# Развитие алгоритма асинхронной дифференциальной эволюции

Е.И. Жабицкая[1,3], М.В. Жабицкий[2]
e-mail: Evgeniya.Zhabitskaya@jinr.ru, [1]ЛИТ ОИЯИ, Дубна
[2]ЛЯП ОИЯИ, Дубна
[3]Университет «Дубна», Дубна

## Введение

Будем рассматривать задачу поиска глобального минимума $\boldsymbol{x}^* = \{x_j\}|_{j=0,\ldots,D-1}$ функции $f(\boldsymbol{x})$: $\Omega \subset \mathbb{R}^D \to \mathbb{R}$:

$$f(\boldsymbol{x}^*) \leqslant f(\boldsymbol{x}) \; \forall \boldsymbol{x} \in \Omega \qquad (1)$$

Здесь $\Omega$ — вещественное пространство поиска решения, возможно с нелинейными ограничениями на допустимые значения параметров. В случае многомодальной функции $f(\boldsymbol{x})$ глобальный минимум может быть найден при помощи алгоритма дифференциальной эволюции ($DE$) [1], одного из из активно развиваемых в последние годы [2, 3] методов поиска глобального минимума. Отсутствие необходимости вычисления производных позволяет успешно применять этот эволюционный алгоритм для решения негладких задач, в том числе большой размерности.

Предложенный авторами в [4, 5] алгоритм Асинхронной Дифференциальной Эволюции ($ADE$) обеспечивает, при сохранении основных положительных качеств классической Дифференциальной Эволюции ($CDE$), более широкие возможности распараллеливания и ускорения вычислений.

## Алгоритм асинхронной дифференциальной эволюции

Метод ДЭ оперирует популяцией, каждый член которой является вектором в пространстве параметров: $P_x = \{\boldsymbol{x}_i\}$, $\boldsymbol{x}_i \in \Omega \subset \mathrm{R}^D$, $i = 0, 1, \ldots, N_p - 1$. Согласно $CDE$ [2] после инициализации начальной популяции, состоящей из $N_p$ векторов, случайно выбранных из $\Omega$, выполняется цикл, на каждом шаге которого ко всем членам текущей популяции применяются операции мутации, кроссовера и отбора для формирования следующего поколения. Смена поколения в $CDE$ происходит синхронно для всех членов популяции. Описанию классической дифференциальной эволюции и ее вариантов посвящены книга [2] и обзор [3].

В новом, предложенном авторами, алгоритме $ADE$ [4] упомянутые операторы мутации, кроссовера и отбора применяются для членов популяции без синхронизации по поколениям (см. схему на рис. 1). После инициализации в цикле алгоритма на первом этапе выбирается *целевой* вектор $\boldsymbol{x}_i$. На этапе мутации для него формируется *мутантный* вектор $\boldsymbol{v}_i$ путем прибавления к *базовому* вектору $\boldsymbol{x}_r$ разности случайно выбранных из текущей популяции векторов $\boldsymbol{x}_p$ и $\boldsymbol{x}_q$ с весом $F$. Далее, на этапе кроссовера (рекомбинации) из координат целевого и мутантного векторов строится *пробный* вектор $\boldsymbol{u}_i$: с вероятностью $C_r$ в качестве координаты пробного вектора берется координата мутантного вектора, с вероятностью $(1-C_r)$ — целевого. Обычно дополнительно налагается условие, чтобы хотя бы одна координата в пробном векторе отличалась от соответствующей координаты целевого вектора, для чего мутируют его случайную координату $j_{\mathrm{rand}}$. Значение целевой функции в пробной точке сравнивается со значением в целевой точке. В популяции остается тот из векторов, для которого значение целевой функции лучше.

Последовательность этапов выбора целевого вектора, мутации, рекомбинации и отбора повторяется в цикле до тех пор, пока не будет выполнен один из критериев остановки. Таким образом, в алгоритме $ADE$ нет обязательного для $CDE$ перебора всех членов популяции. Целевые вектора, для которых будут осуществляться операции мутации, кроссовера и отбора выбираются из популяции независимо, по одному, что упрощает распараллеливание и открывает возможности

```
//Инициализация популяции P_x = {x_i}|_{i=0,N_p-1},
InitializePopulation();        // x_i = {x_{i,j}}|_{j=0,D-1}
do {                           // выбор целевого вектора x_i
    i = ChooseTargetVector();
    //Мутация:    //r ≠ p ≠ q — случайные индексы
    v_i = x_r + F (x_p − x_q);        // мутантный вектор
    //Кроссовер:                //u_{i,j} — пробный вектор
    for (j = 0; j < D; j = j + 1)
      u_{i,j} = { v_{i,j}, if rand(0,1) ⩽ C_r, или j = j_rand
                { x_{i,j}, иначе
    //Отбор:
     if (f(u_i) < f(x_i))          x_i = u_i;
} while (пока не выполнен критерий остановки);
```

Рис. 1: Си-схема алгоритма Асинхронной Дифференциальной Эволюции ($ADE$).

