Iranian Journal of Wargaming
AJA Command and Staff University
http://www.ijwg.ir/
Online ISSN: 2676-7708
Print ISSN: 2676-5284

A Simulation of the Colonel Blotto Game Using Genetic Algorithm

Hamid Bigdeli[1]✉ | Siavash Ganji[2] | Mohammad Taghi Partovi[3]

1. Department of Science and Technology Studies, AJA Command and Staff University, Tehran, Iran. E-mail: H.bigdeli@casu.ac.ir
2. Department of Science and Technology Studies, AJA Command and Staff University, Tehran, Iran. E-mail: sganji@mailfa.com
3. Department of Science and Technology Studies, AJA Command and Staff University, Tehran, Iran. E-mail: mt.partovi@ut.ac.ir

Article Info

Article type:

Research Article

Article history:

Received 15 March 2023
Received in revised form 31 May 2023
Accepted 16 July 2023
Published online 4 September 2023

Keywords:

Game theory, Colonel Blotto, Genetic algorithm, Brute-force algorithm

ABSTRACT

Objective: The allocation of limited resources in the military field is an important challenge that seeks to maximize or minimize. Game theory is used to define resources in terms of conflict and definitions. One of the game theory research issues is Blato's Syringe game, which aims to allocate strategic resources.

Methodology: In this game, two players are given limited resources on a fixed number of battlefields, without knowledge of the opponent's decisions. In this article, this issue is solved by examining the genetic algorithm, which is a meta-heuristic algorithm.

Findings: On the other hand, by introducing the brute-force algorithm and solving all situations, a comparison has been made with the genetic algorithm in terms of execution time and optimality of resource allocation. According to the concluded data, the genetic algorithm works in this problem in 47% of the time similar to the brute-force algorithm and in 42% of the time it works better.

Conclusion: Using the Python programming language and the Django software framework, the Colonel Blato game simulator has been designed and implemented on the web platform, which includes user management pages, game creation, game list and game guide. This simulator has the ability to create problems and choose an opponent as another user, random allocation and genetic algorithm.

Cite this article: bigdeli, H., Partovi, M., & Ganji, S. (2023). A Simulation of the Colonel Blotto Game using Genetic Algorithm. *Iranian Journal of Wargaming*, *6*(12), -. doi: 10.22034/ijwg.2023.409893.1066

Publisher: Command and Staff University
DOI:

Iranian Journal of Wargaming
AJA Command and Staff University
http://www.ijwg.ir/
Online ISSN: 2676-7708
Print ISSN: 2676-5284

شبیه‌سازی بازی سرهنگ بلاتو با استفاده از الگوریتم ژنتیک

حمید بیگدلی[۱]✉ | سیاوش گنجی[۲] | محمدتقی پرتوی[۳]

۱. گروه مطالعات علم و فناوری، دانشگاه فرماندهی و ستاد آجا، تهران، ایران، رایانامه: H.Bigdeli@casu.ac.ir
۲. گروه مطالعات علم و فناوری، دانشگاه فرماندهی و ستاد آجا، تهران، ایران، رایانامه: sganji@mailfa.com
۳. گروه مطالعات علم و فناوری، دانشگاه فرماندهی و ستاد آجا، تهران، ایران، رایانامه: mt.partovi@ut.ac.ir

اطلاعات مقاله

نوع مقاله:
مقالهٔ پژوهشی

تاریخ دریافت:
۱۴۰۱/۱۲/۲۴

تاریخ بازنگری:
۱۴۰۲/۰۳/۱۰

تاریخ پذیرش:
۱۴۰۲/۰۴/۲۵

تاریخ انتشار:
۱۴۰۲/۰۶/۱۳

کلیدواژه‌ها:
نظریه بازی‌ها، سرهنگ بلاتو، الگوریتم ژنتیک، الگوریتم بروت-فورس

چکیده

هدف: تخصیص منابع محدود در حوزه نظامی یک چالش مهم است که به دنبال سود بیشینه یا ریسک کمینه است. برای تخصیص منابع در شرایط تضاد و تعارض از نظریه بازی‌ها استفاده می‌گردد. یکی از مسائل مورد بررسی در نظریه بازی، بازی سرهنگ بلاتو است که با هدف تخصیص منابع استراتژیک است.

روش: در این بازی، دو بازیکن به تعداد ثابتی از میدان‌های جنگ، بدون داشتن اطلاعات در مورد تصمیمات حریف، منابع محدود را تخصیص می‌دهند. در این مقاله با بررسی الگوریتم ژنتیک که یک الگوریتم فراابتکاری است به حل این مسئله پرداخته شده است.

یافته‌ها: از طرفی دیگر با معرفی الگوریتم بروت-فورس و حل تمامی حالات، از نظر زمان اجرا و بهینگی تخصیص منابع با الگوریتم ژنتیک مقایسه‌ای صورت گرفته است. طبق داده‌های نتیجه‌گیری شده، الگوریتم ژنتیک در این مسئله در ۴۷ درصد مواقع مشابه الگوریتم بروت-فورس و در ۴۲ درصد از آن بهتر عمل می‌کند.

نتیجه‌گیری: استفاده از زبان برنامه‌نویسی پایتون و چارچوب نرم‌افزاری جنگو شبیه‌ساز بازی سرهنگ بلاتو در بستر وب طراحی و پیاده‌سازی شده است که شامل صفحات مدیریت کاربر، ساخت بازی، لیست بازی‌ها و راهنمای بازی است. این شبیه‌ساز قابلیت ایجاد مسائل و انتخاب حریف به‌عنوان کاربر دیگر، تخصیص تصادفی و الگوریتم ژنتیک را دارد.

استناد: بیگدلی، حمید، پرتوی، محمدتقی و گنجی، سیاوش. (۱۴۰۲). شبیه‌سازی بازی سرهنگ بلاتو با استفاده از الگوریتم ژنتیک. *دوفصلنامه بازی جنگ*.

ناشر: دانشگاه فرماندهی و ستاد ارتش جمهوری اسلامی ایران

DOI: 10.22034/IJWG.2023.409893.1066

مقدمه

نظریه بازی، یک روش تحلیلی برای مطالعه تعاملات بین افراد و گروه‌ها است که در دهه ۱۹۴۰ میلادی به وجود آمد. این نظریه به شیوه‌ای محاسباتی عمل می‌کند و سعی دارد رفتارهای شخصیتی و تصمیم‌های افراد را در شرایط پیچیده پیش‌بینی کند. نظریه بازی در حوزه‌های مختلف بکار گرفته می‌شود. در علوم اقتصادی، می‌تواند در تحلیل رفتار شرکت‌ها در بازار و همچنین بهینه‌سازی استراتژی‌های تولید و فروش محصولات کمک کند. در علوم سیاسی، می‌تواند در تحلیل تعاملات بین کشورها و نیز تصمیم‌گیری‌های گروهی دولت‌ها در مسائل بین‌المللی مورد استفاده قرار گیرد. همچنین، در علوم رفتاری و روانشناسی، می‌تواند به بررسی رفتار انسان‌ها و تصمیم‌گیری‌های اجتماعی کمک کند. نظریه بازی که برای اولین بار توسط جان فون نویمان و اسکار مورگنشترن در دهه ۱۹۴۰ معرفی شد، به عنوان یک ابزار ریاضی برای بررسی رفتارهای تقابلی با توجه به منافع و اهداف متقابل استفاده می‌شود. یکی از مسائل مهم نظامی اختصاص منابع محدود به قسمت‌های مختلف است به صورتی که بتوان به بیشترین میزان سود یا کمترین میزان ریسک رسید. بازی سرهنگ بلاتو یکی از مسائل نظریه بازی‌ها است که یک مدل از تخصیص منابع استراتژیک است. دو بازیکن مقدار مشخصی از منابع را به تعداد ثابتی از میدان‌های جنگ با تعداد معین و بدون اطلاع از انتخاب‌های حریف اختصاص می‌دهند تا برنده این بازی باشند و از اختصاص منابع محدود خود به کمترین ریسک ممکن برسند. مسئله سرهنگ بلاتو یک مدل ریاضی در نظریه بازی‌ها است که برای تحلیل رقابت میان دو بازیکن استفاده می‌شود. در این مدل، دو بازیکن با یکدیگر رقابت می‌کنند تا نیروهایی را در میان چندین میدان نبرد تقسیم کنند. هدف هر بازیکن کسب بیشترین پیروزی در میدان‌های نبرد است؛ اما چون منابع محدود هستند، هر بازیکن باید تصمیم بگیرد که چگونه منابع خود را بین میدان‌های نبرد تخصیص‌دهد تا پیروزی بیشتری کسب کند. با توجه به اهمیت نظریه بازی در مسائل نظامی و تصمیم‌گیری‌ها، می‌توان با پیاده‌سازی سامانه‌ای بر بستر وب که قابلیت‌های تخصیص‌دهی تصادفی و بازی دونفره را داراست، در بهبود تصمیم‌گیری فرماندهان و افسران و تخصیص منابع تأثیرگذار به آن‌ها کمک کرد. علاوه بر این، از طریق حل بهینه مسئله، این سامانه می‌تواند در تمرین و اخذ تصمیمات چابک و بهینه برای افراد مفید باشد.

