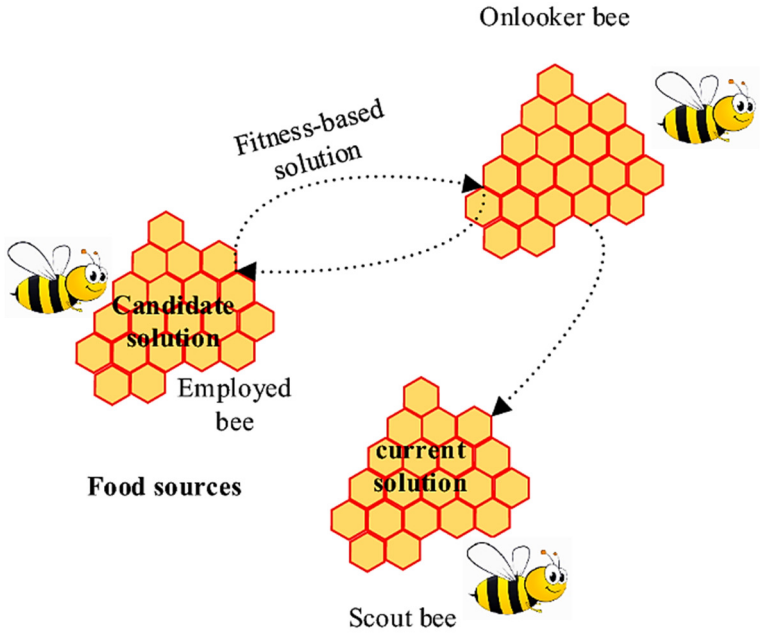
مهدی محمدی - شمارهٔ دانشجویی: ۴۰۱۱۲۰۹۴

الگوریتم کلونی زنبورهای مصنوعی

الگوریتم کلونی زنبورهای مصنوعی (ABC) یک تکنیک بهینه‌سازی است که رفتار جستجوی غذای زنبورهای عسل را شبیه‌سازی می‌کند و با موفقیت در مسائل مختلف عملی اعمال شده است. ABC متعلق به گروه الگوریتم‌های هوش ازدحامی است و توسط Karaboga در سال ۲۰۰۵ پیشنهاد شد.

کلونی زنبورهای مصنوعی از سه گروه زنبورعسل تشکیل شده است: زنبورهای کارگر، ناظر و زنبورهای پیشاهنگ. تمام زنبورهایی که در حال حاضر از یک منبع غذایی بهره‌برداری می‌کنند به‌عنوان کارگر شناخته می‌شوند. زنبورهای کارگر از منبع غذایی بهره‌برداری می‌کنند، اطلاعات مربوط به منبع غذایی را به کندو منتقل می‌کنند و این اطلاعات را با زنبورهای ناظر به اشتراک می‌گذارند. زنبورهای ناظر در کندو منتظر هستند تا اطلاعاتی در مورد منابع غذایی کشف‌شده توسط زنبورهای کارگر به اشتراک گذاشته شود و زنبورهای پیشاهنگ همیشه به دنبال منابع غذایی جدید در نزدیکی کندو خواهند بود. زنبورهای کارگر با رقصیدن در منطقه رقص تعیین‌شده در داخل کندو، اطلاعات مربوط به منابع غذایی را به اشتراک می‌گذارند. ماهیت رقص متناسب با محتوای شهد منبع غذایی است که فقط توسط زنبور رقصنده مورد استفاده قرار می‌گیرد. زنبورهای ناظر رقص را تماشا می‌کنند و با توجه به احتمالی متناسب با کیفیت آن منبع غذایی، منبع غذایی را انتخاب می‌کنند. بنابراین، منابع غذایی خوب در مقایسه با منابع غذایی بد، زنبورهای ناظر بیشتری را جذب می‌کنند. هرگاه از یک منبع غذایی به‌طور کامل بهره‌برداری شود، تمام زنبورهای کارگر مرتبط با آن، منبع غذایی را رها کرده و پیشاهنگ می‌شوند. زنبورهای پیشاهنگ را می‌توان به‌صورت انجام کار اکتشاف (Exploration) تجسم کرد، درحالی‌که زنبورهای کارگر و ناظر را می‌توان به‌صورت انجام کار بهره‌برداری (Exploitation) تجسم کرد.



