# گزارش پروژه درس هوش محاسباتی

# The Cheetah Optimizer

مرتضی ضیابخش  فریماه مه‌آبادی  پانیذ تقی‌پور

۱۱ تیر ۱۴۰۲

# فهرست مطالب

# ۱ توضیح الگوریتم و مراحل و شبه کد آن

این روزها مسائل بهینه سازی توجه زیادی را در حوزه های مختلف به خود جلب کرده‌اند. رویکردهای قطعی کلاسیک، مانند برنامه‌ریزی خطی و برنامه‌نویسی پویا، به طور گسترده برای حل مسائل بهینه‌سازی با توابع و محدودیت‌های هدف محدب و مشتق‌پذیر استفاده شده‌اند. با این حال، این روش‌ها اغلب از گرفتار شدن در راه‌حل‌های بهینه محلی رنج می‌برند، به‌ویژه زمانی که با مسائل دنیای واقعی با ابعاد بالاتر سروکار داریم. برای رفع این محدودیت‌ها، الگوریتم‌های فراابتکاری تصادفی توسعه داده شده‌اند که مسئله را همانند یک جعبه سیاه در نظر می‌گیرند و با حفظ تعادل بین پویش و انتفاع به دنبال جواب مسئله میگردند. این الگوریتم‌ها، ماهیت تصادفی دارند وراه‌حل‌های کلی را تضمین نمی‌کنند و ممکن است در هر اجرا نتایج متفاوتی ایجاد کنند. با توجه به قضیه $no\ free\ lunch$، که بیان می کند هیچ الگوریتم موثر جهانی برای همه مسائل بهینه سازی وجود ندارد، محققان به معرفی الگوریتم های جدید الهام گرفته از طبیعت ادامه می دهند. الگوریتم یوزپلنگ نیز در همین راستا، و با الهام از طبیعت و بر اساس استراتژی‌های شکار یوزپلنگ پیشنهاد شده است.

بنابراین **الگوریتم یوزپلنگ** یک الگوریتم فراابتکاری برگرفته از طبیعت است که برای حل مسائل بهینه سازی در مقیاس‌های بزرگ طراحی شده است. ایده اصلی این الگوریتم از استراتژی یوزپلنگ‌ها برای شکار گرفته شده است.

یوزپلنگ‌ها به طور کلی سه استراتژی اصلی برای شکار طعمه دارند که عبارتند از: جستجو، نشستن و انتظار، و حمله. در این الگوریتم از این استراتژی‌ها استفاده شده. علاوه بر آن، استراتژی رها کردن طعمه و بازگشت به خانه نیز به فرآیند شکار اضافه شده تا تنوع جمعیت و عملکرد همگرایی بهبود یابد.

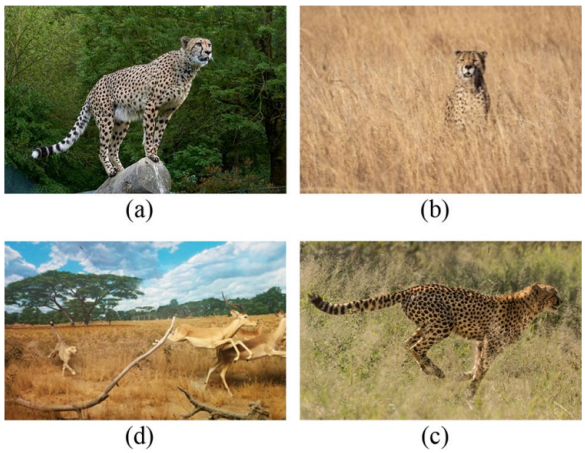


**Figure 1.** Hunting behavior of cheetahs: (**a**) searching for prey (scanning mode), (**b**) sitting-and-waiting (hiding), (**c**) rushing and (**d**) capturing.

