УДК 519.6

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

[1]Федеральный исследовательский центр «Информатика и управление»
Российской академии наук
[2]Российский университет дружбы народов

Исследование эволюционных алгоритмов для решения задачи оптимального управления

Приведен экспериментальный анализ наиболее популярных эволюционных и градиентных алгоритмов для решения задачи оптимального управления. Для корректного сравнения в градиентные методы включен многоточечный поиск, и параметры алгоритмов подобраны так, чтобы все алгоритмы в процессе поиска вычисляли целевую функцию приблизительно одинаковое количество раз. Вычислительный эксперимент проведен на решении задачи оптимального управления мобильным роботом с четырьмя фазовыми ограничениями. В качестве критериев сравнения алгоритмов использованы лучшее найденное значение целевой функции, среднее значение и среднеквадратичное отклонение.

Ключевые слова: оптимальное управление, эволюционные вычисления, алгоритм ADAM, метод роя частиц, пчелиный алгоритм.

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

[1]Federal Research Center «Computer Science and Control», Russian Academy of Sciences
[2]Peoples' Friendship University of Russia (RUDN University)

Study of evolutionary algorithms for the optimal control problem

Experimental analysis of the most popular evolutionary and gradient-based algorithms for the optimal control problem is presented. For correct comparison, the multisolution search is included in the gradient-based methods and the parameters of all algorithms are chosen so that the number of fitness function calculations in the search process by each algorithm is approximately equal. The computational experiment of the optimal control search with four phase constraints is carried out using mobile robot model. As criteria for comparison of algorithms the best found value of the fitness function, the mean value and standard deviation are used.

Key words: optimal control, evolutionary computations, ADAM method, particle swarm optimization, bees algorithm.

1. Введение

Основной подход к численному решению задачи оптимального управления состоит в редукции к задаче нелинейного программирования [1]. Следует отметить некоторые особенности данного подхода. В полученной в результате редукции задаче нелинейного программирования вычисление одного значения целевой функции требует интегрирования системы дифференциальных уравнений, что в большинстве прикладных задач занимает гораздо больше времени, чем преобразование искомого вектора на итерации в процессе поиска. Отсюда следует, что сложность алгоритма в задачах оптимального управления

целесообразнее оценивать по количеству вычислений значений целевой функции, а не по количеству поисковых итераций. Заметим, что в градиентных методах на одной итерации количество вычислений целевой функции равно размерности искомого вектора, а в методах второго порядка – квадрату размерности. Часто в прикладных задачах оптимального управления вычисление одного элемента Гессиана занимает намного больше времени, чем вычисление обратной матрицы для всего Гессиана. Второй важной особенностью задачи оптимального управления является отсутствие сведений о топологических свойствах целевой функции в пространстве искомых параметров. Если целевая функция содержит большое количество локальных экстремумов, то методы нелинейного программирования будут скорее всего находить локальные экстремумы, значение которых не дает информацию о месте расположения глобального оптимума в пространстве искомых параметров. Третьей особенностью поиска численного решения задачи оптимального управления является высокая размерность пространства искомых параметров. Простейшая редукция задачи оптимального управления к задаче нелинейного программирования состоит в дискретизации компонент вектора управления по времени. При таком подходе чем больше точек дискретизации, тем точнее численное решение задачи оптимального управления, но тем больше размерность пространства искомых параметров. Достаточно сложно определить топологические свойства целевой функции, каждое значение которой вычисляется интегрированием системы дифференциальных уравнений, в пространстве большой размерности, равной произведению количества точек дискретизации и размерности вектора управления. Применение для численного решения задачи оптимального управления методов глобальной оптимизации [2] требует покрытия пространства поиска областями и их перебора для анализа. Количество областей с r точками разбиения по каждой оси в пространстве размерности p равно величине $(r+1)^p$. Заметим, что «неподъемная» для перебора на любой вычислительной машине с любым распараллеливанием величина 2^{100} соответствует одной точке разбиения на каждой оси в пространстве искомых параметров размерности 100, которая может быть получена дискретизацией по времени всего пятьюдесятью точками вектора управления из двух компонент.

В конце XX века появились эволюционные алгоритмы, которые создаются до сих пор и имеют достаточно экзотические названия [3]. С вычислительной точки зрения все эволюционные алгоритмы имеют два общих признака: они используют первоначальное множество возможных решений, которое генерируется, как правило, случайно, далее на последующих этапах выполняется изменение или эволюция этого множества за счет создания новых возможных решений на основе использования информации о ранее вычисленных значениях целевой функции для элементов множества возможных решений. При данном определении эволюционных алгоритмов все градиентные методы возможно преобразовать к эволюционным, если использовать в них начальное множество возможных решений, так как для получения нового возможного решения требуется вычисление значения градиента, которые включает вычисление значения целевой функции для текущего возможного решения.