شکل ۱ - زنبورهای کارگر، ناظر و پیشاهنگ.

Exploration یعنی توانایی تولید پاسخ‌های جدید و البته متفاوت، یا آزمایش طرحی که تابه‌حال مشابه آن را نداشتیم. این مفهوم دقیقاً مقابل مفهوم اکتشاف است .در الگوریتم ABC چرا به Exploration نیاز داریم؟ برای اینکه مطمئن شویم همه‌جا را بررسی کرده‌ایم .در الگوریتم ABC چرا به Exploitation نیاز داریم؟ برای اینکه مطمئن شویم به‌اندازه کافی زمان و انرژی برای بهتر شدن راه‌حل صرف کرده‌ایم .ما در بهینه‌سازی به هردوی این‌ها نیاز داریم ولی نمی‌توانیم به‌طور هم‌زمان هر دو را داشته باشیم باید کاری کنیم که در ابتدای کار الگوریتم Exploration بهتری داشته باشیم و در ادامه کار برای اطمینان بیشتر Exploitation بهتری داشته باشیم.

در الگوریتم ABC، هر منبع غذایی یک راه‌حل ممکن برای مسئلهٔ مورد بررسی است و مقدار شهد یک منبع غذایی نشان‌دهنده کیفیت راه‌حل است که با مقدار تناسب نشان داده می‌شود. تعداد منابع غذایی با تعداد زنبورهای کارگر برابر است و برای هر منبع غذایی دقیقاً یک زنبور کارگر وجود دارد و یک زنبور ناظر بسته به مقدار احتمال Pi مرتبط با آن منبع غذایی، منبع غذایی را انتخاب می‌کند.

رفتار هوشمند زنبورهای عسل را می‌توان در چند گام زیر خلاصه کرد:

۱. زنبورها تلاش می‌کنند تا به‌صورت تصادفی در محیط به دنبال منابع غذایی خوب بگردند (مقدار تناسب).
۲. پس از یافتن یک منبع غذایی، آن‌ها تبدیل به زنبورهای کارگر می‌شوند و شروع به استخراج غذا از منبع یافت شده می‌کنند.
۳. زنبور کارگر با شهد به کندو بازمی‌گردد و بار شهد خود را خالی می‌کند. پس از خالی کردن آن، می‌تواند مستقیماً به منبع کشف‌شده خود بازگردد یا اطلاعاتی که درباره منبع غذایی‌اش دارد را با اجرای یک رقص گردون در ناحیه رقص به اشتراک بگذارد.
٤. اگر یک منبع غذایی خالی شد، زنبوران کارگر به پیشاهنگ مبدل شده و به جست‌وجوی تصادفی برای منابع غذایی می‌پردازند.
۵. زنبورهای ناظر در کندو منتظر مانده و زنبورهای کارگر را در منابع غذایی گردآوری کرده‌شان مورد نظارت قرار می‌دهند و از میان منابع غذایی موجود با بیشترین سود، یک منبع را انتخاب می‌کنند.
٦. انتخاب منابع غذایی متناسب با کیفیت آن منبع (مقدار تناسب) است.

بااینکه سه نوع از زنبورهای عسل موجود در کلونی معرفی شدند، در مرحله پیاده‌سازی تنها دو نوع زنبور وجود دارد که زنبورهای کارگر و ناظر هستند. درواقع زنبور پیشاهنگ یک رفتار اکتشافی است که می‌تواند توسط زنبورهای کارگر و ناظر انجام شود.

در الگوریتم ABC، نیمه اول ازدحام شامل زنبورهای کارگر و نیمه دوم زنبورهای ناظر است. تعداد زنبورهای کارگر یا زنبورهای ناظر، با تعداد راه‌حل‌های موجود در ازدحام برابر است. ABC یک جمعیت اولیهٔ توزیع‌شدهٔ تصادفی از SN راه‌حل (منابع غذایی) تولید می‌کند، که در آن SN اندازه ازدحام را نشان می‌دهد.

