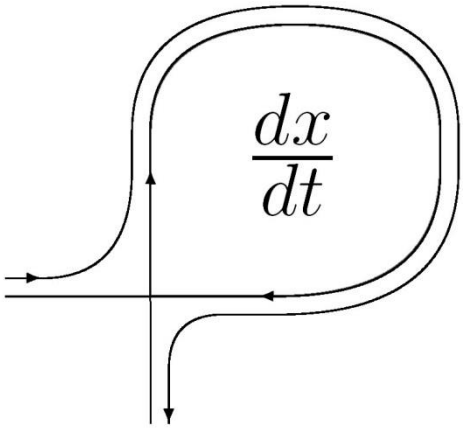

ДИФФЕРЕНЦИАЛЬНЫЕ УРАВНЕНИЯ И ПРОЦЕССЫ УПРАВЛЕНИЯ
N. 1, 2024
Электронный журнал,
рег. Эл № ФС77-39410 от 15.04.2010
ISSN 1817-2172

http://diffjournal.spbu.ru/
e-mail: jodiff@mail.ru

Информационные системы и процессы

Обобщенный алгоритм суммирования перечислителей в задачах дискретной оптимизации в контексте управления мастер-данными

Кузнецов С.В. [1,2*], Кознов Д.В. [2**]

[1]ООО «Юнидата»
[2]Санкт-Петербургский государственный университет

[*] sergey.kouznetsov@gmail.com
[**] d.koznov@spbu.ru

Аннотация. В данной работе выполнена ревизия обобщенного алгоритма для суммирования произвольного числа перечислителей субоптимальных решений задач дискретной оптимизации. Исходные идеи этого алгоритма были предложены российским математиком И. В. Романовским. Однако к настоящему времени открылись перспективы применения концепции перечислителей для задач управления данными и машинного обучения. При этом важной оказалась именно операция суммирования нескольких перечислителей, что позволяет выполнять декомпозицию мультидоменных алгоритмов управления мастер-данными. Также в работе представлены сложностные оценки для различных вариантов алгоритма суммирования произвольного числа перечислителей.

Ключевые слова: субоптимальные задачи дискретной оптимизации, перечислители, граница Парето, управление мастер-данными, мультидоменные алгоритмы

1. Введение

В различных прикладных областях возникают задачи дискретной оптимизации, где требуется найти не только оптимальное решение, но и следующие k лучших решений, то есть решить субоптимальную задачу [1,2]. Отдельно стоит выделить активно развивающиеся отрасли биоинформатики, информационных технологий, и искусственного интеллекта, где приходится

оперировать большими объемами данных из различных источников [3,4]. Для таких задач часто используют прием масштабирования субоптимальных задач, который понижает сложность алгоритмов, позволяя составлять искомые решения из решений более простых задач (меньших по количеству допустимых решений, числу параметров оптимизации, объёму обрабатываемых данных и т.д.).

В [1,5] введено понятие перечислителей, операций с ними, а также показано как они могут использоваться для масштабирования. Одной из ключевых операций при этом является суммирование перечислителей, для которой в [4] представлен алгоритм для случая двух перечислителей. В прикладных задачах, использующих большой набор анализируемых данных, операция суммирования перечислителей позволяет объединять решения задач на разных доменах данных в искомое субоптимальное решение исходной задачи. Однако для этого требуется эффективная реализация суммирования большого числа (десятки и сотни) перечислителей доменов. Анализ подходов и поиск эффективной реализации операции суммирования перечислителей для большого числа слагаемых является целью данной работы.

Статья структурирована следующим образом – рассмотрены алгоритмы попарного суммирования произвольного числа перечислителей, приведены оценки их алгоритмической сложности и потребления памяти (раздел 2). Далее предложен обобщенный алгоритм перебора решений субоптимальной задачи суммы n при $n > 2$ перечислителей на основе многомерной границы Парето, а также дана оценка сложности и потребления памяти (раздел 3). В заключении приведено сравнение представленных алгоритмов для разного количества слагаемых и значений субоптимальности.

2. Суммирование перечислителей субоптимальных задач

В данном разделе представлено два алгоритма суммирования перечислителей решений субоптимальных задач дискретной оптимизации для $n > 2$ на основе попарного сложения и изложена мотивация для поиска более эффективного алгоритма.

Пусть имеется $n > 2$ и набор задач дискретной оптимизации $Dopt_j$ в определениях [4], где $j \in 1:n$. Тогда для любого такого j и соответствующего $Dopt_j$ обозначим через k_j значение субоптимальности задачи $Dopt_j$. Для задачи $DoptSum$, полученной суммированием задач $Dopt_j$, значение субоптимальности определим как $k = \prod_{j=1..n} k_j$.

