RUDN Journal of Engineering researches
Вестник РУДН. Серия: Инженерные исследования
2017 Том 18 № 2 254—265
http://journals.rudn.ru/engineering-researches

DOI 10.22363/2312-8143-2017-18-2-254-265
УДК 004.023, 517.977

ЭВОЛЮЦИОННЫЕ АЛГОРИТМЫ ДЛЯ РЕШЕНИЯ ЗАДАЧИ ОПТИМАЛЬНОГО УПРАВЛЕНИЯ

А.И. Дивеев[1,2], С.В. Константинов[2]

[1] Федеральный исследовательский центр «Информатика и управление»
Российской академии наук
ул. Вавилова, 44, Москва, Россия, 119333
[2] Российский университет дружбы народов,
ул. Миклухо-Маклая, 6, Москва, Россия, 117198

В работе приведено описание некоторых популярных эволюционных алгоритмов: генетического алгоритма, алгоритма дифференциальной эволюции, метода роя частиц и алгоритма летучих мышей. С помощью эволюционных алгоритмов решается задача оптимального управления мобильным роботом. Для сравнения эта же задача решается алгоритмами наискорейшего градиентного спуска и случайного поиска. Вычислительные эксперименты показали, что эволюционные алгоритмы дают результаты решения задачи оптимального управления лучше, чем градиентный алгоритм.

Ключевые слова: задача оптимального управления, эволюционные алгоритмы, генетический алгоритм, алгоритм дифференциальной эволюции, метод роя частиц, алгоритм летучих мышей

Численные методы решения задачи оптимального управления в основном заключаются в редукции задачи к задаче нелинейного программирования и использовании различных градиентных методов для поиска оптимального решения [1]. Следует отметить, что градиентные методы находят точки с нулевым значением градиента. Этому требованию удовлетворяют не только точки локального минимума, но и седловые точки, которых у произвольной функции большой размерности может быть больше, чем точек локального минимума. Например, если рассматривать произвольную функцию в пространстве размерности p, то для точки с нулевым значением градиента может быть 2^p вариантов знаков для вторых производных, среди которых только один вариант, когда все вторые производные больше нуля, т.е. данная точка есть локальный минимум, один вариант, когда все производные меньше нуля, т.е. точка — локальный максимум и $2^p - 2$ вариантов, когда данная точка является седлом. Градиентные методы как правило эффективно работают для выпуклых функций, у которых отсутствуют седловые точки и локальный минимум совпадает с глобальным. В задаче оптимального управления, приведенной к задаче нелинейного программирования, достаточно сложно определить свойства функционала и гарантировать его выпуклость даже в ограниченной области пространства поиска.

Современные поисковые эволюционные алгоритмы [2] не требуют от целевой функции специальных свойств, и находят точки локального минимума, причем в процессе поиска могут находить несколько точек локального минимума и в итоге определяют среди них наилучшую. До недавнего времени единственным препятствием применения эволюционных алгоритмов для решения многомерных задач оптимизации была слабость вычислительной техники. Современные компьютеры позволяют эффективно использовать эволюционные алгоритмы для решения задачи оптимального управления.

Цель данной работы заключается в подтверждении высокой эффективности эволюционных алгоритмов в задачах поиска оптимального управления. В работе рассматривается решение задачи оптимального управления популярными эволюционными алгоритмами: генетическим алгоритмом, алгоритмом дифференциальной эволюции, методом роя частиц и алгоритмом летучих мышей. Для сравнения решаем также эту же задачу алгоритмами случайного поиска и градиентного наискорейшего спуска. Результаты сравниваем по средним значениям функционалов.

Генетический алгоритм

Он был разработан в 80-х годах прошлого столетия [3].

В генетическом алгоритме каждую компоненту вектора параметров $q = [q_1 ... q_8]^{\mathrm{T}}$ кодируем двоичным кодом Грея. Для кодирования используем c бит под целую часть числа и d бит под дробную часть числа. Всего код одного вектора из восьми параметров содержит $8(c + d)$ бит. В результате код каждой компоненты вектора параметров определяет положительное число g_i, $i = 1, ..., 8$, в диапазоне от 0 до 2^c.

