ТЕОРІЯ ОПТИМАЛЬНИХ РІШЕНЬ

УДК 519.8

А.П. ЛИХОВИД

ОБ ОДНОЙ РЕАЛИЗАЦИИ R-АЛГОРИТМА

Рассматривается одна реализация известного r-алгоритма с подключением к системе моделирования AMPL. Приводятся результаты сравнительных численных экспериментов на известных тестовых задачах с входящими в состав системы оптимизационными программами.

Введение. В настоящее время широкое распространение получила система моделирования оптимизационных задач AMPL [1], которая стала стандартом для описания этих задач в среде Интернет и формата входных данных для онлайновых вычислений (см. NEOS-сервер [2]). Поэтому актуальным является подключение к интерфейсу системы моделирования AMPL существующих и новых алгоритмов оптимизации. Преимущества такого подхода состоят в следующем:

к системе AMPL подключены большинство известных оптимизационных программ, что позволяет легко проводить сравнительные эксперименты;

на языке моделирования этой системы реализованы известные наборы тестовых задач, предлагаемые на различных сайтах сети Интернет, что позволяет использовать их для тестирования алгоритмов;

средства системы моделирования позволяют автоматизировать вычисления значений функций и производных.

В этой работе рассматривается реализация известного *r*-алгоритма [3] с подключением к среде AMPL, приводятся результаты сравнительных численных экспериментов на известных тестовых задачах с входящими в состав системы оптимизационными программами.

Краткое описание используемой схемы r-алгоритма. Приведем используемую в данной реализации схему r-алгоритма для минимизации выпуклой функции $f(x)$, определенной на E^n.

Выбираем начальное приближение $x_0 \in E^n$. Положим $B_0 = I_n$ – диагональная единичная матрица $n \times n$. Первый шаг алгоритма производим по формуле $x_1 = x_0 - h_0 \eta_0$, где $\eta_0 = B_0 B_0^T g_f(x_0)$, h_0 – начальное значение шагового множителя.

Пусть в результате вычислений после k $(k = 1, 2, ...)$ шагов процесса получены определенные значения $x_k \in E^n$ и матрицы B_k размером $n \times n$.

Опишем $(k+1)$ -й шаг процесса.

1. Вычисляем следующие величины: $g_f(x_k)$ – субградиент функции $f(x)$ в точке x_k; $r_k = B_k^T (g_f(x_k) - g_f(x_{k-1}))$ – вектор разности двух последовательных субградиентов в преобразованном пространстве.

2. Определяем $\xi_k = r_k / \| r_k \|$ (ξ_k – направление растяжения пространства).

3. Задаем величину β_k, обратную коэффициенту растяжения пространства.

4. Вычисляем $B_{k+1} = B_k R_{\beta_k}(\xi_k)$, где $R_{\beta_k}(\xi_k)$ – оператор растяжения пространства $R_\beta(\xi) = I_n + (\beta - 1)\xi\xi^T$, $\beta = \dfrac{1}{\alpha} < 1$.

5. Находим $\tilde{g}_k = B_{k+1}^T g_f(x_k)$ ($\tilde{g}_k$ – субградиент в преобразованном пространстве.

6. Определяем $x_{k+1} = x_k - h_k B_{k+1} \tilde{g}_k / \| \tilde{g}_k \|$.

7. Переходим к следующему шагу или завершаем работу алгоритма при выполнении условий останова: $\|x_{k+1} - x_k\| \le \varepsilon_x$ или $\|g_f(x_k)\| \le \varepsilon_g$, где ε_x, ε_g – заданные положительные числа.

