СИСТЕМНИЙ АНАЛІЗ

**П.И. СТЕЦЮК**

УДК 519.8

# ТЕОРИЯ И ПРОГРАММНЫЕ РЕАЛИЗАЦИИ *r*-АЛГОРИТМОВ ШОРА[1]

**Аннотация.** Рассмотрены три вычислительные формы *r*-алгоритмов с различным объемом вычислений на одной итерации. Приведены результаты о сходимости предельного варианта *r*-алгоритмов для выпуклых гладких функций и $r_\mu(\alpha)$-алгоритма для выпуклых кусочно-гладких функций. Обсуждены практические аспекты варианта $r(\alpha)$-алгоритмов с постоянным $\alpha$ — коэффициентом растяжения пространства, и адаптивным способом регулировки шага в направлении нормированного антисубградиента в преобразованном пространстве переменных.

**Ключевые слова:** субградиентный метод, наискорейший спуск, разность субградиентов, растяжение пространства, *r*-алгоритм, метод сопряженных градиентов, адаптивный шаг, программная реализация.

## ВВЕДЕНИЕ

В работе [1], посвященной 75-летию со дня рождения Н.З. Шора, рассмотрены три его важнейшие идеи: обобщенный градиентный спуск (1962 г.), использование линейных неортогональных преобразований пространства для улучшения обусловленности овражных функций (1969 г.), двойственный подход к получению и уточнению оценок целевой функции в невыпуклых квадратичных моделях (1985 г.). Также описано применение этих идей в методах и алгоритмах, разработанных в Институте кибернетики им. В.М. Глушкова НАН Украины. Однако в [1] недостаточно внимания уделено *r*-алгоритмам, а их применение очень важно для подтверждения эффективности второй идеи для субградиентных методов с преобразованием пространства. В настоящей статье, приуроченной к 80-летию со дня рождения Н.З. Шора, дан более обширный анализ современных *r*-алгоритмов.

Субградиентные методы с растяжением пространства в направлении разности двух последовательных субградиентов предложены в [2, 3]. Они названы *r*-алгоритмами и являются одним из центральных результатов докторской диссертации Н.З. Шора (1970 г.). Программные реализации *r*-алгоритмов оказались конкурентноспособными как по времени счета, так и по точности результатов с наиболее эффективными методами решения гладких плохо обусловленных задач. Ускоренную сходимость *r*-алгоритмов при минимизации негладких выпуклых функций обеспечивает взаимосвязь в них двух принципов из численных методов оптимизации.

Первый принцип состоит в использовании процедуры наискорейшего спуска в направлении антисубградиента выпуклой функции в преобразованном пространстве переменных. Если поиск минимума функции осуществляется точно, то этот

[1] Работа выполнена при поддержке НАН Украины (проекты № 0117U000327, № 0116U004558) и Volkswagen Foundation (грант N 90 306).

принцип гарантирует монотонность по значениям выпуклой функции для точек минимизирующей последовательности, которая конструируется *r*-алгоритмами, а если приближенно, то «монотонность» по минимизируемой функции заменяется «почти монотонностью».

Однако наискорейший спуск для негладких функций можно зациклить, чтобы избежать этого, используется второй принцип. Последний предназначен для уменьшения степени вытянутости поверхностей уровня овражной функции в преобразованном пространстве переменных. Он состоит в использовании операции растяжения пространства в направлении разности двух последовательных субградиентов, где второй субградиент вычислен в точке минимума функции по направлению первого антисубградиента. В результате этого растяжения уменьшаются поперечные составляющие субградиентов вдоль направления к точке минимума, что обеспечивает достаточно быструю сходимость субградиентного процесса с растяжением пространства.

Комбинации этих принципов при определенной регулировке шага наискорейшего спуска (точного или приближенного) и соответствующем выборе коэффициента растяжения пространства обеспечивают ускоренную сходимость конкретных вариантов *r*-алгоритмов и гарантируют их монотонность (или почти монотонность) по значению минимизируемой функции. Это подтверждается результатами многочисленных применений *r*-алгоритмов в задачах линейного и нелинейного программирования, блочных задачах с различными схемами декомпозиции, при решении минимаксных и матричных задач оптимизации, для вычисления двойственных лагранжевых оценок в многоэкстремальных и комбинаторных задачах оптимизации [4–7].

Далее приведены теоретические результаты и современные программные реализации для *r*-алгоритмов Шора. Статья содержит три раздела: в первом описаны три вычислительные формы *r*-алгоритмов и проанализированы их свойства, во втором приведены теоретические результаты о сходимости *r*-алгоритмов для выпуклых функций, а в третьем — $r(\alpha)$-алгоритм с адаптивной регулировкой шага и его программные реализации.

### 1. ТРИ ВЫЧИСЛИТЕЛЬНЫЕ ФОРМЫ *r*-АЛГОРИТМОВ

Рассмотрим задачу минимизации выпуклой функции $f(x)$, где $x \in E^n$ — вектор из $n$ переменных. Минимальное значение функции обозначим $f^* = f(x^*)$, $x^* \in X^*$. Предположим, что $f(x)$ имеет ограниченное множество минимумов $X^*$, т.е. выполняется условие $\lim\limits_{||x|| \to \infty} f(x) = +\infty$. Это условие обеспечивает корректность регулировки шага в *r*-алгоритмах. Обозначим $\{\alpha_k\}_{k=0}^{\infty}$ набор коэффициентов растяжения пространства, таких что $\alpha_k > 1$.

**Определение 1.** *r*-Алгоритмом для минимизации $f(x)$ называется итеративная процедура нахождения последовательности $n$-мерных векторов $\{x_k\}_{k=0}^{\infty}$ и последовательности $n \times n$-матриц $\{B_k\}_{k=0}^{\infty}$ по следующему правилу:

$$x_{k+1} = x_k - h_k B_k \xi_k,\ B_{k+1} = B_k R_{\beta_k}(\eta_k),\ k = 0, 1, 2, \ldots, \tag{1}$$

где

$$\xi_k = \frac{B_k^{\mathrm{T}} g_f(x_k)}{||B_k^{\mathrm{T}} g_f(x_k)||},\ h_k \geq h_k^* = \arg\min_{h \geq 0} f(x_k - h B_k \xi_k), \tag{2}$$

$$\beta_k = \frac{1}{\alpha_k} < 1,\ \eta_k = \frac{B_k^{\mathrm{T}} r_k}{||B_k^{\mathrm{T}} r_k||},\ r_k = g_f(x_{k+1}) - g_f(x_k). \tag{3}$$

Здесь $x_0$ — стартовая точка; $B_0 = I_n$ — единичная $n \times n$-матрица (в качестве матрицы $B_0$ часто выбирают диагональную матрицу $D_n$ с положительными коэффициентами по диагонали, с помощью которой осуществляется масштабирование переменных); $h_k^*$ — величина шага из условия минимума функции $f(x)$ в направлении нормированного антисубградиента в преобразованном пространстве переменных; $R_\beta(\eta) = I_n + (\beta - 1)\eta\eta^{\mathrm{T}}$ — оператор сжатия пространства субградиентов в нормированном направлении $\eta$ с коэффициентом $\beta = \dfrac{1}{\alpha} < 1$; $g_f(x_k)$ и $g_f(x_{k+1})$ — субградиенты функции $f(x)$ соответственно в точках $x_k$ и $x_{k+1}$. Если на итерации $k$ для процесса (1)–(3) выполнены некоторые критерии (условия) останова, то полагаем $k^* = k$, $x_k^* = x_k$ и заканчиваем работу алгоритма.