فرض کنید X_j یک پاسخ کاندید به تصادفی انتخاب شده باشد ($i \neq j$)، k اندیس بُعد، انتخاب‌شده به‌صورت تصادفی از مجموعهٔ $\{1,2,\ldots,n\}$، و $\Phi_{i,k}$ یک عدد تصادفی در بازهٔ $[-1,1]$ باشد. هنگامی‌که یک پاسخ کاندید جدید X_i تولید می‌شود، یک انتخاب حریصانه استفاده می‌شود. اگر مقدار برازش V_i بهتر از والدش X_i باشد، آنگاه X_i توسط V_i به‌روزرسانی می‌شود؛ در غیر این صورت X_i بدون تغییر باقی می‌ماند. پس‌ازآنکه تمامی زنبورهای کارگر فرایند جستجو را به اتمام رساندند، آن‌ها اطلاعات منابع غذایی خودشان را از طریق یک رقص حرکتی با زنبورهای ناظر به اشتراک می‌گذارند.

زنبور ناظر اطلاعات شهد گرفته‌شده از همه زنبورهای کارگر را ارزیابی می‌کند و منبع غذایی را با احتمال مرتبط با مقدار شهد آن انتخاب می‌کند. این انتخاب احتمالی درواقع مکانیسم انتخاب چرخ رولت است که به‌صورت معادلهٔ زیر توصیف شده است:

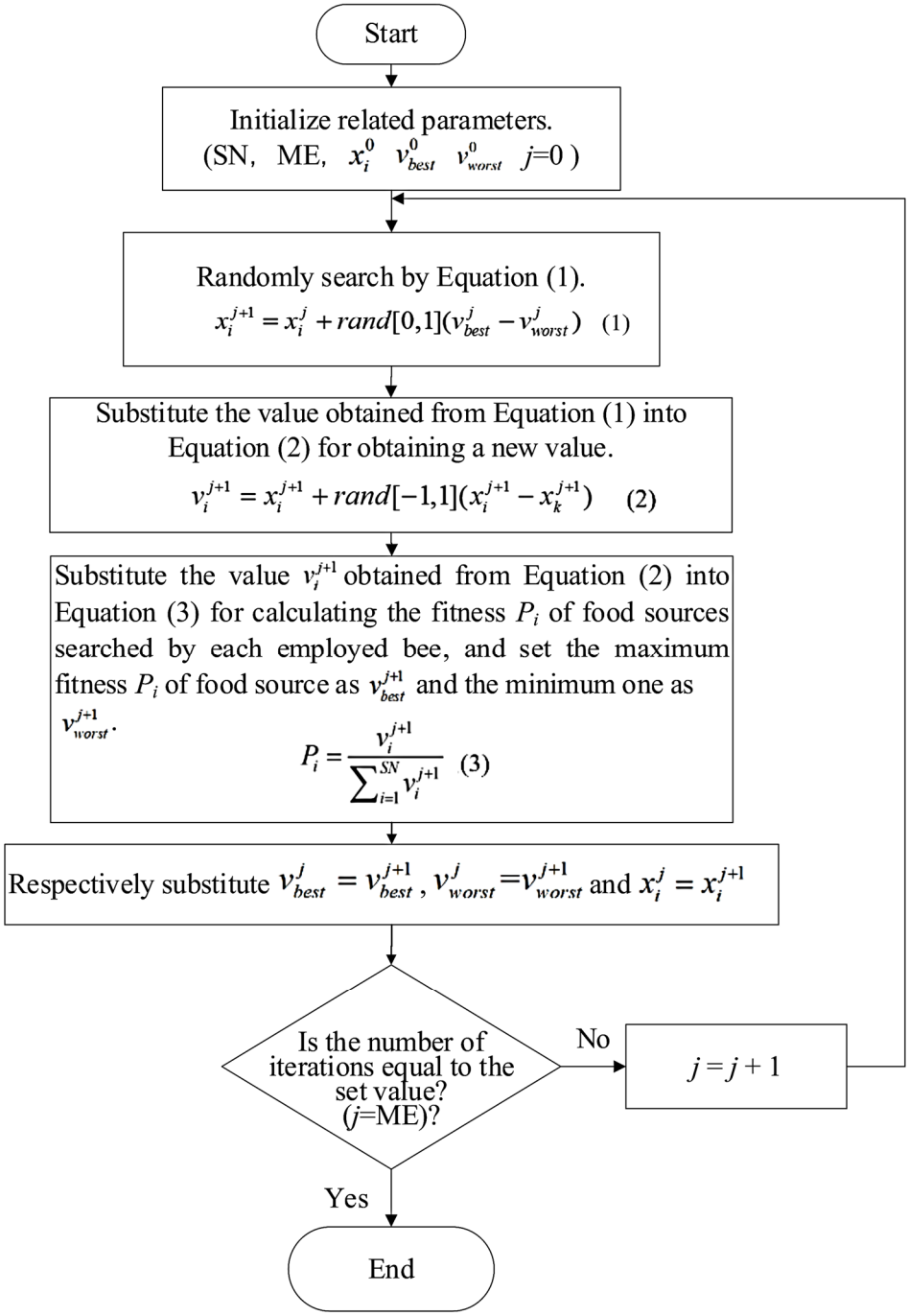
$$P_i = \frac{\mathrm{fit}_i}{\sum_j \mathrm{fit}_j} \qquad (۱)$$

