به نام خدا

موضوع پروژه: Squid Game Optimizer (SGO): یک الگوریتم فراابتکاری جدید

نام اعضای گروه: سارا مظاهری – فائزه رحمتی

نام استاد: جناب آقای دکتر علی جمالیان

**مقدمه:**

در این مقاله، Squid Game Optimizer (SGO) به عنوان یک الگوریتم فراابتکاری جدید با الهام از قوانین اولیه یک بازی سنتی کره ای پیشنهاد شده است.

بازی Squid یک بازی چند نفره با دو هدف اصلی است: هدف مهاجمان تکمیل هدف خود در حالی که تیم ها سعی می کنند یکدیگر را حذف کنند و معمولاً در زمین های باز و بزرگ بدون دستورالعمل تعیین شده برای اندازه و ابعاد بازی می شود.

زمین بازی برای این بازی اغلب شبیه یک ماهی مرکب است و با توجه به زمینه تاریخی، به نظر می رسد تقریباً نصف یک زمین بسکتبال استاندارد باشد.

برای ارزیابی اثربخشی الگوریتم SGO پیشنهادی، از 25 تابع آزمون ریاضی بدون محدودیت با 100 بعد، در کنار شش فراابتکاری رایج دیگر برای مقایسه استفاده می‌شود.

100 اجرای بهینه‌سازی مستقل برای SGO و سایر الگوریتم‌ها با شرایط توقف از پیش تعیین‌شده برای اطمینان از اهمیت آماری نتایج انجام می‌شود.

معیارهای آماری مانند میانگین، انحراف معیار، و میانگین ارزیابی تابع هدف مورد نیاز محاسبه می‌شوند.

مسائل بهینه‌سازی دنیای واقعی تقریباً در هر زمینه وظایف کاملاً چالش برانگیز و مسائل پیچیده در نظر گرفته می‌شوند و آنها را در دسته‌های متفرقه، از جمله محدود یا غیرمحدود، تک یا چند هدفه، پیوسته یا گسسته، و ایستا یا پویا طبقه‌بندی می‌کنند.

توسعه برخی الگوریتم‌های مبتنی بر انسان از شبیه‌سازی رفتارهای انسانی مختلف، از جمله بهینه‌سازی مبتنی بر آموزش (TLBO)51، الگوریتم فرهنگی (CA)52، و جستجوی هارمونی (HS)53 الهام گرفته شده است.

TLBO بر اساس فعالیت های آموزشی و یادگیری بین معلمان و دانش آموزان توسعه یافته است.

در این تحقیق، یک الگوریتم بهینه‌سازی فراابتکاری مبتنی بر جمعیت جدید به نام Squid Game Optimizer (SGO) پیشنهاد شده است.

با استفاده از این رقابت‌ها، هدف این مطالعه بهبود ارزیابی الگوریتم‌های فراابتکاری و افزایش اثربخشی آنها در حل مسائل بهینه‌سازی دنیای واقعی است.

کاربرد این الگوریتم در برخورد با انواع مسائل بهینه‌سازی چالش اصلی این الگوریتم مبتنی بر جمعیت است که در آن کاندیداهای راه‌حل به دو گروه بازیکنان تهاجمی و تدافعی تقسیم می‌شوند در حالی که بازیکن تهاجمی در میان بازیکنان تدافعی می‌رود تا بازی را شروع کند. مبارزه ای که از طریق یک حرکت تصادفی به سمت بازیکنان دفاعی مدل سازی می شود.

**الگوریتم بهینه بازی مرکب**

بازی Squid که با نام ojingeo نیز شناخته می شود، بر اساس یک بازی زمین بازی کودکان کره ای است و اساساً ترکیبی از تگ و هاپسکاچ است.

در همین حال، هیچ توصیه یا دستورالعملی در مورد اندازه و ابعاد زمین بازی وجود ندارد.

بر اساس تاریخچه بازی ماهی مرکب، به نظر می رسد که پلی فلد به شکل ماهی مرکب تقریباً نصف یک زمین بسکتبال معمولی است.

بخش B دقیقاً در مرکز خط افقی در امتداد پایین ماهی مرکب قرار دارد و باید به اندازه کافی جادار باشد تا بسیاری از شرکت کنندگان در آنجا بایستند، که در اصلی یا دروازه ورودی به زمین بازی ماهی مرکب است.

این بازی با ایستادن در بخش A شروع می شود. با این حال، بازیکنان تهاجمی نمی توانند وارد بخش C شوند و بلافاصله بازی را برنده شوند.