На каждой итерации *r*-алгоритмов реализуется субградиентный спуск для выпуклой функции $\varphi(y) = f(B_k y)$ в преобразованном пространстве переменных $y = A_k x$, где $A_k = B_k^{-1}$. Действительно, если обе части формулы $x_{k+1} = x_k - h_k B_k \xi_k$ умножить слева на матрицу $A_k$, то получим

$$y_{k+1} = A_k x_{k+1} = A_k x_k - h_k \xi_k = y_k - h_k \frac{B_k^{\mathrm{T}} g_f(x_k)}{||B_k^{\mathrm{T}} g_f(x_k)||} = y_k - h_k \frac{g_\varphi(y_k)}{||g_\varphi(y_k)||}, \quad (4)$$

где вектор $g_\varphi(y_k) = B_k^{\mathrm{T}} g_f(x_k)$ — субградиент функции $\varphi(y) = f(B_k y)$ в точке $y_k = A_k x_k$ пространства переменных $y = A_k x$. Это легко видеть из того, что субградиент функции $f(x)$ в точке $x_k$ удовлетворяет неравенству

$$f(x) \geq f(x_k) + (g_f(x_k))^{\mathrm{T}} (x - x_k) \quad \forall x \in E^n,$$

откуда, осуществляя замену переменных $x = B_k y$, получаем

$$\varphi(y) \geq \varphi(y_k) + (B_k^{\mathrm{T}} g_f(x_k))^{\mathrm{T}} (y - y_k) = \varphi(y_k) + (g_\varphi(y_k))^{\mathrm{T}} (y - y_k) \quad \forall y \in E^n.$$

Если $h_k = h_k^*$, то формула (4) означает точный поиск минимума функции $\varphi(y) = f(B_k y)$ в направлении нормированного антисубградиента в преобразованном пространстве переменных $y = A_k x$, а если $h_k \approx h_k^*$, — то приближенный поиск. Если функция $f(x)$ является недифференцируемой в точке $x_k$, то возможен случай $h_k = h_k^* = 0$, с которым связаны основные проблемы с условием останова *r*-алгоритмов для негладких функций. Если $h_k = 0$, то это не означает, что в точке $x_k$ спуск следует прекратить. В процессе выполнения серии итераций с нулевым шагом изменяется не очередная точка $x_{k+1} = x_k$, а матрица $B_{k+1}$ и очередной субградиент $g_f(x_{k+1})$. Значит, за счет последовательных растяжений пространства в направлении разности двух субградиентов, которые получены в точке $x_k$, осуществляется поиск подходящего направления убывания функции из точки $x_k$.

Монотонность по минимизируемой функции для *r*-алгоритмов обеспечивает именно растяжение пространства в направлении разности двух последовательных субградиентов. Этим *r*-алгоритмы отличаются от субградиентных методов с растяжением пространства в направлении субградиента, которые для негладких выпуклых функций в принципе не могут быть монотонными по минимизируемой функции. Проиллюстрируем это на примере кусочно-гладкой функции (рис. 1). Пусть имеется точка на границе двух «кусков» кусочно-гладкой поверхности уровня, а градиенты к этим гладким «кускам», вычисленные в данной

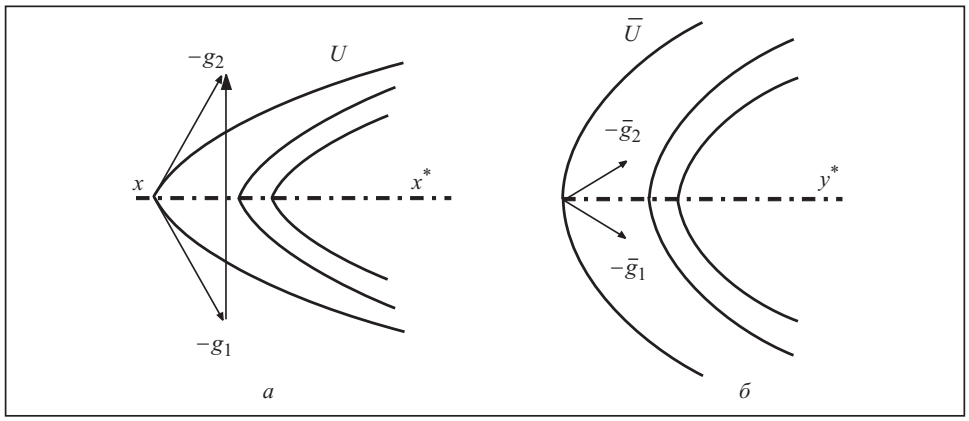


*Рис. 1.* Растяжение пространства по разности двух субградиентов, где антисубградиенты становятся направлениями убывания функции

точке, образуют тупой угол (см. рис 1, *а*). Никакое растяжение пространства в направлении градиентов не способно преобразовать этот угол в острый, величина последнего может лишь приближаться к $\pi/2$, при этом угол остается тупым. Поэтому, применяя растяжение пространства в направлении субградиента, невозможно получить направление убывания функции в виде антиградиента к одному из кусков в растянутом пространстве. В то же время растяжение пространства в направлении разности двух указанных градиентов с достаточным коэффициентом растяжения преобразует тупой угол между градиентами в острый, т.е. соответствующие образы этих антиградиентов в растянутом пространстве становятся направлениями убывания функции (см. рис. 1, *б*).

Итак, семейство *r*-алгоритмов определяется последовательностью коэффициентов растяжения пространства $\{\alpha_k\}_{k=0}^{\infty}$, последовательностью величин шагов $\{h_k\}_{k=0}^{\infty}$ и критериями останова. При этом величина шага выбирается из условия точного (приближенного) минимума функции по направлению антисубградиента в преобразованном пространстве переменных, вследствие чего определяются те два последовательных субградиента, растяжение по разности которых улучшает свойства овражной функции в преобразованном пространстве переменных. В случае определенной регулировки шага и коэффициентов растяжения пространства *r*-алгоритмы являются монотонными (или почти монотонными) по минимизируемой функции.

Метод (1)–(3) называют *B*-формой *r*-алгоритмов; на каждой его итерации корректируется матрица, которая определяет замену переменных $x = By$. Итерация метода требует приблизительно $5n^2$ арифметических операций умножения, определяющих вычислительную трудоемкость итерации (операции сложения не учитываются ввиду их малого влияния на трудоемкость итерации). Из них $3n^2$ умножений необходимо для вычисления векторов $B_k\xi_k$, $B_k^{\mathrm{T}} g_f(x_k)$ и $B_k^{\mathrm{T}} r_k$ (умножение матрицы на вектор), а $2n^2$ умножений — для одноранговой коррекции матрицы $B_{k+1} = B_k R_{\beta_k}(\eta_k)$. Действительно,

$$B_{k+1} = B_k R_{\beta_k}(\eta_k) = B_k (I_n + (\beta_k - 1)\eta_k \eta_k^{\mathrm{T}}) = B_k + (\beta_k - 1)(B_k \eta_k)\eta_k^{\mathrm{T}},$$

откуда легко видеть, что вычисление вектора $\eta = B_k \eta_k$ и построение одноранговой матрицы $\eta\eta_k^{\mathrm{T}}$ требуют $2n^2$ умножений.