для ускорения вычислений.

Способ выбора очередного целевого вектора является специфической чертой *ADE*. Другими факторами, определяющими конкретную стратегию поиска глобального минимума в рамках *ADE*, являются способ выбора базового вектора, число разностных векторов и тип кроссовера. Для идентификации различных вариантов *ADE* введены обозначения $DE/w/x/y/z$, расширяющие принятую в [2] символику. Здесь $w$ соответствует способу выбора целевого вектора: например, это может быть случайный ("rand") или худший ("worst") член популяции. Символ $x$ "отвечает" за способ выбора базового вектора: это может быть как случайный ("rand") или лучший ("best") член популяции, так и другие варианты стратегии. Число разностных векторов, которые добавляются к базовому при формирования мутантного вектора, соответствует $y$. Под $z$ закодирован тип кроссовера. Обычно используется биномиальное (равномерное) скрещивание ("bin").

## Алгоритм асинхронной дифференциальной эволюции с рестартом

Для решения практических задач алгоритм должен находить решение с вероятностью, близкой к единице. Для ДЭ вероятность сходимости к глобальному минимуму возрастает при увеличении размера популяции $N_p$. С другой стороны, для больших популяций характерна медленная скорость сходимости (см. рис. 2). Чтобы получить высокую вероятность нахождения глобального минимума, сохраняя приемлемую скорость сходимости, разработан алгоритм *ADE* с рестартом (*ADE-R*) [6, 7].

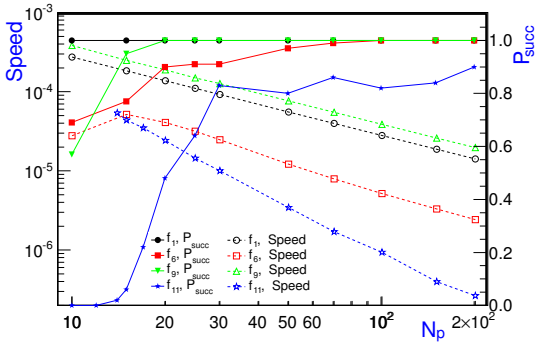


Рис. 2: Скорость сходимости Speed $= \langle N_{FE}^{-1} \rangle$ и вероятность сходимости $P_{\rm succ}$ для $rand/rand/1/bin$ стратегии метода *ADE* для $f_1$, $f_6$, $f_9$ & $f_{11}$ размерности $D = 10$ тестовых функций из CEC2005 [8].

*ADE-R* стартует с популяции небольшого размера $N_{p,\rm init}$. В течение итерационного процесса оцениваются разбросы координат членов популяции $\Delta x_j = \max\limits_{i=0,\ldots,N_p-1}\{x_{i,j}\} - \min\limits_{i=0,\ldots,N_p-1}\{x_{i,j}\}$, и значений целевой функции в популяции $\Delta f = \max\limits_{i=0,\ldots,N_p-1}\{f_i\} - \min\limits_{i=0,\ldots,N_p-1}\{f_i\}$. При выполнении критериев рестарта

$$\exists j \quad \Delta x_j < \varepsilon_{\mathbf{x}} \max_i\{|x_{i,j}|\}, \tag{2}$$

$$\text{или} \quad \Delta f < \varepsilon_{\mathbf{f}} \max_{i=0,\ldots,N_p-1}\{|f_i|\}, \tag{3}$$

алгоритм возобновляет вычисления с увеличенным в $k$ раз размером популяции: $N_p^{\rm new} = kN_p^{\rm old}$. Таким образом, осуществляется адаптация размера популяции в соответствии со сложностью решаемой проблемы [6]. Важно заметить, что i) $\varepsilon_x > 10^{-15}$, $\varepsilon_f > 10^{-15}$ — обусловлено машинной точностью вычислений и не зависит от задачи; ii) верхние границы для $\varepsilon_x$, $\varepsilon_f$ зависят от

