به نام خدا

دانشگاه تهران

دانشکده مهندسی برق و کامپیوتر

پروژه نهایی درس حسابگری زیستی

نام و نام خانوادگی	سارا رستمی
شماره دانشجویی	۸۱۰۱۰۰۳۵۵

فهرست

شکل‌ها

پرسش ۱. الگوریتم Game of Life

Game of Life یک شبکه متعامد بی نهایت و دو بعدی از سلولهای مربعی است که هر سلول یکی از دو حالت ممکن زنده یا مرده را دارد. هر سلول با هشت همسایه خود، که سلول‌هایی هستند که به صورت افقی، عمودی یا مورب مجاور هستند، در تعامل است. شکل زیر گیف است (برای مشاهده حالت متحرک فایل ورد را چک کنید).

شکل ۱ – **Gasper's Glider**

(نمونه‌ای از الگوی **Glider**ها که به خانواده‌ی **spaceships** تعلق دارند)

در هر مرحله از زمان ، transitionهای زیر رخ می دهد:

- هر سلول زنده با کمتر از دو همسایه زنده می میرد، گویی به خاطر کمبود جمعیت.
- هر سلول زنده با دو یا سه همسایه زنده در نسل بعدی زندگی می کند.
- هر سلول زنده با بیش از سه همسایه زنده می میرد، گویی به خاطر جمعیت بیش از حد.
- هر سلول مرده با دقیقاً سه همسایه زنده به یک سلول زنده تبدیل می شود، گویی با تولید مثل.

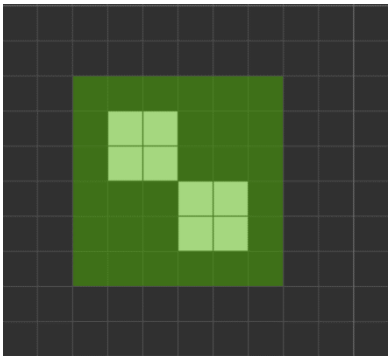
این قوانین به نوعی رفتار اتوماتا را با زندگی واقعی مقایسه می کنند.

انواع مختلفی از الگوهای در Game of Life رخ می دهد که مطابق رفتار آنها طبقه بندی می شوند. انواع الگوهای متداول عبارتند از: still lifes، که از یک نسل به نسل دیگر تغییر نمی کند. oscillators ، که پس از تعداد محدودی از نسل‌ها به حالت اولیه خود باز می گردند. و spaceships، که خود را در سراسر گسترش میدهند.

برای اجرای این الگوریتم در نرم‌افزار Golly در قسمت Set Rule، الگوریتم QuickLife و قاعده‌ی Life را انتخاب می‌کنیم. سپس هر یک از الگوها را با interface ادیت نرم‌افزار وارد می‌کنیم و با زدن گزینه‌ی Next Generation از سربرگ Control نسل بعدی را مشاهده می‌کنیم.

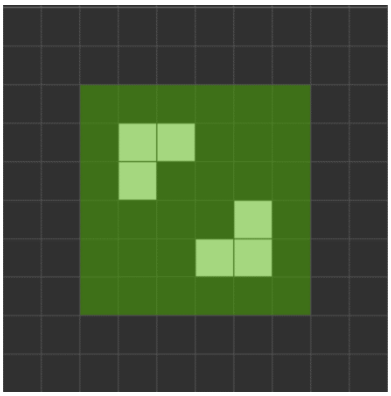
۱-۱. الگوی اول

نسل صفر الگوی اول را در شکل ۲ مشاهده می‌کنید.



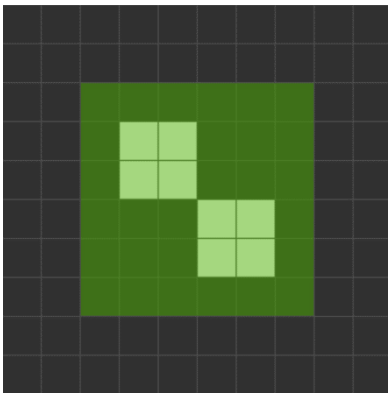
شکل ۲ - نسل ۰ الگوی اول

نسل اول الگوی اول را در شکل ۳ مشاهده می‌کنید.



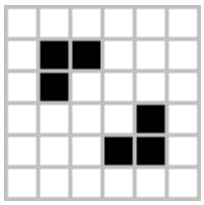
شکل ۳ - نسل ۱ الگوی اول با اجرای قاعده‌ی **Life**

نسل دوم الگوی اول را در شکل ۴ مشاهده می‌کنید.



شکل ۴ - نسل ۲ الگوی اول با اجرای قاعده‌ی **Life**

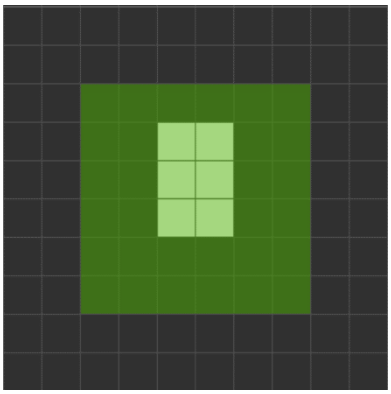
همانطور که می‌بینید الگوی اولیه پس از دو نسل تکرار می‌شود. پس این الگو از نوع Oscillator با period = 2 می‌باشد. به طور خاص اسم این الگو Beacon می‌باشد.



شکل ۵ - الگوی **Beacon** با **period = 2** از خانواده **oscillator**ها

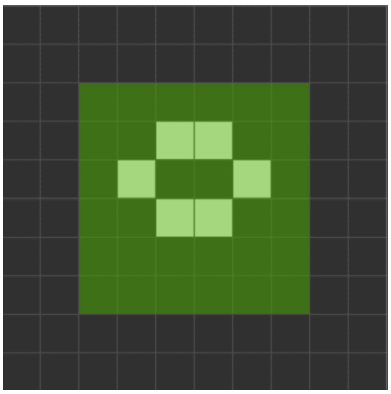
۱-۲. الگوی دوم

نسل صفر الگوی دوم را در شکل ۶ مشاهده می‌کنید.



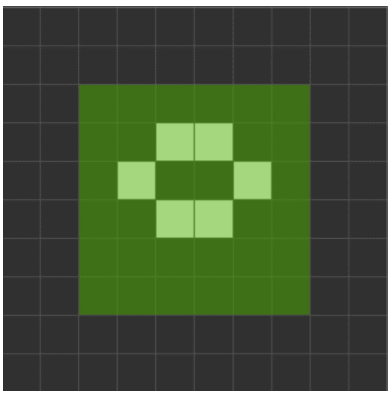
شکل ۶ - نسل ۰ الگوی دوم

نسل ۱ الگوی دوم را در شکل ۷ مشاهده می‌کنید.



شکل ۷ - نسل ۱ الگوی دوم با اجرای قاعده‌ی **Life**

نسل ۲ الگوی دوم را در شکل ۸ مشاهده می‌کنید.

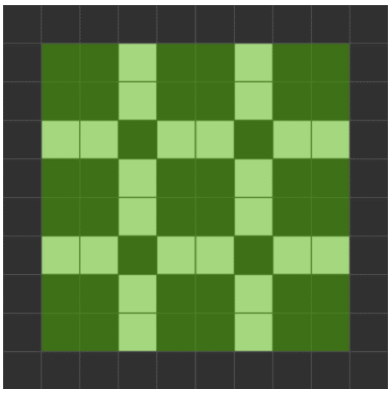


شکل ۸ - نسل ۲ الگوی دوم با اجرای قاعده‌ی **Life**

همانطور که می‌بینید این الگو بعد از یک نسل به یک الگوی ثابت رسید. الگویی که در شکل ۷ و ۸ می‌بینید Bee-hive نام دارد و جزو خانواده‌ی still lifeها می‌باشد.

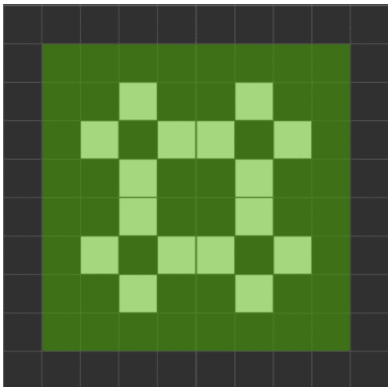
۱-۳. الگوی سوم

نسل ۰ الگوی سوم را در شکل ۹ مشاهده می‌کنید.