В данной реализации коэффициент растяжения пространства α_k выбирался постоянным (в пределах 2–3). Шаговый множитель h_k определялся следующим адаптивным способом регулировки. Задается некоторое натуральное число m (в этой реализации 3), постоянные $q_2 > 1$ и $t_0^0 > 0$ (после k шагов эта величина будет обозначаться соответственно t_k^0). Двигаемся из точки x_k в направлении спуска η с шагом t_k^0 до тех пор, пока не будет выполнено условие завершения спуска по направлению, либо число шагов не станет равным m. Условие завершения спуска состоит в том, что в данной точке $\overline{x}$ $(g_f(\overline{x}), \eta) \le 0$. Если прошло m шагов, а условие завершения спуска не выполнено, то вместо t_k^0 запоминаем $t_k^1 = q_2 t_k^0$, где $q_2 > 1$, и продолжаем спуск в том же направлении с большим шагом. Если после очередного шага условие завершения спуска не

выполнено, то вместо t_k^1 берем $t_k^2 = qt_k^1$ и т. д. Если после заданного числа шагов (в этой реализации 60 шагов) в определенном направлении условие завершения спуска не выполнится, то прерываем процедеру спуска и останавливаем выполнение программы. Величина $t_k^{p_k} = q^{p_k} t_k^0$ ($p_k \in \{0,1,2,\ldots\}$), которая использовалась на последнем шаге, принимается в качестве начальной при спуске в новом направлении из точки x_{k+1}, т.е. $t_{k+1}^0 = t_k^{p_k}$. Если на данной итерации уже после первого шага выполнено условие завершения спуска по направлению, то шаговый множитель умножается на заданное число $q_1 < 1$ (порядка 0,8–0,95) (такой способ регулировки шага хорошо зарекомендовал себя для гладких функций).

Также применялась схема «восстановления» матрицы B_k, т.е. $B_k = I_n$ если $\left\| B_{k+1}^T g_f(x_k) \right\| \le 10^{-16}$.

Программная реализация r-алгоритма и подключение к интерфейсу системы AMPL.

Вышеописанная схема *r*-алгоритма была реализована на языке программирования C++ в виде класса. Программа, которую будем называть amplRalg, была подключена к интерфейсу системы AMPL с помощью средств, доступных на официальном сайте (http://www.ampl.com/DOWNLOADS/index.html). Она предназначена для решения гладких и негладких задач оптимизации. Для решения задач с ограничениями используется метод негладких штрафных функций [4]. Для работы программы на компьютере должна быть установлена система AMPL. Описание математической модели задачи и исходных данных производится в соответствии с требованиями языка моделирования AMPL, при этом значения функций и производных будут вычисляться автоматически системой. Управляющие параметры r-алгоритма задаются в текстовом файле с названием RALG.OPT.

Откомпилированные версии программы amplRalg для систем Windows и Linux и описание их применения можно найти на сайте http://elis.dvo.ru /sites/default/files/OptimiZone/software/ndo-icyb/. Далее представлены результаты сравнительных численных экспериментов и оценки эффективности применения программы для набора тестовых задач.

Сравнительные численные эксперименты.

Для проведения численных экспериментов были выбраны тестовые задачи, доступные на сайте http://orfe.princeton.edu/~rvdb/ampl/nlmodels/. Здесь находятся многие известные наборы тестовых задач, например из CUTEr [5], написанные на языке AMPL. Для сравнительного анализа были выбраны известная программа LOQO (Version 6.06) и SNOPT (Student SNOPT 7.2-8), подключенные к системе моделирования AMPL. LOQO – одна из реализаций

метода внутренних точек, предназначенная для решения задач нелинейного программирования [6], SNOPT – одна из реализаций квазиньютоновских методов [7]. Использовались стандартные значения управляющих параметров для этих программ. Для amplRalg значения параметров для решения тестовых задач были выбраны следующими: $\alpha = 2$ – коэффициент растяжения пространства; $\|x_{k+1} - x_k\| \le 10^{-8}$ – точность при завершении по аргументу (k – номер итерации); $h = 1$ – величина начального шага; значения параметров $q_1 = 1.2$, $q_2 = 0.95$, величина штрафного коэффициента равнялась 1000.

Характеристики компьютера, на котором проводились исследования: компьютер IBM PC Pentium; процессор CPU Pentium 750MHz; оперативная память 256MB; операционная система Windows 2000; система моделирования AMPL (student version).

