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

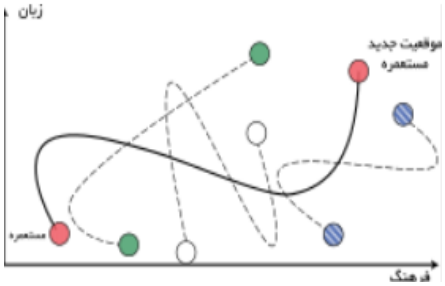
x' = موقعیت مستعمره جدید

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.

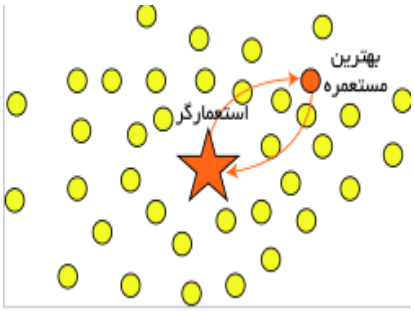


شکل۶: اعمال سیاست انقلاب در الگوریتم رقابت استعماری

- **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.