مبانی نظری و پیشینه‌های پژوهش

بازی سرهنگ بلاتو

بازی سرهنگ بلاتو از تعدادی میدان جنگ تشکیل شده است. دو سرهنگ به رنگ‌های آبی و قرمز در مقابل هم بازی می‌کنند. بازی سرهنگ بلاتو یک بازی جمع-صفر است. در بازی سرهنگ بلاتو سرهنگ آبی و قرمز به یک مقدار سرباز(منابع) در اختیار دارند. تعدادی میدان جنگ مستقل از هم نیز وجود دارد که سرهنگ‌ها باید سربازان خود را به این میدان‌های جنگ اختصاص دهند. این اختصاص به صورت هم‌زمان و بدون اطلاع از انتخاب‌های طرف مقابل باید انجام شود. در هر میدان جنگ، سرهنگی که سربازان بیشتری را به آن میدان اختصاص داده باشد، پیروز آن میدان جنگ می‌شود. در شرایطی که تعداد سربازان برابر است، میدان جنگ پیروزی ندارد. در نهایت کسی که بیشترین میدان‌های جنگ را پیروز شده باشد، برنده است. این بازی بر خلاف شرح بسیار ساده‌ای که دارد مسئله‌ای پیچیده است و حل آن آسان نیست. یکی از استراتژی‌هایی که در ابتدا به ذهن می‌رسد اختصاص سربازان به صورت مساوی است. این استراتژی می‌تواند به باخت منجر شود (امیری و بیگدلی، ۱۳۹۸). به طور مثال ۴ میدان جنگ وجود دارد و هر سرهنگ به تعداد ۸ سرباز در اختیار دارد. سرباز قرمز از استراتژی اختصاص به صورت مساوی استفاده می‌کند اما سرهنگ آبی استراتژی دیگری را انتخاب می‌کند. در جدول (۱) انتخاب‌های سرهنگ‌ها در هر میدان جنگ نمایش داده شده است.

جدول (۱) نمونه‌ای از اختصاص سربازان به میدان‌های جنگ

شماره میدان جنگ	۱	۲	۳	۴
اختصاص سربازان توسط بازیکن قرمز	۲	۲	۲	۲
اختصاص سربازان توسط بازیکن آبی	۳	۳	۲	۰
برنده میدان جنگ	آبی	آبی	برابر	قرمز

با این استراتژی، بازیکن آبی دو میدان جنگ و بازیکن قرمز یک میدان جنگ را پیروز شدند بنابراین بازیکن آبی برنده بازی است. می‌توان دریافت که اختصاص منابع به صورت مساوی استراتژی همیشه برنده‌ای نیست. به طور کلی این مسئله را می‌توان به این‌گونه مدل‌سازی کرد. بازیکن آبی و قرمز با A و B نشان داده شده و فرض می‌شود N عددی

طبیعی است که تعداد منابع هر بازیکن را نشان می‌دهد. n عددی طبیعی است که نشان‌دهنده تعداد میدان‌های جنگ است. $p \in \{A, B\}$ یکی از بازیکنان است و بردار $V_p = (v_p^1, v_p^2, \dots, v_p^n)$ نشان‌دهنده اختصاص منابع هرکدام از بازیکنان است با این محدودیت که $N = \sum_{i=0}^{n} v_p^i , p \in \{A, B\}$ نشان‌دهنده اختصاص به اندازه منابع موجود است. تابع سود به صورتی تعریف می‌شود که جمع نتایج در میدان‌های جنگ را نشان می‌دهد و همچنین به دلیل جمع صفر بودن این بازی، در صورت برابری، سودی برابر ۰٫۵ نصیب هر دو بازیکن خواهد شد.

$$profit_A = \sum_{i=1}^{n} comp(v_A^i, v_B^i)$$

که تابع $comp$ به صورت زیر تعریف می‌شود:

$$comp(x, y) = \begin{cases} 1 & x > y \\ 0 & x < y \\ 0.5 & x = y \end{cases}$$

از آنجایی که این بازی یک بازی جمع-صفر است، سود بازیکن B برابر با $profit_B = n - profit_A$ است.

الگوریتم ژنتیک

الگوریتم‌های فراابتکاری، الگوریتم‌هایی برای حل مسائل بهینه‌سازی پیچیده هستند که از الگوها و راهبردهای طبیعی الهام گرفته‌شده‌اند. در سال‌های اخیر، الگوریتم‌های فراابتکاری برای حل مسائل پیچیده در زمینه‌های مختلف استفاده می‌شود (Kumar, 2014). دو عنصر تشدید و گوناگونی عناصر کلیدی الگوریتم‌های فراابتکاری هستند. تعادل مناسب بین این عناصر برای حل مشکلات واقعی به شیوه‌ای مؤثر موردنیاز است. بیشتر الگوریتم‌های فراابتکاری از فرآیند تکامل بیولوژیکی، رفتارهای جمعیتی و قوانین فیزیک الهام گرفته‌شده‌اند. این الگوریتم‌ها به طور کلی به دو دسته الگوریتم‌های فراابتکاری مبتنی بر تک راه‌حل و الگوریتم‌های فراابتکاری مبتنی بر جمعیت تقسیم می‌شوند. الگوریتم‌های فراابتکاری مبتنی بر تک راه‌حل از جواب کاندید واحد استفاده می‌کنند و این جواب را با استفاده از جستجوی محلی بهبود می‌بخشند (Boussaid, 2013). با این حال، جواب به دست آمده از الگوریتم فراابتکاری مبتنی بر تک راه‌حل مانند الگوریتم گرگ خاکستری ممکن است در راه‌حل بهینه محلی گیر کند (Kumar, 2017). ازجمله الگوریتم‌های

فراابتکاری شناخته‌شده مبتنی بر تک راه‌حل عبارت‌اند از: الگوریتم تبرید شبیه‌سازی‌شده، الگوریتم جستجوی ممنوعه و جستجوی محلی هدایت‌شده. الگوریتم‌های فراابتکاری مبتنی بر جمعیت از چندین جواب نامزد در طول فرآیند جستجو استفاده می‌کنند. این الگوریتم‌های فراابتکاری تنوع جمعیت را حفظ می‌کنند و از گیرکردن جواب‌ها در بهینه محلی جلوگیری می‌کنند. برخی از الگوریتم‌های فراابتکاری مبتنی بر جمعیت معروف عبارت‌اند از: الگوریتم ژنتیک (Michalewicz, 1992)، بهینه‌سازی ازدحام ذرات (Kennedy, 1995)، بهینه‌سازی کلونی مورچه‌ها (Dorigo, 2006). یکی از معروف‌ترین این الگوریتم‌ها، الگوریتم ژنتیک است. الگوریتم ژنتیک یک جستجوی فراابتکاری است که از نظریه تکامل طبیعی چارلز داروین الهام گرفته شده است. این الگوریتم فرآیند انتخاب طبیعی را منعکس می‌کند که در آن مناسب‌ترین افراد برای تولیدمثل انتخاب می‌شوند تا فرزندان نسل بعدی را تولید کنند. به طور خلاصه الگوریتم ژنتیک فرایندی تکرارشونده است که از ایده‌های اصلی فرگشت برای حل مسائل پیچیده بهینه‌سازی الهام گرفته است. ابتدا چندین راه‌حل تصادفی تولید می‌شود. سپس این راه‌حل‌ها ارزیابی می‌شوند تا بهترینشان انتخاب شوند. سپس برای ایجاد جواب‌های بهتر، بهترین جواب‌ها با هم ترکیب می‌شوند تا فرزندانی تولید کنند. این فرآیند تولیدمثل باعث می‌شود راه‌حل‌های بهتری ایجاد شود. سپس تعدادی از این راه‌حل‌ها جهش می‌کنند تا تنوع بیشتری در میان آن‌ها وجود داشته باشد. این فرایند تا زمانی که جواب مطلوبی پیدا شود، تکرار می‌شود. در یک الگوریتم ژنتیک پنج مرحله در نظر گرفته شده است. مرحله اول جمعیت اولیه است که افراد (کروموزوم‌ها) با استفاده از روش‌های تصادفی یا تکاملی ساخته می‌شوند. مرحله دوم تابع تناسب است که هر فرد در جمعیت بر اساس یک تابع تناسب که باید بهینه شود، ارزیابی می‌شود. مرحله سوم انتخاب است که افراد با عملکرد بهتر در ارزیابی، بیشتر احتمال برگزیده شدن برای تولید نسل بعدی را دارند. مرحله چهارم تقاطع است که با استفاده از عملیات ترکیب، فرزندان جدیدی از والدین ایجاد می‌شوند. مرحله پنجم جهش است که با استفاده از عملیات جابه‌جایی ژنتیکی، برخی از افراد جدید از والدین ایجاد می‌شوند (Michalewicz, 1992).

الگوریتم بروت-فورس

الگوریتم‌های بروت-فورس، روش‌های ساده حل مسئله هستند که به قدرت محاسباتی و تلاش در هر امکانی برای حل مسئله تکیه می‌کنند و از تکنیک‌های پیشرفته جهت بهبود کارایی استفاده نمی‌کنند. این الگوریتم‌ها معمولاً زمانی استفاده می‌شوند که اندازه مسئله محدود است یا سادگی پیاده‌سازی مهم‌تر از سرعت است. یکی از مثال‌های کلاسیک این الگوریتم‌ها، مسئله فروشنده دوره‌گرد است که در آن راه‌حل بروت-فورس، محاسبه فاصله کلی برای هر مسیر ممکن و انتخاب کوتاه‌ترین آن‌هاست. پیچیدگی زمانی الگوریتم بروت-فورس $O(mn)$ است. این بدان معناست که با استفاده از الگوریتم بروت-فورس، برای جستجوی یک رشته با n کاراکتر در یک رشته با m کاراکتر، $n * m$ بار تلاش باید صورت گیرد. در علم کامپیوتر، جستجوی بروت-فورس یک تکنیک حل مسئله است که در آن همه حالات ممکن به طور سیستماتیک بررسی می‌شود تا مشخص شود آیا آن‌ها به شرط مسئله صحیح هستند یا خیر. این تکنیک معمولاً زمانی استفاده می‌شود که اندازه مسئله محدود است یا هنگامی که الگوریتم‌های مشخصی برای کاهش مجموعه راه‌حل‌های ممکن وجود دارد (Heule, 2017).

پیشینه پژوهش

مسئله سرهنگ بلاتو یک مسئله کلاسیک در حوزه نظریه بازی‌ها است و تا به حال حل‌های مختلفی برای مسئله اصلی و همچنین مسائل مشابه و گسترش‌یافته آن ارائه شده است. این مسئله نخستین بار توسط امیل بورل، ریاضی‌دان فرانسوی مطرح شد.