В алгоритме первоначально генерируем случайно H двоичных кодов из $8(c + d)$ бит. Предполагаем, что сгенерированные коды представляют собой коды Грея

$$S = \{S_1, ..., S_H\},$$

где $S_i = (s^i_1, ..., s^i_{8(c+d)})$, $s^i_j \in \{0, 1\}$, $i = 1, ..., H, j = 1, ..., 8(c + d)$.

Далее вычисляем значения функционалов для каждого кода возможного решения. Для этой цели первоначально переводим код Грея каждого возможного решения $S_i = (s^i_1, ..., s^i_{8(c+d)})$ в двоичный код $B_i = (b^i_1, ..., b^i_{8(c+d)})$ по формуле

$$b^i_j = \begin{cases} s^i_j, \text{ если } (j-1) \bmod (c+d) = 0 \\ s^i_j \oplus s^i_{j-1} \ — \text{ иначе} \end{cases}, j = 1, ..., 8(c + d).$$

Переводим двоичный код в десятичный для каждых $(c + d)$ бит

$$g^i_k = \sum_{l=1}^{c+d} b^i_{(c+d)(k-1)+l} 2^{c-l}, k = 1, ..., 8.$$

Вычисляем значения вектора параметров $q^i = [q_1^i ... q_8^i]^{\mathrm{T}}$ возможного решения i по формуле

$$q_k^i = \frac{g_k^i}{2^c}(q_i^+ - q_i^-) + q_i^-, \; k = 1, ..., 8.$$

Формируем множество оценок целевых функций

$$F = \{f_1, ..., f_H\}.$$

В алгоритме выполняем одноточечное скрещивание. Отбираем случайно два кода возможных решений

$$S_\alpha = (s_1^\alpha, ..., s_{8(c+d)}^\alpha), \; S_\beta = (s_1^\beta, ..., s_{8(c+d)}^\beta), \; \alpha, \beta \in \{1, ..., H\}.$$

Вычисляем вероятность скрещивания по формуле

$$p_c = \max\left\{\frac{f^-}{f_\alpha}, \; \frac{f^-}{f_\beta}\right\},$$

где $f^- = \min\{f_1, ..., f_H\}$.

Находим случайно точку скрещивания $\sigma \in \{1, ..., 8(c + d)\}$ и выполняем скрещивание. Получаем два новых кода возможных решений:

$$S_{H+1} = (s_1^\alpha, ..., s_{\sigma-1}^\alpha, s_\sigma^\beta, ..., s_{8(c+d)}^\beta);$$

$$S_{H+2} = (s_1^\beta, ..., s_{\sigma-1}^\beta, s_\sigma^\alpha, ..., s_{8(c+d)}^\alpha).$$

Выполняем операцию мутации с заданной величиной вероятности p_m. Для кода нового возможного решения S_{H+1} находим случайно точку мутации $\mu \in \{1, ..., 8(c + d)\}$ и меняем случайно компоненту $S_\mu^{H+1} \in \{0, 1\}$. Тоже повторяем для второго нового возможного решения S_{H+2}.

Оцениваем новые возможные решения по значению минимизируемого функционала, получаем f_{H+1}, f_{H+2}.

Если оценка нового возможного решения лучше наихудшей оценки во множестве возможных решений, то заменяем наихудшее решение на новое возможное решение.

$$S_w \leftarrow S_{H+i}, f_w \leftarrow f_{H+i} \text{ если } f_w > f_{H+i}, \text{ где } f_w = \max\{f_1, ..., f_H\}, \; i = 1, 2.$$

После выполнения K операций скрещивания алгоритм останавливаем и находим наилучшее возможное решение во множестве.

Алгоритм дифференциальной эволюции

Алгоритм дифференциальной эволюции впервые был представлен в 1995 году авторами K. Прайсом (K. Price) и Р. Сторном (R. Storn) [4]. В алгоритме исполь-

зуется адаптивная мутация особей, которая зависит от распределенности особей популяции в пространстве поиска. При сильной распределености особей, алгоритм производит существенные вариации, а при малой распределенности производятся лишь небольшие изменения особей популяции.