При суммировании n задач оказывается важным порядок суммирования задач $Dopt_j$ – от этого порядка зависит сложность алгоритма вычисления итогового перечислителя для $DoptSum$.

Алгоритм 1. Попарное суммирование. Рассмотрим следующий порядок сложения n субоптимальных задач дискретной оптимизации.

$DoptSum = \sum_{j=1}^{n} Dopt_i = (\dots(Dopt_1 + Dopt_2) + Dopt_3) + \cdots + Dopt_n) \dots)$.

Пусть соответствующим образом складываются и перечислители этих задач.

Замечание. Если решения субоптимальной задачи $Dopt_1$, в большинстве своем, меньше, чем $Dopt_2$, то средняя сложность алгоритма перебора решений задачи $Dopt_1 + Dopt_2$ из [4] будет ниже, чем задачи $Dopt_2 + Dopt_1$. При этом асимптотическая сложность останется той же. Это замечание вытекает из того, что в этом случае условие строки 13 алгоритма суммирования (см. листинг 1 в [4]) будет выполняться в большинстве случаев, и, соответственно, алгоритм не будет выполнять шаги после 14-й строки, а значит раньше заканчивать свою работу. Однако асимптотическая сложность останется той же, поскольку можно построить такой «плохой сценарий», который обеспечит выполнение всех шагов алгоритма.

Воспользуемся этим замечанием для модификации Алгоритма 1.

Алгоритм 2. Модифицированное попарное суммирование. Рассмотрим следующий порядок сложения n субоптимальных задач дискретной оптимизации.

1. Отсортируем $Dopt_j$ в порядке возрастания начальных решений ${d_j}^0$, т.е. чтобы выполнилось ${d_{j_1}}^0 \leq .. \leq {d_{j_n}}^0$.
2. Положим: $DoptSum = \sum_{j=1}^{n} Dopt_j = \left(\ldots\left(Dopt_{j_1} + Dopt_{j_n}\right) + Dopt_{j_2}\right) + \cdots + Dopt_{j_{n-1}}) \ldots)$.
   Пусть соответствующим образом складываются и перечислители этих задач.

Заметим, что обычно нет информации о всех решениях $Dopt_1$ и $Dopt_2$, но мы располагаем решениями ${d_1}^0$ и ${d_2}^0$. Из проведенных нами экспериментов на различных задачах и данных вытекает, что обычно ${d_1}^0 \leq {d_2}^0$, и, соответственно, решения задачи $Dopt_1$, в большинстве своем, будут меньше, чем решения $Dopt_2 1$. Это наблюдение, а также замечание, приведённое после Алгоритма 1, влечёт, что Алгоритма 2 имеет меньшую среднюю сложность по сравнению с Алгоритмом 1.

Возможны и другие модификации попарного суммирования. Однако все они будут обладать существенным недостатком: для каждого попарного суммирования перечислителей запоминается контекст его выполнения – все вспомогательные структуры данных, счетчики и вычисленные решения задач-слагаемых, и итоговый алгоритм потребляет значительный объём памяти. Сформулируем это утверждение более строго .

Утверждение 1. Асимптотическая сложность любого алгоритма попарного сложения n перечислителей составляет $O(n * k)$, где k является значением субоптимальности итоговой задачи $DoptSum$. Асимптотическая сложность потребления памяти такого алгоритма составляет $O\ (n * k + 2^{n-1} * |Context|)$, где все вспомогательные данные итогового алгоритма обозначены как $Context$, а размер занимаемой ими памяти – $|Context|$.

В качестве некоторого экспериментального подтверждения Утверждения 1 заметим, что при испытаниях на большом количестве слагаемых ($n > 15$) Алгоритм 1 и Алгоритм 2 некорректно завершались от переполнения памяти. Отметим, что для прикладных задач число перечислителей измеряется десятками и сотнями [4,6], поэтому целесообразно построить алгоритм, потребляющий память более эффективно.

3. Обобщенный алгоритм суммирования перечислителей

Важно, что не требуется сразу вычислять все субоптимальные решения для слагаемых $Dopt_j$, но можно построить «ленивый» процесс перебора (перечислитель), который вычисляет только «следующее» субоптимальное решение нужного слагаемого, что уменьшит размер $Context$. Это важное замечание для случаев небольших значений субпотимальности k задачи $DoptSum$.