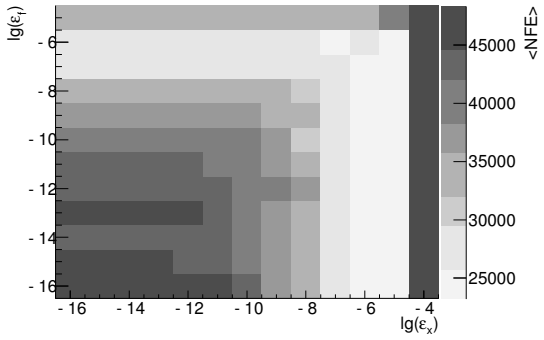


Рис. 3: Среднее количество вычислений функции $\langle N_{FE} \rangle$, необходимое для решения задачи Розенброка $f_6$ ($D = 10$), как функция $\varepsilon_x$ и $\varepsilon_f$. $ADE{-}R/worst/best/1/bin$-стратегия, $k = 2$, $\widehat{P}_{\rm succ} = N_{\rm succ}/N_{\rm total} = 1$.

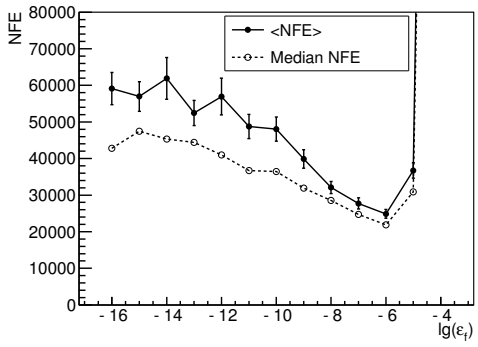


Рис. 4: Среднее количество вычислений функции необходимое для решения задачи Розенброка $f_6$ ($D = 10$) как функция $\varepsilon_f$ (left). без использования критерия $\Delta_x$ (3); $k = 2$. (right). $ADE/worst/best/1/bin$-стратегия, $k$=2, $P_{succ}$=1

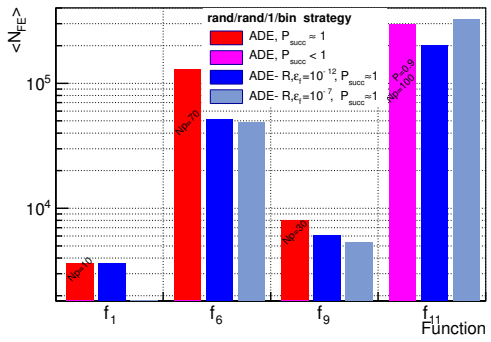


Рис. 5: Скорость сходимости для стратегии *rand/rand/1/bin* для *ADE* и *ADE-R*. Для *ADE*: $N_p{=}N_p^{\rm opt}$ — зависит от целевой функции; $P_{succ}{\simeq}1$ для $f_1$, $f_6$, $f_9$ и $P_{succ}{<}1$ для $f_{11}$. Для *ADE-R*: $\varepsilon_x{=}10^{-12}$, $\varepsilon_f{=}10^{-12}, 10^{-7}$; $N_p^{\rm init}{=}10$, $P_{succ}{\simeq}1\ \forall f$.

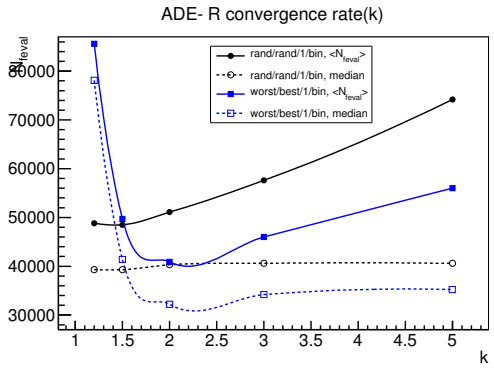


Рис. 6: Зависимость скорости сходимости от мультипликатора популяции $k$ для различных стратегий *ADE-R*, решающих задачу Розенброка $f_6$.

задачи; iii) для задач взвешенного метода наименьших квадратов $\max\varepsilon_f$ может быть оценено *a priori*.

На рисунках 3 и 4 представлены зависимости сходимости для $ADE{-}R/worst/best/1/bin$-стратегии как функции применяемых критериев рестарта. Видно, что даже очень маленькие значения $\varepsilon_x$ и $\varepsilon_f$, равные значению машинного эпсилон, умноженному на $10^3 \ldots 10^4$, гарантируют разумную скорость сходимости.