مک‌دانل[۱] و همکاران به تخصیص استراتژیک منابع در دو میدان جنگ مانند بازی معمولی سرهنگ بلاتو پرداخته‌اند. در این مقاله، دو بازیکن به طور هم‌زمان نیروهای خود را در تنها دو میدان نبرد تخصیص می‌دهند. نیروی بزرگ‌تر در هر میدان نبرد پیروز می‌شود و بازدهی برای یک بازیکن، مجموع ارزش‌های میدان‌های نبرد به دست آمده است. آن‌ها مجموعه‌ای از تعادل‌های نش را در تمام بازی‌های دو میدان نبرد مشخص کرده‌اند. در این مقاله یک توصیف کامل از مجموعه تعادل‌های نش هر بازی متعارف دو میدان نبرد بازی بلاتو و با محدودیت‌های منابع غیرخطی ارائه شده است. علاوه بر این، روشی را برای

[۱] Macdonell

ساخت مجموعه‌های پیوسته از تعادل‌های هر بازی ارائه کرده است و الگوریتمی برای نشان دادن استراتژی‌های تعادل به صورت تصویری ارائه می‌کند.

مطالعات در مقاله رابرسون[۱] و همکاران بازده تعادل منحصربه‌فرد را برای همه پیکربندی‌های (متقارن و نامتقارن) سطوح مجموع نیرو بازیکنان مشخص می‌کند و همچنین مجموعه کامل توزیع‌های حاشیه‌ای تک متغیره تعادل را برای اکثر این پیکربندی‌ها مشخص می‌کند و توزیع‌های چند متغیر تعادلی جدیدی را می‌سازد. این مقاله بازی سرهنگ بلاتو را به چندین روش مهم گسترش می‌دهد. به طور خاص، دشواری بازی سرهنگ بلاتو تا کنون، تمرکز را به تنظیمات متقارن مجموع نیرو بازیکنان محدود کرده بود اما در این مقاله بازی سرهنگ بلاتو را با مشخص کردن بازده‌های تعادلی منحصربه‌فرد برای تمام پیکربندی‌های متقارن و نامتقارن کل نیرو بازیکنان و مشخص کردن مجموعه کامل توزیع‌های حاشیه‌ای تک متغیره تعادل برای اکثر این پیکربندی‌ها، گسترش می‌دهد.

در مقاله شوارتز[۲] و همکاران به حل نسخه دیگری از بازی پرداخته است که راه‌حلی برای بازی ناهمگن سرهنگ بلاتو با بازیکنان نامتقارن و مقادیر میدان نبرد ناهمگن پیشنهاد کرده است. با این فرض که تعداد میدان‌های جنگ کافی و عدم تقارن بین منابع بازیکنان وجود دارد. سپس، بازده‌های تعادلی منحصربه‌فرد و توزیع‌های حاشیه‌ای تک متغیره را همراه با اثبات وجود توزیع مشترک چند متغیره با چنین حاشیه‌هایی مشخص کرده‌اند. این پژوهش با عمومی ساختن میدان‌های جنگ در مقاله پیشین برای داشتن مقادیر متفاوت، بازی سرهنگ بلاتو را تعمیم می‌دهد (زمانی که همه میدان‌های جنگ دارای مقادیر مساوی باشند، تنظیمات به حالت اولیه باز می‌گردد). سپس، نتایج نشان می‌دهد تا انواع شناخته شده بازی سرهنگ بلاتو را به محیط‌هایی می‌توان بسط داد که میدان‌های جنگ از نظر ارزش‌هایشان بسیار متفاوت است. به عنوان مثال، امکان یافتن تعادل بازی‌های چند نفره و اتحاد بین بازیکنان درگیر در بازی‌های ناهمگن در مقابل دشمن مشترک را فراهم می‌کند.

---

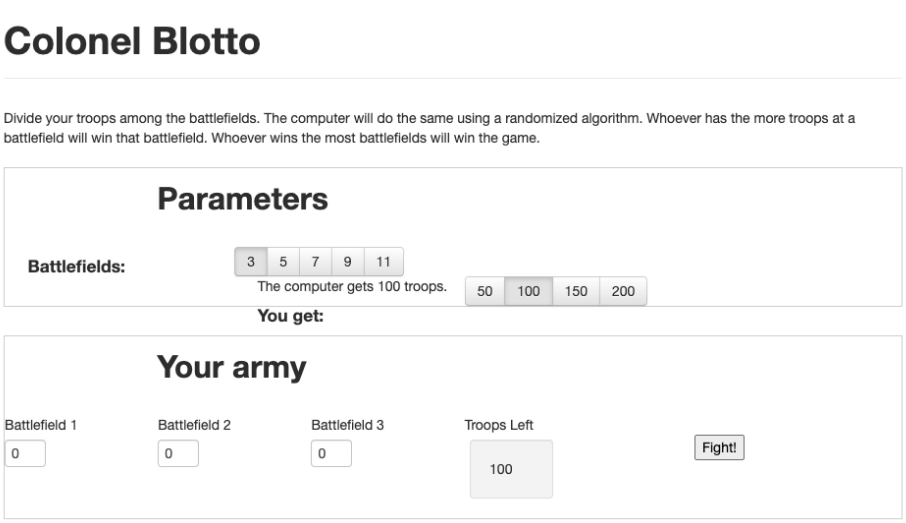
[۱] Roberson

[۲] Schwartz

در مقاله دیگری از یک جستجوی اکتشافی از تکامل بیولوژیکی، الهام گرفته است. از طریق یادگیری تقویتی چندعاملی، الگوریتم عملکرد استراتژی‌های اولیه تصادفی را ارزیابی می‌کند. بیشتر استراتژی‌های عملکردی برای ایجاد استراتژی‌های کارآمدتر ترکیب می‌شوند. جهش‌ها به الگوریتم اجازه می‌دهد تا فضای استراتژی‌های ممکن را به طور مؤثر پویش کند و تنوع گسترده‌ای از انحرافات را در نظر بگیرد. تکرار این فرآیند تا زمانی که هیچ انحراف سودآوری وجود نداشته باشد، استراتژی‌ها را بهبود می‌بخشد. این روش اجازه می‌دهد تا راه‌حل‌های بهینه برای مسائل مربوط به فضاهای استراتژی بسیار بزرگ، مانند بازی سرهنگ بلاتو را شناسایی کند (Vié, 2020).

از حل این مسئله و تبدیل آن به مسائل دیگر نیز بسیار استفاده شده است. به طور مثال برای تولید پارازیت در اینترنت اشیا مدل‌سازی‌هایی از روی این بازی صورت گرفته است (Namvar, 2016). از مدل‌سازی بازی سرهنگ بلاتو در حل مسئله انتخابات ریاست جمهوری استفاده شده است. به این صورت که کاندیدهای ریاست جمهوری بودجه مشخصی داشته و برای اختصاص آن‌ها به شهرهای مختلف به منظور تبلیغات استفاده می‌کنند. این مسئله دقیقاً مسئله بازی سرهنگ بلاتو است که در آن کاندیدها همان سرهنگ‌ها، شهرها همان میادین جنگ و بودجه همان سربازان هستند (Behnezhad, 2018).

در قسمت شبیه‌سازی این بازی تنها یک سایت اینترنتی، قسمتی از آن را پیاده‌سازی کرده است که نمای آن در شکل (۱) آمده است. این شبیه‌ساز به طور کامل مسئله سرهنگ بلاتو را مدل‌سازی نمی‌کند؛ زیرا که اصلاً الگوریتمی ندارد و از تخصیص تصادفی استفاده می‌کند. همچنین تعداد منابع انتخابی کامپیوتر ۱۰۰ عدد به صورت ثابت است که در عمل این شبیه‌ساز را بسیار محدود می‌کند. در انتخاب تعداد میدان‌های جنگ نیز تنها ۵ انتخاب و برای انتخاب منابع بازیکن تنها ۴ انتخاب از پیش تعریف‌شده وجود دارد و هیچ‌گونه آزادی عمل و مشاهده چینش‌های دیگر وجود ندارد (Blotto Simulator, 2023).



شکل (۱) – نمایی از شبیه‌ساز بر خط سرهنگ بلاتو

ابزارهای فنی

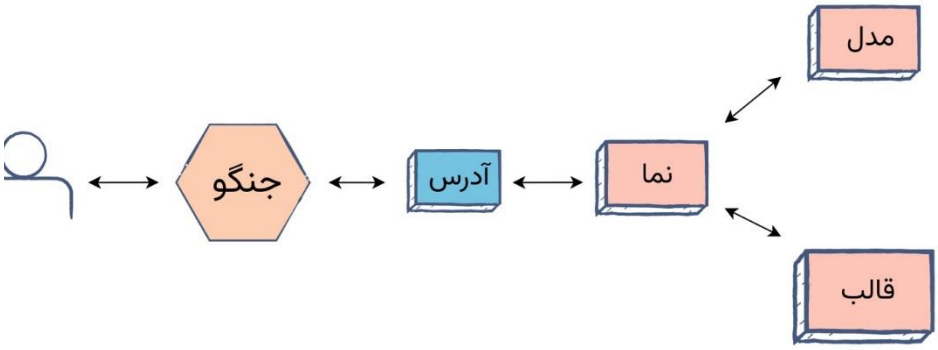
پایتون یک زبان برنامه‌نویسی پویا با قابلیت اجرای کدهای خود در بسیاری از سیستم‌ها و پلتفرم‌ها است. این زبان در دهه ۱۹۸۰ میلادی توسط خویدو فان راسم ابداع شد و به دلیل سادگی، خوانایی و پرکاربرد بودنش، به سرعت محبوبیت یافت و تبدیل به یکی از پراستفاده‌ترین زبان‌های برنامه‌نویسی شد. از دلایل اصلی محبوبیت پایتون، سادگی و خوانایی کدها و همچنین بستر پشتیبانی فراوان برای جامعه برنامه‌نویسان است (Python, 2023).

جنگو یک چارچوب نرم‌افزاری بر پایه پایتون است. جنگو از معماری مدل-نما-قالب پشتیبانی می‌کند.