В настоящей работе мы сравниваем наиболее популярные эволюционные и многоточечные градиентные алгоритмы на примере решения одной прикладной задачи оптимального управления, наибыстрейшего перемещения мобильного робота из одной точки плоскости в другую с учетом четырех фазовых ограничений.

2. Задача оптимального управления

Рассмотрим задачу оптимального управления в наиболее часто используемой для численных решений постановке:

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{u}),$$

где $\mathbf{x} \in \mathbb{R}^n$, $\mathbf{x} = [\ x_1 \quad \dots \quad x_n\ ]^T$ — вектор состояния, $\mathbf{u} \in \mathbb{R}^m$, $\mathbf{u} = [\ u_1 \quad \dots \quad u_m\ ]^T$ — вектор управления, $\mathbf{f}(\mathbf{x}, \mathbf{u}) = [\ f_1(\mathbf{x}, \mathbf{u}) \quad \dots \quad f_n(\mathbf{x}, \mathbf{u})\ ]^T$.

Заданы ограничения на управление:

$$\mathbf{u}^- \leqslant \mathbf{u} \leqslant \mathbf{u}^+,$$

где $\mathbf{u}^-$, $\mathbf{u}^+$ — заданные постоянные векторы ограничений, $\mathbf{u}^- = [\ u_1^- \quad \dots \quad u_m^-\ ]^T$, $\mathbf{u}^+ = [\ u_1^+ \quad \dots \quad u_m^+\ ]^T$.

Заданы начальные условия:

$$\mathbf{x}(0) = \mathbf{x}^0,$$

где $\mathbf{x}^0$ — заданный вектор начальных значений, $\mathbf{x}^0 = [\ x_1^0 \quad \dots \quad x_n^0]^T$.

Заданы терминальные условия:

$$\mathbf{x}(t_f) = \mathbf{x}^f,$$

где $\mathbf{x}^f$ — заданный вектор терминального состояния, $\mathbf{x}^f = [\ x_1^f \quad \dots \quad x_n^f]^T$, t_f — ограниченное время процесса управления:

$$t_f = \begin{cases} t, \text{ если } t < t^+ \text{ и } \|\mathbf{x}(t) - \mathbf{x}^f\| \leqslant \varepsilon, \\ t^+ - \text{ иначе,} \end{cases}$$

где t^+, ε — заданные положительные величины, норма разности векторов может выбираться из особенностей задачи, для модели объекта с соизмеримыми компонентами вектора состояния целесообразно выбрать евклидову норму:

$$\|\mathbf{x}(t) - \mathbf{x}^f\| = \sqrt{\sum_{i=1}^{n} (x_i(t) - x_i^f)^2}.$$

Заданы фазовые ограничения:

$$h_i(\mathbf{x}) \leqslant 0,\ i = 1, \dots, r.$$

Задан функционал качества:

$$J = \|\mathbf{x}(t_f) - \mathbf{x}^f\| + \int_0^{t_f} (f_0(\mathbf{x}, \mathbf{u}) + \sum_{i=1}^{r} \lambda_i \vartheta(h_i(\mathbf{x})) h_i(\mathbf{x})) dt \to \min,$$

где α_i — заданные коэффициенты штрафа, $i = 1, \dots, r$, $\vartheta(A)$ — функция Хэвисайда:

$$\vartheta(A) = \begin{cases} 1, \text{ если } A > 0, \\ 0 - \text{иначе.} \end{cases}$$

Результатом решения задачи оптимального управления является вектор управления с компонентами в форме функций времени $\tilde{\mathbf{u}}(\cdot) = [\ \tilde{u}_1(\cdot) \quad \dots \quad \tilde{u}_m(\cdot)\ ]^T$, поэтому данная задача относится к классу задач бесконечномерной оптимизации. Для применения к решению задачи оптимального управления методов нелинейного программирования необходимо аппроксимировать искомые функции $\tilde{u}_i(\cdot)$, $i = 1, \dots, m$, функциональными зависимостями от конечного числа параметров. Для этой цели часто используют полиномы, ортогональные ряды или кусочно-функциональную аппроксимацию.

Приведем редукцию задачи оптимального управления к задаче нелинейного программирования с помощью кусочно-линейной аппроксимации. Зададим малый интервал $\Delta t > 0$ и определим количество интервалов:

$$M = \left\lfloor \frac{t^+}{\Delta t} \right\rfloor.$$

Значение управления $\tilde{\mathbf{u}}(t) = [\ \tilde{u}_1(t) \quad \dots \quad \tilde{u}_m(t)\ ]^T$ в момент времени t определяем из соотношения

$$\tilde{u}_j(t) = \begin{cases} u_j^-, \text{ если } q(t, j, i, \Delta t) < u_j^-, \\ u_j^+, \text{ если } q(t, j, i, \Delta t) > u_j^+, \\ q(t, j, i, \Delta t) - \text{иначе,} \end{cases}$$

где $i\Delta t \leqslant t < (i+1)\Delta t$,

$$q(t,j,i,\Delta t) = q_{(j-1)M+i} + (q_{(j-1)M+i+1} - q_{(j-1)M+i})\frac{(t-(i-1)\Delta t)}{\Delta t},$$

$i = 1,\ldots,M, j = 1,\ldots,m$.