Классическая схема алгоритма дифференциальной эволюции выглядит следующим образом: если доступная информация о системе не позволяет выбрать начальное приближенное решение, то начальная популяция s_0 заполняется случайным образом.

Далее на каждой итерации в текущей популяции s_k для каждой особи s создается особь-потомок s' на основе трех родительских особей s_1, s_2, s_3, выбранных из этой популяции случайным образом ($s_1 \neq s_2 \neq s_3$). Для этого хромосому q' формируют путем линейной комбинации хромосом q_1, q_2, q_3, которые являются вещественными векторами:

$$q' = q_1 + F(q_2 - q_3),$$

где F — скалярный коэффициент влияния.

После формирования особей s' проверяется их приспособленность и, если она выше чем у родительской особи s, то новая особь заменяет родительскую в популяции:

$$s_i = \begin{cases} s_i', \text{ если } \varphi(s_i') \geq \varphi(s_i) \\ s_i, \text{ иначе.} \end{cases}$$

При формировании новой хромосомы q' может возникнуть ситуация, когда один или несколько ее компонентов (генов) окажутся за пределами допустимых значений $(q_i^-; q_i^+)$. В этом случае вычисляется новое случайное значение q, которым заменяется компонент в хромосоме.

Также существует модификация алгоритма дифференциальной эволюции, в котором в качестве базовой особи для мутации используется не случайно выбранная из текущей популяции, а наиболее приспособленная особь s_{best}. Тогда формирование новой хромосомы q' производится по правилу:

$$q' = q_1 + \lambda(q_{\text{best}} - q_1) + F(q_2 - q_3),$$

где λ — дополнительный коэффициент для повышения эффективности использования наиболее приспособленной особи.

Метод роя частиц

Метод роя частиц основан на имитации социально-поведенческих моделей организованных групп [5]. Метод строится на тех же принципах, на основе которых птицы в стае и рыбы в косяке ведут себя удивительно синхронно, двигаясь в том или ином направлении как единое целое. Авторами метода являются Дж. Кеннеди (J. Kennedy) и Р. Эберхарт (R. Eberhart), которые предложили идею канонического метода роя частиц в 1995 году. В методе роя частиц распределенные в

пространстве параметров задачи оптимизации частицы играют роль агентов. Частицы перемещаются в пространстве параметров и изменяют направление и скорость своего движения на основе определенных правил. На каждой итерации вычисляется соответствующее положению частицы значение целевой функции. Также частице доступна информация о наилучшей частице из числа соседних с ней частиц и информация о лучшем собственном значении на предыдущих итерациях. На основе этой информации определяется правило изменения положения и скорости частицы в пространстве параметров.

Новое положение частицы s_i в момент времени t определяется вектором ее координат q_i, а ее скорость — вектором v_i:

$$q_{i,t+1} = q_{i,t} + v_{i,t+1};$$

$$v_{i,t+1} = \alpha v_{i,t} + P_n[0; \beta] \times (q^b_{i,t} - q_{i,t}) + P_n[0; \gamma] \times (q_{g,t} - q_{i,t}),$$

где $P_n[a; b]$ — n-мерный вектор псевдослучайных значений, равномерно распределенных на интервале $[a; b]$; $q^b_{i,t}$ — вектор координат частицы с лучшим собственным значением на предыдущих итерациях; $q_{g,t}$ — вектор координат лучшей соседней частицы; α, β, γ — свободные параметры алгоритма со следующими рекомендуемыми значениями: $\alpha = 0{,}7298$; $\beta = \gamma = 1{,}4962$.

Алгоритм летучих мышей

Алгоритм разработан в 2010 году [6; 7]. Летучие мыши обладают уникальными средствами эхолокации, которая используется для обеспечения полетов в темноте и обнаружения добычи.

