به نام خدا

الگوریتم گرگ خاکستری

در این گزارش به توضیح الگوریتم بهینه‌سازی گرگ خاکستری که یک از الگوریتم‌های meta-heuristic است می‌پردازیم. این الگوریتم توسط (سید علی میرجلیلی [1]) در سال ۲۰۱۴ ابداع شده است که با الهام از روش شکار گرگ‌های خاکستری پیاده‌سازی و ارایه شده است.
الگوریتم‌های بهینه سازی در دو دهه‌ی اخیر بسیار محبوب شده‌اند که دلیل اصلی این امر در ادامه بیان شده است.

1. سادگی کار با این الگوریتم‌ها
2. انعطاف پذیری
3. عاری از مشتق بودن
4. حل مشکل اکسترمم‌های محلی

از آنجایی که شما می توانید مستقل از مسئله با استفاده از این روش‌ها جواب بهینه را پیدا کنید در رشته‌ها و زمینه‌های مختلف این روش‌ها برای بهینه کردن و پیدا کردن جواب کاربرد دارند.
از جمله الگوریتم‌های معروف که پیاده‌سازی‌ها و کتابخانه‌های متعددی برای آن وجود دارد الگوریتم ژنتیک است.
به طور کلی الگوریتم‌های meta-heuristic در دو دسته قرار می‌گیرند.

1. با یک جواب
2. بر پایه چند جواب

برای مثال الگوریتم simulated Annealing در دسته‌ی اول قرار میگیرد که در آن جستجو با یک جواب پایه شروع می‌شود. ولی الگوریتم ga و گرگ خاکستری با استفاده از **چندین کاندید برای جواب** به دنبال حل مسئله هستند.به نظر این ویژگی فضای خوبی برای پیاده‌سازی الگوریتم به صورت **موازی** فراهم میکند. تا محاسبات با سرعت بیشتری انجام شود.

گرگ‌های خاکستری گونه‌ای از پستانداران هستند که به واسطه زندگی در اجتماع و کار تیمی جز بهترین شکارچیان محسوب می‌شوند.
الگوریتمی که در ادامه به توضیح آن خواهیم پرداخت با استفاده از روش شکار و سلسله مراتبی که در زندگی اجتماعی گرگ‌ها وجود دارد کار می‌کند. در هر گله گرگ حدود ۵ تا ۱۲ گرگ وجود دارند. در هر گله یک گرگ آلفا که مدیریت گله را به عهده دارد وجود دارد. همچنین یک گرگ بتا که جانشین‌گرگ آلفا است و دومین گرگ‌قوی در گله است. گرگ دلتا نیز سومین گرگ قوی محسوب می‌شود و دیگر گرگ‌ها با عنوان اومگا شناخته می‌شوند. بعد از توضیح ساختار سلسله مراتبی در اجتماع گرگ‌ها نوبت به توضیح مراحل شکار می رسد. شکار گله شامل چند مرحله است که در ذیل آمده است:

1. ردیابی، تعقیب و نزدیک‌شدن به طعمه
2. دنبال کردن، حلقه زدن و آزار دادن تا زمانی که طعمه از حرکت به‌ایستد
3. حمله به طعمه

تصویر ۱

در شکار گرگ‌ها امگا از ابتدا از آلفا و سپس از بتا و دلتا پیروی می کنند.(تصویر 2)

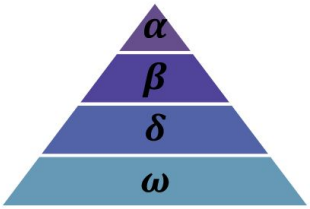


Fig. 1. Hierarchy of grey wolf (dominance decreases from top down).

تصویر 2

مدل‌سازی ریاضی

در این مسئله در هر iteration گرگ‌های آلفا، بتا و دلتا به عنوان نقاطی از مجموعه در نظر گرفته می‌شوند که بهترین مقدار را برای تابع ارزیابی ما دارند.

در ادامه قسمت حلقه زدن به دور طعمه توضیح داده‌شده است که برای این قسمت معادلات زیر بیان شده است.

$$\vec{D} = |\vec{C} \cdot \vec{X}_p(t) - \vec{X}(t)|$$

$$\vec{X}(t+1) = \vec{X}_p(t) - \vec{A} \cdot \vec{D}$$

$$\vec{A} = 2\vec{a} \cdot \vec{r}_1 - \vec{a}$$

$$\vec{C} = 2 \cdot \vec{r}_2$$

در معادلات بالا t شماره تکرار رو مشخص میکند. A و C بردارهای ضرایب هستند و X_p موقعیت طعمه را مشخص میکند و X موقعیت گرگ خاکستری را مشخص می کند. عناصر بردار a به صورت خطی از ۲ به ۰ حرکت می‌کنند و در طول تکرارهای الگوریتم کاهش می‌یابد. و بردارها r نیز بردارهای رندم مستقلی بین ۰ و ۱ هستند.