В результате поиск управления в задаче оптимального управления заменяем поиском вектора постоянных параметров

$$\mathbf{q} = [\ q_1 \quad \ldots \quad q_p\ ]^T,$$

где $p = m(M+1)$.

При поиске значения параметров ограничиваем

$$q_i^- \leqslant q_i \leqslant q_i^+,\ i = 1,\ldots,p,$$

где q_i^-, q_i^+ — заданные значения ограничений на параметры, $q_i^- \leqslant u^-_{\lfloor i/(M+1)\rfloor+1}$, $u^+_{\lfloor i/(M+1)\rfloor+1} \leqslant q_i^+$, $i = 1,\ldots,p$, $\lfloor A\rfloor$ — целая часть числа A.

3. Численные методы решения задач оптимального управления

Для решения задачи оптимального управления используем многоточечные градиентные методы и популярные эволюционные алгоритмы. Для градиентных и эволюционных методов первоначально генерируем множество возможных решений:

$$\mathbf{Q}_0 = \{\mathbf{q}^{0,1},\ldots,\mathbf{q}^{0,H}\}, \tag{1}$$

где $\mathbf{q}^{0,i} = [\ q_1^{0,i} \quad \ldots \quad q_p^{0,i}\ ]^T$,

$$q_j^{0,i} = \xi(q_j^+ - q_j^-) + q_j^-,\ i = 1,\ldots,H,\ j = 1,\ldots,p, \tag{2}$$

величина H зависит от алгоритма и определяется по общему количеству вычислений значений целевой функции в процессе поиска, ξ — случайная равномерно распределенная в интервале от 0 до 1 величина. Среди градиентных методов используем известные методы [4]: наискорейшего градиентного спуска, метод Ньютона–Рафсона, метод Марквардта и новый метод ADAM, который был предложен в 2015 году [5] для решения задачи обучения больших искусственных нейронных сетей. Приведем описание метода ADAM. После генерации начального множества возможных решений (1), (2) задаем векторы $\mathbf{m} = [\ 0 \quad \ldots \quad 0]^T$ и $\mathbf{v} = [\ 0 \quad \ldots \quad 0]^T$ размерностью p. Задаем параметры алгоритма: малые величины ε_1 и ε_2, максимальное число итераций W, константы $\alpha \approx 0.001$, $\beta_1 \approx 0.9$, $\beta_2 \approx 0.999$, счетчик итераций $w = 1$. Вычисляем градиент целевой функции $\nabla\mathbf{J}(\mathbf{q}) = \left[\ \frac{\partial J}{\partial q_1} \quad \ldots \quad \frac{\partial J}{\partial q_p}\ \right]^T$. Вычисляем новые значения векторов $\mathbf{m}$ и $\mathbf{v}$ по формулам

$$m_i \leftarrow \beta_1 m_i + (1-\beta_1)\frac{\partial J}{\partial q_i}, v_i \leftarrow \beta_2 v_i + (1-\beta_2)\left(\frac{\partial J}{\partial q_i}\right)^2,\ i = 1,\ldots,p.$$

Формируем вектор $\mathbf{s} = [\ s_1 \quad \ldots \quad s_p]^T$ по формуле

$$s_i = \frac{Am_i}{\varepsilon_1 + \sqrt{v_i}},\ i = 1,\ldots,p,$$

где $A = \alpha\frac{\sqrt{1-\beta_2^w}}{1-\beta_1^w}$, $w \in [1;W]$ — номер текущей итерации. Вычисляем значение нового вектора параметров:

$$\mathbf{q}^n = \mathbf{q} - \mathbf{s}.$$

Проверяем выполнение условий: если условие $\|\mathbf{q}-\mathbf{q}^n\|<\varepsilon_2$ выполнено, то достигнута заданная точность, завершаем вычисления; если условие $w=W$ выполнено, то достигнуто максимальное число итераций и завершаем вычисления; если оба условия не выполнены, то $\mathbf{q}\leftarrow\mathbf{q}^n$; $w\leftarrow w+1$. Среди эволюционных методов используем генетический алгоритм [6, 7], метод роя частиц [8] и пчелиный алгоритм [9, 10]. Приведем описание метода роя частиц и пчелиного алгоритма, которые не так популярны, как генетический алгоритм. В методе роя частиц на подготовительном этапе задаем размер множества возможных решений H, размер подмножества информаторов N, максимальное число итераций W. Генерируем начальное множество векторов параметров $\mathbf{q}^j=[\ q_1^j\ \ \dots\ \ q_p^j]^T\ ,\ j=1,\dots,H$, по формулам (1), (2). Задаем начальный вектор $\mathbf{v}^j=[\ v_1^j\ \ \dots\ \ v_p^j]^T$ направления изменения векторов параметров:

$$v_i^j=0,\ i=1,\dots,p,\ j=1,\dots,H.$$

Эволюционный процесс поиска оптимального решения продолжаем до достижения максимального числа итераций W. На каждой итерации производим вычисление вектора $\mathbf{q}^b$, доставляющего лучшее значение функционала

$$J(\mathbf{q}^b)=\min_j\{J(\mathbf{q}^j):j=1,\dots,H\},$$

и вычисление опорного вектора $\mathbf{q}^{j_r}$, являющегося лучшим среди N случайно отобранных векторов $\mathbf{q}^{j_1},\dots,\mathbf{q}^{j_N}$:

$$J(\mathbf{q}^{j_r})=\min_k\{J(\mathbf{q}^{j_k}):k=1,\dots,N\},$$

где $j_1,\dots,j_N$ — случайные целые числа в диапазоне от 1 до H. С учётом найденных лучшего $\mathbf{q}^b$ и опорного $\mathbf{q}^{j_r}$ векторов строим новое значение вектора $\mathbf{v}^j$ для каждого вектора параметров $\mathbf{q}^j$:

$$v_i^j\leftarrow\alpha v_i^j+\xi_\beta(q_i^b-q_i^j)+\xi_\gamma(q_i^{j_r}-q_i^j),\ i=1,\dots,p,\ j=1,\dots,H,$$

где α,β,γ — заданные параметры, значение которых подбирают в зависимости от решаемой задачи, $\xi_\beta\in[0;\beta]$ и $\xi_\gamma\in[0;\gamma]$ — случайные величины. Для каждого вектора параметров $\mathbf{q}^j$ вычисляем новый вектор $\tilde{\mathbf{q}}^j=[\ \tilde{q}_1^j\ \ \dots\ \ \tilde{q}_p^j\ ]^T$ по формуле

$$\tilde{q}_i^j=\begin{cases}q_i^-,\ \text{если } q_i^j+\delta v_i^j<q_i^-,\\ q_i^+,\ \text{если } q_i^j+\delta v_i^j>q_i^+,\\ q_i^j+\delta v_i^j-\text{иначе},\end{cases}\ ,\ i=1,\dots,p,\ j=1,\dots,H,$$

где δ — заданный параметр, $\delta\approx1$.

Если $J(\mathbf{q}^j)>J(\tilde{\mathbf{q}}^j)$, то $\mathbf{q}^j\leftarrow\tilde{\mathbf{q}}^j$, $j=1,\dots,H$. Завершаем вычисления при выполнении W итераций. Решением считаем вектор с наилучшим значением целевой функции в итоговом множестве возможных решений.

Пчелиный алгоритм был предложен в 2005 году [10]. В пчелином алгоритме первоначально выявляем определенное число подобластей пространства поиска, в которых значение целевой функции меньше. Данные подобласти исследуем более интенсивно, при этом радиус отбираемых подобластей с каждой итерацией уменьшаем.

В качестве параметров алгоритма задаем размер множества возможных решений H, максимальное число итераций W, количество N элитных возможных решений, $N\approx0.2H$, и количество L перспективных возможных решений, $L\approx0.5H$, количество E новых элитных векторов, $E\approx H$, и количество S новых отобранных векторов, $S\approx0.5H$.

Задаем начальные значения векторов радиусов областей поиска $\mathbf{r}^e=[\ r_1^e\ \ \dots\ \ r_p^e]^T$ и $\mathbf{r}^s=[\ r_1^s\ \ \dots\ \ r_p^s]^T\ ,\ r_i^e>0,\ r_i^s>0,\ i=1,\dots,p$.

Генерируем множество векторов параметров $\mathbf{q}^j=[\ q_1^j\ \ \dots\ \ q_p^j]^T\ ,\ j=1,\dots,H$, по формуле (1).