Это наблюдение, а также анализ вспомогательных структур алгоритма из [4] приводят к его обобщению в русле [5]: хранение $Context$ можно ограничить только индексами решений «вычисляемых для сравнения», то есть хранения обновляемой границы Парето для многомерного множества всех возможных комбинаций индексов решений слагаемых. Такой подход потребляет существенно меньше памяти, так как не хранит весь контекст, а лишь вспомогательные структуры для обсчета и обхода одной многомерной границы Парето. Представим этот алгоритм.

[1] Такое предположение обычно подтверждается на практике, если субоптимальные решения не имеют «разрывов».

Введём ряд обозначений. Для списка задач $Dopt_j$, где $j \in 1:n$ рассмотрим прямое произведение $R = L_1 \times \ldots \times L_n$ отрезков натурального ряда $L_j = 1:k_j$. Положим $B^0 = \{(1,\ldots,1)\} \subset R$. Функцию уровней $G^0: L_2 \times \ldots \times L_n \longrightarrow L_1$ определим так:

$$\text{Для } x = (x_2,\ldots,x_n)\ G^0(x) = \begin{cases} 1, \text{если } \forall\, j \in 2:n \text{ имеем } x_j > 1, \\ k_1, \text{если } \exists\, j \in 2:n \text{ такой, что } x_j = 1. \end{cases} \quad (1)$$

Аналогично [4], i-е решение задачи $Dopt_j$ будем обозначать $d_j{}^i = T_j^i(d_j{}^0)$, при этом уже вычисленные решения задачи $Dopt_j$ сохраняются и потом не вычисляются повторно, поэтому к ним возможен прямой доступ по индексу $\{v_j{}^i, d_j{}^i\} = T_j.get(i)$, где $v_j{}^i$ – это значение целевой функции i-го решения задачи $Dopt_j$.

Ниже представлен Алгоритм 3, который для $m \in 1{:}k$ принимает на вход $T_1,..,T_n$, G^{m-1}, B^{m-1} и выдаёт $\overline{d^m}$, G^m, B^m, то есть решение задачи $DoptSum$, следующее по оптимальности. Отметим, что функции G^m являются обобщением аналогичной функции из [4,5], и их удобно задавать и хранить в виде массива.

Алгоритм 3. Обобщенный алгоритм суммирования перечислителей.

Вход: $T_1, \ldots T_n, G^{m-1}, B^{m-1}$
$G^m = G^{m-1}, B^m = B^{m-1}$
  for each b $\in B^m$
    for $j \in 1$: n
      $\{v_j, d_j\} = T_j.$get(b[j]) /* j -й элемент b ∈ R
      v += v_j
    if $v^m > v$ **then**
      $v^m = v$
      $\overline{d^m} = (d_1,..,d_n)$
      s^m = b
  x = strip (s^m) /* s^m без первого элемента, с индексами 2 : n
  $B^m = B^m \setminus \{s^m\}$
  $y = G^m(x)$
  $G^m(x) = G^{m-1}(x) + 1$
  for $j \in 2$: n
    $x' = x, x'_j = x'_j + 1$
    if $(G^m(x'), x') \notin B^m$ **then** $B^m \cup= \{(G^m(x'), x')\}$
  for l ∈ 2 : n
    $x'' = x, x''_l = x_l - 1$
    if ($G^m(x'') > y + 1$) || ($x_l = 1$) **then**
      $B^m \cup= \{y + 1, x)\}$
Выход: $\overline{d^m}$, G^m, B^m

Докажем, что представленный алгоритм реализует $TSum$, т.е. $\overline{d^m}$ является m-тым субоптимальным решением d^m задачи $DoptSum$. Для этого обобщим Лемму 3 из [4], следуя тем же обозначениям: $S^0 = R, S^m = S^{m-1} \setminus \{s^m\}$.

Лемма. Множество B^m, выдаваемое алгоритмом 3, является границей Парето для множества S^m при m: $0 \le m \le \prod_{j=1}^{n} k_j$.

Доказательство. Для начала покажем, что алгоритм 3 строит множество $B^m \subseteq S^m$, которое можно задать следующим образом:

$$B^m = \{(y, x_2, \dots, x_n) \mid G^m(x_2, .., x_j', .., x_n) > G^m(x), \text{ где}$$
$$x_j' = x_j - 1 \mid \forall j \in 2 : n,\ y = G^m(x)\}. \quad (2)$$

Доказательство (2) проведем индукцией по m. Для B^0 (2) очевидно выполнено. Согласно предположению индукции будем полагать, что оно выполнено для B^{m-1}.

Приступая к построению множества B^m, положим его равным множеству B^{m-1}(строка 2). Далее элемент $s^m = (y, x)$ удаляется из B^m (строка 12), значение функции G^m будет отличаться от G^{m-1} только в x: G^m(x) = $G^{m-1}(x)$ +1 и становится равным $y + 1$ (строка 14).