В процессе поиска добычи летучие мыши генерируют сигналы, имеющие частоту ω_i и громкость a_i. Частота сигналов и их громкость может изменяться в заданных пределах — $[\omega_{\min}; \omega_{\max}]$ и $[0; 1]$ соответственно. Также может изменяться частота повторения сигналов $r_i \in [0; 1]$. На первом шаге метода случайным распределением задаются соответственно начальные значения $q_{i,0}$, скорость $v_{i,0}$, частота сигнала $\omega_{i,0}$, громкость сигнала $a_{i,0}$ и частота повторения сигнала $r_{i,0}$. Определяется вектор координат X^b агента, доставляющего глобально лучшее решение. Выполняется миграционная процедура по следующей схеме:

$$q_{i,t+1} = q_{i,t} + v_{i,t+1};$$

$$v_{i,t+1} = v_{i,t} + \omega_{i,t+1}(q_{i,t} - q^b);$$

$$\omega_{i,t+1} = \omega^{\min} + (\omega^{\max} - \omega^{\min}) P_1[-1; 1],$$

где $P_1[-1; 1]$ — случайное число, равномерно распределенное на интервале $[-1; 1]$.

Далее реализуется случайный локальный поиск по следующей схеме:

$$q'_{i,t} = q_{i,t} + \overline{a} P_n[-1; 1],$$

где $P_n[-1; 1]$ — n-мерный вектор псевдослучайных значений, равномерно распределенных на интервале $[-1; 1]$; $\overline{a}$ — текущее среднее значение громкостей всех агентов популяции:

$$\overline{a} = \frac{1}{p}\sum_{i=1}^{p} a_i.$$

Схема локального поиска повторяется либо до получения вектора $q'_{i,t}$, при котором значение фитнес-функции $\phi(q'_{i,t})$ лучше, чем при $q_{i,t}$, либо до достижения терминального числа повторов схемы. На последнем этапе реализуется эволюция параметров a_i и r_i по правилу:

$$a_{i,t+1} = b_a a_{i,t};$$

$$r_{i,t+1} = r_{i,0}[1 - \exp(-b_r t)],$$

где $b_a \in (0; 1)$, $b_r > 0$ — свободные параметры алгоритма, рекомендуемые значения которых равны 0,9.

Алгоритм случайного поиска

Генерируем случайный вектор параметров $\tilde{q} = [\tilde{q}_1 ... \tilde{q}_8]^{\mathrm{T}}$ по формуле

$$\tilde{q}_k = \xi(q_i^+ - q_i^-) + q_i^-, \; k = 1, ..., 8, \tag{1}$$

где ξ — случайная величина, $\xi \in [0; 1]$.

Вычисляем оценку вектора $\tilde{f}$ по значению минимизируемого функционала.

Далее генерируем новый вектор $q = [q_1 ... q_8]^{\mathrm{T}}$ по формуле (1) и вычисляем его оценку f. Если оценка нового вектора лучше $f < \tilde{f}$, то заменяем вектор и его оценку, $\tilde{q} \leftarrow q, \tilde{f} \leftarrow f$.

Повторяем генерацию вектора и вычисление его оценки R раз. В результате получаем наилучший вектор параметров $\tilde{q} = [\tilde{q}_1 ... \tilde{q}_8]^{\mathrm{T}}$.

Алгоритм наискорейшего градиентного спуска

В соответствии со стратегией алгоритма наискорейшего градиентного спуска [8] генерируем случайный вектор параметров $\tilde{q} = [\tilde{q}_1 ... \tilde{q}_8]^{\mathrm{T}}$ по формуле (1).

Вычисляем значение оценки для полученного вектора параметров

$$\tilde{f} = J(\tilde{q}).$$

Вычисляем вектор градиента целевой функции в точке $\tilde{q} = [\tilde{q}_1 ... \tilde{q}_8]^{\mathrm{T}}$

$$\frac{\partial J(\tilde{q})}{\partial q} = \left[\frac{J(\tilde{q} + \Delta_1) - \tilde{f}}{\delta q} \quad \cdots \quad \frac{J(\tilde{q} + \Delta_8) - \tilde{f}}{\delta q} \right]^{\mathrm{T}},$$

где δq — заданная малая положительная величина,

$$\Delta_i = [\underbrace{0 \quad \ldots \quad 0 \quad \delta q}_{i} 0 \quad \ldots \quad 0]^{\mathrm{T}}.$$

Вычисляем значение второго вектора параметров $\bar{q} = [\bar{q}_1 ... \bar{q}_8]^{\mathrm{T}}$ в направлении антиградиента

$$\bar{q} = \tilde{q} - \frac{\partial J(\tilde{q})}{\partial q}(M\delta q), \tag{2}$$

где первоначально $M = M_0$, M_0 — заданное положительное целое число.