Рассматриваемые *r*-алгоритмы имеют еще одну *B*-форму, которая по сравнению с методом (1)–(3) позволяет сэкономить $n^2$ операций умножения на каждой

итерации. Экономные $r$-алгоритмы представляются итеративной процедурой поиска последовательностей векторов $\{x_k\}_{k=0}^{\infty}$ и матриц $\{B_k\}_{k=0}^{\infty}$ по такому правилу:

$$x_{k+1} = x_k - h_k B_k \frac{\widetilde{g}_k}{||\widetilde{g}_k||},\ B_{k+1} = B_k R_{\beta_k}(\eta_k),\ k = 0, 1, 2, \dots, \tag{5}$$

где

$$h_k \geq h_k^* = \arg\min_{h \geq 0} f\left(x_k - hB_k \frac{\widetilde{g}_k}{||\widetilde{g}_k||}\right),\ g_{k+1}^* = B_k^{\mathrm{T}} g_f(x_{k+1}), \tag{6}$$

$$\beta_k = \frac{1}{\alpha_k},\ \eta_k = \frac{g_{k+1}^* - \widetilde{g}_k}{||g_{k+1}^* - \widetilde{g}_k||},\ \widetilde{g}_{k+1} = R_{\beta_k}(\eta_k) g_{k+1}^*. \tag{7}$$

Здесь $x_0$ — стартовая точка, такая что $x_0 \neq x^*$; $\widetilde{g}_0 = B_0^{\mathrm{T}} g_f(x_0)$, где $B_0$ — невырожденная $n \times n$-матрица; $g_f(x_k)$ и $g_f(x_{k+1})$ — субградиенты функции $f(x)$ в точках $x_k$ и $x_{k+1}$. Если на итерации $k$ выполнены некоторые критерии останова, то полагаем $k^* = k$, $x_k^* = x_k$ и заканчиваем работу алгоритма.

Итерация метода (5)–(7) требует приблизительно $4n^2$ операций умножения: по $2n^2$ операции умножения для вычисления векторов $B_k \widetilde{g}_k$, $B_k^{\mathrm{T}} g_f(x_{k+1})$ и для одноранговой коррекции матрицы $B_{k+1} = B_k R_{\beta_k}(\eta_k)$. Экономия в $n^2$ умножений связана с тем, что $\widetilde{g}_{k+1} = B_{k+1}^{\mathrm{T}} g_f(x_{k+1})$ — субградиент в пространстве переменных $y = A_{k+1}x$, пересчитывается с учетом уже вычисленного $g_{k+1}^* = B_k^{\mathrm{T}} g_f(x_{k+1})$ — субградиента в пространстве переменных $y = A_k x$. Пересчет субградиента $\widetilde{g}_{k+1}$ выполняется по формуле

$$\widetilde{g}_{k+1} = B_{k+1}^{\mathrm{T}} g_f(x_{k+1}) = R_{\beta_k}(\eta_k) B_k^{\mathrm{T}} g_f(x_{k+1}) = R_{\beta_k}(\eta_k) g_{k+1}^* =$$

$$= (I_n + (\beta_k - 1)\eta_k \eta_k^{\mathrm{T}}) g_{k+1}^* = g_{k+1}^* + (\beta_k - 1)(\eta_k^{\mathrm{T}} g_{k+1}^*)\eta_k,$$

которая не требует операции умножения матрицы на вектор. В результате этого пересчета при вычислении нормированного субградиента в преобразованном пространстве по формуле $\xi_k = \dfrac{\widetilde{g}_k}{||\widetilde{g}_k||}$ происходит большее накопление ошибок, чем при вычислении по формуле $\xi_k = \dfrac{B_k^{\mathrm{T}} g_f(x_k)}{||B_k^{\mathrm{T}} g_f(x_k)||}$, как в методе (1)–(3).

Можно записать $r$-алгоритмы в $H$-форме (по типу методов переменной метрики) с помощью симметричной матрицы $H_k = B_k B_k^{\mathrm{T}}$. Им соответствует итеративная процедура отыскания последовательностей векторов $\{x_k\}_{k=0}^{\infty}$ и симметричных матриц $\{H_k\}_{k=0}^{\infty}$ по следующему правилу:

$$x_{k+1} = x_k - h_k \frac{H_k g_f(x_k)}{\sqrt{(g_f(x_k))^{\mathrm{T}} H_k g_f(x_k)}},$$

$$H_{k+1} = H_k + (\beta_k^2 - 1)\frac{H_k r_k r_k^{\mathrm{T}} H_k}{r_k^{\mathrm{T}} H_k r_k},\ k = 0, 1, \dots, \tag{8}$$

где

$$h_k \geq h_k^* = \arg\min_{h\geq 0} f\left( x_k - h\frac{H_k g_f(x_k)}{\sqrt{(g_f(x_k))^{\mathrm{T}} H_k g_f(x_k)}} \right), \qquad (9)$$

$$\beta_k = \frac{1}{\alpha_k} < 1,\ r_k = g_f(x_{k+1}) - g_f(x_k).$$

Здесь $x_0$ — стартовая точка; $H_0 = I_n$ — единичная $n\times n$-матрица; $h_k$ — величина шага, не меньшая $h_k^*$; $g_f(x_k)$ и $g_f(x_{k+1})$ — субградиенты функции $f(x)$ соответственно в точках $x_k$ и $x_{k+1}$. Если на итерации $k$ выполнены условия останова, то полагаем $k^* = k$, $x_k^* = x_k$ и заканчиваем работу алгоритма.

В $H$-форме $r$-алгоритмов формула для пересчета очередного приближения $x_{k+1}$ следует из справедливости следующей цепочки соотношений:

$$B_k \frac{B_k^{\mathrm{T}} g_f(x_k)}{||B_k^{\mathrm{T}} g_f(x_k)||} = \frac{B_k B_k^{\mathrm{T}} g_f(x_k)}{\sqrt{(B_k^{\mathrm{T}} g_f(x_k))^{\mathrm{T}} B_k g_f(x_k)}} =$$

$$= \frac{H_k g_f(x_k)}{\sqrt{(g_f(x_k))^{\mathrm{T}} B_k B_k^{\mathrm{T}} g_f(x_k)}} = \frac{H_k g_f(x_k)}{\sqrt{(g_f(x_k))^{\mathrm{T}} H_k g_f(x_k)}},$$