مدل: مدل قرار است به عنوان رابط داده‌ها عمل کند و وظیفه نگهداری داده‌ها را بر عهده دارد.

نما: نما همان رابط کاربری است، دقیقاً آنچه در مرورگر هنگام رندر کردن یک وب‌سایت مشاهده می‌شود.

قالب: یک قالب شامل بخش‌های ثابت خروجی HTML موردنظر و همچنین برخی نحو خاص است که نحوه درج محتوای پویا را مشخص می‌کند. (Django, 2023)



شکل (۲) - معماری مدل نما قالب در ساختار درخواست جنگو

انجین‌اکس یک سرور وب و سرور پروکسی معروف و متن‌باز است که به دلیل عملکرد بالا، قابلیت مقیاس‌پذیری و قدرتمندی که دارد، شهرت گسترده‌ای پیدا کرده است. این سرور معمولاً برای میزبانی وب‌سایت‌ها، ارائه محتوای استاتیک، توزیع بار و پروکسی کردن درخواست‌ها به سرورهای برنامه‌ریزی شده استفاده می‌شود.

یکی از ویژگی‌های کلیدی انجین‌اکس، توانایی مدیریت تعداد زیادی اتصال هم‌زمان به صورت کارآمد است. این سرور از معماری ناهم‌زمان و رویداد گرا استفاده می‌کند که به آن اجازه می‌دهد درخواست‌های متعدد را به صورت هم‌زمان و بدون مصرف بیش‌ازحد منابع سیستم پردازش کند (Nginx, 2023).

روش‌شناسی پژوهش

حل مسئله سرهنگ بلاتو با الگوریتم ژنتیک

در این بخش به بررسی قسمت‌های مختلف الگوریتم ژنتیک برای حل مسئله سرهنگ بلاتو پرداخته می‌شود. در مرحله اول الگوریتم ژنتیک یعنی مرحله جمعیت اولیه، به تعداد جمعیت، کروموزوم‌هایی به صورت تصادفی ایجاد می‌شود که هر ژن نشان‌دهنده تخصیص نیرو در میدان جنگی متناظر با جایگاه ژن در کروموزوم است.

$$\text{Chromosome1} = [x_1^1, x_2^1, \dots, x_n^1]$$
$$\text{Chromosome2} = [x_1^2, x_2^2, \dots, x_n^2]$$

که در اینجا x_i^j تعداد سربازها در میدان جنگی i ام در کروموزوم j ام است و n تعداد میدان‌های جنگی است.

تابع تناسب در این الگوریتم به این صورت تعریف می‌شود که با توجه به جمعیت موجود، یک کروموزوم به نسبت بارهایی که باقی کروموزوم‌ها را می‌برد مقداردهی می‌شود. به این معنی که اگر ۱۰ روش انتخابی یا همان کروموزوم در جمعیت باشد و این انتخاب بتواند ۴ انتخاب دیگر را ببرد بنابراین تناسب آن ۴ است. در مرحله تقاطع یک عدد تصادفی به‌عنوان اندیس میانی استفاده می‌شود که نشان‌دهنده محل تقاطع است. سپس دو کروموزوم والد از نقطه تقاطع برش خورده و به یکدیگر متصل می‌شوند.

اگر c نقطه انتخاب‌شده برای تقسیم کروموزوم‌ها باشد، دو کروموزوم والد P_1 و P_2 به صورت زیر قابل بیان‌اند:

$$\mathrm{P1} = \left[\mathrm{G_{1,1}}, \mathrm{G_{1,2}}, \dots, \mathrm{G_{1,c-1}}, \mathrm{G_{1,c}}, \mathrm{G_{1,c+1}}, \dots, \mathrm{G_{1,n}}\right]$$

$$P2 = \left[G_{2,1}, G_{2,2}, \dots, G_{2,c-1}, G_{2,c}, G_{2,c+1}, \dots, G_{2,n}\right]$$

همچنین، دو کروموزوم فرزند جدید C_1 و $\mathrm{C_2}$ به صورت زیر ایجاد می‌شوند:

$$\mathrm{C1} = \left[\mathrm{G_{1,1}}, \mathrm{G_{1,2}}, \dots, \mathrm{G_{1,c-1}}, \mathrm{G_{2,c}}, \mathrm{G_{2,c+1}}, \dots, \mathrm{G_{2,n}}\right]$$

$$C2 = \left[G_{2,1}, G_{2,2}, \dots, G_{2,c-1}, G_{1,c}, G_{1,c+1}, \dots, G_{1,n}\right]$$

در این فرمول‌ها، $G_{i,j}$ نمایانگر ژن j ام در کروموزوم i ام است.

اما تفاوتی که در این قسمت با الگوریتم اصلی وجود دارد آن است که ممکن است بعد از تقاطع مقدار سربازهای یک انتخاب با تعداد سربازهای مسئله متفاوت باشد. برای مثال در یک بازی با ۴ میدان نبرد و ۶ سرباز دو والد با کروموزوم‌های ۱،۲،۲،۱ و ۳،۳،۰،۰ انتخاب می‌شود. فرض می‌شود نقطه تقاطع در وسط این دو کروموزوم باشد، بنابراین دو فرزند ۱،۲،۰،۰ و ۳،۳،۲،۱ تولید می‌شود ولی مشکلی که وجود دارد آن است که انتخاب‌های نامعتبری را تشکیل می‌دهند زیرا یک فرزند ۳ سرباز انتخاب کرده و فرزند دیگر ۹ سرباز انتخاب کرده که با ۶ سرباز در تعریف مسئله مغایرت دارد. برای حل این مشکل از یک تابع کمکی تعمیر کروموزوم استفاده شده است. این تابع به این صورت عمل می‌کند که

اگر تعداد سربازهای انتخابی بیشتر از تعداد سربازهای مسئله باشد از بیشینه ژن داخل آن کروموزوم یکی کم کرده و این کار را تا زمانی که جمع سربازهای انتخابی برابر با سربازهای مسئله باشد ادامه می‌دهد. به عنوان مثال انتخاب ۳،۳،۲،۱ نیاز به تعمیر دارد که در مرحله اول به ۲،۳،۲،۱ تبدیل می‌شود با ۸ سرباز سپس به ۲،۲،۲،۱ با ۷ سرباز و در نهایت به ۱،۲،۲،۱ با ۶ سرباز تبدیل می‌شود که سربازهای تعریف‌شده در مسئله است. حال اگر مجموع سربازهای انتخاب‌شده کمتر از سربازهای تعریف‌شده در مسئله باشد تا زمانی که سربازهای انتخابی برابر با سربازهای تعریف‌شده در مسئله باشد، یک سرباز به کمینه ژن در کروموزوم اضافه می‌شود. پس انتخاب ۱،۲،۰،۰ در مرحله اول به ۱،۲،۱،۰ سپس به ۱،۲،۱،۱ و در نهایت به ۲،۲،۱،۱ تبدیل می‌شود.

در مرحله بعد تابع جهش مسئولیت تغییر ژن‌های داخل کروموزوم را با احتمالی دارد. در مسئله سرهنگ بلاتو تغییر یک ژن باعث می‌شود که تعداد سربازهای انتخابی نادرست باشد و مطابق سربازهای تعریف‌شده در مسئله نباشد. تابع جهش برای این مسئله بدین گونه تغییریافته که مقدار دو ژن باهم در کروموزوم تغییر می‌کند. برای تغییر ابتدا یک عدد تصادفی انتخاب می‌شود که اگر از احتمال جهش کمتر بود با این کار یک کروموزوم جهش پیدا می‌کند و انتخاب‌های آن با تعداد سربازها در مسئله تعریف‌شده در تناقض نیست. در شکل (۳) مراحل انجام جهش به صورت شبه‌کد بیان شده است و در آن MUTATION_PROBABILITY احتمال انجام جهش است.

```
function MUTATE(chromosome)
    if random() < MUTATION_PROBABILITY then
        random_index_1 ← random.randint(0, chromosome_length − 1)
        random_index_2 ← random.randint(0, chromosome_length − 1)
        while random_index_1 == random_index_2 do
            random_index_2 ← random.randint(0, chromosome_length − 1)
        end while
        swap(chromosome[random_index_1], chromosome[random_index_2])
    end if
end function
```

شکل (۳) - شبه‌کد جهش در الگوریتم ژنتیک برای مسئله سرهنگ بلات

روند کلی الگوریتم ژنتیک به صورت شبه‌کد در شکل (۴) آمده است.

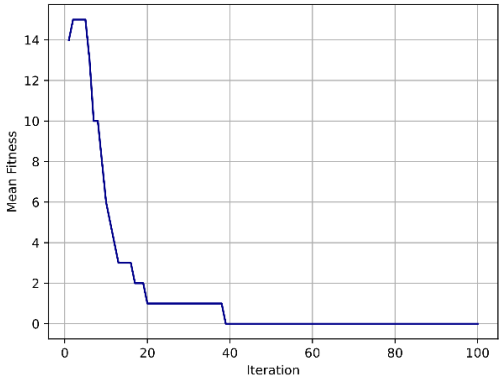
```
function RUN
    while generation < MAX_ITERATION do
        population.sort(key = fitness, reverse = True)
        if fitness(population[0]) > best_fitness then
            best ← population[0]
            best_fitness ← fitness(population[0])
        end if
        for i ← 0to⌊len(population)/2⌋ do
            parent1 ← population[0]
            parent2 ← population[1]
            child1, child2 ← crossover(parent1, parent2)
            mutate(child1)
            mutate(child2)
            children.append(child1)
            children.append(child2)
        end for
        population ← children
        generation ← generation + 1
    end while
    return best
end function
```

شکل (۴) - شبه‌کد روند کلی الگوریتم ژنتیک برای حل مسئله سرهنگ بلاتو

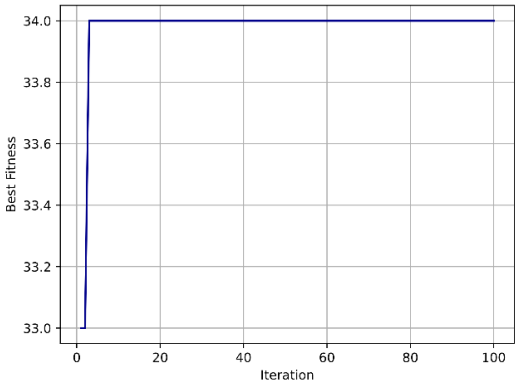
به‌منظور بهینه‌سازی مقادیر و رسیدن به‌سرعت بهتر الگوریتم، اجراهای مختلف از آن در شرایط یکسان گرفته شد تا بررسی شود با تغییر مقادیر و همچنین توابع جهش، در کدام یک به نتیجه بهتر و همچنین هم‌گرایی به جواب بهینه ممکن می‌شود. به همین دلیل اجرای الگوریتم بر روی ۱۰۰ تکرار تنظیم شده است. در هر تکرار تناسب بهترین فرد محاسبه و همچنین میانگین تناسب افراد نیز گزارش شده است. هدف بیشینه کردن بهترین فرد جامعه در حالی است که نرخ نوسان در میانگین تناسب افراد پایین و مقدار آن در بالاترین حد خود باشد.