تاکنون الگوریتم‌های هوش جمعی زیادی از رفتارهای شکار و جستجوی حیوانات در طبیعت الهام گرفته اند، اما یوزپلنگ‌ها ویژگی‌های شکار منحصربه‌فرد خود را دارند، به عنوان مثال برخی از شکارچیان می توانند طعمه را به صورت انفرادی و یا گله ای شکار کنند و سایر اعضا ممکن است در روند شکار شرکت نکنند. علاوه بر این، در برخی موارد، تعداد کمی از شکارچیان می توانند یک منطقه شکار بزرگ را پوشش دهند. این ویژگی‌ها پژوهشگران را برانگیخت تا این رفتارها را از نزدیک مطالعه کنند و یک الگوریتم بهینه‌سازی بر اساس آن ایجاد کنند. بهینه ساز یوزپلنگ (CO) از تکنیک های ساده استفاده می کند و فرآیند شکار را با استفاده از استراتژی های نشستن و انتظار، حمله و بازگشت مدل‌سازی می‌کند. بر خلاف روش های دیگر که بر معادلات پیچیده تکیه می کنند، الگوریتم CO از طریق این استراتژی‌های شکار کارایی خود را افزایش می‌دهد.

## ۱.۱ مدل ریاضی و الگوریتم

هنگامی که یک یوزپلنگ در حال گشت زنی یا اسکن محیط اطراف خود است، امکان تشخیص طعمه وجود دارد. یوزپلنگ با دیدن طعمه ممکن است در جای خود بنشیند و صبر کند تا طعمه به آن نزدیک شود و سپس حمله را آغاز کند. حالت حمله شامل دو فاز دنبال کردن سریع و گرفتن است. یوزپلنگ ممکن است به دلایل مختلفی مانند محدودیت انرژی، فرار سریع طعمه و غیره از شکار دست بکشد.سپس ممکن است برای استراحت به خانه برگردند و شکار جدیدی را شروع کنند. یوزپلنگ با ارزیابی طعمه،

وضعیت، منطقه و فاصله‌اش با طعمه، یکی از این راهکارها را انتخاب می کند. به طور کلی استراتژی‌های استفاده شده در الگوریتم یوزپلنگ شامل این موارد است:

- **جستجو**: یوزپلنگ ها برای یافتن طعمه خود باید در قلمرو خود (فضای جستجو) یا محیط اطراف جستجو کنند.
- **نشستن و انتظار**: زمانی که طعمه شناسایی شده، اما وضعیت مناسب نیست، یوزپلنگ‌ها ممکن است بنشینند و منتظر نزدیک شدن طعمه یا بهتر شدن وضعیت بمانند.
- **حمله**: این استراتژی دو مرحله اساسی دارد:

  - **دنبال کردن سریع**: زمانی که یوزپلنگ تصمیم به حمله می گیرد، با حداکثر سرعت به سمت طعمه می شتابد.

  - **گرفتن**: یوزپلنگ با نزدیک شدن به طعمه از سرعت و انعطاف پذیری خود برای گرفتن طعمه استفاده کرد.

- **رها کردن طعمه و بازگشت به خانه**: دو موقعیت برای این استراتژی در نظر گرفته شده است. اگر یوزپلنگ در شکار طعمه ناموفق بود، باید موقعیت خود را تغییر دهد یا به قلمرو خود بازگردد. همچنین در مواردی که در یک بازه زمانی شکار موفقی نداشته باشد، ممکن است موقعیت خود را به محل آخرین طعمه کشف شده تغییر دهد و اطراف آن را جستجو کند.

در ادامه جزئیات کامل مدل‌های ریاضی شکار که نام برده شدند توضیح داده می‌شوند.

**استراتژی جستجو**

یوزپلنگ‌ها به دو طریق به دنبال طعمه می گردند. یا در حالت نشسته محیط را اسکن میکنند یا به طور فعال در اطراف آن گشت زنی میکنند. حالت اسکن زمانی مناسب‌تر است که طعمه‌ها متراکم‌تر و در حال راه رفتن و چرا در دشت باشند. از طرفی اگر طعمه پراکنده و فعال باشد انتخاب جستجوی فعال که نیاز به انرژی بیشتری نسبت به حالت اسکن دارد بهتر است. بنابراین، در طول دوره شکار، بسته به وضعیت طعمه، پوشش منطقه و وضعیت خود یوزپلنگ، یکی از این دو حالت جستجو ممکن است توسط یوزپلنگ انتخاب شود. برای مدل‌سازی ریاضی این استراتژی جستجوی یوزپلنگ،