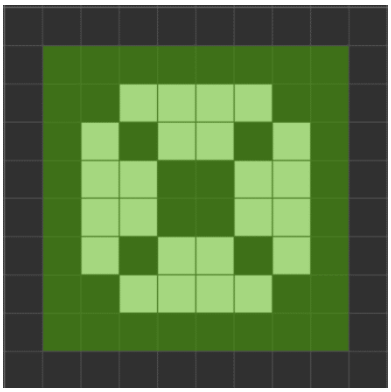
شکل ۹ - نسل ۰ الگوی سوم

نسل ۱ الگوی سوم را در شکل ۱۰ مشاهده می‌کنید.



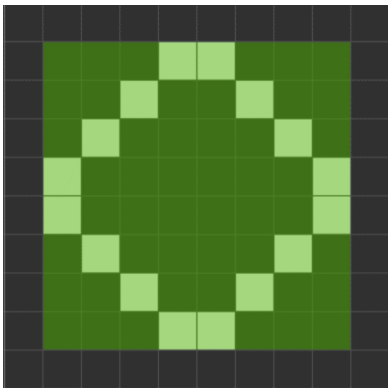
شکل ۱۰- نسل ۱ الگوی سوم با اجرای قاعده‌ی **Life**

نسل ۲ الگوی سوم را در شکل ۱۱ مشاهده می‌کنید.



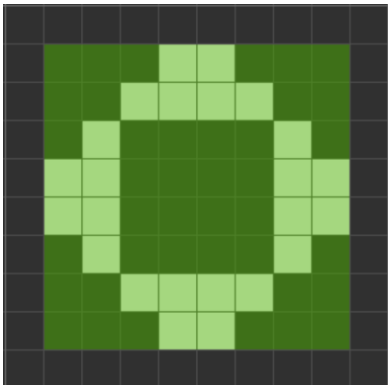
شکل ۱۱ - نسل ۲ الگوی سوم با اجرای قاعده‌ی **Life**

نسل ۳ الگوی سوم را در شکل ۱۲ مشاهده می‌کنید.



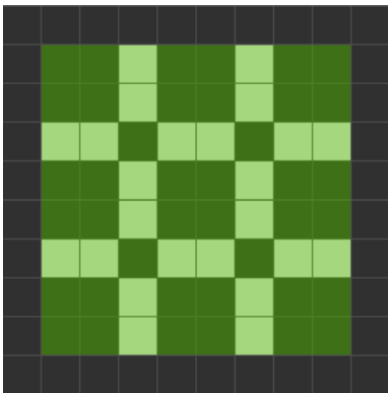
شکل ۱۲- نسل ۳ الگوی سوم با اجرای قاعده‌ی **Life**

نسل ۴ الگوی سوم را در شکل ۱۳ مشاهده می‌کنید.



شکل ۱۳ - نسل ۴ الگوی سوم با اجرای قاعده‌ی **Life**

نسل ۵ الگوی سوم را در شکل ۱۴ مشاهده می‌کنید.

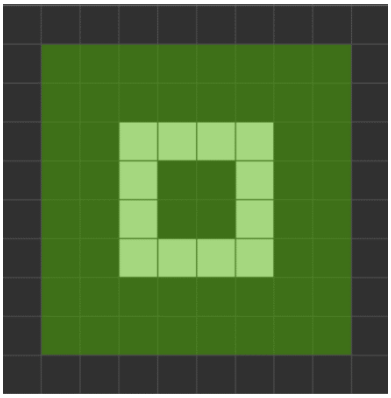


شکل ۱۴ - نسل ۵ الگوی سوم با اجرای قاعده‌ی **Life**

همانطور که می‌بینید الگوی اولیه پس از پنج نسل تکرار می‌شود. پس این الگو نیز از نوع Oscillator منتهی با period = 5 می‌باشد.

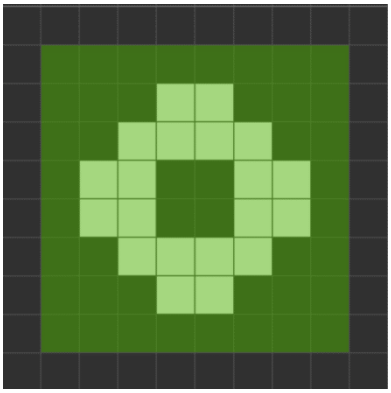
۱-۴. الگوی چهارم

نسل ۰ الگوی چهارم را در شکل ۱۵ مشاهده می‌کنید.



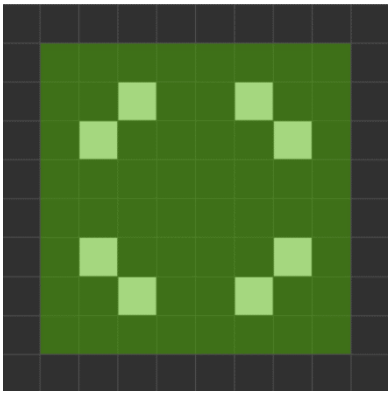
شکل ۱۵- نسل ۰ الگوی چهارم

نسل ۱ الگوی چهارم را در شکل ۱۶ مشاهده می‌کنید.



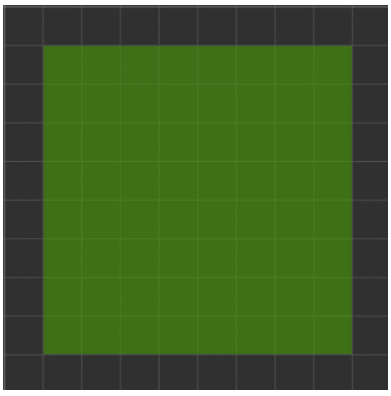
شکل ۱۶- نسل ۱ الگوی چهارم با اجرای قاعده‌ی **Life**

نسل ۲ الگوی چهارم را در شکل ۱۷ مشاهده می‌کنید.



شکل ۱۷- نسل ۲ الگوی چهارم با اجرای قاعده‌ی **Life**

نسل ۳ الگوی چهارم را در شکل ۱۷ مشاهده می‌کنید.



شکل ۱۸- نسل ۳ الگوی چهارم با اجرای قاعده‌ی **Life**

الگوی اولیه پس از ۳ نسل به یک grid خالی رسید. که grid خالی در نسل‌های بعدی نیز خالی می‌ماند.

پرسش ۲. Wolfram 22

قاعده‌ی ۲۲ ولفرام یکی از ۲۵۶ قانون ابتدایی اتوماتای سلولی است که توسط Stephan Wolfram تعریف شده است. این قوانین تکامل یک اتومات سلولی تک بعدی متشکل از یک خط سلول، که هر کدام می تواند در یکی از دو حالت باشد، را کنترل می کند: روشن یا خاموش. در قانون ۲۲، وضعیت بعدی یک سلول با وضعیت فعلی آن و دو سلول مجاور آن تعیین می شود. قواعد Wolfram 22 به این صورت هستند:

۰ <- ۱۱۱

۱ <- ۱۱۰

۱ <- ۱۰۱

۰ <- ۱۰۰

۱ <- ۰۱۱

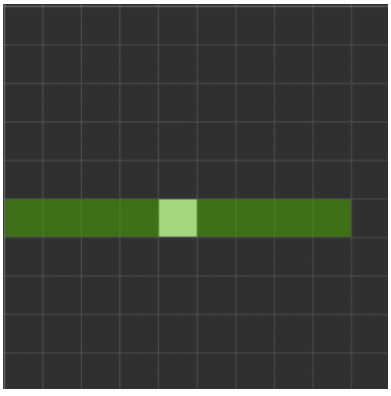
۰ <- ۰۱۰

۱ <- ۰۰۱

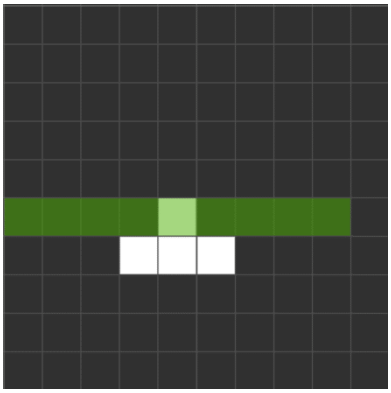
۰ <- ۰۰۰

به طور مثال اگر سلول فعلی و همسایگان چپ و راست آن در یک پیکربندی خاص باشند، وضعیت بعدی سلول مرکزی توسط خروجی مربوطه در قانون تعیین می شود. به عنوان مثال، اگر سلول فعلی و همسایگان آن در پیکربندی "۱۰۱" باشند، سلول مرکزی در نسل بعدی به حالت "۱" تغییر خواهد کرد.