Проверяем ограничения для второго вектора параметров $\bar{q} = [\bar{q}_1 ... \bar{q}_8]^{\mathrm{T}}$.

Если

$$\exists \bar{q}_i \Rightarrow (\bar{q}_i > q_i^+) \vee (\bar{q}_i < q_i^-),\ 1 \le i \le 8, \tag{3}$$

то уменьшаем значение M

$$M \leftarrow M - \delta m, \tag{4}$$

где δm — заданное положительное целое число, $\delta m << M_0$.

Если $M > 0$, то пересчитываем значение второго вектора параметров по формуле (2), иначе завершаем вычисления и считаем решением вектор параметров $\tilde{q} = [\tilde{q}_1 ... \tilde{q}_8]^{\mathrm{T}}$.

Повторяем вычисления по формулам (2), (4), пока выполняются условия (3). В результате получаем второй вектор параметров $\bar{q} = [\bar{q}_1 ... \bar{q}_8]^{\mathrm{T}}$. Вычисляем значение минимизируемого функционала для второго вектора параметров

$$\bar{f} = J(\bar{q}).$$

Вычисляем значения векторов в точках золотого сечения

$$q^{\alpha} = (1 - \gamma)\tilde{q} + \gamma\bar{q},\ q^{\beta} = \gamma\tilde{q} + (1 - \gamma)\bar{q},$$

где

$$g = \frac{\sqrt{5} - 1}{2} \approx 0{,}618034.$$

Векторы параметров $q^{\alpha} = [q_1^{\alpha} ... q_8^{\alpha}]^{\mathrm{T}}$, $q^{\beta} = [q_1^{\beta} ... q_8^{\beta}]^{\mathrm{T}}$ в точках золотого сечения удовлетворяют ограничениям (3)

$$q_i^- \le \min\{\tilde{q}_i, \bar{q}_i\} \le q_i^{\alpha}, q_i^{\beta} \le \max\{\tilde{q}_i, \bar{q}_i\} \le q_i^+,\ i = 1, ..., 8.$$

Вычисляем значения целевой функции для векторов параметров в точках золотого сечения

$$f^{\alpha} = J(q^{\alpha}), f^{\beta} = J(q^{\beta}). \tag{5}$$

Если $f^\alpha < f^\beta$, то

$$\bar{q} = q^\beta, \bar{f} = f^\beta, q^\beta = q^\alpha, f^\beta = f^\alpha, \tag{6}$$

$$q^\alpha = (1-\gamma)\tilde{q} + \gamma\bar{q}, f^\alpha = J(q^\alpha), \tag{7}$$

иначе

$$\tilde{q} = q^\alpha, \tilde{f} = f^\alpha, q^\alpha = q^\beta, f^\alpha = f^\beta, \tag{8}$$

$$q^\beta = \gamma\tilde{q} + (1-\gamma)\bar{q}, f^\beta = J(q^\beta). \tag{9}$$

Повторяем вычисления по формулам (5)—(9), пока выполняются условия

$$\| q^\beta - q^\alpha \| = \sqrt{\sum_{i=1}^{8}\left(q_i^\beta - q_i^\alpha\right)^2} > \delta q. \tag{10}$$

При нарушении условия (10) определяем первый вектор параметров. Если $f^\alpha < f^\beta$, то $\tilde{q} = q^\alpha, \tilde{f} = f^\alpha$, иначе $\tilde{q} = q^\beta, \tilde{f} = f^\beta$. Повторяем вычисления по формулам (5)—(10). Всего выполняем все вычисления заданное количество K_1 раз. Результатом вычислений будет последний вектор параметров $\tilde{q} = [\tilde{q}_1 ... \tilde{q}_8]^T$.