$X_{i,j}^t$ را موقعیت فعلی یوزپلنگ $i$ام در ترتیب (آرایش) $j$ام تعریف میکنیم. که در آن $i$ از $1$ تا $n$ است که $n$ تعداد جمعیت یوزپلنگ‌ها و $j$ نیز از $1$ تا $D$ است که $D$ بعد مسئله بهینه سازی است. در واقع، هر یوزپلنگ موقعیت‌های متفاوتی را در برخورد با طعمه‌های مختلف تجربه می‌کند. هر طعمه یک موقعیت در یک متغیر تصمیم است که مطابق با بهترین راه حل است. در نتیجه، تساوی جستجوی تصادفی زیر برای بروزرسانی موقعیت یوزپلنگ‌ها معرفی میشود:

$$X_{i,j}^{t+1} = X_{i,j}^t + \hat{r}_{i,j}^{-1}.\alpha_{i,j}^t \quad (۱)$$

که در آن $X_{i,j}^{t+1}$ و $X_{i,j}^t$ به ترتیب موقعیت های بعدی و فعلی یوزپلنگ $i$ام در چینش(آرایش) $j$ام هستند. شاخص $t$ نشان دهنده زمان فعلی شکار و $T$ حداکثر مدت زمان شکار است. $\hat{r}_{i,j}^{-1}$ و $\alpha_{i,j}^t$ به ترتیب پارامتر تصادفی سازی و طول گام برای یوزپلنگ $i$ در آرایش $j$ هستند. درواقع جمله دوم عبارت تصادفی سازی است که در آن پارامتر تصادفی r معمولاً عددی تصادفی از یک توزیع نرمال استاندارد است. طول گام $\alpha_{i,j}^t$ را در بیشتر موارد می‌توان روی $0.001 \times \frac{t}{T}$ تنظیم کرد زیرا یوزپلنگ ها جستجوگرهای آهسته‌ای هستند.

### استراتژی نشستن و انتظار

در طول حالت جستجو، طعمه ممکن است در معرض دید یوزپلنگ قرار گیرد. در این شرایط هر حرکت یوزپلنگ ممکن است طعمه را از حضور خود آگاه کند و منجر به فرار طعمه شود. برای جلوگیری از این موضوع، یوزپلنگ ممکن است تصمیم بگیرد (با دراز کشیدن روی زمین یا پنهان شدن در میان بوته ها) کمین کند تا طعمه به اندازه کافی نزدیک شود. بنابراین، در این حالت، یوزپلنگ در موقعیت خود باقی می ماند و منتظر نزدیک شدن طعمه می شود. این رفتار را می توان به صورت زیر مدل کرد:

$$X_{i,j}^{t+1} = X_{i,j}^t \quad (۲)$$

### استراتژی حمله

وقتی یک یوزپلنگ تصمیم به حمله می‌گیرد، با تمام سرعت به سمت طعمه می‌رود. پس از مدتی طعمه متوجه حمله یوزپلنگ می شود و شروع به فرار می کند. یوزپلنگ با چشمان تیزبین خود به سرعت طعمه را تعقیب می کند. به عبارت دیگر، یوزپلنگ موقعیت شکار را دنبال می کند و جهت حرکت خود را به گونه ای تنظیم می کند که

در یک نقطه راه شکار را مسدود کند. از آنجایی که یوزپلنگ با حداکثر سرعت به فاصله کمی از طعمه رسیده است، طعمه باید برای زنده ماندن موقعیت خود را به شکل ناگهانی تغییر دهد، یعنی موقعیت بعدی یوزپلنگ نزدیک آخرین موقعیت شکار است. همچنین، یک یوزپلنگ احتمالاً در استراتژی حمله ای که کاملاً با شکار طبیعی یوزپلنگ‌ها مطابقت دارد، شرکت نمی کند. در روش شکار گروهی، هر یوزپلنگ ممکن است موقعیت خود را بر اساس طعمه در حال فرار و موقعیت رهبر و موقعیت یوزپلنگ کناری تنظیم کند. این تاکتیک‌های حمله یوزپلنگ‌ها به صورت ریاضی به صورت زیر تعریف می شوند:

$$X_{i,j}^{t+1} = X_{B,j}^{t} + \hat{r}_{i,j}.\beta_{i,j}^{t} \quad (۳)$$

که در آن $X_{B,j}^{t}$ موقعیت فعلی طعمه در ترتیب $j$ام است. $\hat{r}_{i,j}$ و $\beta_{i,j}^{t}$ به ترتیب عامل چرخش و ضریب تعامل مرتبط با یوزپلنگ $i$ در ترتیب $j$ هستند. دلیل استفاده از $X_{B,j}^{t}$ در $(3)$ این است که در حالت حمله، استراتژی هجوم یوزپلنگ‌ها با استفاده از حداکثر سرعت به آنها کمک می کند تا در مدت زمان کوتاهی تا حد امکان به موقعیت طعمه نزدیک شوند. از این رو، این مقاله موقعیت جدید یوزپلنگ $i$ام را در حالت حمله بر اساس موقعیت فعلی طعمه محاسبه می کند. در جمله دوم، $\beta_{i,j}^{t}$ تعامل بین یوزپلنگ‌ها یا بین یوزپلنگ و رهبر را در حالت گرفتن را منعکس می‌کند. از نظر ریاضی، این عامل را می توان به عنوان تفاوت بین موقعیت یوزپلنگ کناری، $X_{k,j}^{t}$، و موقعیت یوزپلنگ $i$ام، $X_{i,j}^{t}$ تعریف کرد. ضریب چرخشی $\hat{r}_{i,j}$ یک عدد تصادفی است که برابر با $|r_{i,j}|^{exp(r_{i,j}/2)}sin(2\pi r_{i,j})$ است. این عامل چرخش های تند یوزپلنگ ها را در حالت گرفتن منعکس می کند. در این الگوریتم $r_{i,j}$ اعداد تصادفی از یک توزیع نرمال استاندارد هستند.

## ۲.۱ فرضیات

بر اساس رفتار یوزپلنگ‌ها در شکار، مفروضات زیر برای الگوریتم CO در نظر گرفته شده است:

۱. جمعیت یوزپلنگ‌ها با ردیف‌هایی نشان داده می شود که هر ردیف مربوط به یک یوزپلنگ در حالت‌های متفاوت است. ستون‌ها آرایش(ترتیب) خاصی از یوزپلنگ‌ها را در رابطه با طعمه (بهترین راه حل‌ها برای هر متغیر تصمیم گیری) نشان می‌دهند. یوزپلنگ‌ها با هدف گرفتن بهترین امتیاز از هر متغیر، طعمه خود را دنبال می‌کنند. هدف گرفتن طعمه در هر آرایش است و عملکرد یک یوزپلنگ

بر اساس ارزش تابع برازش آن در همه آرایش ها ارزیابی می شود. عملکرد بالاتر نشان دهنده احتمال بیشتر موفقیت در شکار است.

۲. در یک سناریوی واقعی شکار گروهی، هر یوزپلنگ متفاوت از دیگران واکنش نشان می‌دهد. در هر آرایش یوزپلنگ‌های مختلف ممکن است در حالت حمله، جستجو، نشستن و انتظار یا سایر حالت‌ها باشند. انرژی یوزپلنگ‌ها مستقل از طعمه است. مدل‌سازی متغیرهای تصمیم‌گیری به عنوان آرایش یوزپلنگ‌ها، همراه با پارامترهای تصادفی، از همگرایی زودرس در طول فرآیند تکامل جلوگیری می‌کند. این متغیرهای تصادفی به عنوان منبع انرژی برای یوزپلنگ‌ها در طول شکار عمل می‌کنند و گنجاندن آنها عملکرد بهینه سازی را بهبود می‌بخشد. در استراتژی حمله، جهت یوزپلنگ‌ها به طعمه بستگی دارد، در حالی که در استراتژی جستجو حرکات آنها رفتار کاملا تصادفی را نشان می‌دهد.

۳. رفتار یوزپلنگ در هنگام جستجو یا حمله کاملا تصادفی فرض می‌شود، در حالی که در حالت دنبال کردن سریع و گرفتن، طعمه به شدت تغییر جهت می‌دهد. پارامتر تصادفی سازی و پارامتر ضریب تعامل، همراه با متغیرهای کاملا تصادفی، به یک فرآیند بهینه سازی موثر کمک می‌کنند. این ملاحظات مدلسازی دقیق فرآیند شکار را تضمین می‌کند.

۴. انتخاب استراتژی جستجو یا حمله در فرآیند شکار در ابتدا تصادفی است، اما با گذشت زمان، با کاهش سطح انرژی یوزپلنگ، احتمال استراتژی جستجو بیشتر می‌شود. در برخی موارد، به استراتژی جستجو در گام های اولیه اولویت بیشتری داده می‌شود، و در عوض استراتژی حمله برای مقادیر بیشتری از زمان ($t$) انتخاب می‌شود. با فرض اینکه $r_2$ و $r_3$ دو عدد تصادفی از بازه $[0, 1]$ باشند، اگر $r_2 \leq r_3$ استرتژی نشستن و انتظار انتخاب می‌شود، در غیر این صورت یکی از استرتژی‌های جستجو و حمله بر اساس عدد تصادفی $H = e^{2(-/T)}(2r_1 - 1)$ انتخاب می‌شود. که $r_1$ نیز عددی تصادفی از بازه $[0, 1]$ است.