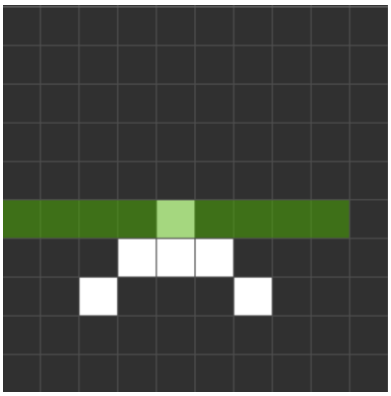
الگوی ورودی و دو نسل بعدی آن را به ازای اجرای قانون ۲۲ به ترتیب در شکل‌های ۱۹ ، ۲۰ و ۲۱ مشاهده می‌کنید:



شکل ۱۹- نسل ۰ الگوی سوال ۲

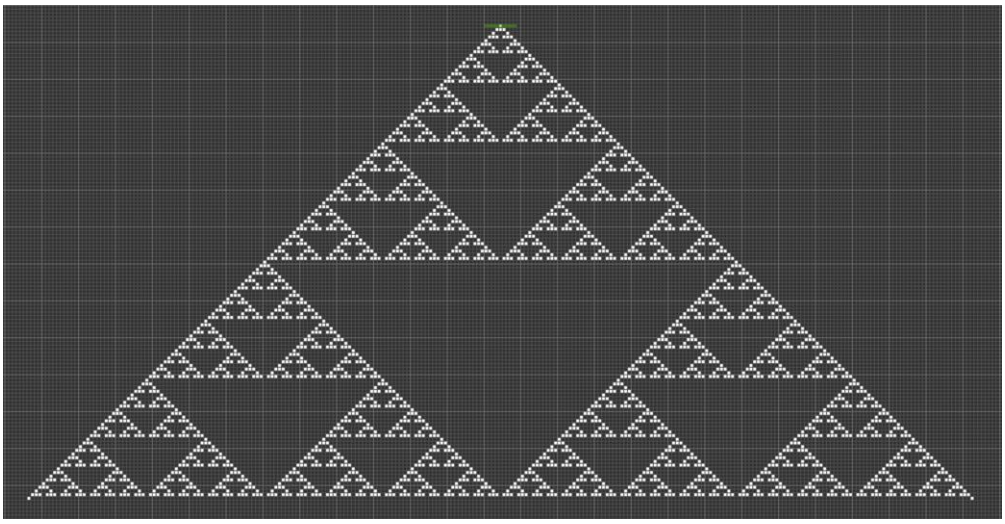


شکل ۲۰- نسل ۰ الگوی سوال ۲ با اجرای قاعده‌ی **Wolfram 22**



شکل ۲۱ - نسل ۰ الگوی سوال ۳ با اجرای قاعده‌ی **Wolfram 22**

همانطور که می‌بینید الگوی اولیه طی هر نسل مقداری گسترش می‌یابد به طوریکه در نسل ۱۲۸ الگویی همانند شکل ۲۲ را خواهیم داشت. به طور کلی قانون Wolfram 22 با توجه به ورودی به اندازه کافی chaotic قادر به رشد به طور explosive است.



شکل ۲۲- نسل ۱۲۸ الگوی سوال ۳ با اجرای قاعده‌ی **Wolfram 22**

پرسش ۳. Wetware Computer

۳-۱. مقدمه

پیشرفت کامپیوترها در طول زمان منجر به تولید دستگاه های کوچکتر، سریعتر و قدرتمندتر شده است. با این حال، محدودیت‌های رایانه‌های متداول مبتنی بر سیلیکون، محققان را بر آن داشته تا راه‌حل‌های جایگزین را بررسی کنند. یک مفهوم امیدوارکننده کامپیوترهای ارگانیک است که به عنوان کامپیوترهای wetware نیز شناخته می‌شود، که دستگاه‌های محاسباتی ساخته شده از مواد آلی مانند نورون‌های زنده هستند. برخلاف رایانه‌های سنتی که به صورت دودویی کار می‌کنند، نورون‌ها می‌توانند در حالت‌های متعددی وجود داشته باشند که به طور بالقوه امکان ذخیره سازی اطلاعات را به میزان قابل توجهی فراهم می‌کنند. در حالی که رایانه های ارگانیک عمدتاً در حد تئوری و نظریه باقی مانده اند، پیشرفت های اخیر نشان می دهد که کاربردهای عملی ممکن است در دسترس باشد. این نشان دهنده تغییر به سمت تحقق پتانسیل رایانه های ارگانیک در آینده نزدیک است.[۱].

سلول‌ها را می‌توان به‌عنوان شکلی از wetwareهای طبیعی در نظر گرفت که با استفاده از معماری ساخت‌یافته خود عمل می کنند، به طوری که اجزای کوچکتر برای دریافت ورودی، پردازش اطلاعات و تولید خروجی با هم کار می کنند. در یک تجزیه و تحلیل ساده، عملکرد سلولی شامل ذخیره اطلاعات و دستورالعمل ها به عنوان DNA، استفاده از RNA به عنوان منبع ورودی که توسط ریبوزوم ها و سایر عوامل پردازش می شود و در نهایت پروتئین ها را به عنوان خروجی تولید می کند [۲].

۳-۲. کاربردهای wetware computer ها

یک روش انقلابی با استفاده از یک ویروس برای ایجاد رایانه های سریعتر

در مطالعه‌ای که در سال ۲۰۱۸ در ژورنال ACS Applied Nano Materials منتشر شده است، محققان دانشگاه‌های MIT و SUTD به کشف مهمی در فناوری رایانه دست یافته‌اند. آنها روشی را با استفاده از یک ویروس به نام باکتریوفاژ M13 برای ایجاد حافظه کامپیوتری بهبود یافته توسعه داده اند. تمرکز کار آنها کاهش تاخیرهای زمانی در انتقال اطلاعات و ذخیره سازی بین تراشه های RAM و هارد دیسک است. تلاش‌های قبلی شامل حافظه تغییر فاز بود، اما مشکلاتی با مصرف انرژی و جداسازی مواد داشت. این تیم از فناوری سیم ریز (tiny wire) برای ساخت سیم‌های اکسید ژرمانیوم-قلع و حافظه در دمای پایین استفاده کردند. این پیشرفت می تواند به رایانه های سریعتر و کارآمدتر منجر شود و تاخیرهای میلی ثانیه ای در ذخیره سازی و انتقال را از بین ببرد [۳].

ایجاد یک کامپیوتر دو هسته ای بیوسنتزی در سلول های انسانی با استفاده از CRISPR

سیستم ویرایش ژن CRISPR کاربرد گسترده ای در زمینه های مختلف داشته است و اکنون می توان زیست شناسی مصنوعی (Synthetic Biology) را نیز به این لیست اضافه کرد. محققان ETH زوریخ از نسخه اصلاح شده CRISPR-Cas9 برای ساختن بیوکامپیوترهای دو هسته‌ای کاربردی در سلول‌های انسانی استفاده کرده اند. این سیستم اصلاح شده از نوع خاصی از پروتئین Cas9 استفاده می کند که به عنوان یک پردازشگر برای خواندن ورودی از توالی های RNA راهنما، تنظیم بیان ژن و تولید پروتئین‌های خاص به عنوان خروجی عمل می کند. محققان با ترکیب اجزای CRISPR-Cas9 از دو باکتری، اولین پردازنده بیولوژیکی دو هسته‌ای را در یک سلول ایجاد کردند [۴]. این بیوکامپیوترها دارای پتانسیل زیادی برای کاربردهای متنوع از جمله تشخیص و درمان بیماری هستند. آنها را می توان طوری برنامه ریزی کرد که سیگنال های بیولوژیکی خاصی را در بدن شناسایی کرده و بر اساس آن پاسخ دهند. برای مثال، بر اساس وجود نشانگرهای زیستی خاص، زیست کامپیوترها می توانند مولکول های تشخیصی یا مواد دارویی تولید کنند. «ارگان‌های محاسباتی (computational organs)» متشکل از میکروبافت‌هایی با میلیاردها سلول، که هر کدام به یک پردازنده دو هسته‌ای مجهز هستند، می‌توانند به طور بالقوه از قدرت محاسباتی ابررایانه‌های دیجیتال پیشی بگیرند و در عین حال انرژی بسیار کمتری مصرف کنند.