Для проведения вычислительных экспериментов было взято 429 задач из тестового набора CUTEr. Максимальное число переменных – 203, ограничений – 256. Эти задачи были как выпуклые, так и невыпуклые. Для сравнения эффективности и надежности программ используется подход, основанный на построении кривых профилей производительности [8]. Профиль производительности представляет собой функцию распределения числа решенных задач и служит как для оценки эффективности, так и надежности исследуемой программы. На рисунке показаны полученные графики кривых профилей производительности, из которых видно, что наилучший результат показала программа SNOPT, а результаты работы программ LOQO и amplRalg близки.

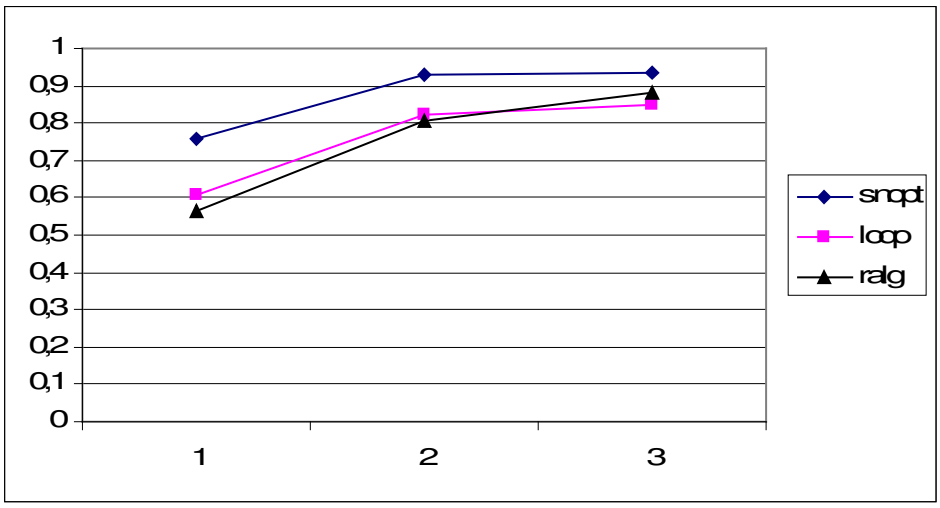

РИСУНОК. Графики профилей производительности

Заключение. Из результатов вычислительных экспериментов можно сделать вывод, что программа amplRalg, использующая метод точных штрафных функций и *r*-алгоритм, сравнима с программой LOQO по надежности и производительности на тестовом наборе из CUTEr.

О.П. Лиховид

ПРО ОДНУ РЕАЛІЗАЦІЮ R-АЛГОРИТМУ

Розглядається одна реалізація відомого r-алгоритму з підключенням до системи моделювання AMPL. Наводяться результати порівняльних чисельних експериментів на відомих тестових задачах з оптимізаційними програмами, що входять до складу системи.

O.P. Lykhovyd

ON ONE REALIZATION OF R-ALGORITHM

A realization of well-known r-algorithm with interface to AMPL modeling system is considered. The results of comparative numerical experiments on known test problems with optimization software implemented to the system are given.

1. *Fourer R., Gay D., Kernighan B.* AMPL: A Modeling Language for Mathematical Programming. – Duxbury Press-Brooks-Cole Publishing Company, –1993. – 351 p.
2. NEOS Server for Optimization. http://www.neos-server.org/neos/
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. *Shor N.Z.* Nondifferentiable Optimization and Polynomial Problems. – Boston Dordrecht. – London: Kluwer Academic Publishers, 1998. – 412 p.
5. CUTEr. A Constrained and Unconstrained Testing Environment.. http://www.cuter.rl.ac.uk/
6. *Vanderbei R.J.* LOQO: An interior point code for quadratic programming // Technical Report SOR-94-15, School of Engineering and Applied Science, Department of Civil Engineering and Operations Research, Princeton University. – 1994.
7. SNOPT 7.2 http://www.sbsi-sol-optimize.com/asp/sol_product_snopt.htm
8. *Dolan E.D., More J.J.* Benchmarking optimization software with performance profiles. Math. Programming, 91:201-214, 2002.

Получено 05.04.2011