نخستین آزمایش با همان الگوریتم اولیه ژنتیک با ۱۰۰ تکرار با مقدار جمعیت ۵۰ و احتمال جهش ۰٫۸ اجرا شده است. این انتخاب از روش بهینه‌سازی پارامتر جستجو استفاده شده که با آزمایش مقادیر مختلف برای پارامترهای جمعیت و درصد احتمال جهش و مشاهده نتایج حاصل‌شده، بهترین پارامترها را گزارش می‌دهد (Montero, 2014). شکل (۵) و شکل (۶) با استفاده از برنامه matplotlib در زبان پایتون تهیه‌شده که به ترتیب نشان‌دهنده میانگین تناسب افراد و بهترین مقدار تناسب در هر تکرار است. همان‌طور که مشخص است بعد از هر تکرار میانگین تناسب کاهش پیدا می‌کند یعنی

افراد با تناسب، از جمعیت جدا می‌شوند و جای خود را به فرزندان خود می‌دهند که احتمال دارد افراد شایسته‌تری نباشند. همچنین از شکل (۶) نیز این‌گونه برداشت می‌شود که بعد از مقداردهی اولیه افراد با بهترین تناسب از دست می‌روند و اجرای بیشتر تکرارها نتیجه را دگرگون نخواهد کرد.



شکل (۵) میانگین مقدار تناسب در هر تکرار برای نخستین آزمایش

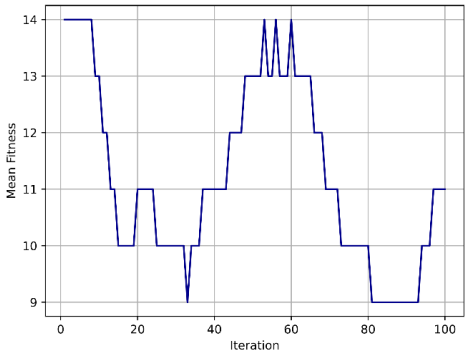


شکل (۶) بهترین مقدار تناسب در هر تکرار برای نخستین آزمایش

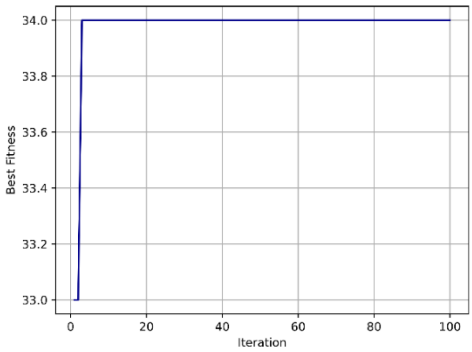
در الگوریتم ژنتیک روش‌های مختلفی برای انتخاب جمعیت در پایان هر تکرار وجود دارد. مشکل آزمایش نخست آن بود که جمعیت اصلی رفته‌رفته جای خود را به افراد شایسته‌تر نمی‌داد. مشکل اصلی آن است که افراد شایسته نباید از جمعیت حذف شوند بلکه باید انتخابی میان افراد صورت بگیرد. روشی که در این آزمایش از آن استفاده‌شده روش حذف بدترین افراد است. به صورتی که فرزندان تولیدشده دیگر جای والدین خود را نمی‌گیرند

بلکه جای افراد جامعه با کمترین شایستگی را می‌گیرند. علاوه بر این در هر تکرار تنها دو عدد از والدین که بیشترین تناسب را دارند انتخاب می‌شوند و از جمعیت نیز حذف نمی‌شوند.

با توجه به شکل (۷) و شکل (۸) مقدار میانگین تناسب در ابتدا ۱۴ است که پس از حدود ۶۰ تکرار باز به این مقدار می‌رسد و سپس دوباره کاهش پیدا می‌کند. بهترین تناسب نیز از ابتدا بهترین مقدار خود را پیدا می‌کند و بنابراین پیشرفتی در تکرارها ایجاد نمی‌شود. یکی از مشکلاتی که در این روش از انتخاب وجود دارد انتخاب بهترین افراد در هر تکرار است. توجیه این رفتار بدین گونه است که به دلیل حذف کردن افراد یا بازی‌هایی که از باقی حالت‌ها می‌بازند فضای حالت را به طور کلی به سمت بازی‌های برنده پیش می‌برد. در حالت کلی این روش به بهبود جواب در الگوریتم ژنتیک کمک می‌کند اما در تعریف تابع تناسب برای مسئله سرهنگ بلاتو، مقدار تناسب به جمعیت بسیار وابسته است. پس برای داشتن جواب بهینه، داشتن میانگین تناسب ثابت بسیار کمک می‌کند زیرا که جمعیت با گوناگونی متفاوت مقدار حالت‌های بیشتری بر اساس بهترین انتخاب را آزمایش می‌کند؛ بنابراین باید از روش‌هایی که دگرگونی را در جامعه حفظ می‌کند استفاده کرد.



شکل (۷) میانگین مقدار تناسب در هر تکرار برای دومین آزمایش



شکل (۸) بهترین مقدار تناسب در هر تکرار برای دومین آزمایش

همان‌طوری که در آزمایش دوم نتیجه‌گیری شد انتخاب جمعیت بعدی نیاز به روشی دارد که گوناگونی را در جمعیت حفظ کند. یکی از این روش‌ها انتخاب به روش نمونه‌برداری جهانی تصادفی است. نمونه‌برداری جهانی تصادفی[۱] تکنیکی است که در الگوریتم‌های ژنتیک برای انتخاب راه‌حل‌های بالقوه مفید برای بازترکیبی استفاده می‌شود که توسط جیمز بیکر معرفی شد. نمونه‌برداری جهانی تصادفی یک الگوریتم نمونه‌برداری تک فازی با حداقل گسترش است و از N نشانگر با فاصله مساوی استفاده می‌کند که N تعداد انتخاب‌های موردنیاز است که در مسئله ژنتیک برای انتخاب والد عدد ۲ است. در این انتخاب، جمعیت به طور تصادفی مخلوط می‌شود و یک عدد تصادفی در محدوده ۰ تا $\frac{\Sigma\, Fitness}{N}$ تولید می‌شود سپس N افراد با ایجاد N نشانگر با فاصله ۱، $[ptr, ptr+1, ptr+2, \ldots, N]$ انتخاب می‌شوند. درواقع انتخاب‌ها بر اساس موقعیت نشان‌گرها شکل می‌گیرد. از آنجایی که افراد کاملاً موقعیت خود را در جمعیت انتخاب می‌کنند، نمونه‌برداری جهانی تصادفی دارای سوگیری صفر است. بر اساس نمونه‌برداری جهانی تصادفی روند کلی الگوریتم به شکلی تغییر می‌کند که تابع make_wheel مسئولیت تولید بخش‌بندی دایره‌ای را بر اساس نسبت تناسب خود به تناسب کل دارد. تابع select که

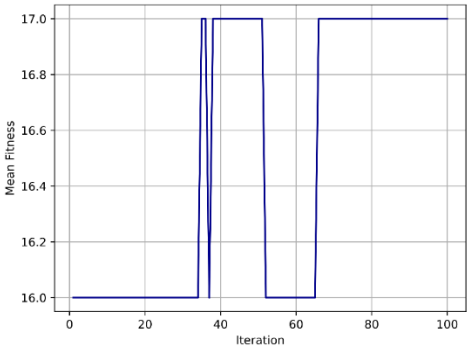
[۱] Stochastic Universal Sampling

شبه‌کد آن در شکل (۹) آمده است نیز مسئولیت ساخت و فراخوانی تابع ساخت چرخ را داشته و سپس بر اساس تعداد نشانگرها آن‌ها را مقداردهی می‌کند تا مشخص شود که کدام والد در کجای جمعیت انتخاب شده است (Sourabh, 2021). لازم به توجه است که نقطه شروع انتخاب نشانگر به صورت تصادفی انتخاب‌شده و باقی نشانگرها بر اساس تعداد انتخاب می‌شوند. در الگوریتم انتخاب از تابع جستجوی دودویی استفاده‌شده که روشی خوب برای جستجوی خطی در یک لیست است. جستجوی خطی یا متوالی، راهی برای یافتن یک عنصر در لیست با جستجوی متوالی عناصر تا جستجو موفقیت‌آمیز است.