اولین کامپیوتر DNA (DNA computer) قابل reprogram شدن

کامپیوترهای DNA مدت‌هاست که وعده سرعت و قدرت بی‌نظیر را داده‌اند، اما عدم انعطاف‌پذیری آن‌ها مانع از مفید بودن آن‌ها شده است. با این حال، پیشرفت اخیر محققان به رهبری دیوید دوتی از UC Davis راه حلی را ارائه می دهد. آنها در مقاله خود که در Nature منتشر شده‌است، روشی را برای reprogram کردن کامپیوترهای DNA با استفاده از یک trigger ساده توصیف می کنند که به مجموعه اصلی مولکول های DNA اجازه می دهد چندین الگوریتم را اجرا کنند. بر خلاف تلاش‌های قبلی، که نیاز به ساخت دقیق توالی‌های DNA خاص برای هر الگوریتم داشت، محققان سیستمی را طراحی کردند که می‌تواند با تولید الگوریتم‌های مختلف، قطعات DNA یکسان را مجبور به تشکیل ساختارهای مختلف کند. با استفاده از این رویکرد، محققان با موفقیت ۲۱ الگوریتم مختلف را ایجاد کردند که قادر به انجام وظایفی مانند تولید الگو، تشخیص مضرب سه، شمارش تا ۶۳ و انتخاب یک leader بودند [۵].

reprogrammable DNA computers در حالی که هنوز در مراحل اولیه هستند، پتانسیل بسیار زیادی دارند. آنها می توانند زمینه هایی مانند رباتیک مولکولی را برای drug delivery متحول کنند. این الگوریتم‌های مولکولی با مونتاژ اجسام پیچیده در سطح نانو، مشابه نحوه تشکیل سلول‌های زنده انواع مختلف سلول از طریق بیان پروتئین، امکانات هیجان‌انگیزی را ارائه می‌کنند. این تحقیق درها را به روی

آینده ای باز می کند که در آن رایانه های DNA می توانند برای طیف گسترده ای از کاربردها با انعطاف پذیری و سازگاری بی سابقه برنامه ریزی شوند.

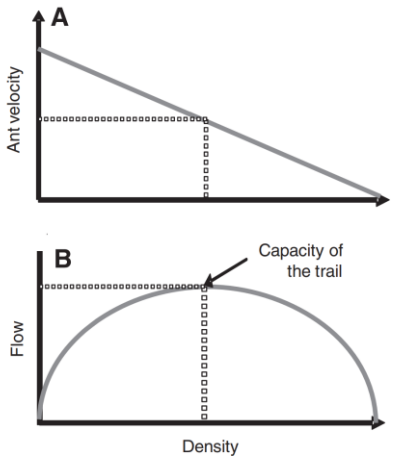
پرسش ۴. Ants Traffic Control Mechanisms

۴-۱. مقدمه

بسیاری از حیوانات در حرکات جمعی شبه جریانی شرکت می کنند. با این حال، در بیشتر گونه ها، جریان یک طرفه است. مورچه ها یکی از گروه‌های نادری از موجودات هستند که در آنها حرکات شبه‌جریانی عمدتاً دو طرفه است. این به سختی کار حفظ یک حرکت صاف و کارآمد می افزاید. با این حال، به نظر می رسد مورچه ها از عهده این کار به خوبی بر می آیند. آیا آنها واقعا توانایی چنین کاری را دارند؟ و اگر چنین است، چگونه چنین ارگانیسم‌های ساده‌ای موفق می شوند جریان ترافیکی روان را حفظ کنند، وقتی حتی انسان ها با این کار مشکل دارند؟ ترافیک در مورچه ها با عابران پیاده یا وسایل نقلیه انسانی چگونه قابل می‌باشد؟ مطالعه تجربی ترافیک مورچه تنها چند سال قدمت دارد اما قبلاً بینش جالبی در مورد سازماندهی و مقررات ترافیکی در حیوانات ارائه کرده است، به ویژه نشان می دهد که یک کلونی مورچه را می توان به عنوان یک سیستم تطبیقی خود سازمان یافته معمولی در نظر گرفت.

۴-۲. مفاهیم ابتدایی ترافیک مورچه‌ها

مورچه ها در مدیریت ترافیک در مسیرهای جستجوی خود با چالشی مشابه مهندسان راه روبرو هستند. برای بهینه سازی استفاده از trail و به حداکثر رساندن تحویل غذا به لانه، مورچه ها باید جریان (flow) خود را نزدیک به ظرفیت مسیر (trail's capacity) حفظ کنند. این ظرفیت توسط عرض مسیر تعیین می شود (شکل ۲۳) و در نهایت توسط ورودی های لانه محدود می شود. اگر شدت ترافیک از ظرفیت مسیر بیشتر شود، انبارهای غذا ممکن است در مقابل ورودی های لانه جمع شوند. جریان برابر با حاصل‌ضرب سرعت و چگالی (یا تراکم) مورچه است و یک مقدارچگالی بهینه برای دستیابی به ظرفیت حداکثر وجود دارد (شکل ۲۳) [۶].



شکل ۲۳ - نمودارهای بنیادی تئوری ترافیک

(**A**) رابطه بین دهد بین سرعت و چگالی ذرات متحرک را نشان می‌دهد و (**B**) رابطه بین جریان و چگالی ذرات متحرک را نشان می‌دهد [۶]

کمتر از این مقدار چگالی، به دلیل کاهش تراکم مورچه، جریان کاهش می یابد و در مقادیر بزرگتر از این چگالی، به دلیل افزایش برخوردهای head-on، جریان کاهش می یابد. بزرگ‌کردن مسیرها می‌تواند نرخ برخورد را کاهش دهد و جریان را حفظ کند، اما این هزینه‌های مختلفی برای کلونی دارد. ساخت مسیرهای بزرگتر پرهزینه است، به ویژه زمانی که تا زیر زمین ادامه داشته باشند. علاوه بر این، کاهش چگالی trial pheromone، جهت گیری مورچه ها را تحت تأثیر قرار می دهد و خطر گم شدن مسیر مورچه ها را افزایش می دهد. علاوه بر این، تماس‌های فیزیکی بین مورچه‌ها را کاهش می‌دهد که نقش مهمی در انتقال اطلاعات و کارایی جستجوی غذا دارند [۷].

۴-۳. سازماندهی مکانی و زمانی ترافیک در مسیرهای جستجوی غذا مورچه‌ها

در صورت وجود نوعی سازماندهی ترافیکی در مسیرها، می توان میزان برخورد در تراکم بالا را کاهش داد و ظرفیت مسیر را افزایش داد. همانند ترافیک وسایل نقلیه، سازمان‌دهی ترافیک در مورچه‌ها می‌تواند به دو شکل متفاوت باشد: ترافیک می‌تواند بر اساس مکانی یا زمانی سازماندهی شود. یک سازماندهی از ترافیک زمانی مشاهده می‌شود که میزانی از تفکیک خطوط رخ می‌دهد، یعنی زمانی که جریان‌های مورچه‌های ورودی و خروجی کاملاً در هم آمیخته نمی‌شوند. البته، میزان جداسازی خطوط مشاهده شده در مسیرهای مورچه ها هرگز به اندازه بزرگراه های ما نیست. با این حال، برای کاهش میزان برخورد رو به رو (head-on) بین مورچه ها و در نتیجه افزایش ظرفیت مسیرهای آنها کافی است [۸].

۴-۴. پژوهش‌های انجام شده

تعداد کمی از مطالعات به این موضوع پرداخته اند که چگونه مورچه ها جریان روانی از ترافیک را حتی با افزایش تعداد مورچه ها در مسیر حفظ می کنند. در حال حاضر، Possoinier، Motsch و همکاران آزمایشی طراحی کرده اند تا بررسی کنند که آیا مورچه ها می توانند جریان ترافیک ثابت خود را در زمانی که مسیر آنها به سمت غذا شلوغ‌تر می‌شود حفظ کنند یا خیر. این آزمایش شامل دستکاری تراکم مورچه ها با استفاده از ترکیبی از کلونی های با اندازه‌های مختلف (از ۴۰۰ تا ۲۵۶۰۰ مورچه آرژانتینی) و تغییر عرض پلی است که مورچه ها را به منبع غذایی آنها متصل می کند. این آزمایش ۱۷۰ بار تکرار شد و داده‌های مربوط به جریان ترافیک، سرعت مورچه‌ها و تعداد برخوردها جمع‌آوری شد. Possoinier غذا را روی پل (bridge) مورچه های آرژانتینی (Linepithema humile) با ۴۰۰ تا ۲۵۶۰۰ عضو قرار داد. عرض پل برای ایجاد ۱۷۰ ترکیب مختلف برای آزمایش واکنش مورچه ها به شرایط متغیر تغییر یافت. نتایج نشان می‌دهد که با وجود، یا شاید به دلیل، مغز کوچک مورچه‌ها، آنها در حرکت در شلوغی بسیار بهتر از

ما هستند. جریان ترافیک زمانی که ۸۰ درصد پل اشغال شده بود حفظ شد. آزمایش‌های قبلی نشان داده‌اند که انسان‌ها، چه با پای پیاده چه با ماشین‌ها، در ۴۰ درصد اشغال یا بالاتر، سرعت خود را از دست می‌دهند. Possoinier و همکارانش در این مطالعه نوشتند: «در تراکم‌های کم، یک رابطه خطی واضح بین تراکم مورچه و جریان وجود دارد، در حالی که در تراکم زیاد، جریان ثابت می‌ماند و ازدحام رخ نمی‌دهد» [۹].