Далее на каждой итерации производим вычисление значений функционала для всех возможных решений $J(\mathbf{q}^j)$, $j = 1, \ldots, H$, и производим сортировку векторов параметров по возрастанию значения целевой функции

$$J(\mathbf{q}^1) \leqslant J(\mathbf{q}^2) \leqslant \ldots \leqslant J(\mathbf{q}^H).$$

Отбираем первые N элитных векторов $\mathbf{q}^j$, $j = 1, \ldots, N$. Для каждого элитного вектора $\mathbf{q}^j$, $1 \leqslant j \leqslant N$, строим набор из E векторов $\mathbf{g}^e = [\ g_1^e \quad \ldots \quad g_p^e]^T$:

$$g_i^e = \begin{cases} q_i^-, \text{ если } q_i^j + \xi_i^e < q_i^-, \\ q_i^+, \text{ если } q_i^j + \xi_i^e > q_i^+, \quad , i = 1, \ldots, p,\ e = 1, \ldots, E, \\ q_i^j + \xi_i^e - \text{иначе}, \end{cases}$$

где $\xi_i^e \in [-r_i^e; r_i^e]$ — величина, случайно распределенная на интервале от $-r_i^e$ до r_i^e, $\mathbf{r}^e$ — заданный вектор радиусов области поиска вокруг элитного вектора параметров, E — число новых векторов $\mathbf{g}^e$ в области элитного вектора $\mathbf{q}^j$.

Если какой-либо вектор $\mathbf{g}^e$ доставляет значение функционала лучше, чем вектор $\mathbf{q}^j$, то заменяем элитный вектор $\mathbf{q}^j$:

$$\text{если } J(\mathbf{g}^e) < J(\mathbf{q}^j), \text{ то } \mathbf{q}^j \leftarrow \mathbf{g}^e,\ J(\mathbf{q}^j) \leftarrow J(\mathbf{g}^e),\ e = 1, \ldots, E.$$

Векторы $\mathbf{q}^j$, $j = N+1, \ldots, L$ называются *отобранными*.

Производим эволюцию параметров для каждого из отобранных векторов. Для каждого $\mathbf{q}^j$, $j = N+1, \ldots, L$, строим набор из S векторов $\mathbf{g}^s = [\ g_1^s \quad \ldots \quad g_p^s]^T$ по формуле

$$g_i^s = \begin{cases} q_i^-, \text{ если } q_i^j + \xi_i^s < q_i^-, \\ q_i^+, \text{ если } q_i^j + \xi_i^s > q_i^+, \quad , s = 1, \ldots, S, \\ q_i^j + \xi_i^s - \text{иначе}, \end{cases}$$

где $\xi_i^s \in [-r_i^s; r_i^s]$ — величина, случайно распределенная на интервале от $-r_i^s$ до r_i^s, $\mathbf{r}^s$ — заданный вектор радиусов области поиска вокруг отобранного вектора параметров.

Если какой-либо вектор $\mathbf{g}^s$ доставляет значение функционала лучше, чем вектор $\mathbf{q}^j$, то его помещаем во множество возможных решений вместо вектора $\mathbf{q}^j$:

$$\text{если } J(\mathbf{g}^s) < J(\mathbf{q}^j), \text{ то } \mathbf{q}^j \leftarrow \mathbf{g}^s,\ J(\mathbf{q}^j) \leftarrow J(\mathbf{g}^s),\ s = 1, \ldots, S.$$

Для остальных векторов $\mathbf{q}^j$, $j = L+1, \ldots, H$, производим генерацию новых случайных значений по формуле (1).

Далее производим уменьшение радиусов области поиска вокруг элитных и отобранных векторов:

$$r_i^e \leftarrow \alpha_e r_i^e,\ r_i^s \leftarrow \alpha_s r_i^s,\ i = 1, \ldots, p,$$

где α_e, α_s — заданные параметры уменьшения областей поиска, $\alpha_e \approx 0.95$, $\alpha_s \approx 0.95$.

Процесс поиска вектора параметров продолжаем до достижения максимального числа итераций W. Лучший вектор в итоговом множестве возможных решений считаем решением задачи.

4. Вычислительный эксперимент

Для теста методов выбрана задача оптимального управления мобильным роботом с фазовыми ограничениями.

Задана математическая модель мобильного робота [11, 12]:

$$\dot{x}_1 = 0.5(u_1 + u_2)\cos(x_3),\ \dot{x}_2 = 0.5(u_1 + u_2)\sin(x_3),\ \dot{x}_3 = 0.5(u_1 - u_2),$$

где $\mathbf{x} = [\ x_1 \quad x_2 \quad x_3\ ]^T$ — вектор состояния, $\mathbf{u} = [\ u_1 \quad u_2\ ]^T$ — вектор управления.