که در آن fit_i مقدار تناسب i^{th} راه‌حل در ازدحام است. همان‌طور که مشاهده می‌شود، هرچه راه‌حل i بهتر باشد، احتمال انتخاب منبع غذایی i^{th} بیشتر می‌شود. اگر موقعیتی را نتوان بیش از تعداد از پیش تعریف‌شده (به نام حد) چرخه بهبود بخشید، آنگاه منبع غذا رها می‌شود. فرض کنید که منبع رهاشده X_i است، و سپس زنبور پیشاهنگ منبع غذایی جدیدی را کشف می‌کند که باید با i^{th} به‌صورت معادله زیر جایگزین شود:

$$x_{i,k} = lb_k + \Phi_{i,k} \times (ub_k - lb_k) \qquad (۲)$$

درجایی که $\Phi_{i,k} = rand(0,1)$ یک عدد تصادفی در بازهٔ $[0,1]$ مبتنی بر یک توزیع نرمال و lb_k و ub_k به ترتیب حدود پایین و بالایی k^{th} بعد هستند.

در بهینه‌سازی وقتی جوابی داریم که به نظر جواب مناسب است سعی می‌کنیم بیشتر روی آن جواب سرمایه‌گذاری کنیم. وقتی یک ایده جالب به ذهن ما می‌رسد سعی می‌کنیم آن را پرورش دهیم و نهایتاً آن را تبدیل به یک پاسخ مناسب‌تر کنیم ایده‌ای که پشت قضیه است این است که ازنظر قواعد بهینه‌سازی یک پاسخ مناسب را مناسب‌تر کنیم که به آن بهره‌برداری (Exploitation) می‌گوییم، یعنی توانایی پرورش پاسخ‌های فعلی برای رسیدن به پاسخ‌های بهتر. در شکل (۲) روند نمای الگوریتم کلونی زنبورهای مصنوعی ترسیم گردید است.



شکل ۲ - روند نمای الگوریتم کلونی زنبورهای مصنوعی.

مثال:

هدف از این مثال یافتن نقطهٔ کمینهٔ سرتاسری تابع راستریگین دومتغیرهٔ

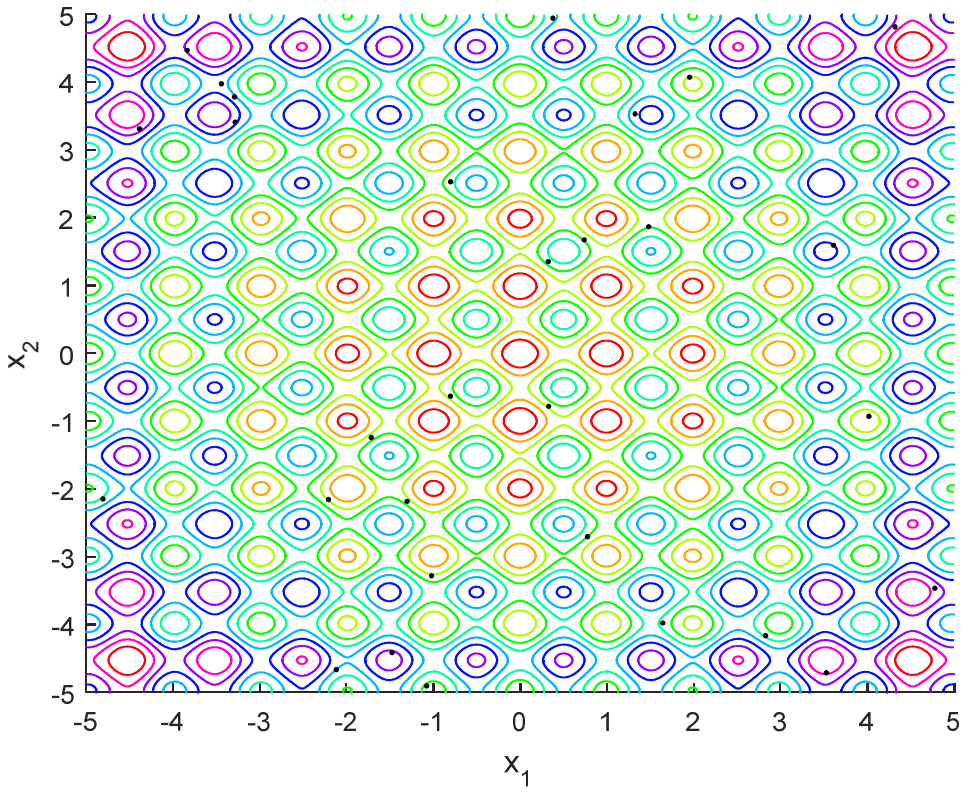
$$f(x) = 20 + x_1^2 + x_2^2 - 10(\cos 2\pi x_1 + \cos 2\pi x_2), -5 \le x_1 \le 5, -5 \le x_2 \le 5 \qquad (۳)$$

با استفاده از الگوریتم ABC است. یادآوری می‌کنیم که تابع راستریگین فوق دارای یک نقطهٔ کمینهٔ سرتاسری در مبدأ است و مقدار تابع در این نقطه نیز برابر صفر است.

