ЭФФЕКТИВНЫЙ АЛГОРИТМ ВЫЧИСЛЕНИЯ БИНАРНОГО ЭПСИЛОН ИНДИКАТОРА, ОСНОВАННЫЙ НА ПОИСКЕ МИНИМУМА В ОРТАНТЕ

Васин А. Ю., магистрант кафедры компьютерных технологий
Университета ИТМО, vasinandrey2010@gmail.com
Буздалов М. В., доцент кафедры компьютерных технологий
Университета ИТМО, mbuzdalov@gmail.com

Аннотация

Бинарный ε-индикатор часто используется для оценки качества решений в многокритериальной оптимизации, а также в проведении самих оптимизаций.

Мы представляем эффективный алгоритм вычисления значения ε-индикатора, который сводит задачу к серии поисков минимума в ортанте. Для последнего мы рассматриваем две реализации: одна основана на древовидной структуре данных, другая основана на подходе «разделяй и властвуй».

Введение

Многие реальные оптимизационные задачи являются многокритериальными, то есть, они требуют максимизации или минимизации нескольких критериев, которые часто конфликтуют. В этой постановке исследователи часто хотят узнать множество Парето-оптимальных решений задачи.

В многокритериальной оптимизации ε-индикатор это функция из одного или более множеств решений в единственное число. Индикаторы в основном используются для двух целей: для оценки качества множества решений, которая особенно полезна при сравнении выходов различных оптимизаторов [1] и для самой оптимизации [2].

Формально *аддитивный бинарный ε-индикатор* это функция от двух множеств точек M и F, которая возвращает наименьшее значение ε, возможно отрицательное, которое нужно прибавить (если критерии оптимизации должны быть максимизированы или иначе вычесть) к каждой координате каждой точки из множества M, так что каждая точка множества F нестрого Парето-доминирована хотя бы одной точкой $p \in M$.

Одно из привлекательных свойств ε-индикатора заключается в том, что его определение подразумевает простой $\theta(|M| \cdot |F| \cdot k)$ алгоритм с малой константой реализации (мы обозначаем количество критериев оптимизации как k). Однако когда количество точек увеличивается (скажем, $|M|, |F| \geq 10^4$), этот алгоритм становится медленным даже для $k = 2$. Данная работа нацелена на улучшение сложившейся ситуации.

Определения

Без потери общности предположим, что мы решаем многокритериальную задачу минимизации с количеством критериев равным k. В данном случае отношение Парето-доминирования определяется на двух точках в пространстве критериев оптимизации следующим образом:

$$a \prec b \leftrightarrow \forall i \in [1; k] a_i \leq b_i \text{ и } \exists i \in [1; k] a_i < b_i$$

$$a \preccurlyeq b \leftrightarrow \forall i \in [1; k] a_i \leq b_i$$

где $a \prec b$ называется *строгим доминированием* и $a \preccurlyeq b$ — *нестрогим доминированием*.

Аддитивный бинарный ε-индикатор, или ε-индикатор для краткости, определен на двух множествах точек M и F и равен наименьшему значению ε, на которое нужно сдвинуть M в сторону оптимальности так, что каждая точка из F нестрого доминирована хотя бы одной точкой из M. В случае минимизации сдвиг точек в сторону оптимальности на ε будет равносилен вычитанию ε из каждой координаты точки. Формальное определение ε-индикатора будет иметь следующую форму:

$$\varepsilon(M, F) = \max_{f \in F} \min_{m \in M} \max_{i \in [1;k]} (m_i - f_i)$$

Мы заканчиваем данную секцию необходимыми определениями, касающимися поиска минимума в ортанте. *Ортант* это часть k-размерного пространства, которая состоит из пересечения k полупространств, где i-ое такое полупространство определено неравенством $x_i \geq b_i$ или $x_i \leq b_i$, где b_i это некоторая константа. Ортант является естественным обобщением луча в одномерном случае, квадранта в двумерном и октанта в трехмерном. В данной статье мы рассматриваем только ортанты, ориентированные в сторону положительной бесконечности, то есть, имеющие вид $[b_1; \infty) \times [b_2; \infty) \times \cdots \times [b_k; \infty)$.

Вычислительная задача *поиска в ортанте* рассматривает множество точек в k-размерном пространстве, часто с некоторыми ассоциированными значениями. В данной задаче необходимо отвечать на запросы связанные с ортантами, которые обычно ориентированы в сторону положительной или отрицательной бесконечности. Виды запросов включают в себя поиск m произвольных точек, принадлежащих ортанту, или поиск суммы, минимума или максимума значений, ассоциированных с точками, принадлежащими ортанту.