а формула для пересчета симметричной матрицы $H_{k+1}$ — из такой цепочки соотношений:

$$H_{k+1} = B_{k+1}B_{k+1}^{\mathrm{T}} = B_k R_{\beta_k}(\eta_k)(B_k R_{\beta_k}(\eta_k))^{\mathrm{T}} = B_k R_{\beta_k}(\eta_k) R_{\beta_k}^{\mathrm{T}}(\eta_k) B_k^{\mathrm{T}} =$$

$$= B_k R_{\beta_k}(\eta_k) R_{\beta_k}(\eta_k) B_k^{\mathrm{T}} = B_k R_{\beta_k^2}(\eta_k) B_k^{\mathrm{T}} = B_k (I_n + (\beta_k^2 - 1)\eta_k \eta_k^{\mathrm{T}}) B_k^{\mathrm{T}} =$$

$$= B_k B_k^{\mathrm{T}} + (\beta_k^2 - 1) B_k \eta_k \eta_k^{\mathrm{T}} B_k^{\mathrm{T}} = H_k + (\beta_k^2 - 1)\frac{B_k B_k^{\mathrm{T}} r_k r_k^{\mathrm{T}} B_k B_k^{\mathrm{T}}}{||B_k^{\mathrm{T}} r_k||^2} =$$

$$= H_k + (\beta_k^2 - 1)\frac{H_k r_k r_k^{\mathrm{T}} H_k}{(B_k^{\mathrm{T}} r_k)^{\mathrm{T}} B_k^{\mathrm{T}} r_k} = H_k + (\beta_k^2 - 1)\frac{H_k r_k r_k^{\mathrm{T}} H_k}{r_k^{\mathrm{T}} B_k B_k^{\mathrm{T}} r_k} = H_k + (\beta_k^2 - 1)\frac{H_k r_k r_k^{\mathrm{T}} H_k}{r_k^{\mathrm{T}} H_k r_k}.$$

Упомянутая $H$-форма $r$-алгоритмов более экономна, чем $B$-форма: по оперативной памяти — почти в два раза, так как необходимо хранить симметричную матрицу, а по трудоемкости — как минимум в $5/3$ раз. Действительно, если даже симметричную матрицу $H_k$ хранить как полную матрицу размера $n\times n$, то итерация метода (8), (9) требует приблизительно $3n^2$ операций умножения, из них $2n^2$ умножений — для вычисления векторов $H_k g_f(x_k)$ и $\eta = H_k r_k$, а $n^2$ умножений — для вычисления одноранговой матрицы $\eta\eta^{\mathrm{T}} = H_k r_k r_k^{\mathrm{T}} H_k$, которая используется при пересчете матрицы $H_{k+1}$. Но этот выигрыш нивелируется тем, что $H$-форма $r$-алгоритмов вычислительно менее устойчива, чем $B$-форма, и при ее реализации нужно учитывать дополнительные условия. Так, например, для метода (8), (9) необходимо контролировать, чтобы матрица $H_k$ была положительно-определенной. В методах (1)–(3) и (5)–(7) такого контроля не требуется, так как вычисления, хотя и неявно, связаны с положительно-определенной матрицей $H_k = B_k B_k^{\mathrm{T}}$.

Вычислительные характеристики рассмотренных форм $r$-алгоритмов по затратам памяти и трудоемкости приведены в табл. 1. И хотя теоретически все три формы $r$-алгоритмов одинаковы, вычислительная устойчивость их компьютерных

**Таблица 1.** Вычислительные характеристики трех форм *r*-алгоритмов

| Форма *r*-алгоритмов | Используемый метод | Оперативная память | Трудоемкость итерации | Устойчивость метода |
|---|---|---|---|---|
| *B*-форма | (1)–(3) | $\sim n^2$ | $\sim 5n^2$ | хорошая (+) |
| Экономная *B*-форма | (5)–(7) | $\sim n^2$ | $\sim 4n^2$ | хорошая |
| *H*-форма | (8), (9) | $\sim n^2/2$ | $\sim 3n^2$ | средняя |

реализаций (см. табл. 1, последний столбец) различна. Для обеих *B*-форм она хорошая, но преимущество имеет метод (1)–(3) (см. табл. 1, отмечено знаком +). Это обусловлено тем, что пересчет субградиента при переходе в очередное пространство переменных, используемый в экономной *B*-форме, способствует накоплению ошибок по отношению к вычислению этого же субградиента методом (1)–(3). Вычислительная устойчивость *r*-алгоритмов в *H*-форме средняя. Это означает, что с их помощью вообще нельзя найти такие приближения к точке минимума, определение которых возможно с помощью *r*-алгоритмов в *B*-форме. Можно использовать *r*-алгоритмы в *H*-форме, если не требуется высокой точности нахождения минимума функции $f(x)$.

## 2. ТРИ ТЕОРЕМЫ О СХОДИМОСТИ *r*-АЛГОРИТМОВ

Для выпуклых функций теоретические результаты о сходимости *r*-алгоритмов связаны с их модификациями: с предельным вариантом *r*-алгоритмов для гладких функций [3] и с $r_\mu(\alpha)$-алгоритмом для негладких функций [8]. Первый результат состоит в том, что предельный вариант *r*-алгоритмов является проективным методом сопряженных градиентов. Второй результат заключается в том, что для задачи минимизации выпуклой дважды непрерывно дифференцируемой функции $f(x)$ предельный вариант *r*-алгоритмов с восстановлением имеет квадратичную скорость сходимости при некоторых условиях гладкости и регулярности $f(x)$. Третий результат состоит в том, что при определенных условиях $r_\mu(\alpha)$-алгоритм сходится к точке минимума для кусочно-гладких выпуклых функций. Данные результаты и условия их определения рассмотрены далее.

В предельном варианте *r*-алгоритмов коэффициент растяжения пространства полагается бесконечным (при этом $\beta_k = 0$, $k = 0, 1, \ldots$), а шаг выбирается из условия минимума функции $f(x)$ по направлению антисубградиента ($h_k = k_k^*$, $k = 0, 1, \ldots$). Если $\beta_k = 0$, то оператор $R_{\beta_k}(\eta_k)$ определяется формулой

$$R_0(\eta_k) = I_n - \eta_k \eta_k^{\mathrm{T}}, \tag{10}$$

где $\eta_k = \dfrac{B_k r_k}{||B_k r_k||}$, $k = 1, 2, \ldots, n$. Формула (10) означает, что $R_0(\eta_i)$ — оператор проектирования на подпространство, ортогональное вектору $\eta_i$. Произведение операторов $\prod_{i=1}^{k} R_0(\eta_i)$ не зависит от порядка сомножителей, является самосопряженным оператором и осуществляет проекцию на подпространство ортогональным дополнением к линейной оболочке взаимно ортогональных векторов $\eta_i$, $i = 1, 2, \ldots, k$. Отсюда следует такая теорема.

**Теорема 1** [3]. Для предельного варианта *r*-алгоритмов на некоторой итерации $k^* \le n$ обязательно выполнится условие $B_{k^*}^{\mathrm{T}} g_f(x_{k^*}) = 0$.

Предельный вариант *r*-алгоритмов является проективным методом сопряженных градиентов. Для неотрицательно-определенной квадратичной функции он находит точку минимума $x^*$ за количество итераций, не превышающее $n$ — размерности вектора переменных. Из теоремы 1 следует, что после $k^*$ итераций для предельного варианта *r*-алгоритмов продолжать вычисления невозможно ввиду того, что направление движения в преобразованном пространстве становится нулевым. Следовательно, для минимизации гладких функций $f(x)$ в предельном варианте *r*-алгоритма после выполнения условия $B_{k^*}^{\mathrm{T}} g_f(x_{k^*}) = 0$ необходимо применять «восстановление», т.е. после некоторого количества итераций, не превышающего $n$, требуется «восстанавливать» матрицу $B_k$, заменяя ее единичной матрицей $I_n$. Построенный таким образом алгоритм называют предельным вариантом *r*-алгоритмов с восстановлением. Справедлива следующая теорема.

