АЛГОРИТМ ВЕТВЕЙ И ГРАНИЦ ДЛЯ ЗАДАЧИ КОММИВОЯЖЕРА НЕ ЯВЛЯЕТСЯ АЛГОРИТМОМ ПРЯМОГО ТИПА

А. Н. Максименко[1]

[1] ЯрГУ им. П.Г. Демидова,
ул. Советская, 14, 150000 Ярославль, Россия
E-mail: maximenko.a.n@gmail.com

Аннотация. В настоящей работе рассматривается понятие линейного разделяющего алгоритма прямого типа, введенное В. А. Бондаренко в 1983 г. До недавнего времени считалось, что класс алгоритмов прямого типа является широким и включает в себя многие классические комбинаторные алгоритмы, в том числе, алгоритм ветвей и границ для задачи коммивояжера, предложенный J. D. C. Little, K. G. Murty, D. W. Sweeney, C. Karel в 1963 г. Мы покажем, что этот алгоритм не является алгоритмом прямого типа. Рис. 1, алг. 3, библиогр. 12.

Ключевые слова: метод ветвей и границ, задача коммивояжера, линейное разделяющее дерево, кликовое число, алгоритм прямого типа.

1. Введение

В 2015–2018 гг. было опубликовано несколько работ [1–5], основными результатами которых являются оценки кликовых чисел графов многогранников, ассоциированных с различными задачами комбинаторной оптимизации. Основной мотивацией для таких оценок является следующий тезис: “It is known that this value characterizes the time complexity in a broad class of algorithms based on linear comparisons”[1)] [5]. А именно, речь идет о классе алгоритмов прямого типа, впервые введенном в [6]. В качестве подтверждения этого тезиса в [2, 3] говорится о том, что этот класс включает алгоритмы сортировки, жадный алгоритм, динамическое программирование и метод ветвей и границ[2)]. Доказательства того, что эти алгоритмы (а также алгоритм Эдмондса для задачи о паросочетаниях) являются алгоритмами прямого типа, впервые были опубликованы в диссертации [7] (см. также монографию [8]). В 2014 г. в [9] было показано, что алгоритм Куна—Манкреса для задачи о назначениях (а вместе с ним и алгоритм Эдмондса) не принадлежит к этому

[1)] «Известно, что эта величина характеризует сложность по времени в широком классе алгоритмов, основанных на линейных сравнениях»

[2)] Но ссылки на источник с соответствующими доказательствами не приводятся.

классу. Там же был описан часто используемый на практике способ модификации алгоритмов, выводящий их из класса алгоритмов прямого типа. Ниже мы докажем, что классический алгоритм ветвей и границ для задачи коммивояжера [10, 11] тоже не принадлежит к этому классу. Тем самым будет показано, что теорема 2.6.3 из диссертации [7] (теорема 3.6.6 из многографии [8]) не может быть доказана в оригинальной постановке. Это позволяет сделать вывод о том, что класс алгоритмов прямого типа не является столь широким, как предполагалось ранее.

Текст статьи организован следующим образом. В разделе 2 приводится псевдокод классического алгоритма ветвей и границ для задачи коммивояжера. В разделе 3 вводятся основные понятия концепции алгоритмов прямого типа и два ключевых определения: алгоритма прямого типа и алгоритма «прямого типа». В разделе 4 показано, что классический алгоритм ветвей и границ для задачи коммивояжера не является алгоритмом прямого типа, а в разделе 5 — что он не является алгоритмом «прямого типа».

2. Алгоритм ветвей и границ для задачи коммивояжера

Рассмотрим полный орграф $G = (V, A)$ с множеством вершин $V = [n] = \{1, 2, \dots, n\}$ и дуг $A = \{(i, j) \,|\, i, j \in V,\ i \neq j\}$. Каждой дуге $(i, j) \in A$ поставлено в соответствие число $c_{ij} \in \mathbb{Z}$, называемое *длиной дуги*. *Длиной подмножества* $H \subseteq A$ будем называть суммарную длину входящих в него дуг: $\mathrm{len}(H) = \sum_{(i,j)\in H} c_{ij}$. Задача коммивояжера состоит в том, чтобы найти $H^* \subseteq A$, являющееся гамильтоновым контуром в G и имеющее минимальную длину $\mathrm{len}(H^*)$.