چرا مورچه‌ها آنطور که در موقعیت‌های ترافیکی معمول انتظار می‌رود، گیر نکردند یا مسدود نشدند؟ این می تواند ناشی از سازماندهی فضایی و زمانی جریان در چگالی بالا باشد [۶]. زمانی که جریان مورچه های ورودی و خروجی کاملاً در هم آمیخته نباشند و جداسازی خطوط رخ دهد، ترافیک به صورت مکانی سازمان یافته در نظر گرفته می شود [۶ و ۸].

پرسش ۵. الگوریتم PSO

هدف ما در این سوال این است که یک مسئله‌ی بهینه‌سازی دلخواه را با استفاده از الگوریتم PSO و TPSO حل کنیم. برای پیاده‌سازی الگوریتم TPSO، ۳ کلاس اصلی تعریف می‌کنیم. کلاس Particle، کلاس Tribe و کلاس TPSO. در کلاس particle جزئیات مربوط به ذرات از جمله position، بهترین جواب پیدا شده توسط همسایه، بهترین جواب پیدا شده توسط خود ذره و امثال اینا مشخص شده‌است. همچنین در ایم کلاس تابع update_particle قرار داده شده‌است که قاعده‌ی update کلی ذرات در آن نیست. بلکه تنها به روزرسانی موقعیت و سرعت ذرات را در شرایطی که ذرات از borderها خارج می‌شوند را پیاده‌سازی می‌کند. در کلاس Tribe اطلاعات مربوط به قبیله‌ها قرار دارند. در این کلاس تابع update_particles وجود دارد که قاعده‌ی اصلی PSO در آن قرار دارد. این قاعده‌ی به روزرسانی برای ذره‌ی iم به ترتیب برای position و velocity (که جهات حرکت ذره را مشخص می‌کند) صورت زیر تعریف می‌شود:

$$P_i^{t+1} = P_i^t + V_i^t \quad (1)$$

$$P_i^{t+1} = c_0 * V_i^t + c_1 * r_1 * \left(P_{localbest(i)}^t - P_i^t\right) + c_2 * r_2 * \left(P_{neighborbest(i)}^t - P_i^t\right) \quad (2)$$

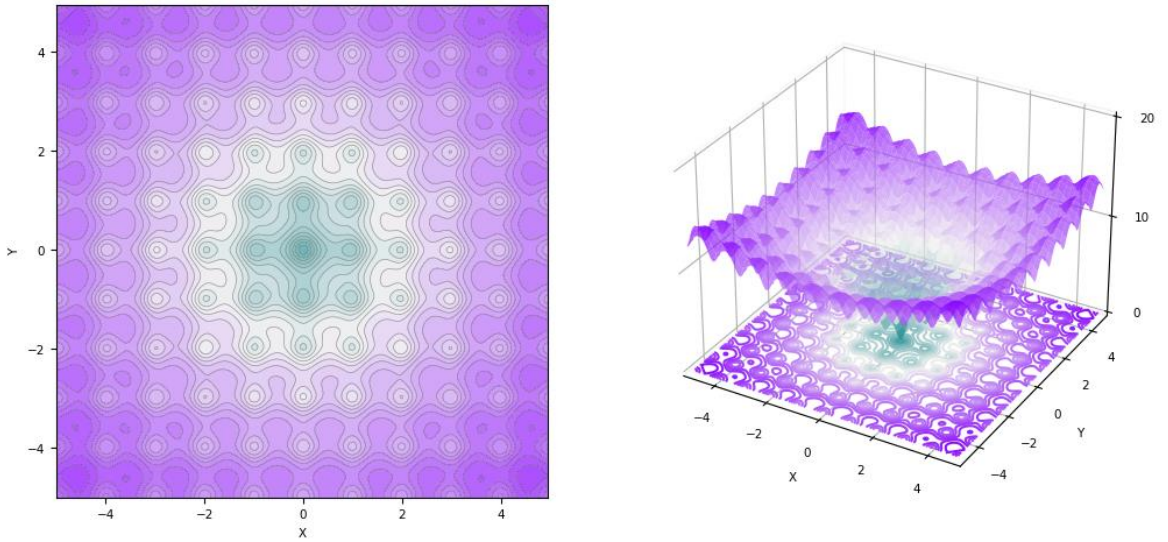
همانطور که در فرمول بالا می‌بینید، ما عبارت global را حذف کردیم (چون ضریب c3 را برابر با صفر قرار دادیم). چرا که در TPSO، ما با استفاده از best هر tribe سعی می‌کنیم ذرات ره به سمت tribeای که جواب بهتری پیدا کرده هدایت کنیم. مقادیر c0 ،c1 و c2 را به ترتیب برابر با ۰/۷، ۱ و ۲ قرار دادیم. در فرمول (۲)، term اول Inertia term می‌باشد که نشان‌دهنده‌ی میزان تاثیرگذاری سرعت فعلی ذره در سرعت ذره در time step بعدی را مشخص می‌کند. Term دوم،cognitive term نام دارد که بهترین نقطه پیدا شده توسط خود ذره را در نظر می‌گیرد. Term سوم، social term نام دارد که بهترین نقطه پیدا شده توسط همسایه‌های ذره را لحاظ می‌کند. پارامترهای r1 و r2 را هم با مقادیر رندوم بین صفرو یک مقداردهی کردیم.

مسئله‌ی بهینه‌سازی‌ای که ما به این منظور استفاده کردیم، مسئله‌ی پیدا کردن نقطه‌ی مینیمم در تابع Ackley می‌باشد. این مسئله یکی از توابع معروف مورد استفاده برای تست بهینه‌سازی است. این تابع به صورت زیر تعریف می‌شود:

$$f(x,y) = -20\exp\left[-0.2\sqrt{0.5(x^2+y^2)}\right] - \exp[0.5\,(\cos 2\pi x + \cos 2\pi y)] + e + 20$$

شکل ۲۴- تابع هزینه **Ackley**

نمایش این تابع در فضای دوبعدی و سه بعدی را در شکل زیر می‌بینید. در شکل سه بعدی، دو بعد x و y نشان دهنده‌ی متغیرهای این تابع هستند. یعنی مسئله ما در فضای دو بعدی تعریف می‌شود. و بعد z نشان دهنده‌ی مقدار تابع به ازای مقادیر x و y متناظر می‌باشد.



شکل ۲۵ - فضای حالت تابع **Ackley**

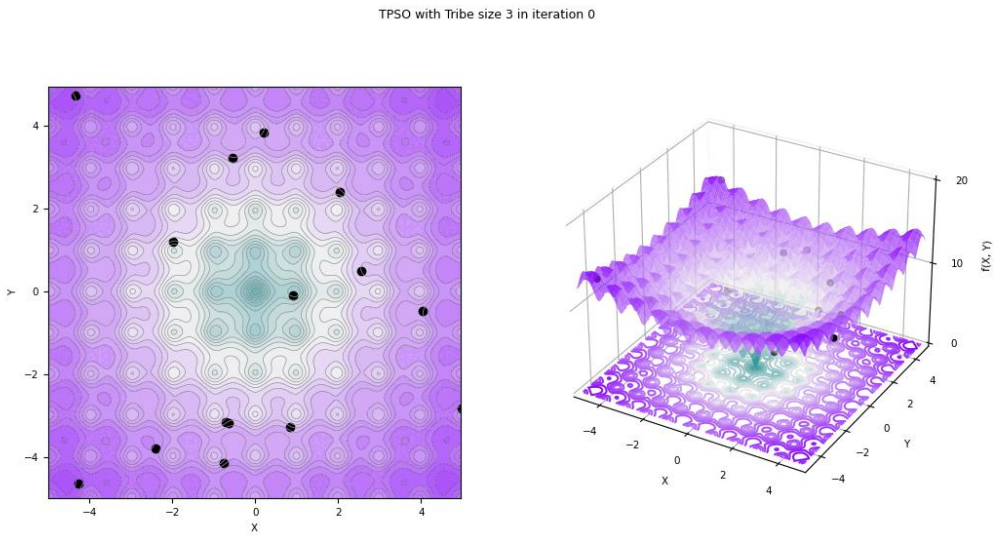
همانطور که در شکل بالا نیز مشخص است، تابع ackley دارای local minimaهای زیادی است و دارای یک Global minimum در نقطه‌ی (0,0) می‌باشد.