۵. استراتژی‌های اسکن و نشستن و انتظار در الگوریتم CO مترادف هستند و نشان دهنده ثابت ماندن یوزپلنگ (عامل جستجو) است.

۶. اگر رهبر در چند فرآیند شکار متوالی موفق به شکار نشود، موقعیت یک یوزپلنگ که به طور تصادفی انتخاب شده است به عنوان آخرین موقعیت شکار موفق

شناخته شده (موقعیت شکار) تغییر می‌کند. حفظ موقعیت طعمه میان یک جمعیت کوچک، بخش انتفاع الگوریتم را افزایش می‌دهد.

۷. هر گروه از یوزپلنگ‌ها به دلیل محدودیت انرژی، یک محدودیت زمانی برای شکار دارند. اگر گروهی نتواند در یک دوره شکار به موفقیت دست یابد، طعمه فعلی را رها کرده و برای استراحت به محدوده خانه خود (موقعیت اولیه) باز می گردد. جایگاه رهبر نیز در این فرآیند به روز می شود. این استراتژی از به دام افتادن الگوریتم در نقاط بهینه محلی جلوگیری می کند.

۸. در هر تکرار، تنها بخشی از اعضا در فرآیند تکامل شرکت می‌کنند.

## ۳.۱ شبه کد

شبه کد زیر معادل توضیحات داده شده در بخش‌های قبل است:

**Algorithm 1:** *The CO Algorithm*

Define the problem data, dimension ($D$), and the initial population size ($n$)
Generate the initial population of cheetahs $X_i (i = 1, 2, \dots, n)$ and evaluate the fitness of each cheetah
Initialize the population's home, leader and prey solutions
$t \leftarrow 0$
$it \leftarrow 1$
$MaxIt \leftarrow$ desired maximum number of iterations
$T \leftarrow 60 \times \lceil D/10 \rceil$
**while** $it \leq MaxIt$ **do**
    Select $m$ $(2 \leq m \leq n)$ members of cheetahs randomly
    **for** each member $i \in m$ **do**
    Define the neighbor agent of member $i$
        **for** each arbitrary arrangement $j \in \{1, 2, \dots, D\}$ **do**
            Calculate $\hat{r}$, $\check{r}$, $\alpha$, $\beta$, and $H$
            $r_2, r_3 \leftarrow$ random numbers are chosen uniformly from 0 to 1
            **if** $r_2 \leq r_3$ **then**
                $r_4 \leftarrow$ a random number is chosen uniformly from 0 to 3
                **if** $H \geq r_4$ **then**
                    Calculate the new position of member $i$ in arrangement $j$ using Equation (3) //Attack
                **else**
                    Calculate the new position of member $i$ in arrangement $j$ using Equation (1) //Search
                **end**
            **else**
                Calculate the new position of member $i$ in arrangement $j$ using Equation (2) //Sit-and-wait
            **end**
        **end**
        Update the solutions of member $i$ and the leader
    **end**
    $t \leftarrow t + 1$
    **if** $t > rand \times T$ and the leader position doesn't change for a time, **then** //Leave the prey and go back home
        Implement the leave the prey and go back home strategy and change the leader position
        Substitute the position of member $i$ by the prey position
        $t \leftarrow 0$
    **end**
    $it \leftarrow it + 1$
    Update the prey (global best) solution
**end**

# ۲ توضیح پیاده سازی الگوریتم

بخش پیاده سازی این الگوریتم شامل چهار فایل است:

- main.py
- CO.py
- fitness_functions.py
- output.txt

فایل CO.py شامل پیاده سازی الگوریتم یوزپلنگ است و فایل fitness_functions.py شامل توابع هدفی است که میخواهیم آنها را بر روی الگوریتم اجرا کنیم. فایل main.py الگوریتم یوزپلنگ را روی توابع هدف اجرا میکند و خروجی مناسب را تولید میکند. و در نهایت فایل output.txt شامل خروجی یک نمونه اجرای برنامه است.