Для удобства дальнейшего обсуждения поместим числа c_{ij} в матрицу $C = (c_{ij})$. Диагональным элементам c_{ii} припишем максимально возможные длины, $c_{ii} := \infty$, чтобы исключить их влияние на работу алгоритма, и будем предполагать, что $\infty - b = \infty$ для любого числа $b \in \mathbb{Z}$. Через $\mathrm{I}(M)$ будем обозначать множество индексов строк матрицы M, а через $\mathrm{J}(M)$ обозначим множество индексов столбцов матрицы M. В начале работы алгоритма $\mathrm{I}(C) = \mathrm{J}(C) = V$. Через $M(S, T)$ обозначим подматрицу матрицы M, лежащую на пересечении строк $S \subseteq \mathrm{I}(M)$ и столбцов $T \subseteq \mathrm{J}(M)$.

Сам алгоритм подробно описан в [11, раздел 4.1.6] и [10]. Мы приводим лишь его псевдокод — алгоритм 1. Отдельно, в алгоритме 2 описан процесс редуцирования строк и столбцов матрицы, а в алгоритме 3 — способ выбора такого нулевого элемента матрицы, при замене которого на бесконечность сумма редукций матрицы максимальна.

Алгоритм 1. Метод ветвей и границ для задачи коммивояжера

Глобальные: гамильтонов контур Hopt с минимальной длиной; его длина lopt. До начала работы алгоритма $\mathsf{lopt} := \infty$.

Вход : матрица длин M; множество дуг Arcs, обязательных для включения в контур; текущая сумма всех редукций sum. В самом начале работы алгоритма $\mathsf{M} := C$, $\mathsf{Arcs} := \varnothing$, $\mathsf{sum} := 0$.

```
1  Procedure BranchBound(M, Arcs, sum)
       /* Редуцируем матрицу M                                   */
2      Reduction(M, sum)
3      if sum ⩾ lopt then
4          завершить текущий экземпляр процедуры
       /* Выбираем оптимальный нулевой элемент матрицы M         */
5      (i, j) := ChooseArc(M)
       /* Разбираем случаи, когда контур содержит дугу (i, j)    */
6      if |I| = 3 then
           /* Находим единственный гамильтонов контур            */
7          H := HamiltonCycle(Arcs ∪ {(i, j)})
8          if len(H) < lopt then
9              Hopt := H
10             lopt := len(H)
11     else
           /* Вычеркиваем i-ю строку и j-й столбец               */
12         Mnew := M(I(M) \ {i}, J(M) \ {j})
           /* Находим запрещенную дугу                           */
13         (l, k) := ForbiddenArc(Arcs,(i,j))
14         Mnew[l,k] := ∞
15         BranchBound(Mnew, Arcs ∪ {(i, j)}, sum)
       /* Разбираем случаи, когда контур не содержит дугу (i, j)
          */
16     M[i,j] := ∞
17     BranchBound(M, Arcs, sum)

18 Function HamiltonCycle(Arcs)
19     Найти гамильтонов контур, содержащий все дуги из Arcs.

20 Function ForbiddenArc(Arcs,(i, j))
21     Найти пару вершин l и k, являющихся концом и началом
        наибольшего (по включению) пути в Arcs, содержащего (i, j).
```

Алгоритм 2. Редуцирование строк и столбцов матрицы

Вход : матрица M; текущая сумма всех редукций sum.
Выход : редуцированная матрица M; измененная sum.

```
1  Procedure Reduction(M, sum)
       /* Редуцируем строки матрицы M                          */
2      for i ∈ I(M) do
3          m := ∞
           /* Находим m = m(i) = min_{j∈J(M)} M[i,j]           */
4          for j ∈ J(M) do
5              if m > M[i,j] then m := M[i,j]
6          sum := sum + m
7          for j ∈ J(M) do M[i,j] := M[i,j] − m
       /* Редуцируем столбцы матрицы M                         */
8      for j ∈ J(M) do
9          m := ∞
10         for i ∈ I(M) do
11             if m > M[i,j] then m := M[i,j]
12         sum := sum + m
13         for i ∈ I(M) do M[i,j] := M[i,j] − m
```

3. Алгоритмы прямого типа

При изложении основ теории алгоритмов прямого типа мы будем придерживаться [7] (см. также [8]).