نکته: کدهای نوشته شده در فایل نوت‌بوک colab به نام Q5.ipynb قرار دارند. فایل پایتون utils.py حاوی توابع کمکی مورد نیاز برای نمودارهای سه بعدی می‌باشد.

۵-۱. TPSO با سه ذره در هر tribe

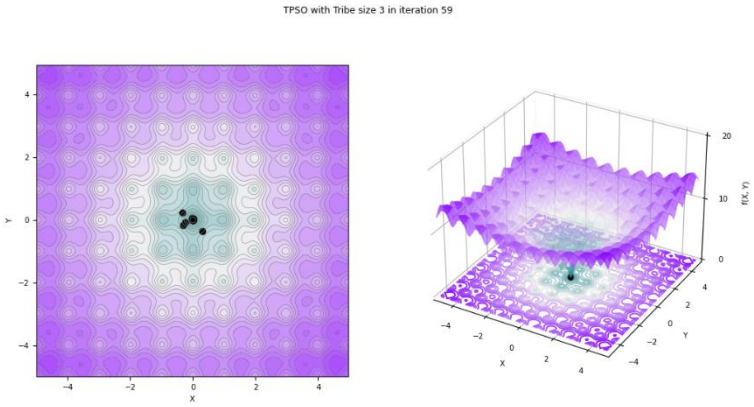
برای این قسمت ۵ تا قبیله ساختیم به طوریکه هر قبیله شامل ۳ ذره می‌باشد. ذرات عضو هر قبیله را در ابتدای الگوریتم به طور رندوم assign می‌کنیم. پس در کل ۱۵ ذره داریم. حال الگوریتم را با مقادیر تعیین شده برای پارامترها و متغیرهای مسئله اجرا می‌کنیم. در کل می‌گذاریم الگوریتم برای ۶۰تا iteration اجرا شود.

نحوه حرکت ذرات طی iterationهای مختلف را در گیف زیر (شکل ۲۶) می‌بینید (اگر در فایل pdf به صورت یک تصویر نشان داده می‌شود، لطفا به فایل ورد یا گیف‌های پیوست شده مراجعه کنید).



شکل ۲۶- اجرای الگوریتم **Tribal PSO** با قبیله‌هایی با سایز ۳

همانطور که در شکل بالا می‌بینید الگوریتم بعد حدود ۲۱ iteration موفق به همگرا شدن در نقطه بهینه سراسری می‌شود. شکل ۲۷ نشان‌دهنده‌ی محل ذرات در iteration آخر می‌باشد.



شکل ۲۷- محل ذرات در **iteration** شصتم الگوریتم **TPSO** با قبیله‌های با سایز ۳

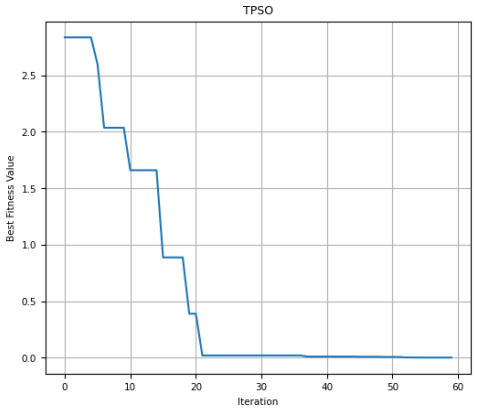
شکل ۲۷ نشان می‌دهد الگوریتم به همگرایی خوبی رسیده است. نقطه نهایی گزارش شده در الگوریتم را در شکل ۲۸ مشاهده می‌کنید.

```
Global Best Solution: [0.00012896 0.00011331]
Global Best Fitness: 0.00048632819885741085
```

شکل ۲۸- جواب نهایی پیدا شده توسط **TPSO** با قبیله‌هایی با سایز ۳

همانطور که در شکل ۲۸ می‌بینید این پاسخ با پاسخ اصلی این مسئله‌ی بهینه‌سازی تفاوت چندانی ندارد. پس الگوریتم ما به خوبی توانسته این مسئله را حل کند.

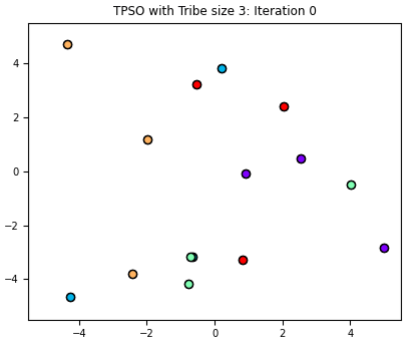
این موضوع در نمودار زیر نیز مشهود است. نمودار بهترین میزان fitness پیدا شده در هر iteration را درشکل ۲۹ می‌بینید.



شکل ۲۹- نمودار **Global Best** در هر **iteration** برای **TPSO** با سایز قبیله ۳

دقت شود که ما term مربوط به Global Best را در قاعده‌ی به روزرسانی خود نداشتیم. این مقدار را با مقایسه‌ی best هر tribe با سایر tribeها بدست می‌آوریم. طبق نمودار شکل ۲۹، الگوریتم به خوبی توانسته به مینیمم سراسری همگرا شود.

برای مشاهده‌ی ذرات هر tribe به طور جداگانه و نحوه حرکت آن‌ها در الگوریتم، یک scatter plot دو بعدی در فضای حالت مسئله کشیدیم که در آن ذرات هم‌رنگ به یک tribe تعلق دارند. نحوه جابه‌جایی ذرات طی هر iteration را در گیف شکل ۲۸ مشاهده می‌کنید (اگر در فایل pdf به صورت یک تصویر نشان داده می‌شود، لطفا به فایل ورد یا گیف‌های پیوست شده مراجعه کنید). دقت شود که این شکل دقیقا متناظر با گیف شکل ۲۶ می‌باشد. در اجرای الگوریتم هر دو نوع نمودار را به منظور visualization نحوه عملکرد الگوریتم رسم کردیم.

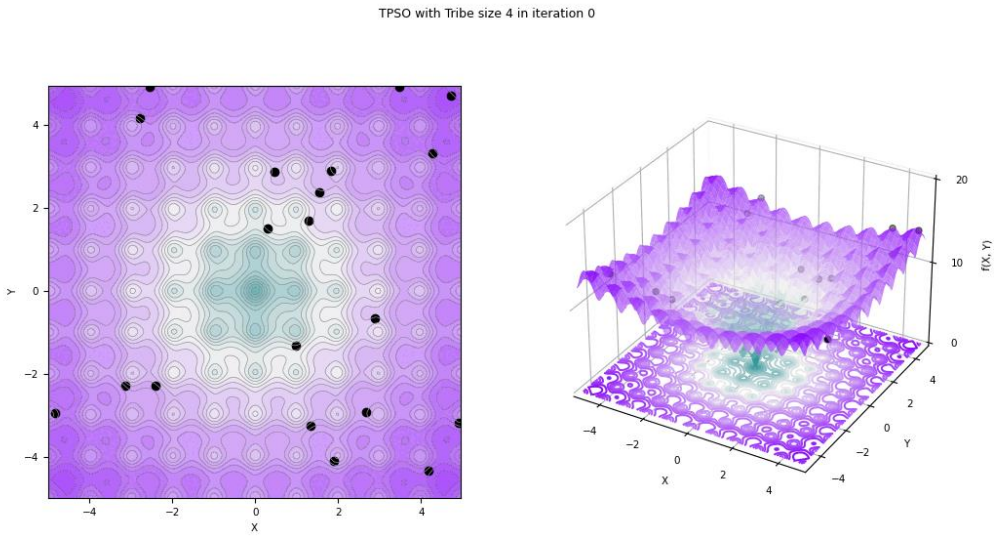


شکل ۳۰- اجرای الگوریتم **Tribal PSO** با قبیله‌هایی با سایز ۳ با نمایش قبیله‌ها

۵-۲. TPSO با چهار ذره در هر tribe

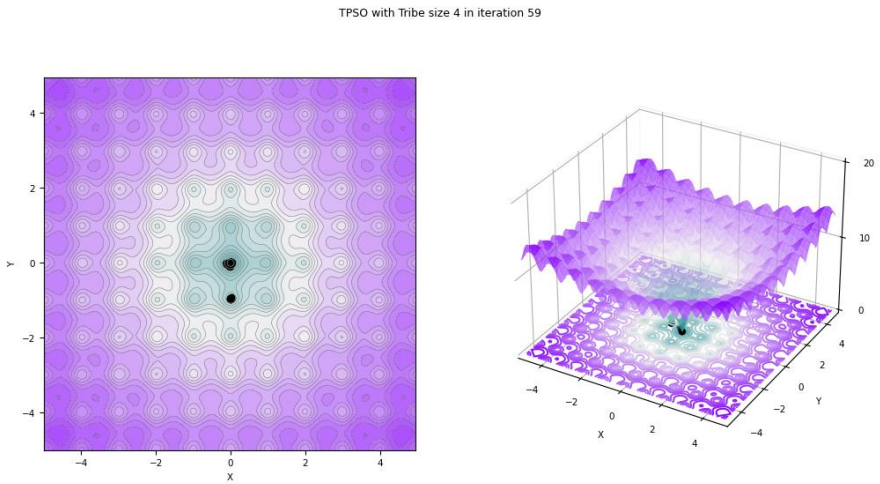
این بار فقط کافیست پارامتر ورودی تعداد ذرات در هر قبیله را ۴ قرار دهیم و همه‌ی مراحل گزارش شده در قسمت الف را تکرار کنیم. نتایج به این صورت خواهند بود:

نحوه حرکت ذرات طی iterationهای مختلف را در گیف زیر (شکل ۳۱) می‌بینید (اگر در فایل pdf به صورت یک تصویر نشان داده می‌شود، لطفا به فایل ورد یا گیف‌های پیوست شده مراجعه کنید).