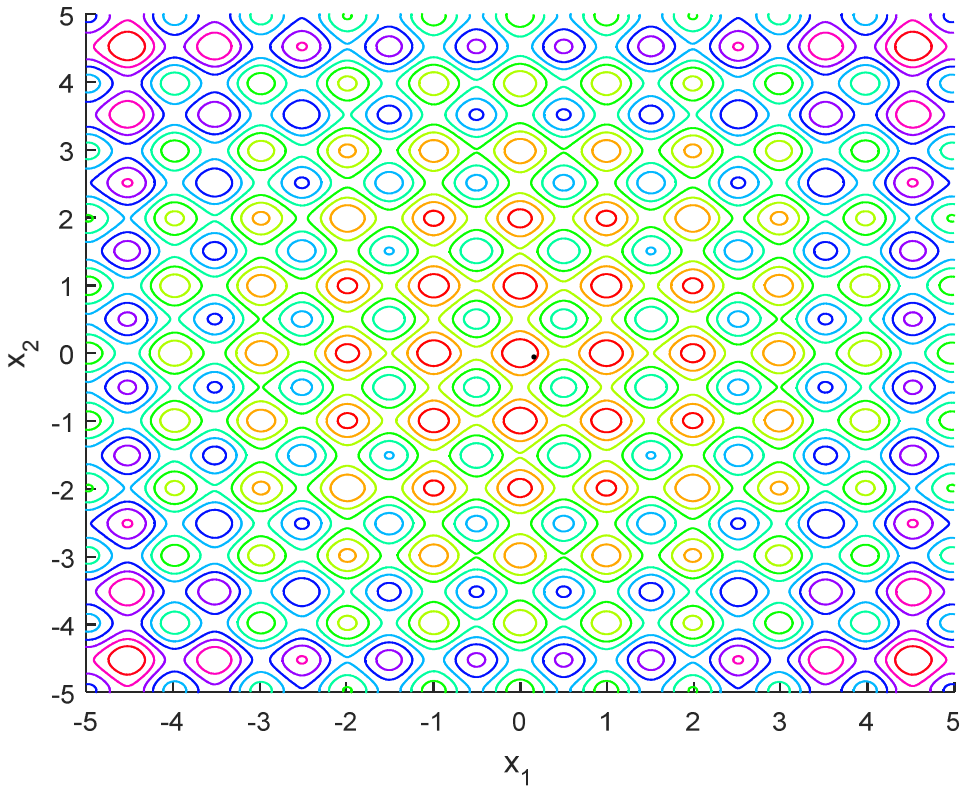
```
clear all

ul = -5; % کران پایین متغیرهای طراحی
uh = 5; % کران بالای متغیرهای طراحی

x = [ul:0.01:uh];
y = [ul:0.01:uh];
for i = 1:length(x)
    for j = 1:length(y)
        f(i, j) = 20 + x(i)^2 + y(j)^2 - 10 * (cos(2*pi*x(i)) + cos(2*pi*y(j)));
    end
end

figure(1);
clf;
hold on
contour(x, y, f, 10)
xlabel('x_1')
ylabel('x_2')

axis([ul, uh, ul, uh])
colormap(hsv)

figure(2);
clf;
hold on
contour(x, y, f, 10)
xlabel('x_1')
ylabel('x_2')

axis([ul, uh, ul, uh])
colormap(hsv)

SN = 30; % تعداد زنبورهای کلونی
n = 2; % تعداد متغیرهای مسئله بهینه‌سازی
a = .1;

% مقداردهی اولیه
x = ul + (uh - ul) * rand(n, SN);

figure(1)
plot(x(1, :), x(2, :), 'k.')

lim = 10;

for k = 1:1000

    % تعیین موقعیت زنبورهای کارگر
    for z = 1:SN
        temp = ceil(SN*rand);
        v(1, z) = x(1, z) + (-a + 2 * a * rand) .* (x(1, z) - x(1,
ceil(SN*rand)));
        while temp == z
            temp = ceil(SN*rand);
```

```
            v(1, z) = x(1, z) + (-a + 2 * a * rand) .* (x(1, z) - x(1,
ceil(SN*rand)));
        end

        temp = ceil(SN*rand);
        v(2, z) = x(2, z) + (-a + 2 * a * rand) .* (x(2, z) - x(2,
ceil(SN*rand)));
        while temp == z
            temp = ceil(SN*rand);
            v(2, z) = x(2, z) + (-a + 2 * a * rand) .* (x(2, z) - x(2,
ceil(SN*rand)));
        end
    end

    % محاسبه تابع تناسب
    temp1 = 20 + x(1, :).^2 + x(2, :).^2 - 10 * (cos(2*pi*x(1, :)) + cos(2*pi*x(2,
:)));

    % زنبورهای ناظر
    for z = 1:length(temp1)
        if temp1(z) >= 0
            fitness(z) = 1 / (1 + temp1(z));
        else
            fitness(z) = 1 + abs(temp1(z));
        end
    end

    pm = fitness / sum(fitness);

    % ذخیرهٔ بهترین نتایج به‌دست‌آمده تاکنون
    fbest(k) = min(20+x(1, :).^2+x(2, :).^2-10*(cos(2*pi*x(1, :)) + cos(2*pi*x(2,
:))));

    % محاسبهٔ بهترین راه‌حل به‌دست‌آمده تاکنون
    if fbest(k) <= min(fbest)
        x_best = v(:, find(fitness == max(fitness), 1));
    else
        fbest(k) = min(fbest);
    end

    % پیاده‌سازی چرخ رولت
    for z = 1:SN
        s = 0;
        temp = rand;
        count = 0;
        while s < temp
            s = s + pm(count+1);
            count = count + 1;
        end
        x(:, z) = v(:, count); % تعیین سمت‌های جدید
    end


    % بررسی کنید که آیا زنبورها راه‌حل‌های بهتری پیدا می‌کنند یا نه. اگرنه، آن‌ها باید
    % تبدیل به زنبورهای پیشاهنگ شوند
```