Отметим, что в итоге в множество B^m войдут следующие элементы $\{(y', x') \mid x' = (x_2, \dots, x_j + 1, \dots, x_n) \mid \forall j \in 2 : n,\ y' = G^m(x')\}$ (строки 15-17), поскольку значение $G^m(x)$ на шаге m увеличено на единицу, а согласно предположению индукции для B^{m-1} выполнено $G^m(x') \le y < G^m(x)$, поскольку $(y, x) \in B^{m-1}$.

Поскольку элемент (y, x) удален из B^m, то для множества элементов $\{x'' = (x_2, \dots, x_j - 1, \dots, x_n) \mid x_j > 0, \forall j \in 2 : n\}$ выполнено $G^m(x'') > y + 1$ значит элемент $(y + 1, x)$ должен войти в B^m, поскольку $G^m[x] = y + 1$. Это условие и определяет вхождение элемента $(y + 1, x)$ в B^m, что реализовано в строках 18–21.

Таким образом мы показали, что B^m удовлетворяет (2).

Перейдём к доказательству утверждения леммы и проведём его индукцией по m. Для множества S^0 это утверждение очевидно, поскольку очевидно, что $B^0 \subset S^0$. При этом B^0 состоит из одного элемента $(1, .., 1)$, который доминирует все другие элементы S^0.

Предположим, утверждение выполнено для множества S^{m-1} при $m > 0$. Множество S^m получается удалением из S^{m-1} элемента $s^m = (y, x)$, который входил в B^{m-1} (строки 3, 10). При этом элемент s^m также удаляется из B^m (строка 12), а в B^m добавляются только элементы из S^{m-1} (строки 17, 21), значит, $B^m \subseteq S^m$.

Для каждого $j \in 2 : n$ рассмотрим $x'' = (x_2, \dots, x_j - 1, \dots, x_n) \mid x_j > 0$. При этом возможны два случая: $G^m(x'') = y + 1$ или $G^m(x'') > y + 1$. Случай $G^m(x'') < y + 1$ невозможен по построению G^m. При этом будем полагать, что $x_j > 1$, поскольку если $x_j = 1$, то в множество B^m на шаге m войдет элемент $(y + 1, x)$, и таким образом, очевидно, что B^m будет границей Парето для множества S^m.

Случай 1. $G^m(x'') = y + 1$ (строка 13). Это означает, что элемент $(y + 1, x)$, определяемый новым значением $G^m(x)$, не попадает в множество B^m в силу (2). Однако там содержится элемент $(y + 1, x'')$, и, тем самым, доминируются элементы множества $\{(w, u) \mid x_j - 1 \le u_j,\ y + 1 \le w\}$ и элемент $(G^m(x'), x')$, который доминирует следующее множество $\{(w, u) \mid x_j + 1 \le u_j,\ G^m(x') \le w\}$. Пересекая эти множества и принимая во внимание, что элемент (y, x) уже исключен из B^m, а множество B^{m-1} было границей Парето согласно предположению индукции, мы видим, что $B^m = \{B^{m-1} \setminus \{(y, x)\}\} \cup \{(G^m(x'), x')\}$ является границей Парето множества S^m.

Случай 2. $G^m(x'') > y + 1$. В этом случае в множество B^m войдет новый элемент $(y + 1, x)$ (строка 21). Осталось лишь проверить те элементы, которые им не доминируются, а именно $E = \{(w, u) \mid x_j \le u_j,\ y = w\}$. Помним о том, что $(G^m(x'), x') \in B^m$, а значит, элементы $\{(w, u) \mid x_j + 1 \le u_j,\ G^m(x') \le w\}$ уже им доминируются. Тогда, принимая во внимание, что (y, x) уже исключен из S^m и $G^m(x') \le y$, получаем, что множество $E \setminus \{(y, x)\}$ доминируется элементом $(G^m(x'), x')$, а он уже включен в B^m. Таким образом и в этом случае B^m является границей Парето множества S^m ■

Теорема. Пусть $DoptSum = \sum_{j=1}^{n} Dopt_j$. Тогда Алгоритм 3 задает $TSum$, т.е. $\overline{d^m} = d^m$ и $SUM(P_1, .., P_n)$ является перечислителем.

Доказательство проходит аналогично доказательству Теоремы 1 в [4] поскольку применимы такие же рассуждения о переборе элементов многомерной границы Парето (строки 3–10), а то, что B^m ей является для всего множества индексов решений, еще не возвращенных на $m-1$ предыдущих шагах (S^m) доказано в Лемме выше.