Задача параметрического оптимального управления мобильным роботом

Задана математическая модель мобильного робота:

$$\dot{x}_1 = u_1 \cos(x_3); \tag{11}$$

$$\dot{x}_2 = u_1 \sin(x_3); \tag{12}$$

$$\dot{x}_3 = u_2, \tag{13}$$

где $x = [x_1\ x_2\ x_3]^T$ — вектор состояния; $u = [u_1\ u_2]^T$ — вектор управления.

Заданы ограничения на управление:

$$u_i^- \le u_i \le u_i^+, \ i = 1, 2.$$

Заданы начальные условия:

$$x_i(0) = x_i^0, \ i = 1, 2, 3. \tag{14}$$

Заданы терминальные условия:

$$x_i = x_i^f, \ i = 1, 2, 3$$

Задан функционал

$$J = t_f + \sqrt{\left(x_1(t_f) - x_1^f\right)^2 + \left(x_2(t_f) - x_2^f\right)^2 + \left(x_3(t_f) - x_3^f\right)^2}, \tag{15}$$

где

$$t_f = \begin{cases} t, \text{ если } \sqrt{\left(x_1(t)-x_1^f\right)^2+\left(x_2(t)-x_2^f\right)^2+\left(x_3(t)-x_3^f\right)^2} \le \varepsilon, \\ t^+ \ — \ \text{иначе} \end{cases}$$

t^+, ε — заданные положительные величины.

Необходимо найти управление в классе кубических полиномов

$$\tilde{u}_1(t) = \begin{cases} u_1^-, \text{ если } \ q_1+q_2t+q_3t^2+q_4t^3 < u_1^- \\ u_1^+, \text{ если } \ q_1+q_2t+q_3t^2+q_4t^3 > u_1^+, \\ q_1+q_2t+q_3t^2+q_4t^3 \ — \ \text{иначе} \end{cases} \tag{16}$$

$$\tilde{u}_2(t) = \begin{cases} u_2^-, \text{ если } \ q_5+q_6t+q_7t^2+q_8t^3 < u_2^- \\ u_2^+, \text{ если } \ q_5+q_6t+q_7t^2+q_8t^3 > u_2^+, \\ q_5+q_6t+q_7t^2+q_8t^3 \text{ иначе} \end{cases} \tag{17}$$

где значения коэффициентов полиномов ограничены

$$q_i^- \le q_i \le q_i^+, \ i = 1, \ldots, 8.$$

Найденные оптимальные коэффициенты кубических полиномов должны обеспечивать минимальное значение заданному функционалу

$$J = t_f + \sqrt{\left(x_1(t_f)-x_1^f\right)^2+\left(x_2(t_f)-x_2^f\right)^2+\left(x_3(t_f)-x_3^f\right)^2} \to \min.$$

Сравнительный эксперимент

Для проведения сравнительного эксперимента для каждого возможного решения формировались полиномы для управления в виде (16), (17), интегрировалась система дифференциальных уравнений (11)—(13) из заданных начальных условий (14) и вычислялась оценка по значению функционала (15).

В вычислительном эксперименте использовались следующие параметры модели (11)—(13): $x_1^0 = 10$, $x_2^0 = 10$, $x_3^0 = 0$, $x_1^f = 0$, $x_2^f = 0$, $x_3^f = 0$, $u_1^- = -10$, $u_2^- = -10$, $u_1^+ = 10$, $u_2^+ = 10$, $t^+ = 4$, $\varepsilon = 0{,}02$, $q_i^- = -8$, $q_i^+ = 8$, $i = 1, \ldots, 8$.

В генетическом алгоритме использовали следующие значения параметров: $H = 512$, $K = 2^{14} = 16384$, $c = 4$, $d = 12$, $p_m = 0{,}7$.

В алгоритме случайного поиска количество сгенерированных векторов параметров было $R = 40000$.

В алгоритме наискорейшего градиентного спуска использовали следующие параметры: $q = 0{,}0001$, $M_0 = 4096$, $\delta m = 8$, $K_1 = 128$.

В методе роя частиц размер популяции составлял $H = 16$, число итераций — $W = 2048$.

В методе, инспирированным летучими мышами, размер популяции составлял $H = 32$, число итераций — $W = 1024$.