شکل ۳۱- اجرای الگوریتم **Tribal PSO** با قبیله‌هایی با سایز ۴

همانطور که در شکل بالا می‌بینید الگوریتم بعد حدود ۴۰ iteration موفق به همگرا شدن در نقطه بهینه سراسری می‌شود. شکل ۳۲ نشان‌دهنده‌ی محل ذرات در iteration آخر می‌باشد.



شکل ۳۲- محل ذرات در **iteration** شصتم الگوریتم **TPSO** با قبیله‌های با سایز ۳

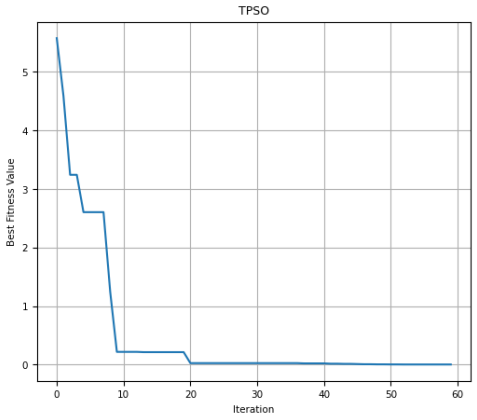
شکل ۳۲ نشان می‌دهد الگوریتم به همگرایی خوبی رسیده است. نقطه نهایی گزارش شده در الگوریتم را در شکل ۳۳ مشاهده می‌کنید.

```
Global Best Solution: [-0.00012485  0.00029298]
Global Best Fitness: 0.0009034862476338112
```

شکل ۳۳- جواب نهایی پیدا شده توسط **TPSO** با قبیله‌هایی با سایز ۴

همانطور که در شکل ۳۳ می‌بینید این پاسخ با پاسخ اصلی این مسئله‌ی بهینه‌سازی تفاوت چندانی ندارد. و همچنین تفوت چندانی با پاسخ پیدا شده توسط TPSO با سایز قبیله ۳ ندارد.پس الگوریتم ما با تعداد ۴ ذره در قبیله هم به خوبی توانسته این مسئله را حل کند.

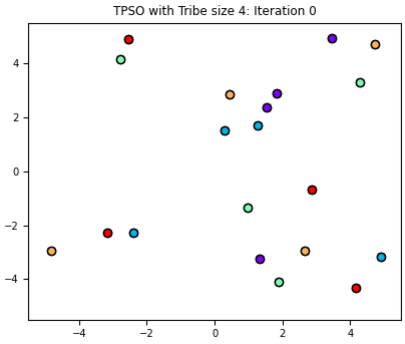
این موضوع در نمودار زیر نیز مشهود است. نمودار بهترین میزان fitness پیدا شده در هر iteration را درشکل ۳۴ می‌بینید.



شکل ۳۴- نمودار **Global Best** در هر **iteration** برای **TPSO** برای قبیله با سایز ۴

طبق نمودار شکل ۳۴، الگوریتم به خوبی توانسته به مینیمم سراسری همگرا شود. در مقایسه‌ی این حالت با حالتی که ۳ ذره در هر قبیله بود، می‌بینیم که در این حالت الگوریتم کمی زودتر همگرا می‌شود. در اینجا در iteration = 20 همگرا شده درحالیکه یا سایز قبیله ۳ بعد از iteration = 20 همگرا شده بود.

نحوه جابه‌جایی ذرات هر tribe طی هر iteration را در گیف شکل ۳۵ مشاهده می‌کنید (اگر در فایل pdf به صورت یک تصویر نشان داده می‌شود، لطفا به فایل ورد یا گیف‌های پیوست شده مراجعه کنید). دقت شود که این شکل دقیقا متناظر با گیف شکل ۳۱ می‌باشد.

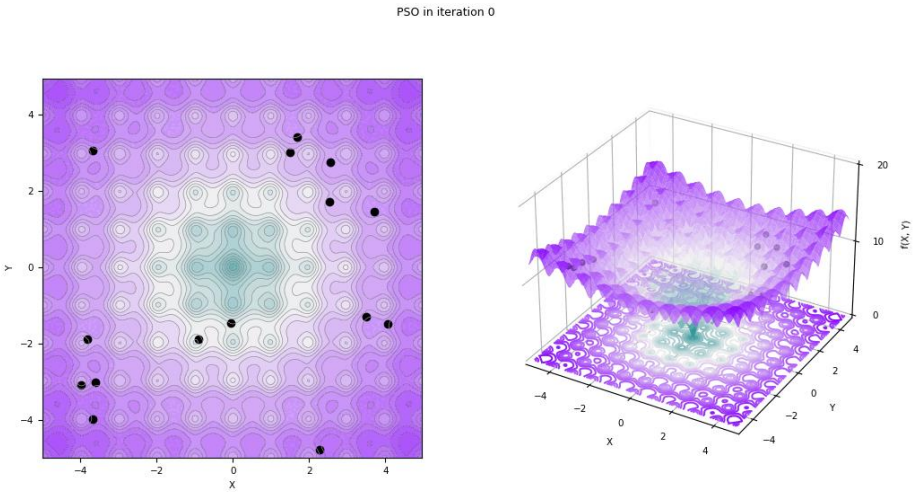


شکل ۳۵- اجرای الگوریتم Tribal PSO با قبیله‌هایی با سایز ۳ با نمایش قبیله‌ها

۵-۳. PSO با یال‌های رندوم به تعداد یالهای قسمت یک

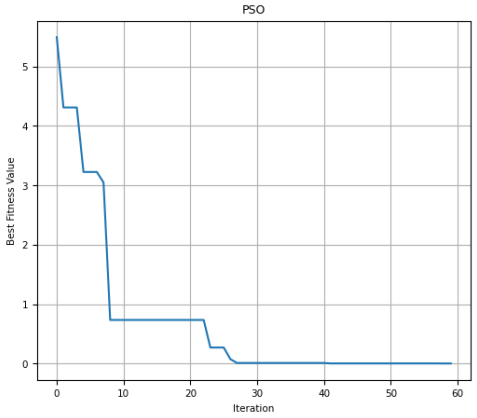
این بار الگوریتم مقداری تغییر پیدا می‌کند و مفهوم tribe از بین می‌رود. چرا که در هر Tribe بین تمام ذرات موجود در tribe یال وجود دارد. به عبارتی هر tribe یک زیرگراف کامل است و tribeها با یالی (information link) به طور رندوم به هم وصل می‌شوند. ولی در این حالت، همان تعداد یالی که در بخش ۱ داشتیم (۱۵ یال به ازای ۵ تا قبیله‌ی با سایز ۳ و ۵ یال برای اتصال قبایل بهم. در مجموع ۲۰ یال) را به طور رندوم بین هر دو ذره به طور تصادفی انتخاب شده وصل می‌کنیم و گراف نهایی ذرات لزوماً شامل زیرگراف کامل نخواهد بود. به این منظور دیگر کلاس Tribe را نخواهیم داشت و به جای آن دو کلاس Particle و PSO را خواهیم داشت.

اجرای این الگوریتم روی مسئله‌ی بهینه سازی تعریف شده (مینیمم سازی تابع Ackley) را در شکل زیر مشاهده می‌کنید (اگر در فایل pdf به صورت یک تصویر نشان داده می‌شود، لطفا به فایل ورد یا گیف‌های پیوست شده مراجعه کنید).