همانطور که مشخص است موقعیت گرگ با توجه به موقعیت هدف تغییر می‌کند و شما چند موقعیت‌های احتمالی که ممکن است رخ دهد را در تصویر ۳ مشاهده می‌کنید.

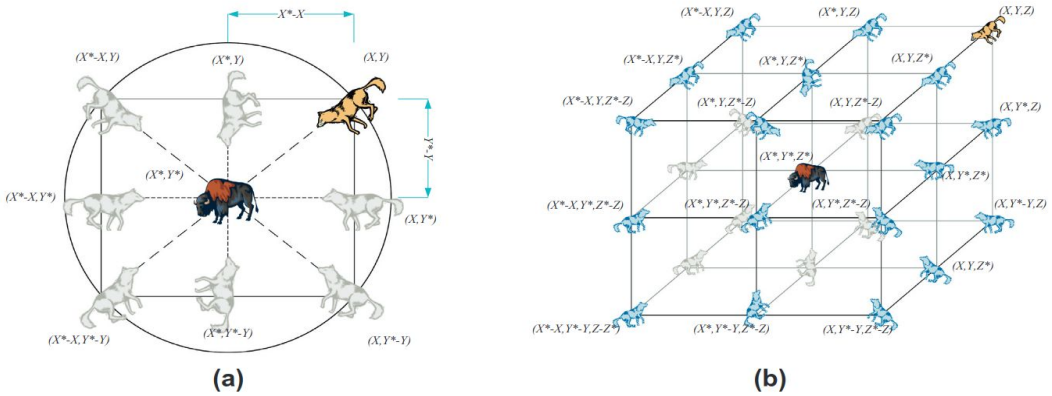


Fig. 3. 2D and 3D position vectors and their possible next locations.

تصویر ۳

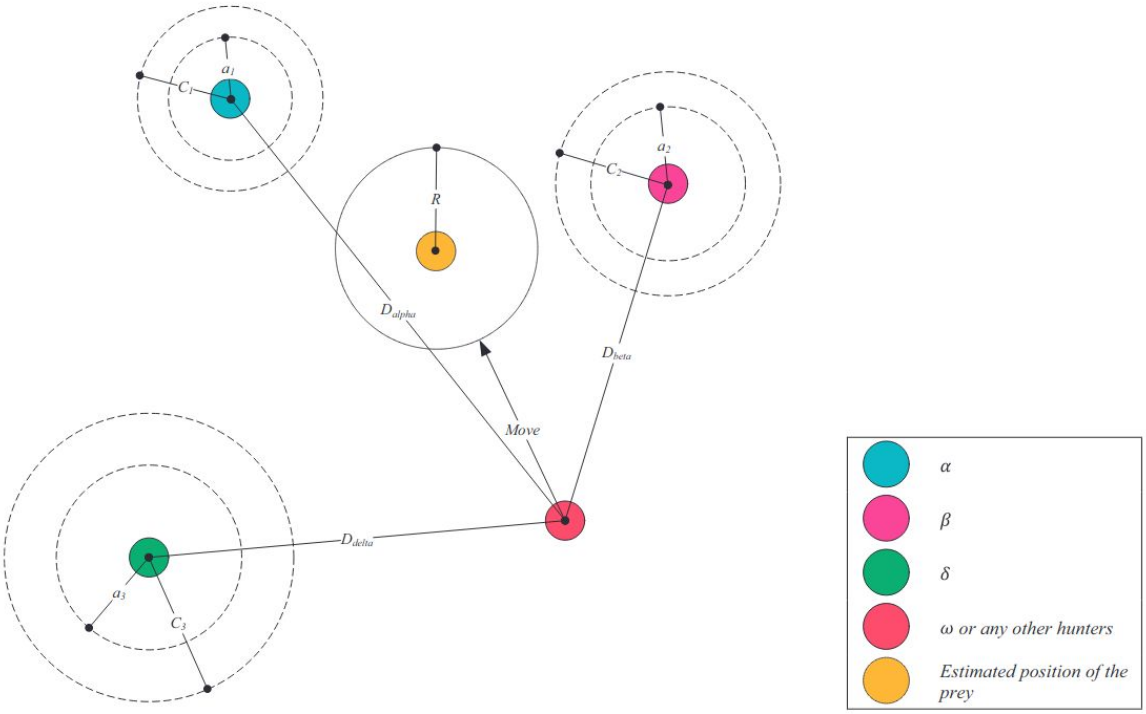
در حیات وحش شکار تنها با دستور گرگ آلفا صورت میگیرد و در بعضی مواقع بتا و دلتا نیز به آلفا کمک می‌کنند. از آنجایی که ما مکان شکار یا در واقع نقطه optimum را نمی‌دانیم گرگ‌هایی که بهتری موقعیت را داشته باشند به عنوان آلفا و بتا و دلتا فرض می‌شوند و دیگر گرگ‌ها مجبور به پیروی از آن‌ها هستند. برای مدل سازی این بخش معادلات زیر ارایه شده است.

