Алгоритм для вычисления нижних оценок на матожидание числа запросов для решения задач оптимизации

Буздалов М. В., к.т.н., доцент кафедры компьютерных технологий Университета ИТМО, mbuzdalov@gmail.com[1]

Аннотация

В работе предлагается алгоритм, позволяющий вычислять нижние оценки на матожидание числа запросов для решения задачи оптимизации по матрично-векторной конфигурации задачи при заданном размере. Демонстрируется работа алгоритма на двух известных модельных задачах дискретной оптимизации OneMax и Mastermind, имеющих важное значение в теории эволюционных вычислений. Известные нижние оценки для указанных задач эмпирически улучшены.

Введение

Эволюционные алгоритмы, а также подобные им алгоритмы оптимизации, решают задачи оптимизации с помощью запросов к оптимизируемой функции, рассматриваемой как «черный ящик»: об оптимизируемой функции не делается никаких предположений, кроме, возможно, принадлежности функции определенному классу. Для понимания того, насколько хорошо алгоритмы такого вида могут решать те или иные задачи в принципе, и насколько существующие алгоритмы (например, эволюционные) далеки от идеала, изучают сложность задач оптимизации.

В теории эволюционных вычислений *сложность задачи оптимизации* (англ. black-box complexity) определяют как матожидание числа запросов (в худшем случае по всем экземплярам задачи), необходимых для нахождения оптимума данного экземпляра задачи, при использовании лучшего из возможных алгоритмов оптимизации (при этом алгоритм должен принадлежать тому или иному классу алгоритмов). Рассматривают несколько различных видов сложности, при этом каждый

[1]Работа выполнена при государственной финансовой поддержке ведущих университетов Российской Федерации (субсидия 074-U01).

из этих видов определяется тем, какой класс алгоритмов рассматривается:

- *неограниченная* сложность (англ. unrestricted black-box complexity) не накладывает никаких ограничений на алгоритмы оптимизации [5];
- *несмещенная*, или *непредвзятая* сложность (англ. unbiased black-box complexity) рассматривает алгоритмы, использующие только несмещенные (англ. unbiased) операторы порождения запросов [6];
- также рассматриваются сложности для алгоритмов с ограниченной памятью, для алгоритмов, использующих только операторы сравнения над значениями оптимизируемой функции, и другие варианты.

Рассматриваемые классы алгоритмов определяются, главным образом, тем, что эволюционные и другие подобные алгоритмы действительно используют несмещенные операторы (например, стандартный оператор мутации или однородный кроссовер), многие из них используют только операторы сравнения значений приспособленности, и т.д. В свою очередь, изучение сложности задач оптимизации и понимание того, почему время работы распространенных алгоритмов не достигает известных оценок, помогает разрабатывать новые, более эффективные алгоритмы [3].

Как правило, напрямую оценить сложность задачи оптимизации затруднительно, поэтому отдельно доказывают верхние и нижние оценки сложности. *Верхние оценки* доказываются путем создания алгоритма оптимизации, принадлежащего рассматриваемому классу алгоритмов, но специально разработанного для решения рассматриваемой задачи (но тем не менее способного лишь делать запросы к оптимизируемой функции), и доказательству верхней оценки на его время работы или на матожидание времени работы (под временем работы понимается число запросов, необходимых алгоритму для нахождения оптимума функции). *Нижние оценки* доказываются путем нахождения причины, по которой любой алгоритм из рассматриваемого класса неспособен найти оптимум за меньшее время, чем нижняя оценка. Чаще всего нижние оценки получаются на основании информационно-теоретических соображений.

Настоящая работа посвящена нахождению *нижних* оценок на *неограниченную* сложность задач оптимизации. При этом знание о

структуре задачи оптимизации записывается следующим образом: для размера задачи N вводится T_N типов вершин дерева решений алгоритма оптимизации, матрица A размера $T_N \times T_N$, описывающая взаимоотношения этих типов, и вектора B размера T_N, каждый компонент которого описывает максимальное число различных возможных оптимумов в поддереве вершины дерева решений соответствующего типа. В работе предлагается алгоритм, который по значениям N, A и B вычисляет нижнюю оценку на сложность задачи при размере N.

Основные идеи и описание алгоритма

Одним из основных инструментов доказательства нижних оценок служит *принцип минимакса Яо* (англ. Yao's minimax principle) [7]. Этот принцип утверждает, что матожидание времени работы вероятностного алгоритма (на наихудших входных данных) не меньше, чем время работы лучшего детерминированного алгоритма (усредненное по всем входным данным). Таким образом, для получения нижней оценки на матожидание времени работы наилучшего алгоритма достаточно получить нижнюю оценку на среднее время работы лучшего детерминированного алгоритма.