С целью унификации изложения матрица длин дуг C далее будет называться *вектором*[3)] *входных данных* или просто *входом*. Решение задачи коммивояжера, т.е. гамильтонов контур $H \subseteq A$, будет представляться в виде 0/1-вектора $\boldsymbol{x} = (x_{ij})$, имеющего ту же размерность, что и C. Координаты этого вектора $x_{ij} = 1$, при $(i,j) \in H$, и $x_{ij} = 0$ иначе. Через X обозначаем множество всех 0/1-векторов $\boldsymbol{x}$, соответствующих гамильтоновым контурам в рассматриваемом орграфе G. Таким образом, при фиксированном входе C задача коммивояжера состоит в поиске решения $\boldsymbol{x}^* \in X$ такого, что $\langle \boldsymbol{x}^*, C\rangle \leqslant \langle \boldsymbol{x}, C\rangle \ \forall \boldsymbol{x} \in X$. Далее будем называть такое решение $\boldsymbol{x}^*$ *оптимальным относительно входа* C. Следуя [7, определение 1.1.2], совокупность всех таких оптимизационных задач, образованную фиксированным множеством допустимых решений X (в случае задачи коммивояжера, X однозначно определяется числом

[3)] Элементы матрицы всегда можно выписать в строку или столбец.

Алгоритм 3. Выбор дуги

Вход : матрица M.
Выход : дуга (i^*, j^*), при запрещении которой нижняя оценка длины гамильтонова контура максимальна.

```
1  Function ChooseArc(M)
2      w := −1
3      for i ∈ I(M) do
4          for j ∈ J(M) do
5              if M[i,j] = 0 then
6                  m := ∞
                   /* Находим m = min_t M[i,t]                    */
7                  for t ∈ J(M) \ {j} do
8                      if m > M[i,t] then m := M[i,t]
9                  k := ∞
                   /* Находим k = min_t M[t,j]                    */
10                 for t ∈ I(M) \ {i} do
11                     if k > M[t,j] then k := M[t,j]
                   /* Сравниваем m + k с текущим рекордом w       */
12                 if m + k > w then
13                     w := m + k
14                     (i*, j*) := (i, j)
```

вершин орграфа G) и всевозможными входными векторами C, будем называть *задачей* X. Два допустимых решения $\boldsymbol{x}, \boldsymbol{y} \in X$ задачи X называются *смежными*, если найдется вектор C такой, что они, и только они, являются оптимальными относительно C. Подмножество $Y \subseteq X$ называется *кликой*, если любая пара $\boldsymbol{x}, \boldsymbol{y} \in Y$ смежна.

Выпуклая оболочка $\mathrm{conv}(X)$ называется *многогранником задачи* X. Так как X в задаче коммивояжера является подмножеством вершин единичного куба, то X совпадает с множеством вершин многогранника $\mathrm{conv}(X)$. В этой терминологии два решения $\boldsymbol{x}, \boldsymbol{y} \in X$ смежны тогда и только тогда, когда смежны соответствующие вершины многогранника $\mathrm{conv}(X)$ [7]. Известно [12], что все вершины многогранника коммивояжера попарно смежны при $n < 6$, где n — число вершин орграфа G, в котором требуется найти оптимальный гамильтонов контур.

Алгоритмы прямого типа относятся к классу линейных разделяющих

алгоритмов, которые удобно представлять в виде линейных разделяющих деревьев.

Определение 1 ([7, определение 1.3.1]). *Линейным разделяющим деревом* задачи $X \subset \mathbb{Z}^m$ называется ориентированное дерево, обладающее следующими свойствами:

а) в каждый узел, за исключением одного, называемого корнем, входит ровно одна дуга; дуг, входящих в корень, нет;
б) для каждого узла либо имеется две выходящих из него дуги, либо таких дуг нет вообще; в первом случае узел называется внутренним, во втором — внешним, или листом;
в) каждому внутреннему узлу соответствует некоторый вектор $B \in \mathbb{Z}^m$;
г) каждому листу соответствует некоторый элемент из X (нескольким листьям может соответствовать один и тот же элемент множества X);
д) каждой дуге d соответствует число $\operatorname{sgn} d$, равное 1 либо -1; две дуги, выходящие из одного узла, имеют различные значения;
е) для каждой цепи $W = B_1 d_1 B_2 d_2 \dots B_k d_k \boldsymbol{x}$, соединяющей корень и лист (в обозначении цепи перечислены соответствующие ее узлам векторы B_i; дуга d_i выходит из узла B_i, $i \in [k]$), и для любого входа C из неравенств $\langle B_i, C\rangle \operatorname{sgn} d_i \geqslant 0$, $i \in [k]$, следует, что решение $\boldsymbol{x}$ является оптимальным относительно C.