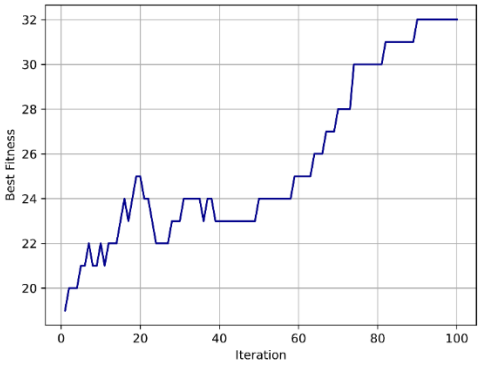
```
function SELECT(number)
    wheel ← make_wheel()
  step ← 1.0/number
  parents ← []
  r ← random()
  parents.append(binary_search(wheel, r))
  while len(parents) < number do
      r ← r + step
      if r > 1 then
          r = r module  1
      end if
      parents.append(binary_search(wheel, r))
  end while
  return parents
nd function
```

شکل (۹) تابع انتخاب در نمونه‌برداری جهانی تصادفی

در پایان اجرا بر اساس سومین آزمایش و انتخاب نمونه‌برداری جهانی تصادفی نتایج در شکل (۱۰) و شکل (۱۱) قابل مشاهده است. میانگین تناسب به صورت یکسان بین ۱۶ و ۱۷ است و نوسان شدیدی ندارد که نتیجه می‌شود میانگین تناسب افراد در یک محدوده است. همچنین به دلیل اینکه در شکل (۱۰) بهترین تناسب بالا می‌رود نشان از بهبود نتایج در تکرارهای الگوریتم است.



شکل (۱۰) میانگین مقدار تناسب در هر تکرار برای سومین آزمایش



شکل (۱۱) بهترین مقدار تناسب در هر تکرار برای سومین آزمایش

حل مسئله سرهنگ بلاتو با الگوریتم بروت-فورس

در این بخش به حل مسئله سرهنگ بلاتو با بررسی تمام فضای حالت پرداخته شده است. در این روش حل، تمام فضای حالت ساخته‌شده و سپس هر انتخاب با تمام انتخاب‌های دیگر بازی می‌کند. در این الگوریتم به دلیل آنکه هر انتخاب با تمام انتخاب‌های دیگر مقایسه می‌شود از مرتبه زمانی $O(n^2)$ است. با این حال که تعداد فضای حالت در مقادیر بالای تعداد سرباز و میدان‌های جنگی بالا می‌رود محاسبه تمام فضای حالت نیز بسیار زمان‌بر خواهد بود. با این حال که برای ۷۰ سرباز جنگی در ۶ میدان نبرد حدود ۱۷ میلیون حالت وجود دارد. برای پیدا کردن جواب با این روش نیاز به

۷,۸۸۶,۵۴۳,۱۷۲,۱۰۰ پردازش نیاز است. رایانه‌های امروزی حدود ۱۰۰۰۰۰۰ پردازش را در یک ثانیه انجام می‌دهد؛ بنابراین برای پیدا کردن راه‌حل با این روش نیاز به حدود ۲۹۷۸۸۶۵۴۳ ثانیه یعنی حدود ۹ سال نیاز است که زمان بسیار زیادی با این مقادیر ورودی مسئله است. برای حل مسئله ابتدا تمام فضای حالت ایجاد می‌شود. تابع دیگری که استفاده شده است تابع is_first_win است که برای مشخص کردن برنده دو حالت استفاده می‌شود. اگر اولین حالت نسبت به حالت دوم برنده باشد مقدار درست برگردانده می‌شود و اگر مساوی یا بازنده باشد مقدار نادرست برگردانده می‌شود. شبه‌کد الگوریتم بروت-فورس در شکل (۱۲) آمده است.

```
function GET_WINS(all_combination)
    for i in all_combination do
        wins ← 0
     for j in all_combination do
         if is_first_win(i, j) then
             wins ← wins + 1
         end if
         if wins in results then
             results[wins].append(i)
         else
             results[wins] ← [i]
         end if
     end for
  end for
  return max(results)
end function
```

شکل (۱۲) شبه‌کد روند اجرای کلی الگوریتم بروت-فورس

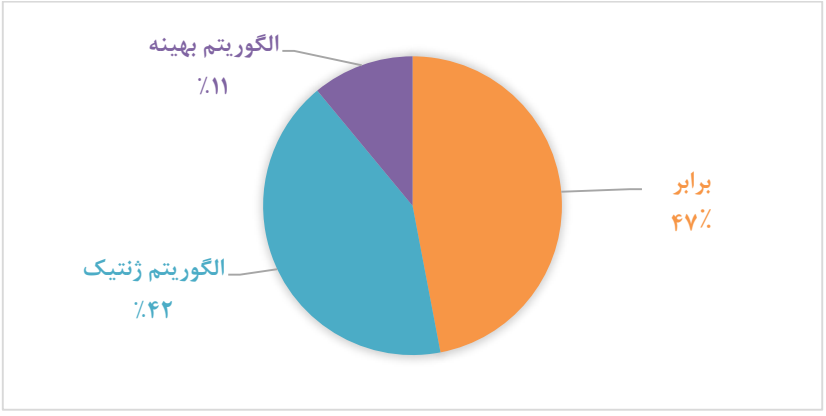
تجزیه و تحلیل داده‌ها

در ابتدا الگوریتم بروت-فروس برای تعداد میدان‌های نبرد و نیروهای محدود اجرا شد. به نظیر آن الگوریتم ژنتیک برای همان میدان‌های نبرد و نیروها اجرا شد و نتایج این دو الگوریتم با یکدیگر مقایسه شد. این اجرا برای هر دو الگوریتم در شرایط یکسان انجام شد. هر دو این الگوریتم‌ها به زبان پایتون پیاده‌سازی شده و بر روی رایانه‌ای با ۸ هسته پردازنده از نوع Intel Core i7 و ۸ گیگابایت حافظه انجام شد. این دو الگوریتم از منظر زمان اجرا و برد و باخت و یا مساوی هرکدام مورد بررسی قرار گرفت. در جدول (۲) نمونه‌هایی از ۱۰۰ اجرا آورده شده است.

جدول (۲) نمونه‌هایی از نتیجه مقایسه بین دو الگوریتم ژنتیک و بروت-فورس

میدان نبرد	نیروها	زمان اجرا الگوریتم بروت-فورس	زمان اجرا الگوریتم ژنتیک	برنده
۶	۹	۱ ثانیه	کمتر از ۱ ثانیه	ژنتیک
۶	۱۰	۳ ثانیه	کمتر از ۱ ثانیه	بهینه
۶	۲۰	۱۸ دقیقه و ۵۹ ثانیه	کمتر از ۱ ثانیه	ژنتیک
۶	۲۱	۲۸ دقیقه و ۱۷ ثانیه	کمتر از ۱ ثانیه	ژنتیک
۶	۲۲	۴۳ دقیقه و ۴ ثانیه	کمتر از ۱ ثانیه	برابر
۶	۲۳	۱ ساعت و ۳ دقیقه و ۴ ثانیه	کمتر از ۱ ثانیه	برابر
۶	۲۴	۱ ساعت و ۳۳ دقیقه و ۲۹ ثانیه	کمتر از ۱ ثانیه	برابر
۶	۲۵	۲ ساعت و ۱۴ دقیقه و ۲۹ ثانیه	کمتر از ۱ ثانیه	برابر

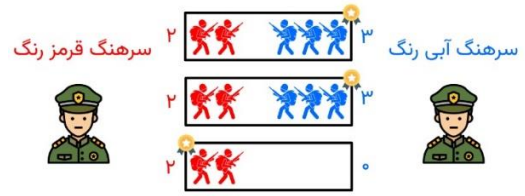
همان‌طوری که مشاهده می‌شود ادامه الگوریتم بروت-فورس برای میدان‌های نبرد و نیروهای بیشتر بسیار زمان‌بر است و برای الگوریتم ژنتیک تعداد میدان‌های و نیروها تفاوتی نمی‌کند و نتایج در چندین میلی‌ثانیه محاسبه می‌شود. پر واضح است که در این آزمایش‌ها برنده‌ی زمانی، با اختلاف بسیار الگوریتم ژنتیک است. با این حال باید دید نتایج الگوریتم ژنتیک نیز قابل استناد است؟ به این منظور در شکل (۱۳) این ۱۰۰ آزمایش بررسی شده است.



شکل (۱۳) درصد برد در الگوریتم ژنتیک و بروت-فروس در بازی سرهنگ بلاتو

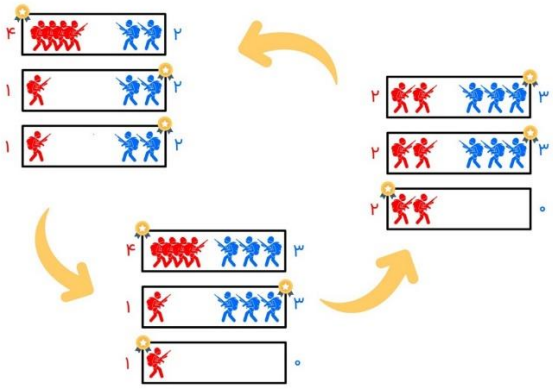
همان‌طوری که از نتایج پیداست در ۴۷ درصد مواقع این دو الگوریتم نتایج یکسانی را تولید می‌کنند. این بدین معناست که الگوریتم ژنتیک توانسته در ۴۷ درصد مواقع همانند الگوریتم بروت-فورس پاسخ دهد؛ اما تفاوتی که آشکار است زمان پاسخ‌دهی سریع