В свою очередь, для получения указанной выше нижней оценки не обязательно находить лучший детерминированный алгоритм. Любому детерминированному алгоритму можно взаимно однозначно сопоставить дерево решений, в вершинах которого находятся запросы к оптимизируемой функции, а исходящие из вершины ребра соответствуют ответам на эти запросы (то есть, значениям функции). Если ответом является значение, соответствующее оптимуму задачи, то алгоритм останавливается; при этом удобно считать, что соответствующий оптимум «записан» в соответствующем узле. Пример такого дерева приведен на рис. 1.

Если считать, что каждая точка пространства поиска взаимно однозначно соответствует одному и только одному экземпляру задачи (как это часто бывает в теоретических исследованиях), то нижнюю оценку на время работы лучшего детерминированного алгоритма можно построить, заполняя вершины дерева решений точками пространства поиска жадно (то есть, выбирая в каждый момент вершину с минимальной глубиной). Средняя глубина вершин, содержащих точки (назовем такие вершины *помеченными*), и будет нижней оценкой.

Таким образом, для получения нижней оценки нужно знать возможную форму дерева решений. В работе [1] для этой цели предлага-

лось каким-либо образом разбить вершины дерева решений на T типов, так чтобы не более A_{ij} различных ответов на запрос, находящийся в вершине типа i, вело в вершины типа j. Пример дерева с ограничениями этого вида приведен на рис. 2. В той же работе была доказана теорема, позволяющая с использованием полученной матрицы A получить нижнюю оценку на среднюю глубину помеченных вершин в дереве такого вида.

В настоящей работе предлагается ввести еще один набор ограничений — в поддереве вершины типа i не должно быть более B_i помеченных вершин. Ограничения такого вида не позволяют дереву разрастаться вширь слишком сильно, тем самым увеличивая среднюю глубину помеченных вершин и давая более сильную нижнюю оценку. Пример дерева с дополнительными ограничениями приведен на рис. 3.

Автору работы пока не удалось сформулировать и доказать аналог теоремы о нижней оценке сложности задачи из работы [1]. Вместо этого предлагается алгоритм, который вычисляет требуемые оценки, исходя из фактических значений матрицы A и вектора B. Алгоритм заполняет дерево по одной строчке, при этом считается и сохраняется для повторного использования максимальное число помеченных вершин глубины k в поддереве вершины типа i для всех i и k, для которых указанная величина не равна нулю. Исходный код алгоритма на языке *Java* доступен в репозитории GitHub по адресу `https://github.com/mbuzdalov/papers/tree/master/2016-gecco-bbcomp-algo/src/MatrixTheorem.java`.

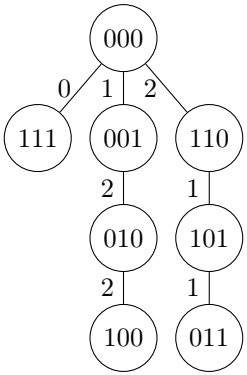

Рис. 1: Детерминированный алгоритм оптимизации, представленный в виде дерева решений. В качестве примера выбрана задача OneMax для $n = 3$. Ответы 3, соответствующие оптимуму, не показаны на рисунке.

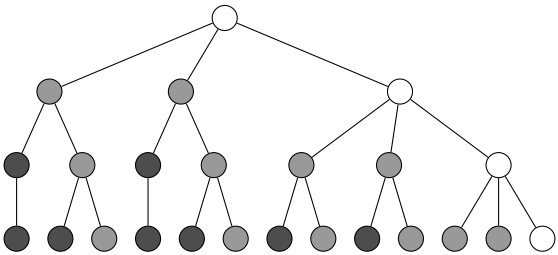

Рис. 2: Пример дерева решений с типизированными вершинами и ограничениями матричного вида. Белые вершины имеют тип 1, светло-серые вершины имеют тип 2, темно-серые вершины имеют тип 3. При этом $A_{11} = 1$, $A_{12} = 2$, $A_{22} = 1$, $A_{23} = 1$, $A_{33} = 1$, а все остальные элементы матрицы равны нулю.