**Теорема 2** [3]. Пусть функция $f(x)$, определенная в $E^n$, дважды непрерывно дифференцируема в некоторой окрестности $S$ точки минимума $x^*$, причем в этой окрестности матрица вторых производных (гессиан) $H(x)$ удовлетворяет условию Липшица

$$||H(x) - H(x')|| \le L||x - x'||, \; x, x' \in S. \tag{11}$$

Кроме того, $H(x^*)$ — положительно-определенная матрица. Тогда для точки $x^*$ найдется такая окрестность $S' \subseteq S$, что если $x_0 \in S'$, то найдется такое число $c > 0$, что $||x_n - x^*|| \le c||x_n - x^*||^2$, где $x_n$ — точка, полученная после $n$ шагов работы предельного варианта *r*-алгоритма с восстановлением (если для некоторого $k^* < n$ получилось $B_{k^*}^{\mathrm{T}} g_f(x_{k^*}) = 0$, то полагаем $x_n = x_{k^*}$).

Итак, для задачи минимизации выпуклой дважды непрерывно дифференцируемой функции $f(x)$ предельный вариант *r*-алгоритмов с восстановлением матрицы $B_k$ после каждых $n$ итераций имеет квадратичную скорость сходимости при условиях гладкости и регулярности $f(x)$, указанных в теореме 2.

Отметим, что для минимизации гладких выпуклых функций *r*-алгоритмы занимают промежуточное место между методом наискорейшего спуска и алгоритмами квазиньютоновского типа с переменной метрикой. Действительно, если $\beta_k = 1$ и $h_k = k_k^*$, то растяжений пространства не требуется и *r*-алгоритм переходит в метод наискорейшего спуска. Если $\beta_k = 0$ и $h_k = k_k^*$, то получаем предельный вариант *r*-алгоритмов, который является проективным методом сопряженных градиентов и для квадратичной выпуклой функции сходится не более чем за $n$ итераций. Если $\beta_k = \beta < 1$ и $h_k = k_k^*$, то получим вариант *r*-алгоритмов, где восстановление матрицы $B_k$ не требуется, а алгоритм будет сходиться быстрее, чем метод наискорейшего спуска. Этим в значительной мере объясняется замечательное свойство *r*-алгоритмов, которое заключается в том, что их конкретные реализации показывают очень хорошие результаты при минимизации овражных выпуклых функций.

В работах Н.З. Шора $r_\mu(\alpha)$-алгоритм создавался для минимизации почти дифференцируемых функций, класс которых приведен в [9] и является более широким, чем класс выпуклых функций. В связи с этим Б. Мордухович, М. Солодов и М. Тодд в предисловии спецвыпуска журнала «Optimization Methods and Software» [10], посвященному Н.З. Шору, отметили: «В 1972 г. Н.З. Шор ввел фундаментальное понятие обобщенного дифференциала для локально липшицевых функций, которое он назвал «множество почти градиентов». Оно определено как совокупность предельных точек обычных градиентов липшицевой непрерывной функции, которая является почти всюду дифференцируемой по классической теоре-

ме Радемахера. Это предельное множество впоследствии широко применялось под названием $B$-градиента и $B$-якобиана вектор-функций при разработке негладких версий ньютоновского метода. Стоит отметить, что в этой же статье Шор также ввел и использовал понятие выпуклой оболочки множества почти градиентов, которую он назвал «множеством обобщенных почти градиентов». Это множество было переоткрыто Ф. Кларком и стало широко известным в недифференцируемой оптимизации как обобщенный градиент Кларка для липшицевых функций».

Подробно изложены результаты о сходимости $r_\mu(\alpha)$-алгоритма для минимизации почти дифференцируемых функций в работах [8] и [4, с. 102–113]. Поэтому далее описано только к чему приводят эти результаты в задачах минимизации кусочно-гладких функций вида

$$f(x) = \max_{1 \le i \le m} f_i(x), \tag{12}$$

где $f_i(x)$ — гладкие выпуклые функции, $i = 1, 2, \ldots, m$.

Функция (12) является выпуклой. Воспользуемся для ее минимизации $r_\mu(\alpha)$-алгоритмом. Обозначим $G_f(x) = \left\{ \bigcup_{i \in I(x)} g_{f_i}(x) \right\}$, где $I(x) = \{i \mid f_i(x) = f(x)\}$, $g_{f_i}(x)$ — градиент функции $f_i(x)$ в точке $x$.

Пусть $\alpha > 1$ — коэффициент растяжения пространства; $\mu$ — константа, такая что $0 \le \mu < 1$; $x_0$ — начальная точка; $g_f(x_0)$ — субградиент функции $f$ в точке $x_0$ (не любой субградиент, а выбранный из множества $G_f(x_0)$, т.е. $g_f(x_0) \in G_f(x_0)$); $B_0$ — невырожденная $n \times n$-матрица. Рассматриваемый $r_\mu(\alpha)$-алгоритм является итерационной процедурой построения последовательностей векторов $\{x_k\}_{k=0}^{\infty}$ и матриц $\{B_k\}_{k=0}^{\infty}$, где переход от $k$-й итерации к $(k+1)$-й осуществляется по следующему правилу.

1. Вычисляем очередную точку вида

$$x_{k+1} = x_k - \rho_k B_k g_{\varphi_k}(y_k), \tag{13}$$

где $g_{\varphi_k}(y_k) = B_k^{\mathrm{T}} g_f(x_k)$, а величина шага $\rho_k$ выбирается из условий:

а) на отрезке $[0, \rho_k]$ функция

$$\varphi_k(\rho) = f(x_{k+1}(\rho)) \tag{14}$$

не возрастает;

б) существует $g \in G_f(x_{k+1})$ такой, что

$$\frac{(B_k^{\mathrm{T}} g)^{\mathrm{T}} g_{\varphi_k}(y_k)}{\| B_k^{\mathrm{T}} g \| \, \| g_{\varphi_k}(y_k) \|} \le \mu. \tag{15}$$

2. Пересчитываем матрицу преобразования пространства

$$B_{k+1} = B_k R_\beta(\eta_k) = B_k + (\beta - 1)(B_k \eta_k) \eta_k^{\mathrm{T}},$$

где

$$\eta_k = \frac{B_k^{\mathrm{T}} r_k}{\| B_k^{\mathrm{T}} r_k \|}, \quad r_k = g - g_f(x_k), \quad \beta = \frac{1}{\alpha} < 1.$$

3. Переходим к очередной итерации с $x_{k+1}$, $B_{k+1}$ и $g_f(x_{k+1}) = g$.

**Комментарий.** Если обе части формулы (13) умножить слева на матрицу $A_k = B_k^{-1}$, то получим $y_{k+1} = A_k x_{k+1} = A_k x_k - \rho_k g_{\varphi_k}(y_k) = y_k - \rho_k g_{\varphi_k}(y_k)$. Таким образом, формула (13) фактически реализует шаг субградиентного спуска для функции $\varphi_k(y) = f(B_k y)$, где $\rho_k$ — неотрицательный шаг на $k$-й итерации (может равняться нулю). При $\mu = 0$ величина $\rho_k^*$ — это шаг наискорейшего спус-

ка в направлении антисубградиента. Он связан с $h_k^*$ (шагом наискорейшего спуска в направлении нормированного антисубградиента) по формуле $h_k^* = \rho_k^* || B_k^{\mathrm{T}} g_f(x_k) ||$.