شکل ۳۶- اجرای الگوریتم **PSO** با تعداد یال ۲۰

همانطور که می‌بینید الگوریتم نسبت به بخش یک کمی دیرتر همگرا می‌شود. این موضوع در نمودار بهترین fitness در هر iteration مشاهده می‌کنید (شکل ۳۷). الگوریتم بخش یک در حدود iteration بیست و یکم همگرا می‌شود در حالیکه در این حالت حدود iteration بیست و هشتم این اتفاق میفتد. به عبارتی الگوریتم TPSO از نظر سرعت همگرایی کمی بهتر از الگوریتم PSO تغییر یافته عمل می‌کند.



شکل ۳۷ - نمودار **Global Best** در هر **iteration** برای **PSO** با تعداد یال برابر با بخش ۱

در شکل ۳۸ پاسخ نهایی پیدا شده توسط این الگوریتم برای مسئله‌ی بهینه‌سازی Ackley را مشاهده می‌کنید.

```
Global Best Solution: [0.00018533 0.00052217]
Global Best Fitness: 0.0015753502283328835
```

شکل ۳۸- جواب نهایی پیدا شده توسط **PSO** با تعداد یال برابر با بخش ۱

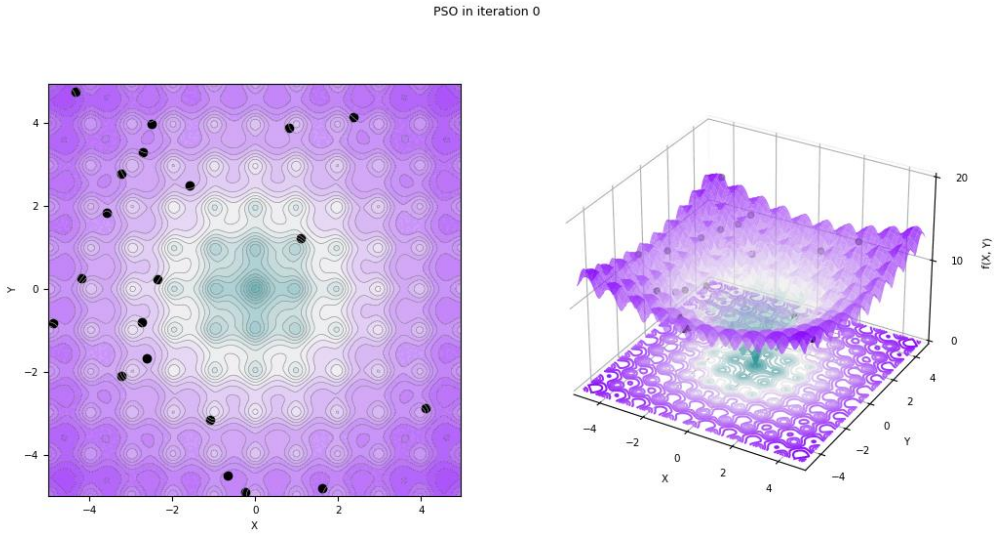
این الگوریتم طی ۶۰ دور، موفق به یافتن جواب 0.0015 شد در حالیکه الگوریتم TPSO بخش یک با همین تعداد دور موفق به یافتن جواب 0.00048 شد. یعنی از نظر بهینگی پاسخ نیز الگوریتم TPSO بهتر عمل می‌کند.

پس، الگوریتم TPSO بخش یک برای این مسئله‌ی بهینه‌سازی بهتر عمل می‌کند (نسبت به الگوریتم PSO یی که همان تعداد یال را به طور رندوم بین ذرات می‌کشد). هم از نظر بهینگی جواب و هم از نظر سرعت همگرایی.

۵-۴. PSO با یال‌های رندوم به تعداد یالهای قسمت دو

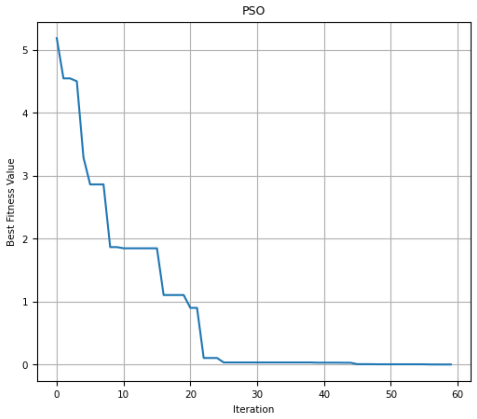
در این قسمت الگوریتم PSO معرفی شده در بخش قبل (بخش ۳) را با تعداد یال تعیین شده در بخش ۲ (یعنی برای Tribeهایی با سایز ۴) اجرا می‌کنیم. تعداد یال در این حالت ۳۵ تا خواهد بود (۵ تا قبیله و هر قبیله ۶ تا یال داشت و برای اتصال این ۵ قبیله بهم، به ۵ یال دیگر نیاز داشتیم).

اجرای این الگوریتم روی مسئله‌ی بهینه سازی تعریف شده (مینیمم سازی تابع Ackley) را در شکل زیر مشاهده می‌کنید (اگر در فایل pdf به صورت یک تصویر نشان داده می‌شود، لطفا به فایل ورد یا گیف‌های پیوست شده مراجعه کنید).



شکل ۳۹- اجرای الگوریتم **PSO** با تعداد یال ۳۵

همانطور که می‌بینید الگوریتم نسبت به بخش دو کمی دیرتر همگرا می‌شود. این موضوع در نمودار بهترین fitness در هر iteration مشاهده می‌کنید (شکل ۴۰). الگوریتم در بخش دو حدود iteration بیستم همگرا می‌شود در حالیکه در این حالت حدود iteration بیست و پنجم این اتفاق میفتد.



شکل ۴۰ - نمودار **Global Best** در هر **iteration** برای **PSO** با تعداد یال برابر با بخش ۲

در شکل ۴۱ پاسخ نهایی پیدا شده توسط این الگوریتم برای مسئله‌ی بهینه‌سازی Ackley را مشاهده می‌کنید.

```
Global Best Solution: [ 0.00058232 -0.00032509]
Global Best Fitness: 0.0018981715324675186
```

شکل ۴۱- جواب نهایی پیدا شده توسط **PSO** با تعداد یال برابر با بخش ۲

این الگوریتم طی ۶۰ دور، موفق به یافتن جواب 0.0018 شد در حالیکه الگوریتم TPSO بخش یک با همین تعداد دور موفق به یافتن جواب 0.0009 شد. یعنی از نظر بهینگی پاسخ نیز الگوریتم TPSO کمی بهتر عمل می‌کند.

پس، الگوریتم TPSO بخش دو برای این مسئله‌ی بهینه‌سازی بهتر عمل می‌کند (نسبت به الگوریتم PSO یی که همان تعداد یال را به طور رندوم بین ذرات می‌کشد). هم از نظر بهینگی جواب و هم از نظر سرعت همگرایی.

مراجع

[۱] Van Hooijdonk, R. (2021). How close are we to organic computers? Richard Van Hooijdonk Blog. https://blog.richardvanhooijdonk.com/en/how-close-are-we-to-organic-computers/

[۲] Bray, D. (2009). Wetware: a computer in every living cell. Yale University Press

[۳] Loke, D. K., Clausen, G. J., Ohmura, J. F., Chong, T. C., & Belcher, A. M. (2018). Biological-templating of a segregating binary alloy for nanowire-like phase-change materials and memory. ACS Applied Nano Materials, 1(12), 6556-6562.

[۴] Kim, H., Bojar, D., & Fussenegger, M. (2019). A CRISPR/Cas9-based central processing unit to program complex logic computation in human cells. Proceedings of the National Academy of Sciences, 116(15), 7214-7219.

[۵] Woods, D., Doty, D., Myhrvold, C., Hui, J., Zhou, F., Yin, P., & Winfree, E. (2019). Diverse and robust molecular algorithms using reprogrammable DNA self-assembly. Nature, 567(7748), 366-372.

[۶] Fourcassié, V., Dussutour, A., & Deneubourg, J. L. (2010). Ant traffic rules. Journal of Experimental Biology, 213(14), 2357-2363.

[۷] Dussutour, A., Fourcassié, V., Helbing, D., & Deneubourg, J. L. (2004). Optimal traffic organization in ants under crowded conditions. Nature, 428(6978), 70-73.

[۸] Couzin, I. D., & Franks, N. R. (2003). Self-organized lane formation and optimized traffic flow in army ants. Proceedings of the Royal Society of London. Series B: Biological Sciences, 270(1511), 139-146.

[۹] Poissonnier, L. A., Motsch, S., Gautrais, J., Buhl, J., & Dussutour, A. (2019). Experimental investigation of ant traffic under crowded conditions. Elife, 8, e48945.