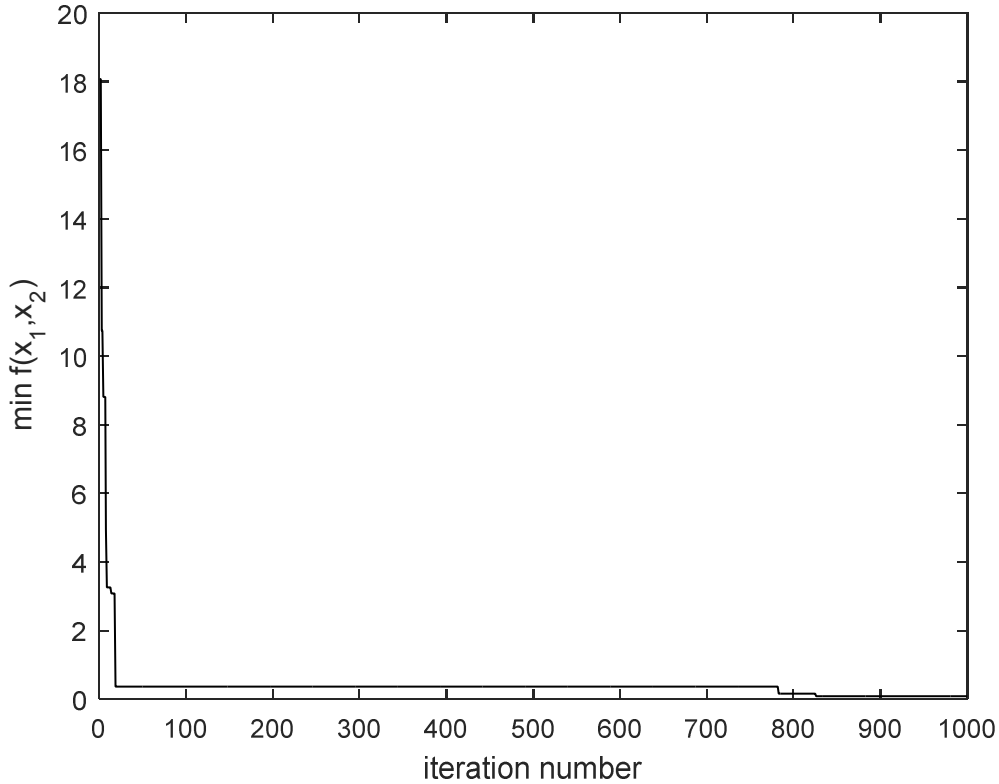
```
    if k > lim && all(diff(fbest(end-lim:end))) == 0
        x = ul + (uh - ul) * rand(n, SN);
    end
end

figure(2)
plot(x_best(1), x_best(2), 'k.')

figure(3)
plot(fbest, 'k')
xlabel('iteration number')
ylabel('min f(x_1,x_2)')

x_best
```



شکل ۳ - پراکندگی زنبورها در فضای مسئله راستریگین پس از مقداردهی اولیه.



شکل ٤ - نقطهٔ کمینهٔ سرتاسری تابع راستریگین در انتهای فرایند بهینه‌سازی، با استفاده از الگوریتم زنبورعسل مصنوعی.



شکل ٥ – نمودار کمینه‌سازی تابع راستریگین در برابر تعداد تکرار.

مراجع:

[1] Gao, W.; Liu, S.; Huang, L. A Global Best Artificial Bee Colony Algorithm for Global Optimization. J. Comput. Appl. Mathemat. 2012, 236, 2741-2753.

[2] D. Karaboga, An idea based on honey bee swarm for numerical optimization, Erciyes University, Technical Report-TR06, Kayseri, Turkey, 2005.

[3] D. Karaboga, B. Basturk A powerful and efficient algorithm for numerical function optimization : artificial bee colony (ABC) algorithm Journal of Global Optimization, 39 (2007), pp. 459-471.

[4] D. Karaboga, B. Basturk On the performance of artificial bee colony (ABC) algorithm Applied Soft Computing, 8 (2008), pp. 687-697.

[5] D. Karaboga, B. Basturk A comparative study of artificial bee colony algorithm Applied Mathematics and Computation, 214 (2009), pp. 108-132.