Таким образом, в рамках теории линейных разделяющих алгоритмов внимание уделяется только тем операциям, где выполняется проверка условий вида $\langle B, C\rangle \geqslant 0$, где C — вектор входных данных. Так, например, в строке 5 алгоритма 2 на самом первом шаге цикла проверяется неравенство $\infty > C_{11}$; на втором шаге проверяется условие $C_{11} > C_{12}$, и т. д. А в функциях `HamiltonCycle` и `ForbiddenArc` алгоритма 1, с точки зрения линейных разделяющих алгоритмов, не происходит ничего интересного, так как не выполняются никакие сравнения с элементами вектора входных данных.

Процесс работы линейного разделяющего алгоритма для фиксированного вектора входных данных C представляет собой некоторую цепь $B_1 d_1 B_2 d_2 \dots B_m d_m \boldsymbol{x}$, соединяющую корень B_1 и некоторый лист $\boldsymbol{x}$ соответствующего линейного разделяющего дерева. Листом в нашем случае является гамильтонов контур (точнее, его характеристический вектор), являющийся оптимальным относительно C.

Пусть B — некоторый внутренний узел в линейном разделяющем дереве рассматриваемого алгоритма, а X — множество всех допустимых решений (множество меток всех листьев). Обозначим через X_B, $X_B \subseteq X$, множество меток всех листьев этого дерева, которым предшествует узел

B, а через X_B^+ и X_B^- обозначим подмножества множества X_B, соответствующие двум выходящим из B дугам. Очевидно, $X_B = X_B^+ \cup X_B^-$. Обозначим через $R_B^- = X_B^+ \setminus X_B^-$ множество меток, отбрасываемых при переходе по «отрицательной» дуге. По аналогии определим множество меток $R_B^+ = X_B^- \setminus X_B^+$, отбрасываемых при переходе по «положительной» дуге.

Определение 2 ([7, определение 1.4.2])**.** Линейное разделяющее дерево называется деревом *прямого типа*, если для любого внутреннего узла B и для любой клики $Y \subseteq X$ выполняется неравенство

$$\min\{|R_B^+ \cap Y|, |R_B^- \cap Y|\} \leqslant 1. \tag{1}$$

Непосредственно из определения следует, что высота дерева прямого типа (то есть число сравнений, используемых алгоритмом в худшем случае) для задачи X не может быть меньше, чем $\omega(X) - 1$, где $\omega(X)$ — кликовое число множества X [7, теорема 1.4.3].

Если же мы хотим доказать, что некий алгоритм не является алгоритмом прямого типа, достаточно указать клику Y, состоящую из четырех решений, и узел B такие, что $|R_B^+ \cap Y| = |R_B^- \cap Y| = 2$.

Для каждого $\boldsymbol{x} \in X$ определим *конус исходных данных*

$$K(\boldsymbol{x}) = \{C \mid \langle \boldsymbol{x}, C\rangle \leqslant \langle \boldsymbol{y}, C\rangle,\ \forall \boldsymbol{y} \in X\}.$$

Т. е. $K(\boldsymbol{x})$ состоит из всех векторов C таких, что $\boldsymbol{x}$ оптимален относительно C.

Определение 3 ([7, определение 1.4.4])**.** Линейное разделяющее дерево называется деревом *«прямого типа»*, если каждая цепь $B_1 d_1 B_2 d_2 \ldots B_k d_k \boldsymbol{x}$, соединяющая корень и лист, удовлетворяет условиям:

(*) для любого $\boldsymbol{y} \in X$, смежного с $\boldsymbol{x}$, найдется такой номер $i \in [k]$, что условия $\langle B_i, C\rangle \operatorname{sgn} d_i > 0$ и $C \in K(\boldsymbol{y})$ несовместны;

(**) для любого $i \in [k]$ из несовместности условий

$$\langle B_i, C\rangle \operatorname{sgn} d_i > 0 \qquad \text{и} \qquad C \in K(\boldsymbol{y})$$

для $\boldsymbol{y}$, смежного с $\boldsymbol{x}$, и из телесности конуса

$$K(\boldsymbol{x}) \cap \{C \mid \langle B_i, C\rangle \operatorname{sgn} d_i \leqslant 0\}$$

следует, что ветвь, начинающаяся в узле B_i с дугой $-d_i$, имеет хотя бы один лист, помеченный $\boldsymbol{x}$.