## ۱.۲ فایل CO

این فایل شامل تابع `cheetah_optimizer` است. که با دریافت اندازه جمعیت اولیه (n) و حد بالا (ub) و پایین (lb) و بعد مسئله (dim) و تابع هدف آن (fitness_function) الگوریتم یوزپلنگ را اجرا میکند و در نهایت بهترین جواب پیدا شده را به همراه تعداد تکرار انجام شده برمیگرداند. همچنین در طول اجرای برنامه پس از هر ۵۰۰ تکرار، بهترین پاسخ بدست آمده را در خروجی چاپ میکند.

برای تولید اعداد تصادفی و کار با آرایه‌ها از کتابخانه `numpy` استفاده شده است.

در ابتدای برنامه برای انجام راحت‌تر محاسبات متغیرهای lu و bu را به بعد dim میرسانیم تا جمع و ضرب آنها با بقیه متغیرها راحت‌تر شود (خط ۵ تا ۷).

سپس برای راه‌اندازی اولیه، بهترین جواب را بینهایت (inf) فرض میکنیم و یک جمعیت اولیه با استفاده از موقعیت‌های تصادفی بین lb و ub میسازیم و برازندگی (cost) هر یوزپلنگ را محاسبه میکنیم. همچنین متغیر `best_solution` را با بهترین جوابی که در این مرحله پیدا می‌شود آپدیت میکنیم. (خط ۱۱ تا ۱۸)

سپس در خط‌های ۲۰ تا ۳۰ متغیرهای اولیه مورد نیاز را طبق توضیحات الگوریتم مقدار دهی اولیه میکنیم.

از خط ۳۱ حلقه اصلی الگوریتم شروع میشود. این حلقه while حداکثر به اندازه max_it تکرار می‌شود. در ابتدای حلقه تعداد m یوزپلنگ به طور تصادفی انتخاب می‌شوند تا در فرآیند شکار شرکت کنند.

سپس برای هر یوزپلنگ ابتدا یوزپلینگ کناری آن را برای محاسبه ضریب تعامل در ادامه انتخاب میکنیم، و بعد موقعیت یوزپلنگ و یوزپلنگ کناری و رهبر و میزان برازندگی رهبر را در متغیرهای مربوطه ذخیره میکنیم. بخش بعدی توسط نویسنده‌های مقاله برای بهبود متغیرهای kk و X فراهم شده که میتواند باعث بهبود عملکرد الگوریتم شود. پس از آن با استفاده از تابع permutation یک ترتیب (آرایش) تصادفی میسازیم همچنین متغیر Z را به عنوان $X^{t+1}$ مقدار دهی اولیه میکنیم و سپس برای هر مقدار j از ترتیبی که ساختیم ابتدا مقادیر تصادفی سازی ($\hat{r}$) و طول گام ($\alpha$) و r۱ (r) و r۲ (r_check) و ضریب تعامل ($\beta$) و متغیر تصمیم ($H$) را طبق توضیحات قبلی میسازیم و سپس با استفاده از آنها و با دستورات if و else مشخص میکنیم که کدام یک از استراتژی‌های جستجو و حمله و نشستن و انتظار باید اجرا شود. و با مشخص شدن استراتژی مورد نظر مقدار Z[j] یا همان $X_{i,j}^{t+1}$ با استفاده از متغیرهایی که پیش‌تر تعریف کردیم و ۳ تساوی‌ای که برای الگوریتم یوزپلنگ تعریف شده بود آپدیت می‌شود. پس از اینکه این کارها برای تمام مقادیر j انجام شدند مقادیر ذخیره شده در Z با حدهای بالا و پایین چک می‌شوند و درصورتی که مقداری در بازه تعریف شده قرار نداشت با یک مقدار تصادفی از آن بازه تعویض می‌شود. سپس میزان برازندگی پاسخ جدیدی که محاسبه شده (Z) با استفاده از fitness_function محاسبه می‌شود و اگر از جواب قبلی بهتر بود با آن جایگزین می‌شود و همچنین بهترین پاسخ پیدا شده تا اینجا نیز در صورت نیاز آپدیت می‌شود. (خط‌های ۳۴ تا ۱۱۲)

سپس شرایط رها کردن طعمه و بازگشت به خانه چک می‌شود. در ابتدا شرط محدودیت بازه زمانی با استفاده از زمان طی شده (t) و محدودیت زمان (T) چک می‌شود و در صورتی که برقرار بود بررسی می‌شود که آیا شکار قبلی با شکست مواجه شده یا خیر، که در صورت برقرار بودن این شرط یک رهبر دیگر انتخاب میشود و همچنین تعدادی از یوزپلنگ‌ها به مکان اولیه خود باز میگردند و زمان t نیز بازنشانی می‌شود. (خط ۱۱۶ تا ۱۳۹)