Введение константы $\mu \geq 0$ дает возможность рассматривать алгоритмы, в которых поиск минимума по направлению проводится приближенно и так, чтобы очередная точка находилась ближе, чем точка минимума функции по направлению. Заметим, что в *r*-алгоритмах из разд. 1 это не разрешалось. Действительно, если $\mu \geq 0$, то условия (14), (15) задают ограничение на косинус острого угла между двумя последовательными субградиентами, по разности которых реализуется растяжение пространства.

Справедлива следующая теорема.

**Теорема 3** [8]. Пусть $f(x)$ — функция вида (12), такая что $\lim\limits_{\|x\| \to \infty} f(x) = +\infty$, и последовательность $\{x_k\}_{k=0}^{\infty}$, генерируемая $r_\mu(\alpha)$-алгоритмом, удовлетворяет условию

$$\lim_{k \to \infty} ||x_{k+1} - x_k|| = 0. \tag{16}$$

Если $x^*$ — изолированная точка минимума, а точка $x_0$ такая, что выпуклое множество $\{x : f(x^*) \leq f(x) \leq f(x^0)\}$, содержащее $x_0$ и $x^*$, не содержит, кроме $x^*$, других точек $z$, в которых семейство $G_f(z)$ линейно зависимо, то последовательность $\{x_k\}_{k=0}^{\infty}$ сходится к точке $x^*$.

Проблема обоснования сходимости $r_\mu(\alpha)$-алгоритмов для всего класса выпуклых функций в настоящее время не решена. Одна из причин, по которой для негладких функций сложно провести доказательство $r_\mu(\alpha)$-алгоритма, связана с неоднозначным выбором антисубградиента для последующего направления движения из точек негладкости, где субградиенты (из множества $G_f$) линейно зависимы и ни один из антисубградиентов не является направлением убывания функции. В [11] показано, что для выпуклой кусочно-линейной функции такие точки негладкости могут являться ловушками для минимизирующей последовательности рассматриваемого варианта $r_\mu(\alpha)$-алгоритма.

В работе [12] Н.З. Шор наметил пути для обоснования модификаций $r_\mu(\alpha)$-алгоритма. Здесь рассматриваются монотонные модификации *r*-алгоритмов, для которых в качестве критерия останова выбрано необходимое и достаточное условие оптимальности для выпуклых функций: $0 \in \partial f(x)$, где $\partial f(x)$ — субдифференциал. Пусть $r_\mu(\alpha)$-алгоритм дополнен следующим правилом: если в текущей точке минимум по направлению реализуется при нулевом шаге, то для очередной итерации направление движения в преобразованном пространстве выбирается аналогично наискорейшему спуску для выпуклых функций. Модифицированный таким способом $r_\mu(\alpha)$-алгоритм для почти дифференцируемых кусочно-гладких функций всегда гарантирует выход из любой точки, в которой существует направление убывания функции, и, более того, дает возможность утверждать, что получен оптимум, если кратчайшим до выпуклой оболочки почти градиентов окажется нулевой вектор. Такая схема теоретически представляет интерес, так как вопросы сходимости при этом фактически сводятся к выяснению существования в точке направления убывания функции. Но их так же, как и $r_\mu(\alpha)$-алгоритм, можно считать «идеализированными», поскольку для реализации наискорейшего спуска с высокой точностью необходимо значительное количество вычислений значений функции и ее субградиента.

В практических вариантах *r*-алгоритмов шаг $h_k$ выбирается так, чтобы выполнялось неравенство $h_k > h_k^*$, где $h_k^*$ соответствует минимуму функции. Для

них «приближенный» поиск минимума функции направлен на уменьшение общего количества вычислений значений функции и ее субградиента и выполняется так, чтобы на одну итерацию алгоритма приходилось в среднем два-три таких вычисления. Эффективной реализацией такой стратегии является $r(\alpha)$-алгоритм с адаптивной регулировкой шага, который преодолевает «ловушки» для минимизирующей последовательности $r_\mu(\alpha)$-алгоритма.

### 3. ВАРИАНТ $r(\alpha)$-АЛГОРИТМА С АДАПТИВНОЙ РЕГУЛИРОВКОЙ ШАГА И ЕГО ПРОГРАММНЫЕ РЕАЛИЗАЦИИ

Одним из эффективных является $r(\alpha)$-алгоритм с адаптивной регулировкой шага, где $\alpha$ — постоянный коэффициент растяжения пространства, а величина шага $h_k$ настраивается в процессе выполнения одномерного спуска в направлении нормированного антисубградиента в преобразованном пространстве переменных. Настройка шага осуществляется с помощью четырех параметров: $h_0 > 0$ — величина начального шага (используется на первой итерации, а на каждой последующей эта величина уточняется); $q_1$ — коэффициент уменьшения шага ($q_1 \le 1$), если условие завершения спуска по направлению выполняется за один шаг; $q_2$ — коэффициент увеличения шага ($q_2 \ge 1$); через каждые $n_h$ шагов одномерного спуска ($n_h > 1$) величина шага увеличится в $q_2$ раз. Условие завершения спуска по направлению выполняется, как только обнаружена точка $x_{k+1}$, для которой выполняется условие

$$(x_{k+1} - x_k)^{\mathrm{T}} g_{k+1}(x_{k+1}) \ge 0. \tag{17}$$

Условие (17) легко проверяется, так как в силу положительности шага равносильно выполнению неравенства

$$(B_k B_k^{\mathrm{T}} g_f(x_k))^{\mathrm{T}} g_f(x_{k+1}) \le 0. \tag{18}$$

Оно означает, что угол между двумя последовательными субградиентами в преобразованном пространстве переменных будет неострым. Действительно, неравенство (18) можно записать как условие

$$(B_k B_k^{\mathrm{T}} g_f(x_k))^{\mathrm{T}} g_f(x_{k+1}) \le 0, \tag{19}$$

откуда

$$(g_\varphi(y_k))^{\mathrm{T}} g_\varphi(y_{k+1}) \le 0, \tag{20}$$

где $g_\varphi(y_k) = B_k^{\mathrm{T}} g_f(x_k)$ и $g_\varphi(y_{k+1}) = B_k^{\mathrm{T}} g_f(x_{k+1})$ являются субградиентами функции $\varphi(y) = f(B_k y)$ в точках $y_k = A_k x_k$ и $y_{k+1} = A_k x_{k+1}$ преобразованного пространства переменных $y = A_k x$, $A_k = B_k^{-1}$. Поскольку предполагается, что $\lim\limits_{\|x\| \to \infty} f(x) = +\infty$, то после конечного числа шагов адаптивного спуска в направлении нормированного антисубградиента обязательно выполнится условие завершения спуска (19), (20).

Итеративный процесс в $r(\alpha)$-алгоритме с адаптивной регулировкой шага продолжается до выполнения некоторого критерия останова, где ключевую роль играют параметры $\varepsilon_x$ и $\varepsilon_g$. Алгоритм останавливается в точке $x_{k+1}$, если выполнено $\| x_{k+1} - x_k \| \le \varepsilon_x$ (останов по аргументу) или условие $\| g_f(x_{k+1}) \| \le \varepsilon_g$ (останов по норме субградиента, используемый для гладких функций). Кроме них применяются еще стандартный останов, если превышено максимальное количество итераций, и аварийный останов, который сигнализирует о том, что либо функция $f(x)$ неограниченна снизу, либо начальный шаг $h_0$ слишком мал и его необходимо увеличить.