Сравнение скорости сходимости для стратегии *rand/rand/1/bin* для ADE и ADE-R при двух разных значениях $\varepsilon_x$ представлено на рис. 5

Зависимость скорости сходимости от мультипликатора популяции $k$ также обсуждалась в работе [6]. Она представлена на рис. 6. По нашему мнению, разумными значениями для $k$ являются значения $k \in (1.2 \ldots 3.0)$, причем для большинства задач $k \in [1.5, 2.0]$ будет наилучшим.

## Параллельная реализация *ADE*

При оптимизации целевых функций, требующих значительных вычислений, выгодно перейти к параллельной реализации алгоритма *ADE* (рис. 7). Нами реализована параллельная версия программы в стандарте OpenMP [4] и в стандарте передачи сообщений MPI [9, 13, 10] в рамках модели ведущий/ведомый (master/worker). Мастер оптимизации запрашивает пробные вектора из алгоритма и направляет их на расчет на ведомые процессоры. По мере расчета значений целевых функций результаты асинхронно передаются мастером в программу алгоритма. Класс *ADE* реализован таким образом, что для каждого пробного вектора отслеживается индекс соответствующего целевого вектора. Оператор отбора осуществляет сравнение вычисленного значения функции для пробного вектора с соответствующим значением для вектора текущей популяции с тем же индексом. Таким образом, в алгоритме корректно обрабатывается случай, когда первоначальный целевой вектор был замещен в популяции другим пробным вектором, рассчитанным с помощью более быстрого вычислительного потока. Освободившемуся ведомому процессору сразу же выставляется задание на расчет значения целевой функции следующего пробного вектора, что позволяет полностью и эффективно использовать все имеющиеся вычислительные мощности.

Для типичной оптимизационной задачи была получена модельная зависимость ускорения алгоритма *ADE* как функция доступного числа процессоров $N_{\rm proc}$ (рис. 8). Благодаря асинхронизации достигнуто лучшее ускорение по сравнению с классической дифференциальной эволюцией [4]. В [9] аналогичные оценки приведены для $ADE - R$.

В [13] (см. рис. 9) эффективность параллельной MPI-реализации алгоритма *ADE* тестировалось на LINUX-кластере ЛИТ ОИЯИ. Расчет запускался многократно с одними и теми же границами начальных и допустимых значений варьируемых параметров на разном количестве вычислительных узлов: $N_{\rm proc} \in [2; 128]$. В представ-

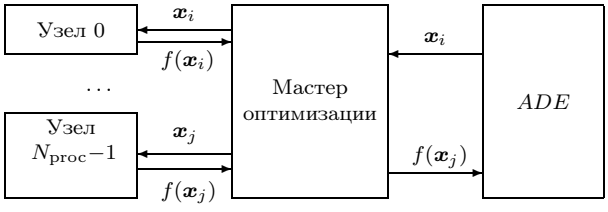


Рис. 7: Схема параллельной реализации алгоритма *ADE*

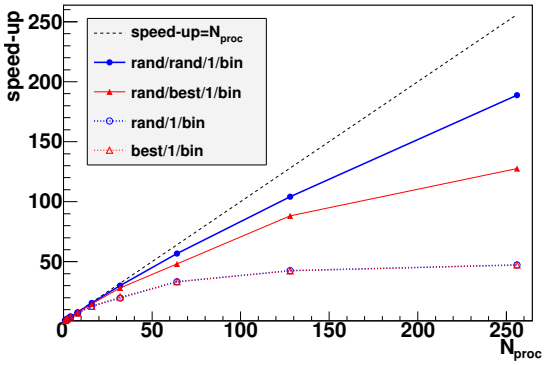


Рис. 8: Ускорение при параллельном расчете *ADE* [4]

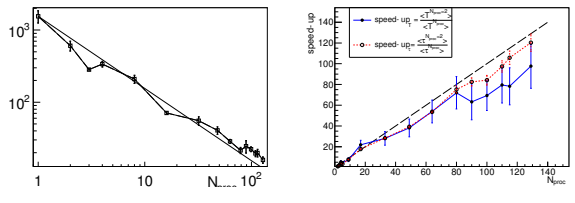

Рис. 9: Среднее время $\langle T \rangle$, затраченное на вычисления (а) и ускорение (speed-up $= \frac{T^{N_{\rm proc}=2}}{T^{N_{\rm proc}}}$) (б) в зависимости от числа задействованных процессоров $N_{\rm proc}$.