В алгоритме дифференциальной эволюции размер популяции составлял $H = 64$, число итераций — $W = 1024$, число скрещиваний на каждой итерации — $S = 32$.

Для каждого алгоритма проводилось по 10 испытаний. Вычисления выполняли на компьютере с процессором Intel® Core™, i7-2640M, CPU@2.80 GHz. Результаты вычислений, представляющие собой продолжительность перехода из начального в конечное положение для каждого алгоритма (таблица) показали, что все эволюционные алгоритмы решают рассмотренную задачу оптимального управления приблизительно одинаково и лучше, чем алгоритм случайного поиска. Алгоритм наискорейшего градиентного спуска при решении данной задачи работает неустойчиво и не находит в большинстве случаев приемлемого решения. Среди эволюционных алгоритмов наилучшие результаты показа генетический алгоритм.

Таблица

Результаты вычислительного эксперимента
[The results of computational experiments]

Испытание	GA	DE	PSO	BIA	RS	FGD
1	4,9349	6,5507	4,8288	4,3877	5,9813	9,7561
2	4,1093	4,6874	4,8866	4,7422	6,1006	5,4075
3	4,6475	4,7433	4,4649	5,3366	5,2015	18,2190
4	4,4244	5,8050	4,6277	4,6821	5,7782	11,3907
5	4,2987	5,8152	4,4758	4,4747	4,6759	22,5127
6	4,0434	6,4671	4,7797	4,1769	5,2597	28,3650
7	4,1218	4,6943	4,8279	4,5676	5,0867	15,1068
8	4,6303	4,9991	4,9260	5,4031	5,0530	17,1744
9	4,2148	4,6895	4,7935	4,3658	5,9370	30,7676
10	4,4533	5,0282	4,6172	5,5366	5,1588	13,9237
Сред.	4,3878	5,3480	4,7228	4,7673	5,4233	17,2624

Финансирование:
Работа выполнена при поддержке гранта РФФИ № 16-29-04224-офи_м.

СПИСОК ЛИТЕРАТУРЫ

[1] *Полак Э.* Численные методы оптимизации. М.: Мир, 1974. 376 с.
[2] *Карпенко А.П.* Современные алгоритмы поисковой оптимизации. Алгоритмы, вдохновленные природой. М.: Издательство МГТУ им. Н.Э. Баумана, 2014. 448 с.
[3] *Goldberg D.E.* Genetic Algorithms in Search, Optimization, and Machine Learning. Addison-Wesley. 1989. 412 p.
[4] *Storn R., Price K.* Differential Evolution — A Simple and Efficient Heuristic for Global Optimization over Continuous Spaces / Journal of Global Optimization. 1997. No. 11. P. 341—359.
[5] *Kennedy J., Eberhart R.* Particle Swarm Optimization / Proceedings of IEEE International Conference on Neural Networks IV. 1995. P. 1942—1948.
[6] *Yang Xin-She.* A New Metaheuristic Bat-Inspired Algorithm, in: Nature Inspired Cooperative Strategies for Optimization (NISCO 2010). Studies in Computational Intelligence. Berlin: Springer, 2010. Vol. 284. P. 65—74.

[7] *Карпенко А.П.* Популяционные алгоритмы глобальной поисковой оптимизации. Обзор новых и малоизвестных алгоритмов // Информационные технологии. 2012. № 7. С. 1—32.
[8] *Пантелеев А.В., Летова Т.А.* Методы оптимизации в примерах и задачах: учеб. пособие. М.: Высшая школа, 2005. 544 с.

История статьи:
Дата поступления в редакцию: 28 февраля 2017
Дата принятия к печати: 13 марта 2017

Для цитирования:
Дивеев А.И., Константинов С.В. Эволюционные алгоритмы для решения задачи оптимального управления // *Вестник Российского университета дружбы народов. Серия «Инженерные исследования»*. 2017. Т. 18. № 2. С. 254—265.