Заданы ограничения на управление: $-10 \leqslant u_1 \leqslant 10$, $-10 \leqslant u_2 \leqslant 10$. Заданы начальные условия: $x_1^0 = 10$, $x_2^0 = 10$, $x_3^0 = 0$. Заданы терминальные условия: $x_1^f = 0$, $x_2^f = 0$, $x_3^f = 0$. Заданы четыре фазовых ограничения:

$$h_1(x) = 1.5 - \sqrt{(x_1 - 2.5)^2 + (x_2 - 2.5)^2} \leqslant 0,$$
$$h_2(x) = 1.5 - \sqrt{(x_1 - 7.5)^2 + (x_2 - 7.5)^2} \leqslant 0,$$
$$h_3(x) = 3 - \sqrt{(x_1 - 2)^2 + (x_2 - 8)^2} \leqslant 0,$$
$$h_4(x) = 3 - \sqrt{(x_1 - 8)^2 + (x_2 - 2)^2} \leqslant 0.$$

Задан функционал качества:

$$J = t_f + \sqrt{\sum_{i=1}^{3}(x_i(t_f) - x_i^f)^2} + \\ + \int_0^{t_f} (\vartheta(h_1(x))h_1(x) + \vartheta(h_2(x))h_2(x) + \vartheta(h_3(x))h_3(x) + \vartheta(h_4(x))h_4(x))dt \to \min,$$

где $t_f = \begin{cases} t, \text{ если } \sqrt{\sum_{i=1}^{3}(x_i(t) - x_i^f)^2} \leqslant \varepsilon, \\ t^+ - \text{иначе.} \end{cases}$

Заданы параметры модели: $t^+ = 2.5$, $\varepsilon = 0.01$, $\Delta t = 0.25$, $M = \lfloor 4/0.25 \rfloor = 16$, $p = 2(M+1) = 34$, $q_i^- = -20$, $q_i^+ = 20$, $i = 1, \ldots, p$, $\lambda_j = 1$, $j = 1, \ldots, 4$.

Необходимо найти управление в виде

$$\tilde{u}_k(t) = \begin{cases} u_k^-, \text{ если } q_{i+M(k-1)} + (q_{i+M(k-1)+1} - q_{i+M(k-1)})\left(\frac{t}{\Delta t} - i + 1\right) < u_k^-, \\ u_k^+, \text{ если } q_{i+M(k-1)} + (q_{i+M(k-1)+1} - q_{i+M(k-1)})\left(\frac{t}{\Delta t} - i + 1\right) > u_k^+, \\ q_{i+M(k-1)} + (q_{i+M(k-1)+1} - q_{i+M(k-1)})\left(\frac{t}{\Delta t} - i + 1\right) - \text{иначе,} \end{cases}$$

где $i\Delta t \leqslant t < (i+1)\Delta t$, $i = 1, \ldots, M$, $j = M+1, \ldots, 2M$, $k = 1, 2$, $q_i^- \leqslant q_i \leqslant q_i^+$, $i = 1, \ldots, p$.

Решением задачи является вектор параметров $\mathbf{q} = [\ q_1 \quad \ldots \quad q_p\ ]^T$.

Для всех методов был проведен вычислительный эксперимент с общим числом вычислений целевого функционала около 130 000. Параметры алгоритмов для экспериментов имели следующие значения: в методе наискорейшего градиентного спуска количество одномерных поисков $W = 60$, размер множества возможных решений $H = 50$; в методе Ньютона–Рафсона количество итераций $W = 5$, $H = 10$; в методе метод Марквардта $W = 10$, $H = 5$; в методе ADAM $\delta q = 0.0001$, $\varepsilon_1 = 0.0001$, $\varepsilon_2 = 0.0001$, $\alpha = 0.001$, $\beta_1 = 0.9$, $\beta_2 = 0.999$, $W = 200$, $H = 16$; в генетическом алгоритме $H = 256$, вероятность мутации 0.7, число бит под целую часть числа 4, под дробную часть 16, число возможных скрещиваний $W = 530$; в методе роя частиц $H = 32$, $\alpha = 0.7278$, $\beta = 0.5$, $\gamma = 0.1$, $\delta = 1$, $W = 4000$; в пчелином алгоритме $\alpha^e = 0.95$, $\alpha^s = 0.95$, $r_i^e = 4$, $r_i^s = 4$, $i = 1, \ldots, p$, $W = 400$, $H = 30$, $N = 6$, $L = 15$, $E = 30$, $S = 15$.

В эксперименте также использовали случайный поиск, в котором генерировали случайно 130 000 векторов и определяли из них лучший. В каждом эксперименте выполняли по 10 запусков. Результаты эксперимента приведены в табл. 1, где использованы следующие обозначения: GA — генетический алгоритм, PSO — метод роя частиц, BA — пчелиный алгоритм, FGD — метод наискорейшего градиентного спуска, NR — метод Ньютона–Рафсона, MQ — метод Марквардта, AD — метод ADAM, RS — случайный поиск, Ср. — среднее значение, СКО — среднеквадратичное отклонение. Во всех экспериментах определяли наилучшее, среднее и среднеквадратичное значения. В каждой колонке указаны места, занятые алгоритмом по каждому показателю. В последней колонке представлена сумма мест по трем показателям.