الگوریتم ژنتیک است. از طرفی الگوریتم ژنتیک در ۴۲ درصد مواقع توانسته بهتر از الگوریتم بروت-فورس پاسخ بدهد. تنها درصد کمی یعنی ۱۱ درصد الگوریتم بروت-فورس موفق به شکست الگوریتم ژنتیک شده که در این مواقع الگوریتم ژنتیک در موقعیت‌های بیشینه محلی افتاده و جهش نسل‌ها کمکی به بهتر شدن آن نداشته است. به منظور شناخت و بررسی الگوریتم بروت-فورس در زمان‌هایی که الگوریتم ژنتیک بازنده است، نمونه‌ای مورد بررسی قرار می‌گیرد تا نشان دهد که جواب الگوریتم ژنتیک چه مقدار خوب است. به عنوان مثال در موقعیتی که ۳ میدان نبرد و ۲۹ نیرو وجود دارد الگوریتم بروت-فورس انتخاب [۹,۱۰,۱۰] را انجام می‌دهد. الگوریتم ژنتیک نیز انتخاب [۷,۸,۱۴] را انجام می‌دهد. همان‌طوری که مشخص است الگوریتم ژنتیک بازنده است اما این انتخاب به چه میزان بد است؟ با بررسی در تمامی حالات این مثال، الگوریتم بروت-فورس با اختصاص این مقادیر در ۲۸۰ حالت از حالت‌های کلی امکان برد را دارد. الگوریتم ژنتیک در مقایسه با تعداد حالات کلی در ۲۶۶ حالت از حالت‌های کلی امکان برد را دارد. با توضیحات ذکرشده الگوریتم ژنتیک در مقایسه با الگوریتم بروت-فورس چندان ضعیف عمل نکرده است. سوال مهمی که وجود دارد آن است که چگونه الگوریتم ژنتیک بهتر از الگوریتم بروت-فورس عمل می‌کند؟ آیا الگوریتم بروت-فورس، بهترین جواب را خروجی نمی‌دهد؟ در بسیاری از مواقع مسئله سرهنگ بلاتو جواب بهینه ندارد و تنها می‌توان جوابی را پیشنهاد داد که به صورت احتمالاتی از باقی جواب‌ها بهتر باشد؛ بنابراین الگوریتم بروت-فورس به تنهایی نمی‌تواند در بیشتر اوقات جوابی تولید کند که همیشه برنده باشد. برای مثال دو سرهنگ به رنگ‌های قرمز و آبی در ۳ میدان نبرد با هم در نبرد هستند. هر کدام از آن‌ها ۶ نیرو در اختیار دارند و باید این نیروها را به سه میدان نبرد تخصیص دهند. نحوه این تخصیص در شکل (۱۴) نشان داده شده است.



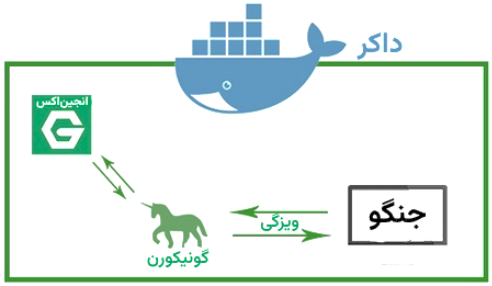
شکل (۱۴) نمونه‌ای از انتخاب‌ها برای ۳ میدان نبرد و ۶ نیرو

سرهنگ آبی با اختصاص دادن ۳ نیرو در میدان اول و دوم توانست برنده آن دو میدان شود. سرهنگ قرمز با تخصیص ۲ نیرو در هر میدان تنها توانست میدان نبرد سوم را برنده شود و در نهایت سرهنگ آبی با برنده شدن دو میدان نبرد، برنده کل بازی شد؛ اما در این مثال جواب بهینه‌ای وجود ندارد و برای هر جوابی یک راه‌حل بهتری وجود دارد. در ۳ میدان نبرد با ۶ نیرو یک استراتژی برد ممکن است در مقابل استراتژی دیگر برنده نباشد. با توجه به این موضوع الگوریتم ژنتیک با ذات تصادفی خود و همچنین بررسی مقداری از جمعیت (نه کل حالت‌ها) به جواب‌های برنده‌تری نسبت به حالت اصلی می‌رسد. نمونه این دور بازگشتی در شکل (۱۵) آمده است.



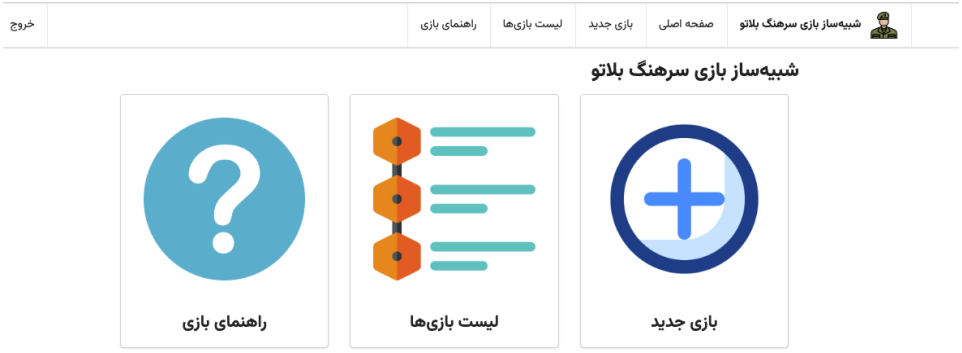
شکل (۱۵) نمونه‌ای از دور در راه‌حل‌های برنده در ۳ میدان نبرد با ۶ نیرو

شبیه‌ساز بازی سرهنگ بلاتو با زبان پایتون و چارچوب نرم‌افزاری جنگو به همراه ویزگی و انجین‌اکس به همراه داکر پیاده شده است، معماری مطابق شکل (۱۶) دارد.



شکل (۱۶) معماری شبیه‌ساز مسئله سرهنگ بلاتو

این شبیه‌ساز قابلیت ساخت و تغییر در کاربران و ایجاد نام کاربری و رمزعبور را دارد. برای کنترل مرکزی کاربران از سیستم مدیریت کاربران جنگو استفاده‌شده که فرایند ساخت و تغییر کاربران را تسهیل می‌بخشد. قسمت راهنمای بازی شامل چهار صفحه توضیحات در مورد پیشینه مسئله سرهنگ بلاتو، نمونه‌ای از بازی سرهنگ بلاتو، توضیحات در مورد شبیه‌ساز و در نهایت نکات تکمیلی است. صفحه لیست بازی حاوی لیست تمام بازی‌هایی است که کاربر انجام داده و یا به عنوان حریف در آن شرکت دارد. نمای صفحه اصلی شبیه‌ساز در شکل (۱۷) آمده است.



شکل (۱۷) صفحه اصلی شبیه‌ساز بازی سرهنگ بلاتو

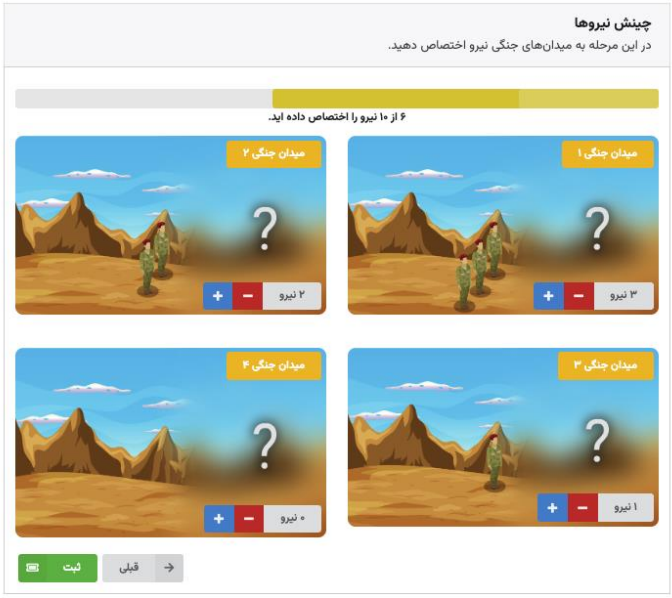
صفحه بازی جدید که در شکل (۱۸) آمده است به کاربر این امکان را می‌دهد تا یک بازی جدید در شبیه‌ساز سرهنگ بلاتو تعریف کند. این صفحه از سه مرحله تشکیل شده است. در مرحله نخست کاربر تعداد میدان‌های جنگی، تعداد نیروهای خود و تعداد نیروهای حریف را مشخص می‌کند.

شکل (۱۸) نمایی از مرحله اول ساخت بازی در شبیه‌ساز بازی سرهنگ بلاتو

در مرحله دوم که در شکل (۱۹) قابل مشاهده است، حریف مقابل انتخاب می‌شود. حریف می‌تواند کاربر دیگری از سیستم یا الگوریتم ژنتیک و یا تخصیص تصادفی باشد. تخصیص تصادفی از روش خاصی پیروی نمی‌کند و تنها به صورت تصادفی نیروها را به میدان‌ها تخصیص می‌دهد. پرواضح است که این روش مطلوبی نیست و در موارد بسیاری ناکارآمد است اما ممکن است به صورت شانسی در یک بازی تخصیص خوبی را تصادفاً انتخاب کند. در الگوریتم ژنتیک با بررسی روش‌های برد یک مسئله و بهبود آن توسط جهش جمعیتی یک استراتژی برای برد با نیروهای خود در میدان‌های نبرد پیدا می‌کند. همچنین می‌توان با کاربر دیگری در شبیه‌ساز بازی کرد و درخواست بازی را برای فرد ارسال کرد.

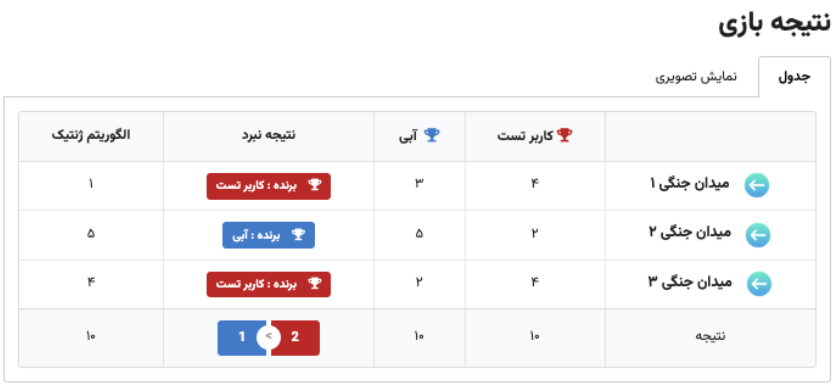
شکل (۱۹) نمایی از مرحله دوم صفحه ساخت بازی در شبیه‌ساز سرهنگ بلاتو

در مرحله سوم با توجه به انتخاب‌های مرحله اول به صورت گرافیکی میدان‌های جنگی نمایش داده می‌شود و کاربر باید به میدان‌های جنگی نیرو اختصاص دهد. نمایی از این صفحه در شکل (۲۰) آمده است. این صفحه حاوی یک نوار پیشرفت از اختصاص‌هاست تا کاربر را مطلع کند که چه میزان نیرو اختصاص داده و چه میزان نیرو در اختیار دارد. در ادامه نیز میدان‌های جنگی با اختصاص نیروهایشان موجود است.



