Методы оптимизации

Русначенко Николай

7 Января, 2016

1 Решение задач многоэкстремальной оптимизации на основе популяционных алгоритмов

Дана полимодальная целевая функция, в общем случае недифференцируемая $f(X)$, которая интерпретируется в контексте поисковых методов как функция фитнеса $\varphi(X)$. Заданы также функции ограничений в виде гиперпараллелепипеда $P(x_{i,j}^{min} \leq x_{i,j} \leq x_{i,j}^{max}, i = \overline{1,n}, j = \overline{1,m})$ и $g_k(x_j) \leq 0, k = \overline{1,p}$, определяющие множество допустимых решений D.

Требуется найти глобальный экстремум $X^* \in X_e$(X_e - множество локальных экстремумов) на множестве , т.е. такое решение $X^* \in X_e \subset D$, которое удовлетворяет следующим условиям:

$$F(X^*) = \inf_{x \in D}(f(x_{1_e}), \ldots, f(x_{N_{p_e}}))$$

$$D = \begin{cases} P(x_{i,j}^{min} \leq x_{i,j} \leq x_{i,j}^{max}, i = \overline{1,n}, j = \overline{1,m}) \\ g_k(X_j) \leq 0, k = \overline{1,p} \end{cases}$$

Двухэкстремальная функция. Множество допустимых решений:

$$f(x_1, x_2) = ax_1^2 + bx_2^2 + c\cos(x_1 - 0.4)$$

Исследовать характер решений в зависимости от параметров при значениях:

$$a = (1.0, 3.0, 5.0)$$

$$b = (2.0, 4.0, 8.0)$$

$$c = (10, 20, 30)$$

Границы рассматриваемой области:

$$[x_i^{min}, x_i^{max}] = [-10, 10]$$

1.1 Решение задачи

Алгоритм решения поставленной задачи состоит из следующих этапов:

1. Задать размер популяции Np; весовой коэффициент F; параметр операции скрещивания CR; максимальное количество популяций M;
2. Формирование начально популяции из Np векторов;
3. $m = 0$;
4. $j = 1$;
5. выбираем вектор-мишень $X_t = X_j$;
6. Случайный выбор X_a, X_b, X_c для формирование вектора $X_{c_1} : X_{c_1} = X_c + F(X_a - X_b)$. Если какая-либо из координат выходит за границы, то необходимо сгенерировать случайную величину равномерным распределением из области $[x_i^{min}, x_i^{max}]$;
7. Составить X_s на основе мутации X_t и X_{c_1};
8. Проверить приближение к точки экстремума (минимума) в X_s:
   (a) $f(X_s) < f(X_t)$, то замена вектора X_t на X_s;
   (b) $f(X_s) >= f(X_t)$, то вектор мишень остается в множестве;
9. Если $j < Np$, то $j = (j + 1)\%Np$;
10. Проверить число совершаемых мутаций $m < M$, если так, то переходим на шаг 5. Иначе, завершаем работу алгоритма.

Точность нахождения глобального экстремума зависит от параметров Np и M. Чем выше значения — тем больше вероятность нахождения.

1.2 Результат работы программы

Программа была протестирована для следующего набора входных параметров:

1. $a = 1.0, b = 2.0, c = 10;$
2. $a = 3.0, b = 4.0, c = 20;$
3. $a = 5.0, b = 8.0, c = 30.0.$

Листинг 1: "Результат работы программы для случая 1"

```
p =
   -2.2854527    -0.0012022
minf = -3.7543
```

Листинг 2: "Результат работы программы для случая 2"

```
p =
   -2.0711e+00    -4.8892e-06
minf = -2.8019
```

Листинг 3: "Результат работы программы для случая 3"

```
p =
   -2.0082e+00    -6.0373e-07
minf = -2.1229
```

1.3 Реализация программы на языке *Octave*

Листинг 4: "Реализация программы"

```
a = 5; b = 8; c = 30;
f = @(x) a * x(1)^2 + b * x(2)^2 + c * cos(x(1) - 0.4)
CR = unifrnd(0,1);
F = 1; Np = 10; M = 2000; xmin = -10; xmax = 10;
vp = zeros(Np, 2);
for i=1:size(vp)(1)
    for j=1:size(vp)(2)
        vp(i, j) = unifrnd(xmin, xmax);
    end
end

function xs = mutate(vp, xt_index, F, CR, xmin, xmax)
    % Produce xc1
    xt = vp(xt_index);
    x = randperm(size(vp)(1));
    x([xt_index]) = [];
    xa = x(1); xb = x(2); xc = x(3);
    xc1 = vp(xc, :) .+ F.*(vp(xa, :) .- vp(xb, :));
    for j=1:size(vp)(2)
        if (xc1(j) < xmin || xc1(j) > xmax)
            xc1(j) = unifrnd(xmin, xmax);
        end
    end
    % Mutate xc1 and xt
    xs = zeros(1, size(vp)(2));
    for i=1:(size(vp)(2) - 1)
        r = unifrnd(0,1);
        if (r <= CR)
            xs(i) = xc1(i);
        else
            xs(i) = xt(i);
        end
    end
    xs(size(vp)(2)) = xc1(size(vp)(2));
end
% Main algorithm
j = 0
for i=1:M
    j = mod(j, Np) + 1;
    xt = vp(j, :)
    xs = mutate(vp, j, F, CR, xmin, xmax)
    if (f(xs) < f(xt))
        vp(j,:) = xs;
    end
    vp
end
% Find best result
minf = f(vp(1, :))
p = vp(1,:)
for i=2:Np
    if f(vp(i, :)) < minf
        p = vp(i,:)
        minf = f(vp(i,:))
    end
end
% Show results
p
minf
```