با این حال، زمانی که هر بازیکنی از بخش های D، A، B یا E خارج می شود، این مزیت دفاعی به پایان می رسد. مشابه بازیکنان تهاجمی، بازیکنان دفاعی باید با یک پا بیرون از زمین بپرند و تیم آنها زمانی برنده می شود که با موفقیت تمام بازیکنان های تهاجمی را حذف کنند.

**نحوه پیاده سازی الگوریتم**

در این پیاده سازی لیستی به نام بازیکنان داریم که نام شرکت کنندگان را ذخیره می کند. ما چندین تابع داریم:

* ()add\_player برای اضافه کردن بازیکنان به لیست
* ()remove\_player برای حذف بازیکنان از لیست
* ()select\_winner برای انتخاب تصادفی یک برنده از لیست بازیکنان

حلقه اصلی بازی به کاربر اجازه می دهد تا با انتخاب گزینه هایی برای اضافه کردن بازیکنان، حذف بازیکنان، انتخاب برنده یا ترک بازی با بازی تعامل داشته باشد.

ابتدا ماژول تصادفی را وارد می کنیم که به ما امکان می دهد اعداد تصادفی تولید کنیم و انتخاب های تصادفی انجام دهیم.

در مرحله بعد، یک لیست خالی به نام بازیکنان را برای ذخیره نام شرکت کنندگان مقداردهی اولیه می کنیم.

ما سه تابع تعریف می کنیم:

add\_player(name): این تابع نام بازیکن را به عنوان آرگومان می گیرد و با استفاده از متد ()append به لیست بازیکنان اضافه می کند.

remove\_player(name): این تابع نام بازیکن را به عنوان آرگومان می گیرد و با استفاده از متد ()remove آن را از لیست بازیکنان حذف می کند.

()select\_winner: این تابع با استفاده از تابع ()random.choice یک برنده را از لیست بازیکنان انتخاب می کند و نام برنده را چاپ می کند.

پس از تعریف توابع، یک حلقه while وارد می کنیم که تا زمانی که کاربر تصمیم به خروج از بازی بگیرد، اجرا می شود.

در داخل حلقه، ما منویی از گزینه‌ها را برای بازیکن نمایش می‌دهیم تا از بین آنها انتخاب کند:

"1. افزودن پخش کننده": از کاربر می خواهد نام یک بازیکن را وارد کند و تابع ()add\_player را فراخوانی می کند تا بازیکن را به لیست بازیکنان اضافه کند.

"2. Remove Player": از کاربر می خواهد که نام یک بازیکن را وارد کند و تابع ()remove\_player را فراخوانی می کند تا بازیکن را از لیست بازیکنان حذف کند.

"3. Select Winner": تابع ()select\_winner را فراخوانی می کند تا به طور تصادفی یک برنده از لیست بازیکنان انتخاب شود و نام آنها چاپ شود.

"4. ترک": از حلقه while خارج می شود و بازی را به پایان می رساند.

اگر کاربر یک انتخاب نامعتبر وارد کند، یک پیام خطا نمایش داده می شود و حلقه ادامه می یابد.

هنگامی که کاربر با انتخاب گزینه 4 ترک بازی را انتخاب کرد، حلقه while شکسته می شود و برنامه به خط بعدی ادامه می یابد.

در نهایت، یک پیام تشکر برای تایید پایان بازی چاپ می کنیم.

این پیاده سازی یک چارچوب اساسی برای یک بازی الهام گرفته از Squid Game فراهم می کند که در آن بازیکنان می توانند اضافه شوند، حذف شوند و برنده به طور تصادفی انتخاب شود. ذکر این نکته ضروری است که این یک نسخه ساده شده است و پیچیدگی و ویژگی های بازی Squid را ندارد.

**نحوه بازی**

برای شرکت در بازی می توانید گزینه 1 یعنی «افزودن بازیکن» را انتخاب کنید. وقتی از شما خواسته شد نام خود را وارد کنید. این برنامه نام شما را به لیست بازیکنان اضافه می کند.

اگر می‌خواهید خودتان را از بازی حذف کنید، می‌توانید گزینه ۲ یعنی «حذف بازیکن» را انتخاب کنید. وقتی از شما خواسته شد نام خود را وارد کنید. این برنامه نام شما را از لیست بازیکنان حذف می کند.

هنگامی که همه بازیکنان اضافه شدند، می توانید گزینه 3، "انتخاب برنده" را انتخاب کنید. این برنامه به صورت تصادفی یک برنده را از لیست بازیکنان انتخاب کرده و نام آنها را نمایش می دهد.

اگر می‌خواهید بازی را ترک کنید، می‌توانید گزینه 4 یعنی «خروج» را انتخاب کنید. برنامه به پایان می رسد و شما یک "متشکرم برای بازی Squid Game!" پیام