Деревья «прямого типа» с деревьями прямого типа объединяет тот факт, что их высота тоже ограничена снизу величиной $\omega(X) - 1$ [7, теорема 1.4.5].

Чтобы доказать, что алгоритм 1 не является алгоритмом «прямого типа», мы ограничимся проверкой условия (*) из этого определения.

А именно, мы укажем вполне конкретный входной вектор C^*, который однозначно определит некоторую цепь $B_1d_1B_2d_2\dots B_kd_k\boldsymbol{x}$. Далее будет выбран $\boldsymbol{y}\in X$, смежный с $\boldsymbol{x}$, для которого условия $\langle B_i,C\rangle\operatorname{sgn}d_i>0$ и $C\in K(\boldsymbol{y})$ совместны при любом $i\in[k]$. Обратим особое внимание на то, что нам нужно будет проверить совместность условий $\langle B_i,C\rangle\operatorname{sgn}d_i>0$ и $C\in K(\boldsymbol{y})$ отдельно для каждого $i\in[k]$, вне зависимости от результатов других сравнений. То есть для каждого $i\in[k]$ достаточно указать C_i такой, что $\langle B_i,C_i\rangle\operatorname{sgn}d_i>0$ и $C_i\in K(\boldsymbol{y})$.

4. Алгоритм 1 не является прямым

Рассмотрим задачу коммивояжера в полном орграфе на 5 вершинах. Множество допустимых решений X такой задачи состоит из двадцати четырех 0/1-векторов, соответствующих гамильтоновым контурам в этом орграфе. Все 24 решения попарно смежны [12].

Предположим, что элементы матрицы длин дуг $C\in\mathbb{Z}^{5\times5}$ удовлетворяют следующим условиям:

$$\begin{array}{lll} c_{12}\leqslant c_{13}, & c_{12}\leqslant c_{14}, & c_{12}\leqslant c_{15},\\ c_{21}\leqslant c_{23}, & c_{21}\leqslant c_{24}, & c_{21}\leqslant c_{25},\\ c_{31}>c_{32}, & c_{32}>c_{34}, & c_{34}>c_{35}. \end{array}\tag{2}$$

В самом начале работы рассматриваемого алгоритма выполняется процедура редуцирования этой матрицы (алгоритм 2). Мы ограничимся рассмотрением этапа редуцирования строк. В результате последовательных сравнений в первой строке выбирается наименьший элемент (в данном случае c_{12}) и вычитается из всех её элементов. Далее выбирается минимальный элемент во второй строке, им оказывается c_{21}, и минимальный элемент в третьей строке — c_{35}. После этого алгоритм переходит к проверке неравенства

$$c_{41}>c_{42}\tag{3}$$

(сравнение $\infty>c_{41}$ присутствует в алгоритме исключительно для краткости описания и не несет никакой информации). Соответствующий узел линейного разделяющего дерева алгоритма обозначим B. Ясно, что алгоритм попадает в этот узел дерева, если, и только если для входного вектора C выполняются условия (2).

Рассмотрим характеристические вектора четырех гамильтоновых контуров:

$$\boldsymbol{x} = \begin{pmatrix} & 0 & 0 & 1 & 0 \\ 0 & & 0 & 0 & 1 \\ 1 & 0 & & 0 & 0 \\ 0 & 1 & 0 & & 0 \\ 0 & 0 & 1 & 0 & \end{pmatrix}, \quad \boldsymbol{y} = \begin{pmatrix} & 0 & 0 & 0 & 1 \\ 0 & & 1 & 0 & 0 \\ 1 & 0 & & 0 & 0 \\ 0 & 1 & 0 & & 0 \\ 0 & 0 & 0 & 1 & \end{pmatrix},$$

$$\boldsymbol{z} = \begin{pmatrix} & 0 & 1 & 0 & 0 \\ 0 & & 0 & 0 & 1 \\ 0 & 1 & & 0 & 0 \\ 1 & 0 & 0 & & 0 \\ 0 & 0 & 0 & 1 & \end{pmatrix}, \quad \boldsymbol{w} = \begin{pmatrix} & 0 & 0 & 0 & 1 \\ 0 & & 0 & 1 & 0 \\ 0 & 1 & & 0 & 0 \\ 1 & 0 & 0 & & 0 \\ 0 & 0 & 1 & 0 & \end{pmatrix}.$$