Сведение к поиску минимума в ортанте

Для эффективного вычисления ε-индикатора, заметим, что значение эпсилон для каждой точки из F может быть вычислено независимо от других точек:

$\forall f \epsilon F$ пусть $v_f = \varepsilon(M, f) = \min_{m \in M} \max_{i \in [1;k]}(m_i - f_i)$,

и затем следует найти максимум из них:

$$\varepsilon(M, F) = \max_{f \in F} v_f$$

Для вычисления $\varepsilon(M, f)$, мы можем разбить M на произвольные подмножества $M_1, \dots, M_k$ такие, что $\bigcup_{i \in [1;k]} M_i = M$, решить задачу независимо для каждого подмножества и взять минимум. Заметим, что разбиение на подмножества может зависеть от f произвольно. Мы определяем M_j как множество точек m такое, что $\max_{i \in [1;k]}(m_i - f_i) = m_j - f_j$. Тогда определение $\varepsilon(M, f)$ может быть переписано как:

$$\begin{aligned}
\varepsilon(M, f) &= \min_{m \in M} \max_{i \in [1;k]}(m_i - f_i) \\
&= \min_{j \in [1;k]} \min_{m \in M_j} \max_{i \in [1;k]}(m_i - f_i) \\
&= \min_{j \in [1;k]} \min_{m \in M_j}(m_j - f_j) \\
&= \min_{j \in [1;k]} \left((\min_{m \in M_j} m_j) - f_j \right)
\end{aligned}$$

Следующим шагом будет являться осознание того, что собой представляют множества M_j. Для выбранной точки f и любой точки $m \in M_j$ мы можем записать следующие неравенства на основе определения M_j:

$$\forall i \neq j \; m_j - f_j \geq m_i - f_i,$$

что эквивалентно:

$$\forall i \neq j \; m_j - m_i \geq f_j - f_i. \tag{1}$$

Определим проекцию $P_j: \boldsymbol{R}^k \to \boldsymbol{R}^{k-1}$ следующим образом:

$$P_j(x_1, \ldots, x_k) = (x_j - x_1, \ldots, x_j - x_k)$$

где равная нулю координата $x_j - x_j$ не включена в получившуюся точку.

Используя проекцию P_j, мы можем переписать (1) как:

$$\forall i \neq j \; m_j - m_i \geq f_j - f_i \leftrightarrow P_j(f) \preccurlyeq P_j(m),$$

в итоге получаем:

$$\varepsilon(M, f) = \min_{j \in [1;k]} \left(\left(\min_{\substack{m \in M \\ P_j(f) \preccurlyeq P_j(m)}} m_j \right) - f_j \right).$$

Внутренний минимум ничто иное, как результат поиска минимума в ортанте для множества точек $\{P_j(m) : m \in M\}$, где значение m_j связано с точкой $P_j(m)$. Так как проекция P_j строится одинаково для всех точек $f \in F$, мы можем построить одно множество проекций точек для каждого j и выполнить поиск для всех f.

Описанный алгоритм может работать с любым алгоритмом для поиска минимума в ортанте. Ключевым фактом (который имеет положительное влияние на производительность) является то, что все точки и все запросы известны заранее, так что может быть использован эффективный офлайн алгоритм.

Алгоритм на основе динамических деревьев

В [3, Теореме 3.3] приведена структура данных для d-размерного поиска минимума в ортанте, которая поддерживает только операцию активации для модификации структуры, и она может быть реализована с использованием $O(n \log^{d-1} n)$ времени и памяти на предподготовку, $O(n \log^{\mathrm{d}-1} n \log\log n)$ суммарного времени активации и $O(n \log^{\mathrm{d}-1} n \log\log n)$ времени на единичный запрос. Такая структура данных использует деревья ван Эмде Боаса [4] для достижения $O(\log\log n)$ времени работы для наименьших измерений. Однако эти деревья или имеют огромный дополнительный расход памяти, или требуют сжатия хранимых значений в небольшие диапазоны целых чисел.

Хотя мы находим упомянутые выше проблемы решаемыми, мы выбрали более простую структуру данных дерево Фенвика [5]. Сложность необходимых операций в нем составляет $O(\log n)$, и оно весьма оптимально во времени работы и занимаемой памяти на современных компьютерах. Итоговая структура данных требует $O(n \log^{\mathrm{d}} n)$ времени на предподготовку и активацию, $O(n \log^{\mathrm{d}-1} n)$ памяти и $O(\log^{\mathrm{d}} n)$ времени на единичный запрос.

Для вычисления значения ε-индикатора для k-размерных точек, мы производим k различных *(k-1)*-размерных офлайн поисков минимума в ортанте. Общее время работы $O(k \cdot (nk + n\log n + n\log^{k-2} n))$.

Алгоритм «разделяй и властвуй»

Чтобы понять идею работы подхода «разделяй и властвуй» для данной задачи, рассмотрим рис. 1. В данном двумерном рисунке белые точки соответствуют запросам, а черные точки соответствуют точкам данных. Вертикальная прямая $x_2 = s_2$ разбивает множество белых точек Q на два множества Q_L и Q_R, и множество черных точек P на два множества P_L и P_R таким образом, что:

— $\forall q \in Q_L \ q_2 < s_2$; $\forall q \in Q_R \ q_2 > s_2$;

— $\forall p \in P_L \ p_2 < s_2$; $\forall p \in P_R \ p_2 > s_2$;

— $|Q_L| + |P_L| = |Q_R| + |P_R|$.