ленной статистике каждая комбинация запускалась $N = 10$ или 20 раз. На рис. 9 а) видно, что среднее время, реально затраченное на вычисления, падает практически линейно при росте числа задействованных вычислительных узлов. Ускорение вычислений (speed-up) с учетом возрастания необходимого количества вычислений функции при $N_{\rm proc} = 128$ составило $98 \pm 21$ раз. Рис. 9 б) демонстрирует близкое к линейному ускорение вплоть до $n_{\rm proc} = 128$ параллельно задействованных вычислительных узлов.

Метод *ADE-R* был успешно применен при решении ряда оптимизационных задач, возникающих в реальных физических исследованиях ([11, 12, 13] и др.). Эти задачи сводились к минимизации целевых функций со сложным многомодальным или оврагоподобным рельефом, поэтому их решение классическими методами поиска локального минимума затруднено.

Методы *ADE* и *ADE-R* использовались для определения параметров микроскопического оптического потенциала упругого рассеяния $\pi$-мезонов на ядрах [11, 12].

Параллельные алгоритмы *ADE* и *ADE-R* а также *ADE* с адаптивным кроссовером [10] применялись при исследовании структуры однослойных везикул DMPC [13].

## Заключение

Предложенный алгоритм прост в применении, имеет малое количество контрольных параметров ($N_p$, $F$ и $C_r$), применим для решения задач большой ($D = 10 \ldots 100 \ldots$) размерности. В *ADE* не используются производные, поэтому с его помощью можно решать недифференцируемые задачи. Алгоритм устойчив при оптимизации многомодальных функций. Асинхронизация ускоряет алгоритм при параллельных вычислениях, при этом сохраняются сравнимые с классической ДЭ вероятность и скорость сходимости [4]. Правильный подбор параметров алгоритма [5] и применение методов, позволяющих при необходимости возобновить вычисления [6, 7], помогают снизить вероятность вырождения и повысить вероятность сходимости к глобальному минимуму.

## Список литературы


[1] K.V. Price, R.M. Storn. // J. of Global Optimization. 1997. V. 11. P. 341.

[2] K.V. Price, R.M. Storn, J.A. Lampinen. Differential Evolution: A Practical Approach to Global Optimization. Springer-Verlag. Berlin Heidelberg. 2005.

[3] S. Das, P.N. Suganthan. IEEE Trans. Evol. Comput. 2011. V. 15. P. 4.

[4] Zhabitskaya, E.I., Zhabitsky, M.V. Lecture Notes in Computer Science, Springer, **7125**, 2012, pp. 328–333.

[5] Zhabitskaya, E.I. //Lecture Notes in Computer Science, **7125**, 2012, pp. 322–327.

[6] E. Zhabitskaya, M. Zhabitsky // I. Dimov, I. Farag, and L. Vulkov (Eds.): Fifth Conference on Numerical Analysisand Applications (NAA 2012), // LNCS 8236, pp. 555Ц561, 2013.

[7] Жабицкая, Е. И., Жабицкий, М. В. // Труды шестнадцатой научной молодых ученых и специалистов ОИЯИ, ОМУС XVI, 2012, с. 50–53.

[8] P.N. Suganthan, et al. Problem definitions and evaluation criteria for the CEC05 special session on real-parameter optimization. Technical report, Nanyang Technological University. Singapore. 2005.

[9] E. I. Zhabitskaya, M. V. Zhabitsky. Математическое моделирование, Т. 24, No. 12, 2012, С. 33–37.

[10] E. I. Zhabitskaya, M. V. Zhabitsky. Proceeding of the 15th Annual Conference on Genetic and Evolutionary Computation, USA, New York, 2013, pp. 455–462.

[11] Жабицкая, Е. И., Жабицкий, М. В. // Труды шестнадцатой научной молодых ученых и специалистов ОИЯИ, ОМУС XVI, 2012, с. 46–49.

[12] Жабицкая, Е.И., Жабицкий, М.В., Земляная, Е.В., Лукьянов, В.К. // Компьютерные Исследования и Моделирование, Т. 4 No 3, 2012, С. 585Ц595

[13] E. Zhabitskaya, E. Zemlyanaya, M. Kiselev // MMCP-2013. Book of Abstracts, 2012 с. 189.