По результатам экспериментов по всем трем показателям лучшим оказался пчелиный алгоритм. Все эволюционные алгоритмы нашли решения лучше, чем случайный поиск. Из градиентных алгоритмов многоточечные методы Марквардта и ADAM дали приемлемые результаты по значениям целевой функции.

На рис. 1 представлена траектория движения робота на плоскости для лучшего найденного решения. Значение функционала для полученного возможного решения составляет 2.5100, число вызовов целевой функции — 132 032.

Таблица 1

Сравнительный анализ результатов эксперимента

Алгоритм	Лучший результат		Среднее значение		СКО		Сумма
BA	2.5100	1	2.7306	1	0.0911	1	3
PSO	2.5302	2	2.8506	2	0.1827	3	7
ADAM	2.8185	4	3.0375	3	0.1198	2	9
GA	2.5304	3	3.0671	4	0.3988	5	12
MQ	2.8852	5	3.3991	5	0.3412	4	14
FGD	3.0267	6	3.8374	6	0.6052	7	19
RS	3.3242	7	4.4672	7	0.5515	6	20
NR	6.1135	8	8.3549	8	1.7682	8	24

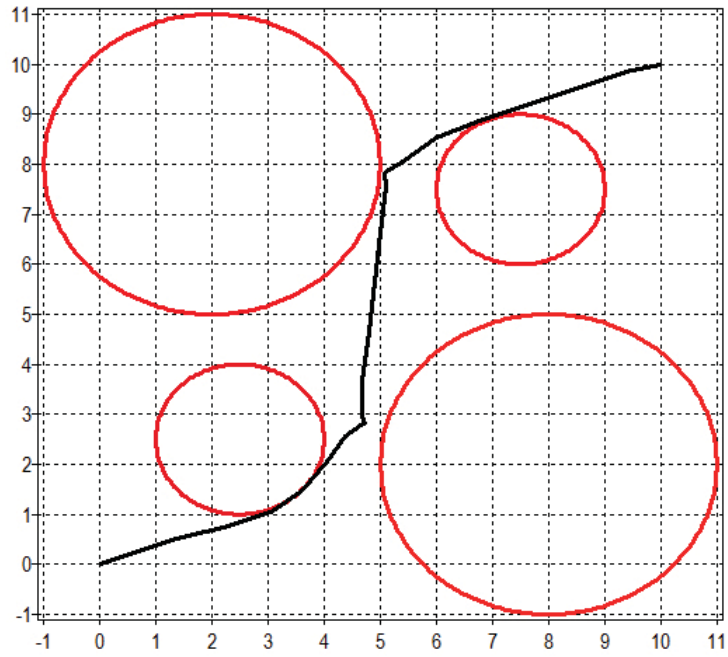

Рис. 1. Траектория движения мобильного робота на плоскости

Найденный наилучший вектор параметров имеет следующее значение:

$$\mathbf{q} = [\ 1.8824 \quad -18.5686 \quad -2.7510 \quad -19.4642 \quad 18.1277 \quad 19.2901 \quad 12.7768 \quad -5.6104$$
$$-15.9746 \quad -7.3604 \quad -5.5767 \quad 16.4129 \quad 4.4010 \quad -11.9245 \quad -6.0492 \quad -18.3288 \quad 2.8640$$
$$-7.8236 \quad -11.8720 \quad -18.2929 \quad 15.5538 \quad 13.1515 \quad 15.1281 \quad -12.5568 \quad -6.5664 \quad -3.4499$$
$$-15.7867 \quad 13.4663 \quad 4.2882 \quad -4.4374 \quad -16.7430 \quad 4.0273 \quad -16.1006 \quad -19.2252\ ]^T.$$

5. Заключение

В результате анализа задачи оптимального управления, редуцированной к задаче нелинейного программирования, определены основные особенности и проблемы численного решения задачи оптимального управления методами нелинейного программирования, которые заключаются в сложности вычисления целевой функции, неопределенности топологических свойств целевой функции в пространстве искомых параметров и большой размерностью пространства поиска.

Сравнительный вычислительный эксперимент решения различными эволюционными и многоточечными градиентными алгоритмами задачи оптимального управления с фазовыми ограничениями для математической модели мобильного робота третьего порядка показал, что популярные эволюционные алгоритмы, метод роя частиц, пчелиный алгоритм и

генетический алгоритм дают результаты лучше, чем многоточечные классические градиентные методы при том же количестве вычислений целевой функции. Многоточечный новый градиентный метод ADAM дает результаты, сравнимые с результатами, полученными эволюционными алгоритмами. Лучшим из всех используемых в эксперименте алгоритмов оказался пчелиный алгоритм по всем оцениваемым показателям, наилучшему найденному значению целевой функции, по среднему значению и среднеквадратичному отклонению. Все эволюционные алгоритмы показали значительно лучшие результаты, чем алгоритм случайного поиска при том же количестве вычислений значений целевой функции.

