان کلی سیستم بالاست اما در سیستم پل قابلیت اطمینان بیشترین مقدار ممکن را به خود اختصاص داده است.

در جدول زیر نتایج این الگوریتم ترکیبی را با الگوریتم رقابت استعماری و ژنتیک که هر کدام به طور جداگانه در مسائل به عنوان روش حل استفاده شده‌اند، مقایسه می‌کنیم:

جدول ۵: نتایج قابلیت اطمینان با الگوریتم ژنتیک

GA			
Parameter	Series system	Series – parallel system	Bridge system
R	0.931578	0.9728	0.999879
n	(3,2,2,3,3)	(2,2,2,2,4)	(3,3,3,3,1)
	0.779427	0.7854	0.814090
	0.869482	0.8429	0.864614
r	0.902674	0.8853	0.890291
	0.714038	0.9179	0.701190
	0.786896	0.8703	0.734731

جدول ۶: نتایج قابلیت اطمینان با الگوریتم رقابت استعماری

ICA			
Parameter	Series system	Series – parallel system	Bridge system
R	0.931679	0.9845	0.999889
n	(3,2,2,3,3)	(2,2,2,2,4)	(3,3,2,4,1)
	0.779874	0.8220	0.82764257
	0.872057	0.8436	0.85747845
r	0.903426	0.8912	0.91419677
	0.71096	0.8986	0.64927372

ICA			
Parameter	Series system	Series – parallel system	Bridge system
	0.786902	0.8682	0.704092

جدول ۷: نتایج قابلیت اطمینان با الگوریتم ترکیبی رقابت استعماری و ژنتیک

ICA-GA			
Parameter	Series system	Series – parallel system	Bridge system
R	0.952417	0.9998	0.999974
n	(3,3,3,3,3)	(6,4,1,1,1)	(3,4,3,3,1)
	0.7994	0.8134	0.8223
	0.8070	0.5871	0.8404
	0.8648	0.8886	0.9083
r	0.7206	0.5750	0.6511
	0.7931	0.7065	0.7875

هسیه و همکاران مسئله تخصیص مازاد را به کمک الگوریتم ژنتیک و آفونسو و همکاران این مسئله را با استفاده از الگوریتم رقابت استعماری حل کرده‌اند. با توجه به نتایج، ترکیب این دو الگوریتم نتیجه بهتری در هر سه سیستم خواهد داشت. در واقع، الگوریتم ترکیبی حاضر، باعث افزایش بیشتری در قابلیت اطمینان نسبت به استفاده از هر کدام از این الگوریتم‌ها به طور جداگانه خواهد شد.

نتیجه‌گیری

تخصیص قابلیت اطمینان مناسب به هر یک از اجزا و استفاده از اجزا مازاد به صورت موازی در کنار اجزای اصلی سیستم، به طور همزمان یکی از راه‌های ارتقا قابلیت اطمینان است. این نوع تخصیص تحت عنوان تخصیص قابلیت اطمینان _ مازاد شناخته می‌شود. از این رو، در این مقاله برای افزایش قابلیت اطمینان از این راهکار برای افزایش قابلیت اطمینان استفاده شد. وجود عبارات پیچیده غیرخطی در مسئله بهینه‌سازی مورد بحث، لزوم استفاده از روش‌های فرا ابتکاری مناسب در مقابله با این پیچیدگی‌ها را بیش از پیش نشان می‌دهد. برای حل مدل از ترکیب الگوریتم رقابت استعماری و ژنتیک استفاده شده است؛ چرا که در بسیاری از زمان‌ها راه حل‌های ترکیبی راه‌حل‌های مناسب‌تری هستند و می‌توان با استفاده از یک الگوریتم نقاط ضعف الگوریتم دیگر را پوشش داد و با استفاده از نقاط قوت الگوریتم‌های ترکیب‌شونده می‌توان عملکرد بهتری را در حل مسائل بهینه‌سازی مختلف داشت. نتایج عددی نشان می‌دهد که این رویه حل، بسیار کاراست و عملکرد بهتری نسبت به استفاده از هر کدام از الگوریتم ها به تنهایی خواهد داشت.