И хотя $r(\alpha)$-алгоритм с адаптивной регулировкой шага не гарантирует монотонного убывания функции, однако, как показали эксперименты, возрастание функции происходит достаточно редко. Подробные рекомендации по выбору коэффициента растяжения пространства и параметров адаптивной регулировки шага, приведенные в [13, с. 45–47], заключаются в том, что адаптивный способ регулировки шага должен увеличивать точность поиска минимума функции по направлению в процессе счета, а количество шагов по этому направлению не превышать в среднем двух-трех на одну итерацию.

Если $r(\alpha)$-алгоритм с адаптивной регулировкой шага применять для минимизации негладких функций, то рекомендуется следующий выбор параметров: $\alpha = 2 \div 4$, $h_0 = 1.0$, $q_1 = 1.0$, $q_2 = 1.1 \div 1.2$, $n_h = 2 \div 3$. Если известна оценка расстояния от начальной точки $x_0$ до точки минимума $x^*$, то начальный шаг $h_0$ целесообразно выбирать сравнимым с $||x_0 - x^*||$. При минимизации гладких функций рекомендуемые параметры аналогичные, за исключением $q_1$ ($q_1 = 0.8 \div 0.95$). Это обусловлено тем, что дополнительное дробление шага способствует увеличению точности поиска минимума функции по направлению, что при минимизации гладких функций обеспечивает более быструю скорость сходимости. При таком выборе параметров, как правило, число спусков по направлению редко превышает два, а за $n$ шагов точность по функции улучшается в три-пять раз. Параметры останова $\varepsilon_x, \varepsilon_g \sim 10^{-6} \div 10^{-5}$ при минимизации выпуклой функции, даже существенно овражной структуры, обеспечивают нахождение точки $x_k^*$ — приближения к точке $x^*$, для которого значение функции достаточно близко к оптимальному ($\frac{f(x_k^*) - f(x^*)}{|f(x^*)|+1} \sim 10^{-6} \div 10^{-5}$ — для негладких и $\frac{f(x_k^*) - f(x^*)}{|f(x^*)|+1} \sim 10^{-12} \div 10^{-10}$ — для гладких функций). Это подтверждается результатами многочисленных тестовых и реальных расчетов.

В настоящее время $r(\alpha)$-алгоритм с адаптивной регулировкой шага и ее модификациями реализован рядом компьютерных программ на языках программирования Фортран, Си, C++, C# и Octave. В их основу положены $B$-формы $r$-алгоритмов. С помощью экономной $B$-формы (5)–(7) и ее модификаций разработаны компьютерные программы **ralg** (Фортран, Си, C++), **ralgb4** (Фортран, Octave), **SolveOpt** (Фортран, Си). На основе $B$-формы метода (1)–(3) разработана компьютерная программа **ralgb5** (Фортран, Octave, C++ и C#). Для $r$-алгоритмов в $H$-форме компьютерные программы широко не применяются в силу того, что они оказались неэффективными для достаточно точного решения задач оптимизации. Их использовали, главным образом, для решения специальных задач с простыми ограничениями (линейные равенства и двусторонние границы на переменные), для которых $H$-форма $r$-алгоритмов в отличие от $B$-формы предоставляет ряд удобств при реализации метода проекции субградиента.

Одной из первых разработана фортрановская программа **ralg** (автор Н.Г. Журбенко). В 70–80 гг. прошлого века эта программа активно использовалась для оптимизации негладких функций в Институте кибернетики АН Украины и в других организациях. Так, например, пользователи программы Д.И. Соломон (г. Кишинев, Институт математики) и Е.М. Киселева (Днепропетровский университет) впоследствии защитили докторские диссертации под руководством Н.З. Шора [14, 15]. В 90-е годы на основе программы **ralg** была создана фортрановская программа **ralgb4** (автор П.И. Стецюк), в которой использовалась модификация $r$-алгоритма из [16]. С помощью программы **ralg** А.В. Кунцевич разработал комплекс программ **SolveOpt** (языки Фортран и Си), где использовал усложненную адаптивную регулировку шага и критерии останова

$|x_{k+1}^i - x_k^i| \le \delta_x |x_{k+1}^i|$ и $|f(x_{k+1}) - f(x_k)| \le \delta_f |f(x_{k+1})|$ с заданными достаточно малыми $\delta_x$ и $\delta_f$ [17]. В конце XX – начале XXI веков программа **ralg** явилась основой для разработки компьютерных программ на языках программирования Си, C++ (авторы Н.Г. Журбенко и А.П. Лиховид).

В 2007–2008 гг. разработана фортрановская программа **ralgb5** (автор П.И. Стецюк), в которой используется метод (1)–(3). Ее название связано с тем, что в основу программы положена $B$-форма $r$-алгоритма, которая требует $5n^2$ арифметических операций умножения на каждой итерации. В 2010 г. программу **ralgb5** переписали на языке Octave, ее код приведен в [18, с. 384–385]. Octave-программа **ralgb5** оказалась быстрее одноименной фортрановской программы, если решались задачи для тысячи и более переменных. Это обусловлено тем, что библиотека BLAS (Basic Linear Algebra Subprograms) для языка Octave позволяет быстрее выполнять матрично-векторные операции в $r$-алгоритмах, чем исполняемый код при оптимизирующих опциях компилятора Фортрана. В настоящее время программа **ralgb5** реализована А.П. Лиховидом на языках C++, C# и используется в программных имплементациях алгоритмов решения различного рода задач нелинейного программирования.

В 2016 г. по аналогии с программой **ralgb5** разработана Octave-программа **ralgb4** [19]. Она использует метод (5)–(7), который реализует экономную $B$-форму $r$-алгоритмов. Octave-программа **ralgb4** требует $4n^2$ арифметических операций умножения на каждой итерации. Ее вычислительные свойства оказались схожими с вычислительными свойствами программы **ralgb5**. Octave-функции **ralgb4** и **ralgb5** можно использовать как оптимизационные ядра при реализации на языке Octave алгоритмов решения задач нелинейного программирования. На их основе легко разрабатывать оптимизационные ядра на языке MATLAB для решения вычислительных задач, которые сводятся к проблемам минимизации негладких выпуклых функций или гладких выпуклых функций с овражной структурой поверхностей уровня.

Octave-функции **ralgb4** и **ralgb5** можно легко адаптировать к языкам Фортран и Си, используя библиотеку базовых подпрограмм линейной алгебры BLAS (Basic Linear Algebra Subprograms) или библиотеку математических прикладных программ IntelR Math Kernel Library (IntelR MKL), которые оптимизированы под современные вычислительные машины. Это значительно ускоряет методы для решения больших задач (с тысячью и более переменными), например, за счет использования вычислительных мощностей графического процессора, в разы превышающие вычислительные мощности классических процессоров. Так, существенное ускорение можно получить, применив для расчетов технологию CUDA на графических ускорителях. Это подтверждает гибридная реализация $r$-алгоритма [20] при которой достигается сокращение времени решения задач с 1000–8000 переменными от 14 до 18 раз.