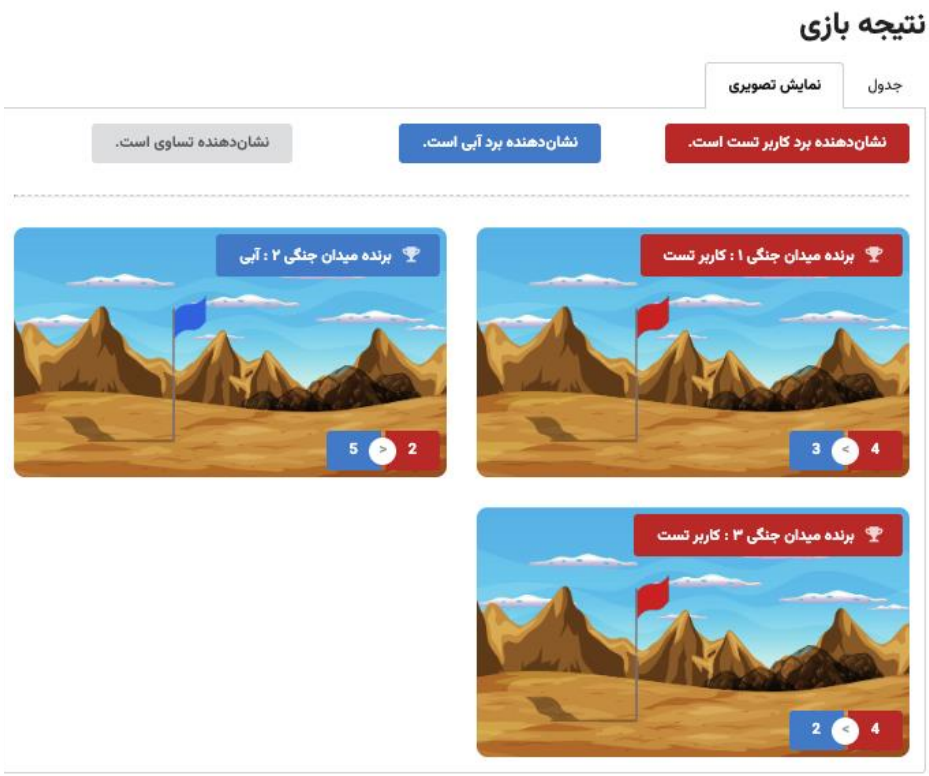
شکل (۲۰) نمایی از مرحله سوم ساخت بازی جدید و اختصاص نیروها

پس از انجام هر بازی می‌توان مجدد نتایج آن را از صفحه لیست بازی مشاهده کرد. این صفحه دو نمایش جدولی همچون شکل (۲۱) و تصویری همانند شکل (۲۲) دارد.



	کاربر تست	آبی	نتیجه نبرد	الگوریتم ژنتیک
میدان جنگی ۱	۴	۳	برنده : کاربر تست	۱
میدان جنگی ۲	۲	۵	برنده : آبی	۵
میدان جنگی ۳	۴	۲	برنده : کاربر تست	۴
نتیجه	۱۰	۱۰	1 < 2	۱۰

شکل (۲۱) نتیجه بازی به صورت جدولی



شکل (۲۲) نتیجه بازی به صورت تصویری

نتیجه‌گیری و پیشنهادها

در این تحقیق سعی شد تا روشی برای حل مسئله سرهنگ بلاتو پیدا شود و سپس راه‌حل مورد نظر با راه‌حل بهینه مقایسه و در انتها شبیه‌سازی برای حل این مسئله طراحی و پیاده‌سازی شود. حل این مسئله با الگوریتم ژنتیک که الگوریتمی فراابتکاری است انجام شد. الگوریتم دیگری با عنوان الگوریتم بروت-فورس نیز معرفی شد و با مقایسه با الگوریتم ژنتیک صحت و کارایی الگوریتم ژنتیک ارزیابی شد. در ادامه شبیه‌ساز مسئله سرهنگ بلاتو با به‌روزترین زبان و چارچوبّ برنامه‌نویسی یعنی زبان پایتون و چارچوب نرم‌افزاری جنگو پیاده‌سازی شد. این شبیه‌ساز به افسران کمک می‌کند تا در شرایط تخصیص منابع تصمیمات بهتر و چابک‌تری اخذ نمایند. یکی از محدودیت‌های موجود زمان بسیار زیاد الگوریتم بروت-فورس است. در واقع به دلیل NP-Hard بودن مسئله سرهنگ بلاتو ایجاد تمامی حالات و حل آن در میدان‌های جنگی و نیروها با تعداد بالا بسیار زمان‌بر و عملاً

نشدنی است به همین دلیل در این مقایسه‌ها تنها به بررسی ۱۰۰ حالت پرداخته شده است. نتایج حاصل به صورتی است که الگوریتم ژنتیک در ۴۲ درصد بهتر از الگوریتم بروت-فورس، در ۱۱ درصد عملکرد بدتر و در ۴۷ درصد عملکرد مشابه را دارد. به منظور پیشنهادها برای ادامه‌ی این مقاله می‌توان از سلسله الگوریتم‌های دیگر فراابتکاری استفاده کرد و نتیجه‌ها را با الگوریتم ژنتیک مقایسه کرد و این شبیه‌ساز را با الگوریتم‌های دیگر کامل کرد. پیشنهاد دیگر تعمیم مسئله سرهنگ بلاتو است. تعمیم مسئله سرهنگ بلاتو به این صورت است که به جای اختصاص یک نیروی خاص با قدرت واحد، چندین نیرو در اختیار دو طرف باشد. به عنوان مثال علاوه بر یک نیرو واحد چندین نیروی متفاوت با قدرت‌های گوناگون وجود داشته باشد.

منابع

- امیری، کامبیز؛ بیگدلی، حمید. (۱۳۹۸). بازی‌های سرهنگ بلاتو و سرهنگ ریچارد، دوفصلنامه بازی جنگ، ۲ (۴)، ۷۸-۹۴.
- بیگدلی، حمید و موسی‌زاده، محمد. (۱۴۰۲). فرآیند تحلیل سلسله مراتبی در مدل‌سازی و حل بازی‌های ماتریسی در محیط نوتروسوفیک و کاربرد آن در مسائل نظامی .*علوم و فنون نظامی*. ۱۹ (۶۴):۳۳-۵.
- فرهنگ، سجاد و آروند، حمید. (۱۴۰۲). تأثیر کاربرد فناوری‌های شبیه‌سازی دیجیتال بر یادگیری شناختی اجتماعی رفتار اخلاقی در سازمان‌های نظامی .*آینده‌پژوهی دفاعی*. ۸ (۲۹): ۱۶۰-۱۳۵.
- Blotto Simulator. (2023). Retrieved from *http://success-equation.com/blotto.html*
- Behnezhad, S., Blum, A., Derakhshan, M., HajiAghayi, M., Mahdian, M., Papadimitriou, C. H. Stark, P. B. (2018). *From Battlefields to Elections: Winning Strategies of Blotto and Auditing Games*. Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms, 2291–2310.
- Boussaid, I., Lepagnot, J., & Siarry, P. (2013). *A survey on optimization metaheuristics*. Information Sciences, 237, 82–117.
- Django. (2023). Retrieved from *https://www.djangoproject.com*
- Dorigo, M., Birattari, M., & Stutzle, T. (2006). *Ant colony optimization*. IEEE Computational Intelligence Magazine, 1(4), 28–39.
- Heule, M. J. H., & Kullmann, O. (2017). *The Science of Brute Force*. Commun. ACM, 60(8), 70–79.

- Kennedy, J., & Eberhart, R. (1995). *Particle swarm optimization*. Proceedings of ICNN'95 - International Conference on Neural Networks, 4, 1942–1948 vol.4.
- Kumar, V., Chhabra, J. K., & Kumar, D. (2014). *Parameter adaptive harmony search algorithm for unimodal and multimodal optimization problems*. Journal of Computational Science, 5(2), 144–155.
- Kumar, V., & Kumar, D. (2017). *An astrophysics-inspired Grey wolf algorithm for numerical optimization and its application to engineering design problems*. Advances in Engineering Software 112, 231–254.
- Macdonell, S. T., & Mastronardi, N. (2015). *Waging simple wars: a complete characterization of two-battlefield Blotto equilibria*. Economic Theory, 58(1), 183–216.
- Michalewicz (1992). *Genetic algorithms + data structures = evolution programs*. Springer-Verlag.
- Montero, E., Riff, M.-C., & Neveu, B. (2014). *A beginner's guide to tuning methods*. Applied Soft Computing, 17, 39–51.
- Namvar, N., Saad, W., Bahadori, N., & Kelley, B. (2016). *Jamming in the Internet of Things: A Game-Theoretic Perspective*. 2016 IEEE Global Communications Conference (GLOBECOM), 1–6.
- NginX. (2023). Retrieved from *https://www.nginx.com/*
- Python. (2023). Retrieved from *https://www.python.org/*
- Roberson (2006). The colonel blotto game. Economic Theory, 29, 1-24.
- Schwartz, G., Loiseau, P., & Sastry, S. S. (2014). *The heterogeneous Colonel Blotto game*. 2014 7th International Conference on NETwork Games, COntrol and OPtimization (NetGCoop), 232–238.
- Sourabh K., Chauhan, S. S., & Kumar, V. (2021). *A review on genetic algorithm: past, present, and future*. Multimedia Tools and Applications, 80(5), 8091–8126.
- Vié, A. (2020). A genetic algorithm approach to asymmetrical blotto games with heterogeneous valuations. SSRN Electronic Journal.