منابع

Garg, H., Rani, H., Sharma, S., & Vishwakarma, Y. (2014). Intuitionistic fuzzy optimization technique for solving multi-objective reliability optimization problems in interval environment. Expert Syst Appl, 57-67

Zoulfagharia, H., Zeinal Hamadania, A., & Abouei Ardakanb, M. (2015). Multi-objective availability-redundancy allocation problem for a system with repairable and non-repairable components. Decision Science Letters, 289-302.

Chern, M. (1992). On the Computational Complexity of Reliability Redundancy Allocation in a Series System. Operations Research Letters, 11, 309-315.

Yalaoui, A., chatelet, E., & Chu, C. (2005). A new dynamic programming method for reliability_redundancy allocation in a parallel-series system. IEEE Transactions on Reliability, 2(54), 254-261.

.Billionnet, A. (2008). Redundancy allocation for series-parallel systems using integer linear programming. IEEE Transactions on Reliability(57), 507-16.

Ha, C., & Kuo, W. (2006). Reliability redundancy allocation:An improved realization for nonconvex nonlinear programming problems. European Journal of Operational Research, 1(127), 24-38.

Ramirez-Marquez, J. E., Coit, D. W., & Konak, A. (2004). Redundancy allocation for series-parallel systems using a max-min approach. Iie Transactions, 36(9), 891-898

Lins, I., & Droguett, E. (2011). Redundancy allocation problems considering systems with imperfect repairs using multi-objective genetic algorithms and discrete event simulation. Simul. Model. Pract. Theory(19), 362-281.

Ardakan, M. A., & Hamadani, A. Z. (2014)Reliability optimization of series–parallel systems with mixed redundancy strategy in subsystems. Reliability Engineering & System Safety, 130, 132-139.

Garg, H., & Sharma, S. (2013). Multi-objective reliability-redundancy allocation problem using particle swarm optimization. Comput Ind Eng, 64(1), 247-355.

Zou, D., Gao, L., Li, S., & Wu, J. (2011). An effective global harmony search algorithm for reliability problems. Expert Syst ApP, 4644-4648.

Garg, H. (2015). An approach for solving constrained reliability-redundancy allocation problems using cuckoo search algorithm. Beni-Suef University Journal of Basic and Applied Sciences, 4(1), 14-25.

Sharifi, M., & Mombeini, H. (2015). The effects of technical and organizational activities on redundancy allocation problem with choice of selecting redundancy strategies using imperialist competition algorithm. The Business & Management Review, 6(2).

Khorshidi, H. A., Gunawan, I., & Ibrahim, M. Y. (2015). On reliability evaluation of multistate weighted k-out-of-n system using present value. The Engineering Economist, 60(1), 22-39.

Levitin, G., Xing, L., & Dai, Y. (2014). Cold vs. hot standby mission operation cost minimization for 1-out-of-N systems. European Journal of Operational Research, 234(1), 155-162

Wang, Y., & CAI, Z. (2009). A hybrid multi-swarm particle swarm optimization to Solve constrained optimization problems. Frontiers of Computer Science in China, 3(1), 52-38

Heiling, L., & Vob, S. (2014). A scientometric analysis of redundancy allocation literature(1969–2013):Technical report. Institute of information Systems,University of Hamburg.

Sheikhalishahi, M., Ebrahimipour, V., & Shiri, H. (2013). A hybrid GA–PSO approach for reliability optimization in redundancy allocation problem. Int J Adv Manuf Technol, 317-338.

Moghadam, M. R., Afsar, a., & Sohrabi, b. (1384). Modeling of supply chain with genetic algorithm approach. (In Persian فارسی)

Valente, J., & Goncalves, J. (2009). A genetic algorithm approach for the singlemachine scheduling prolem with linear earliness and quadratic tradiness penalties. Computers and Operations Engineering, 36, 2707-2715.

.Liu, Y. L. (2002). Reactive power optimization by GA/SA/TS combined algorithms. International Journal of Electrical Power & Energy Systems, 2002., 24(9), 765-769.

Khorani, A. V., Farzad Razavi, B., & Ahsan Ghoncheh, C. (2010). A new hybrid evolutionary algorithm based on ICA and GA: Recursive-ICA-GA. The International Conference on Artificial Intelligence.