Работа выполнена при поддержке гранта РФФИ №17-08-01203-а.

Литература

1. *Евтушенко Ю.Г.* Оптимизация и быстрое автоматическое дифференцирование. М.: ВЦ РАН, 2013.
2. *Евтушенко Ю.Г.* Численный метод поиска глобального экстремума функций (перебор на неравномерной сетке) // Ж. вычисл. матем. и матем. физ. 1971. Т. 11, № 6. С. 1390–1403.
3. *Карпенко А.П.* Современные алгоритмы поисковой оптимизации. Алгоритмы, вдохновленные природой. М.: МГТУ им. Н. Э. Баумана, 2014.
4. *Пантелеев А.В., Летова Т.А.* Методы оптимизации в примерах и задачах. М.: Высшая школа, 2005.
5. *Kingma D.P., Ba J.* Adam: A Method for Stochastic Optimization // 3rd International Conference for Learning Representations. arXiv:1412.6980v8 [cs.LG]. 2015.
6. *Holland J.N.* Adaptation in Natural and Artificial Systems. — Michigan: Univ. Michigan Press, 1975.
7. *Goldberg D.E.* Genetic Algorithms in Search, Optimization, and Machine Learning. Addison-Wesley, 1989.
8. *Kennedy J., Eberhart R.* Particle Swarm Optimization // Proceedings of IEEE International Conference on Neural Networks IV. 1995. P. 1942–1948.
9. *Карпенко А.П., Селиверстов Е.Ю.* Глобальная оптимизация методом роя частиц. Обзор // Информационные технологии. 2010. № 2. С. 25–34.
10. *Pham D.T.* [*et al.*]. The Bees Algorithm — A Novel Tool for Complex Optimisation Problems // 2nd I*PROMS Virtual International Conference «Intelligent Production Machines and Systems». 2006. P. 454–459.
11. *Рапопорт Л.Б.* Оценка области притяжения в задаче управления колесным роботом // АиТ. 2006. № 9. С. 69–89.
12. *Пестерев А.В.* Синтез линеаризующего управления в задаче стабилизации движения автомобилеподобного робота вдоль криволинейного пути // Изв. РАН. ТиСУ. 2013. № 5. С. 153–165.

References

1. *Yevtushenko Yu.G.* Optimization and rapid automatic differentiation. Moscow: Computing Center of RAS, 2013. (in Russian).
2. *Yevtushenko Yu.G.* A numerical method for searching for a global extremum of functions (search on an uneven grid). Journal of Computational Mathematics and Mathematical Physics. 1971. V. 11, N 6. P. 1390–1403. (in Russian).

3. *Karpenko A.P.* Modern algorithms of searching optimization. Algorithms inspired by the nature. Moscow: MSTU named after N. E. Bauman, 2014. (in Russian).

4. *Panteleev A.V., Letova T.A.* Optimization methods in examples and tasks. Moscow: Higher School, 2005. (in Russian).

5. *Kingma D.P., Ba J.* Adam: A Method for Stochastic Optimization. 3rd International Conference for Learning Representations. arXiv:1412.6980v8 [cs.LG]. 2015.

6. *Holland J.N.* Adaptation in Natural and Artificial Systems. Michigan: Univ. Michigan Press, 1975.

7. *Goldberg D.E.* Genetic Algorithms in Search, Optimization, and Machine Learning. Addison-Wesley, 1989.

8. *Kennedy J., Eberhart R.* Particle Swarm Optimization. Proceedings of IEEE International Conference on Neural Networks IV. 1995. P. 1942–1948.

9. *Karpenko A.P., Seliverstov E.Yu.* Global optimization by the particle swarm optimization method. Overview. Information technology. 2010. N 2. P. 25–34. (in Russian).

10. *Pham D.T.* [*et al.*]. The Bees Algorithm — A Novel Tool for Complex Optimisation Problems. 2nd I*PROMS Virtual International Conference «Intelligent Production Machines and Systems». 2006. P. 454–459.

11. *Rapoport L.B.* Evaluation of the attraction domain in the control problem of the wheeled robot. Automation and Telemechanics. 2006. N 9. P. 69–89. (in Russian).

12. *Pesterev A.V.* Synthesis of Linearizing Control in the Problem of Stabilizing Motion of an Automobile-like Robot along a Curved Path. Proceedings of the RAS. Theory and Control Systems. 2013. N 5. P. 153–165. (in Russian).

Поступила в редакцию 10.07.2017