Доказательство выполнения условия (2) определения перечислителя из [4] для $TSum$ несколько сложнее. В силу выполнения условия (1) для перечислителей-слагаемых $\forall d_S = (d_1,..,d_n) \in D_S \backslash \{{d_S}^0\}$ можно указать такой индекс j, что $d_j = T_j(d'_j)$ и построить такое d'_S, что $d'_S = \{d_1,..,d_{j-1},d'_j,d_{j+1},..,d_n\}$. Таким образом, элемент d_S оказывается достижим с помощью конечного числа суперпозиций $TSum$ над значением d'_S.■

С точки зрения производительности важно, что нам удалось ввести функцию G^m, отображающую значения из n-мерного массива в отрезок, и поэтому нет необходимости тратить ресурсы на обход этого массива. Отметим важность эффективной реализации вычисления функции G^m для большого числа складываемых перечислителей. Хранение значений G^m в виде многомерного целочисленного массива неэффективно для больших n. Поэтому использована так называемая техника «ведерного хранения» реализации G^m в виде хеш-массива, в котором за счет подбора параметров (число «ведер» и хранимых в нем элементов) можно добиться константного времени поиска элемента [7]. Последнее наблюдение является ключевым для оценки сложности Алгоритма 3 для определенных сценариев.

Утверждение 2. Асимптотическая сложность Алгоритма 3 составляет $O(n*k)$. Асимптотическая сложность потребления памяти Алгоритма 3 составляет $O(\ n*k+|Context|)$.

4. Заключение

Выше представлены алгоритмы попарного суммирования произвольного числа перечислителей субоптимальных решений задач дискретной оптимизации, даны их оценки сложности и объема потребляемой памяти. Также представлен обобщенный алгоритм суммирования на основе пересчёта многомерной границы Парето, который потребляет значительно меньше памяти и эффективен при большом количестве слагаемых, а также при небольших значениях субоптимальности.

Наши эксперименты показали, что Алгоритм 3 ожидаемо быстрее выдаёт решения для больших $n > 10$. Также он эффективен для поиска небольшого числа первых решений, т.е. при малых k. Однако для поиска всех решений и произвольном значении k Алгоритм 2 оказывается более эффективным при условии, что ему предоставлен достаточный объём памяти.

Литература

[1] Романовский И. В. Субоптимальные решения. Петрозаводск: Изд-во Петрозаводского университета. 1998.

[2] Брумштейн Ю. М., Тарков Д. А., Дюдиков И. А. Анализ моделей и методов выбора оптимальных совокупностей решений для задач планирования в условиях ресурсных ограничений и рисков //Прикаспийский журнал: управление и высокие технологии. – 2013. – №. 3. – С. 169–180.

[3] Кузнецов С., Константинов А., Скворцов Н. Ценность ваших данных. Изд-во Альпина PRO, 2022.

[4] Кузнецов С. В. Суммирование перечислителей в задачах дискретной оптимизации в контексте управления мастер-данными // Дифференциальные уравнения и процессы Управления. – 2023. – №. 4. – С. 42–52.

[5] Романовский И. В., Кузнецов С. В. Обобщенный алгоритм суммирования перечислителей субоптимальных решений // Вестник Санкт-Петербургского университета. Математика. Механика. Астрономия. – 2005. – №. 2. – С. 74–87.

[6] Marler R. T., Arora J. S. Survey of multi-objective optimization methods for engineering // Structural and multidisciplinary optimization. – 2004. – Т. 26. – С. 369–395.

[7] Denardo E. V., Fox B. L. Shortest-route methods: reaching, pruning, and buckets // Operations Research. 1979. Vol. 27. P. 161–186.

Summation of the enumerators in the discrete optimization problems

Kuznetsov S. V. [1,2*], Koznov D. V. [2**]

[1] Unidata LLC
[2] Saint-Petersburg State University

*sergey.kouznetsov@gmail.com
**d.koznov@spbu.ru

Abstract. The publication presents an approach to the use of discrete optimization algorithms, in particular, the search for suboptimal solutions. The theory of enumerators, proposed by the famous Leningrad mathematician I.V. Romanovsky, and the operation of their summation, which is proposed to be used to create multi-domain suboptimal algorithms, are considered. The paper presents an efficient algorithm to sum enumerators based on the recalculation of the Pareto boundary. Motivations for using the proposed algorithm within the framework of a well–known task in the field of Master Data Management are given.

Keywords: enumerators, suboptimal problems, discrete optimization, Pareto boundary, master data management, multi-domain algorithms.