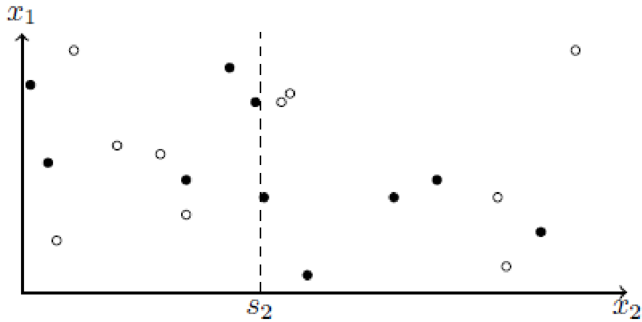

Рисунок 1: Иллюстрация подхода «разделяй и властвуй» для поиска минимума в ортанте

Заметно, что для любой точки запроса $q \in Q_R$, можно рассматривать только точки $p \in P_R$, так как для каждой $p \in P_L$ верно, что $p_2 < q_2$ и p не покрывается запросом q. Так же для каждого запроса $q \in Q_L$ и каждой точки $p \in P_R$ верно, что $q_2 < p_2$, так что алгоритм может больше не проверять данное условие.

Если мы вызовем нашу процедуру $Algo(P, Q, d)$, где d текущее измерение, то она будет состоять после поиска медианы (значение s_2 в примере выше) из проведения разбиений и трех рекурсивных вызовов, а именно, $Algo(P_L, Q_L, d)$, $Algo(P_R, Q_R, d)$, $Algo(P_R, Q_L, d-1)$. Если записать время работы как $T(Algo(P, Q, d)) = T_d(|P| + |Q|)$, мы можем оценить его следующим образом:

$$T_d(n) \leq O(n) + 2T_d(n/2) + T_{d-1}(n/2),$$

что при известном $T_{d-1}(n) = O(n\log^{\mathrm{d}-2} n)$ дает нам $T_d(n) = O(n\log^{\mathrm{d}-1} n)$.

Эмпирическая оценка

Мы эмпирически оценили предложенные алгоритмы с поисками минимума в ортанте, а также наивного алгоритма для вычисления ε-индикатора. Мы использовали размерности в интервале $[2;\ 6]$. Также мы использовали сдвигаемое и фиксированное множества одинакового размера. Эти размеры были выбраны из набора $\{100, 310, 1000, 3100, 10000, 31000\}$.

Из анализа результатов мы заметили, что предложенный алгоритм с обеими версиями алгоритма поиска минимума в ортанте имеет лучшую асимптотику по сравнению с наивным алгоритмом.

Для размерностей 2 и 3 предложенный алгоритм превосходит наивный алгоритм для всех рассматриваемых размеров задачи, хотя для $n = 100$ и $k = 3$ времена работы почти совпадают. Для этих размерностей не наблюдается заметной разницы между алгоритмами поиска минимума в ортанте.

Для больших размерностей можно заметить два явления. Во-первых, чем выше размерность, тем хуже поведение реализации предложенного алгоритма с использованием динамического дерева по сравнению с реализацией на основе подхода «разделяй и властвуй». Во-вторых, предложенный алгоритм (здесь мы рассматриваем реализацию на основе подхода «разделяй и властвуй») начинает превосходить наивный алгоритм лишь с какого-то конкретного размера задачи: с приблизительно 350 для $k = 4$, с 1000 для $k = 5$, и с 3000 для $k = 6$.

Заключение

Мы представили алгоритм для эффективного вычисления аддитивного бинарного ε-индикатора. Данный алгоритм основан на сведении этой проблемы к серии поисков минимума в ортанте. Для проведения последнего мы использовали два подхода: один из них основан на классической древовидной структуре данных для ортанта и запросов на интервале, второй основан на многомерном подходе “разделяй и властвуй”.

Литература

1. Performance assessment of multiobjective optimizers: An analysis and review / E. Zitzler, L. Thiele, M. Laumanns, C. M. Fonseca, V. Grunert da

Fonseca. // IEEE Transactions on Evolutionary Computation — 2003 — C. 117–132.
2. Indicator-based selection in multiobjective search / E. Zitzler, S. Künzli // Parallel Problem Solving from Nature PPSN VIII number 3242 in Lecture Notes in Computer Science — 2004 — C. 832–842.
3. Scaling and related techniques for geometry problems / H. N. Gabow, J. L. Bentley, R. E. Tarjan // Proceedings of the sixteenth annual ACM symposium on Theory of computing — 1984 — C. 135–143.
4. Design and implementation of an efficient priority queue / P. Van Emde Boas, R. Kaas, E. Zijlstra // Mathematical Systems Theory — 1976 — C. 99–127.
5. A new data structure for cumulative frequency tables / P. M. Fenwick // Software: Practice and Experience — 1994 — C. 327–336.