## ЗАКЛЮЧЕНИЕ

В рамках семейства субградиентных методов с растяжением пространства в направлении разности двух последовательных субградиентов получены достаточно эффективные реализации $r$-алгоритмов. Разработанные модификации $r$-алгоритмов можно использовать при минимизации выпуклых негладких функций из различных областей приложений. Так как свойства гладкой функции с очень быстро изменяющимся градиентом схожи со свойствами негладкой функции, то $r$-алгоритмы имеют ускоренную сходимость при оптимизации овражных гладких функций. При минимизации гладких функций они оказываются конкурентноспособными с наиболее удачными реализациями методов сопряженных направлений и методов квазиньютоновского типа.

С помощью программных реализаций $r(\alpha)$-алгоритма с адаптивным шагом можно находить достаточно точные приближения к точке минимума выпуклой функции. Если коэффициент растяжения пространства выбрать таким, чтобы он хорошо согласовался с параметрами адаптивной регулировки шага в направлении нормированного антисубградиента в преобразованном пространстве переменных, то для выполнения одних и тех же критериев останова можно значительно сократить количество итераций и количество вычислений значения функции и субградиента (градиента). Это зависит от конкретного вида минимизируемой функции, степени ее овражности и масштаба переменных.

Однако теоретическое обоснование *r*-алгоритмов проведено недостаточно полно. И хотя количество итераций для нахождения минимального значения $f^*$ с точностью $\varepsilon$ для выпуклых функций от $n$ переменных эмпирически оценивается как $N = O\left( n \log \frac{1}{\varepsilon} \right)$, в настоящее время актуально суждение Н.З. Шора «... Теория всего класса алгоритмов с растяжением пространства далека от совершенства. Нам кажется достаточно реалистичной целью построение такого алгоритма, который по своей практической эффективности не уступал бы *r*-алгоритму и был столь же хорошо обоснован, как метод эллипсоидов». Исследования в данном направлении активно продолжаются.

СПИСОК ЛИТЕРАТУРЫ


1. Сергиенко И.В., Стецюк П.И. О трех научных идеях Н.З. Шора. *Кибернетика и системный анализ*. 2012. № 1. С. 4–22.
2. Шор Н.З. Методы минимизации недифференцируемых функций и их приложения: Автореф. дис. ... докт. физ-мат. наук. Киев, 1970. 44 с.
3. Шор Н.З., Журбенко Н.Г. Метод минимизации, использующий операцию растяжения пространства в направлении разности двух последовательных градиентов. *Кибернетика*. 1971. № 3. С. 51–59.
4. Шор Н.З. Методы минимизации недифференцируемых функций и их приложения. Киев: Наук. думка, 1979. 200 с.
5. Shor N.Z. Nondifferentiable optimization and polynomial problems. Boston; Dordrecht; London: Kluwer Academic Publishers. 1998. 412 p.
6. Shor N.Z., Stetsyuk P.I. Lagrangian bounds in multiextremal polynomial and discrete optimization problems. *Journal of Global Optimization*. 2002. Vol. 23, N 1. P. 1–41.
7. Шор Н.З., Журбенко Н.Г., Лиховид А.П., Стецюк П.И. Развитие алгоритмов недифференцируемой оптимизации и их приложения. *Кибернетика и системный анализ*. 2003. № 4. С. 80–94.
8. Шор Н.З. Исследование сходимости метода градиентного типа с растяжением пространства в направлении разности двух последовательных субградиентов. *Кибернетика*. 1975. № 4. С. 48–53.
9. Шор Н.З. О классе почти дифференцируемых функций и одном методе минимизации функций этого класса. *Кибернетика*. 1972. № 4. С. 65–70.
10. Optimization methods and software. *Special Issue "Nonsmooth optimization and related topics", dedicated to the memory of professor Shor N."* 2008. Vol. 23, N 1. P. 3–4. DOI: 10.1080/10556780701652368.
11. Стецюк П.И. К вопросу сходимости *r*-алгоритмов. *Кибернетика и системный анализ*. 1995. № 6. С. 173–177.
12. Шор Н.З. Монотонные модификации *r*-алгоритмов и их приложения *Кибернетика и системный анализ*. 2002. № 6. С. 74–96.
13. Шор Н.З., Стеценко С.И. Квадратичные экстремальные задачи и недифференцируемая оптимизация. Киев: Наук. думка, 1989. 208 с.

14. Шор Н.З., Соломон Д.И. Декомпозиционные методы в дробно-линейном программировании. Кишинев: Штиинца, 1989. 204 с.

15. Киселева Е.М., Шор Н.З. Непрерывные задачи оптимального разбиения множеств: теория, алгоритмы, приложения. Киев: Наук. думка, 2005. 564 с.

16. Шор Н.З., Стецюк П.И. Использование модификации *r*-алгоритма для нахождения глобального минимума полиномиальных функций. *Кибернетика и системный анализ*. 1997. № 4. С. 28–49.

17. Kappel F., Kuntsevich A.V. An implementation of Shor's *r*-algorithm. *Computational Optimization and Applications*. 2000. Vol. 15, N 2. P. 193–205.

18. Стецюк П.И. Методы эллипсоидов и *r*-алгоритмы. Кишинэу: Эврика. 2014. 488 с.

19. Стецюк П.И. Субградиентные методы ralgb5 и ralgb4 для минимизации овражных выпуклых функций. *Вычислительные технологии*. 2017. № 2. С. 127–149.

20. Стецюк П.І., Хіміч О.М., Сидорук В.О. Реалізація *r*-алгоритму на графічних процесорах. *Компьютерная математика*. Киев: Ин-т кибернетики им. В.М. Глушкова НАН Украины, 2016. № 2. С. 100–109.

*Надійшла до редакції 03.04.2017*

## П.І. Стецюк

### ТЕОРІЯ ТА ПРОГРАМНІ РЕАЛІЗАЦІЇ *r*-АЛГОРИТМІВ ШОРА

**Анотація.** Розглянуто три обчислювальні форми *r*-алгоритмів з різним обсягом обчислень на одній ітерації. Наведено результати про збіжність граничного варіанта *r*-алгоритмів для опуклих гладких функцій і $r_\mu(\alpha)$-алгоритму для опуклих кусково-гладких функцій. Обговорено практичні аспекти варіанта $r(\alpha)$-алгоритмів з постійним $\alpha$ — коефіцієнтом розтягу простору і адаптивним способом регулювання кроку в напрямку нормованого антисубградієнта в перетвореному просторі змінних.

**Ключові слова:** субградієнтний метод, найшвидший спуск, різниця субградієнтів, розтяг простору, *r*-алгоритм, метод спряжених градієнтів, адаптивний крок, програмна реалізація.

## P.I. Stetsyuk

### THEORY AND SOFTWARE IMPLEMENTATIONS OF SHOR'S *r*-ALGORITHMS

**Abstract.** Three computational forms of *r*-algorithms with different amount of computation per iteration are considered. The results on the convergence of the limit variant of *r*-algorithms for convex smooth functions and the $r_\mu(\alpha)$-algorithm for convex piecewise smooth functions are presented. Practical aspects of the variant of $r(\alpha)$-algorithms with a constant coefficient of space dilation $\alpha$ and an adaptive method for step adjustment in the direction of the normalized anti-subgradient in the transformed space of variables are discussed.

**Keywords:** subgradient method, steepest descent, difference of subgradients, space dilation, *r*-algorithm, conjugate gradient method, adaptive step, software implementation.

**Стецюк Петр Иванович,**
доктор физ.-мат. наук, заведующий отделом Института кибернетики им. В.М. Глушкова НАН Украины, Киев, e-mail: stetsyukp@gmail.com.