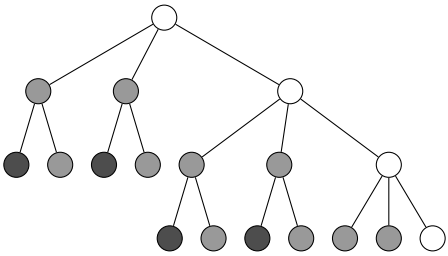

Рис. 3: Пример дерева решений с дополнительными ограничениями. В дополнении к ограничениям из рис. 2, максимальное число помеченных вершин в поддереве вершины типа 2 равно трем.

Примеры использования алгоритма

В табл. 1 приведены результаты работы алгоритма для разработанных автором матрично-векторных конфигураций, описывающих задачу OneMax. Для этой задачи имеется тривиальная нижняя оценка сложности $n/\log_2 n$ и верхняя оценка $2n/\log_2 n$. В указанной таблице показано, что с помощью *простой* конфигурации нижнюю оценку удалось ощутимо улучшить, а с помощью *сложной* конфигурации улучшение еще более существенно.

В табл. 2 приведены аналогичные оценки для задачи Mastermind [2]. Тривиальная нижняя оценка для этой задачи при размере n составляет n, наилучшая верхняя оценка составляет $O(n \log\log n)$ [4]. Из результатов таблицы видно, что сложная конфигурация, вероятнее всего, улучшает нижнюю оценку с n до $8/7n + o(n)$.

Таблица 1: Результаты для задачи OneMax

n	Нижняя	Простая конф.		Сложная конф.		Верхняя
	трив.	Знач.	Время	Знач.	Время	оценка
1000	100.999	112.927	0.000	121.174	2.031	200.687
1001	100.999	112.955	0.000	121.435	2.032	200.858
2000	183.000	201.982	0.000	215.713	23.93	364.771
2001	183.000	201.989	0.001	215.816	22.25	364.929
3000	260.000	285.833	0.002	303.867	84.01	519.447
3001	260.000	285.899	0.001	303.916	82.40	519.598
4000	335.000	365.994	0.001	388.440	291.8	668.573
4001	335.000	365.997	0.001	388.622	288.5	668.720
5000	407.000	443.999	0.002	470.649	822.1	813.821
5001	407.000	443.999	0.001	470.769	827.5	813.965

Таблица 2: Результаты для задачи Mastermind

n	Простая конф.		Сложная конф.		Отношение
	Знач.	Время	Знач.	Время	$r - 8/7$
100	100.990	0.003	121.456	0.114	+0.0717
200	200.995	0.001	238.452	0.512	+0.0494
300	300.997	0.001	354.271	2.165	+0.0380
400	400.997	0.002	469.219	5.518	+0.0302
500	500.998	0.001	583.490	12.67	+0.0241
600	600.998	0.002	697.511	26.67	+0.0197
700	700.999	0.003	811.272	46.08	+0.0161
800	800.999	0.004	924.819	79.58	+0.0132
900	900.999	0.004	1038.09	118.9	+0.0106
1000	1001.00	0.005	1151.11	183.9	+0.0083
1100	1101.00	0.006	1264.04	283.4	+0.0063
1200	1201.00	0.007	1376.72	428.2	+0.0044
1300	1301.00	0.009	1489.27	619.6	+0.0027
1400	1401.00	0.012	1601.89	890.9	+0.0014
1500	1501.00	0.012	1714.17	1233	−0.0001

Литература

[1] M. Buzdalov, M. Kever, and B. Doerr. Upper and lower bounds on unrestricted black-box complexity of $\textsc{Jump}_{n,\ell}$. In *Evolutionary Computation in Combinatorial Optimization*, number 9026 in Lecture Notes in Computer Science, pages 209–221. 2015.

[2] V. Chvátal. Mastermind. *Combinatorica*, 3(3):325–329, 1983.

[3] B. Doerr, C. Doerr, and F. Ebel. From black-box complexity to designing new genetic algorithms. *Theoretical Computer Science*, 567:87–104, 2015.

[4] B. Doerr, R. Spönel, H. Thomas, and C. Winzen. Playing Mastermind with many colors. In *Proceedings of the Twenty-Fourth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 695–704, 2013.

[5] S. Droste, T. Jansen, and I. Wegener. Upper and lower bounds for randomized search heuristics in black-box optimization. *Theory of Computing Systems*, 39(4):525–544, 2006.

[6] P. K. Lehre and C. Witt. Black-box search by unbiased variation. In *Proceedings of Genetic and Evolutionary Computation Conference*, pages 1441–1448. ACM, 2010.

[7] A. C.-C. Yao. Probabilistic computations: Toward a unified measure of complexity. In *18th Annual Symposium on Foundations of Computer Science*, pages 222–227, 1977.