Сведения об авторах:
Дивеев Асхат Ибрагимович, доктор технических наук, профессор, заведующий сектором проблем кибернетики Федерального исследовательского центра «Информатика и управление» Российской академии наук, профессор департамента Механики и мехатроники инженерной академии Российского университета дружбы народов. *Сфера научных интересов*: вычислительные методы для решения задач управления. *Контактная информация*: e-mail: aidiveev@mail.ru

Константинов Сергей Валерьевич, старший преподаватель департамента Механики и мехатроники инженерной академии Российского университета дружбы народов. *Сфера научных интересов*: методы оптимизации, эволюционные алгоритмы, генетические алгоритмы, вычислительные методы решения задач оптимального управления. *Контактная информация*: e-mail: konstantinov_sv@rudn.university

EVOLUTIONARY ALGORITHMS FOR THE PROBLEM OF OPTIMAL CONTROL

A.I. Diveev[1,2], S.V. Konstantinov[2]

[1] Institution of Russian Academy of Sciences Dorodnicyn Computing Centre of RAS
Vavilova str., 40, Moscow, Russia, 119333
[2] Peoples' Friendship University of Russia (RUDN University)
Miklukho-Maklaya str., 6, Moscow, Russia, 117198

The paper describes some of the popular evolutionary algorithms: genetic algorithms, differential evolution method, particle swarm optimization and bat-inspired method. With the help of these algorithms the problem of optimal control of a mobile robot is solved. For comparison the same problem is solved with the algorithm of fast gradient descent and random search. The computational experiments showed that evolutionary algorithms provide more accurate results for the optimal control problems than fast gradient descent algorithm.

Key words: optimal control problem, evolutionary algorithms, genetic algorithm, differential evolution method, particle swarm optimization, bat-inspired method

REFERENCES

[1] Polak E. *Chislennye metody optimizatsii*. M.: Mir, 1974. (In Russ).
[2] Karpenko A.P. *Sovremennye algoritmy poiskovoi optimizatsii. Algoritmy, vdokhnovlennye prirodoi*. M.: Izdatel'stvo MGTU im. N.E. Baumana, 2014. (In Russ).
[3] Goldberg D.E. Genetic Algorithms in Search, Optimization, and Machine Learning. Addison-Wesley, 1989.
[4] Storn R., Price K. Differential Evolution — A Simple and Efficient Heuristic for Global Optimization over Continuous Spaces / *Journal of Global Optimization*. 1997. No. 11. P. 341—359.
[5] Kennedy J., Eberhart R. Particle Swarm Optimization / *Proceedings of IEEE International Conference on Neural Networks IV*. 1995. P. 1942—1948.
[6] Yang Xin-She. A New Metaheuristic Bat-Inspired Algorithm, in: Nature Inspired Cooperative Strategies for Optimization (NISCO 2010). *Studies in Computational Intelligence*. Berlin: Springer, 2010. Vol. 284. P. 65—74.
[7] Karpenko A.P. Populyatsionnye algoritmy global'noi poiskovoi optimizatsii. Obzor novykh i maloizvestnykh algoritmov. *Informatsionnye tekhnologii*. 2012. No. 7 P. 1—32. (In Russ).
[8] Panteleev A.V., Letova T.A. *Metody optimizatsii v primerakh i zadachakh: ucheb. posobie*. M.: Vysshaya shkola, 2005. (In Russ).

Article history:
Received: 28 February 2017
Accepted: 13 March 2017

For citation:
Diveev A.I., Konstantinov S.V. (2017) Evolutionary algorithms for the problem of optimal control. *RUDN Journal of Engineering Researches*, 18(2), 254—265.

Bio Note:
Askhat I. Diveev, Doctor of technical sciences, professor, chief of sector of Cybernetic problems, Federal Research Centre "Computer Science and Control" of Russia Academy of Sciences, professor of department Mechanics and mechatronics, Engineering Academy, Peoples' Friendship University of Russia (RUDN University). *Research interests*: Computational methods for problems of control. *Contact information*: e-mail: aidiveev@mail.ru

Sergey V. Konstantinov, senior lecturer of department Mechanics and mechatronics, Engineering Academy, Peoples' Friendship University of Russia (RUDN University). *Research interests*: Optimization algorithms, evolutionary algorithms, genetic algorithms, computational methods for problems of optimal control. *Contact information*: e-mail: konstantinov_sv@rudn.university