در انتها نیز بهترین جواب در طی تمام مراحل در صورت نیاز بروزرسانی می‌شود و همچنین اگر در یک تکرار مضرب ۵۰۰ بودیم بهترین جواب تا آن مرحله در خروجی چاپ می‌شود. (خط ۱۴۲ تا ۱۵۱)

پس از پایان حلقه while مقدار بهترین جواب و تعداد تکرارها بازگردانده می‌شود.

## ۲.۲ فایل fitness_functions

این فایل شامل توابع هدف است و یک تابع اصلی به نام `fitness_function` در آن تعریف شده که با دریافت نام تابع هدف مورد نیاز ( F۱ یا F۲ یا ... ) خود تابع هدف (`F_obj`) و مقادیر حد پایین (`LB`) و حد بالا (`UB`) و بعد مسئله (`Dim`) را که مورد نیاز الگوریتم یوزپلنگ هستند برمیگرداند.

## ۳.۲ فایل main

این فایل شامل لیست `POPULATION_SIZE` است که اندازه جمعیت اولیه‌ای که برای حل توابع هدف F۱ تا F۲۳ استفاده می‌شود را مشخص می‌کند. این فایل از توابع `fitness_function` و `cheetah_optimizer` که در فایل‌های قبل تعریف شده‌اند استفاده میکند و پاسخ هر مسئله به همراه تعداد تکرار رسیدن به جواب را در خروجی چاپ میکند. (مسئله مینیمم سازی توابع هدف است).

## ۴.۲ فایل output

این فایل شامل یک نمونه از خروجی برنامه main است.

# ۳ نتایج اجرای الگوریتم

| نام تابع | بعد مسئله | # تکرار | # جمعیت | جواب برنامه | جواب بهینه |
|---|---|---|---|---|---|
| $F1$ | 10 | 20001 | 10 | $2.1810949585806744 \times 10^{-134}$ | 0 |
| $F2$ | 10 | 25001 | 10 | $1.533094721631549 \times 10^{-40}$ | 0 |
| $F3$ | 10 | 50001 | 10 | $3.627803698869468 \times 10^{-27}$ | 0 |
| $F4$ | 10 | 50000 | 10 | $5.27023478401582 \times 10^{-9}$ | 0 |
| $F5$ | 10 | 11112 | 22 | 0.006775473583644024 | 0 |
| $F6$ | 10 | 49999 | 10 | $1.1247430875221457 \times 10^{-30}$ | 0 |
| $F7$ | 10 | 5264 | 40 | 1.7127963504981056 | 0 |
| $F8$ | 10 | 19997 | 10 | $-4189.828872724336$ | $-418.9829 \times n$ |
| $F9$ | 10 | 16667 | 40 | 0.9949590917825049 | 0 |
| $F10$ | 10 | 9092 | 30 | 0.0563568407576267 | 0 |
| $F11$ | 10 | 33332 | 40 | 0.049212707576478665 | 0 |
| $F12$ | 10 | 16664 | 30 | 0.7352588238435913 | 0 |
| $F13$ | 10 | 9091 | 30 | $1.039911484028557 \times 10^{-26}$ | 0 |
| $F14$ | 2 | 2499 | 50 | 0.99800383779445 | 1 |
| $F15$ | 4 | 2668 | 100 | $1.0907510333799825 \times 10^{-24}$ | 0.00030 |
| $F16$ | 2 | 2498 | 50 | $-1.0316284534898774$ | $-1.0316$ |
| $F17$ | 2 | 910 | 50 | 0.397887357729763 | 0.398 |
| $F18$ | 2 | 2499 | 50 | 2.999999999999926 | 3 |
| $F19$ | 3 | 716 | 100 | $-3.8627821478207554$ | $-3.86$ |
| $F20$ | 6 | 2309 | 70 | $-3.2031020502634227$ | $-.32$ |
| $F21$ | 4 | 1083 | 300 | $-10.153199679058229$ | $-10.1532$ |
| $F22$ | 4 | 432 | 200 | $-10.402940566818664$ | $-10.4028$ |
| $F23$ | 4 | 716 | 200 | $-10.536409816692048$ | $-10.5363$ |