$$\vec{D}_\alpha = |\vec{C}_1 \cdot \vec{X}_\alpha - \vec{X}|, \vec{D}_\beta = |\vec{C}_2 \cdot \vec{X}_\beta - \vec{X}|, \vec{D}_\delta = |\vec{C}_3 \cdot \vec{X}_\delta - \vec{X}|$$

$$\vec{X}_1 = \vec{X}_\alpha - \vec{A}_1 \cdot (\vec{D}_\alpha), \vec{X}_2 = \vec{X}_\beta - \vec{A}_2 \cdot (\vec{D}_\beta), \vec{X}_3 = \vec{X}_\delta - \vec{A}_3 \cdot (\vec{D}_\delta)$$

$$\vec{X}(t+1) = \frac{\vec{X}_1 + \vec{X}_2 + \vec{X}_3}{3}$$

معادلات بیان می کنند که موقعیت بعدی گرگ خاکستری در واقع مکانی است است که در دایره‌ای که مرکز آن دایره به عنوان موقعیت شکار توسط سه گرگ برتر تخمین زده شده است و در تصویر ۴ می توانید این موضوع را به صورت دو بعدی مشاهده کنید.



تصویر ۴

به دلیل وجود مقدارهار رندم در معادلات ممکن است جابجایی‌ها به صورت مشخصی همگرا نباشند و یا اصلا جابجایی صورت نگیرد و یا هم چنین از هدف دور شوند. که این موردها باعث می‌شود الگوریتم پر optimum های محلی گیر نکند و به نزدیک شدن به جواب ادامه دهد.

در ادامه شبه کدی که برای الگوریتم آماده‌شده است را مشاهده می‌کنید که با استفاده از فرمول‌ها و مدل‌سازی‌های بالا پیاده‌سازی می‌شود و پیاده‌سازی این الگوریتم با زبان python در [2] موجود است و همچنین در [3] گیف مربوط به اجرای الگوریتم که با استفاده از الگوریتم به حل یک مسئله پرداخته شده است قرار گرفته است.

Initialize the grey wolf population X_i *(i = 1, 2, ..., n)*
Initialize a, A, and C
Calculate the fitness of each search agent
X_α*=the best search agent*
X_β*=the second best search agent*
X_δ*=the third best search agent*
while *(t < Max number of iterations)*
    for *each search agent*
        Update the position of the current search agent by equation (3.7)
    end for
    Update a, A, and C
    Calculate the fitness of all search agents
    Update X_α, X_β, *and* X_δ
    t=t+1
end while
return X_α

در [1] چندین تابع برای بررسی عملکرد این روش پیشنهاد شده است که این تابع‌ها به عنوان آزمون [4] در دیگر مقالات و روش‌ها نیز استفاده می‌شوند. در پروژه درس ما با پیاده‌سازی چندین تابع از دسته‌های مختلف از توابع بیان‌شده برای آزمون به بررسی نتایج کار خود می‌پردازیم و بررسی میکنیم که موازی‌سازی ما تا چه میزان به بهبود سرعت اجرا کمک‌کرده است و در ادامه باید تا الگوریتم را با استفاده از زبان C نیز پیاده‌سازی کنیم.

پیوست

1. https://www.sciencedirect.com/science/article/abs/pii/S0965997813001853
2. https://github.com/7ossam81/EvoloPy/blob/master/optimizers/GWO.py
3. https://raw.githubusercontent.com/HaaLeo/swarmlib/master/doc/example.gif
4. https://en.wikipedia.org/wiki/Test_functions_for_optimization