Нетрудно проверить, что входные векторы

$$C_x = \begin{pmatrix} & 0 & 6 & 1 & 6 \\ 0 & & 6 & 6 & 1 \\ 3 & 2 & & 1 & 0 \\ 6 & 0 & 6 & & 6 \\ 6 & 6 & 0 & 6 & \end{pmatrix}, \quad C_y = \begin{pmatrix} & 0 & 6 & 6 & 1 \\ 0 & & 1 & 6 & 6 \\ 3 & 2 & & 1 & 0 \\ 6 & 0 & 6 & & 6 \\ 6 & 6 & 6 & 0 & \end{pmatrix},$$

$$C_z = \begin{pmatrix} & 0 & 1 & 6 & 6 \\ 0 & & 6 & 6 & 1 \\ 6 & 3 & & 1 & 0 \\ 0 & 6 & 6 & & 6 \\ 6 & 6 & 6 & 0 & \end{pmatrix}, \quad C_w = \begin{pmatrix} & 0 & 6 & 6 & 1 \\ 0 & & 6 & 1 & 6 \\ 6 & 3 & & 1 & 0 \\ 0 & 6 & 6 & & 6 \\ 6 & 6 & 0 & 6 & \end{pmatrix}$$

удовлетворяют условиям (2), а для каждого $\boldsymbol{t} \in \{\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z}, \boldsymbol{w}\}$ и для любого $\boldsymbol{s} \in X \setminus \{\boldsymbol{t}\}$ выполняется неравенство $\langle \boldsymbol{t}, C_t \rangle = 5 < \langle \boldsymbol{s}, C_t \rangle$. Следовательно, все четыре вектора входят в множество меток X_B всех листьев дерева алгоритма, которым предшествует узел B.

Покажем, что $\boldsymbol{z}$ и $\boldsymbol{w}$ входят в множество меток R_B^+, отбрасываемых при выполнении неравенства (3), а $\boldsymbol{x}$ и $\boldsymbol{y}$ входят в множество меток R_B^-, отбрасываемых при невыполнении неравенства (3).

Предположим, что для входной матрицы C выполнены условия (2) и неравенство (3). Тогда $\langle \boldsymbol{z}, C \rangle > \langle \boldsymbol{z'}, C \rangle$ для

$$\boldsymbol{z'} = \begin{pmatrix} & 0 & 1 & 0 & 0 \\ 1 & & 0 & 0 & 0 \\ 0 & 0 & & 0 & 1 \\ 0 & 1 & 0 & & 0 \\ 0 & 0 & 0 & 1 & \end{pmatrix}.$$

Аналогично, $\langle \boldsymbol{w}, C\rangle > \langle \boldsymbol{w'}, C\rangle$ для

$$\boldsymbol{w'} = \begin{pmatrix} & 0 & 0 & 0 & 1 \\ 1 & & 0 & 0 & 0 \\ 0 & 0 & & 1 & 0 \\ 0 & 1 & 0 & & 0 \\ 0 & 0 & 1 & 0 & \end{pmatrix}.$$

Таким образом, $\boldsymbol{z}, \boldsymbol{w} \in R_B^+$.

Предположим, что для C выполнены условия (2), но не выполнено неравенство (3). Тогда $\langle \boldsymbol{x}, C\rangle > \langle \boldsymbol{x'}, C\rangle$ для

$$\boldsymbol{x'} = \begin{pmatrix} & 1 & 0 & 0 & 0 \\ 0 & & 0 & 0 & 1 \\ 0 & 0 & & 1 & 0 \\ 1 & 0 & 0 & & 0 \\ 0 & 0 & 1 & 0 & \end{pmatrix},$$

и $\langle \boldsymbol{y}, C\rangle > \langle \boldsymbol{y'}, C\rangle$ для

$$\boldsymbol{y'} = \begin{pmatrix} & 1 & 0 & 0 & 0 \\ 0 & & 1 & 0 & 0 \\ 0 & 0 & & 0 & 1 \\ 1 & 0 & 0 & & 0 \\ 0 & 0 & 0 & 1 & \end{pmatrix}.$$

следовательно, $\boldsymbol{z}, \boldsymbol{w} \in R_B^+$.

Таким образом, условие (1) для данного узла B не выполнено, и алгоритм 1 не является алгоритмом прямого типа.

5. Алгоритм 1 не является «прямым»

При анализе алгоритма 1, как линейного разделяющего дерева, нам будут встречаться только неравенства следующего вида:

$$\langle B^+, C\rangle - \langle B^-, C\rangle > 0, \tag{4}$$

где $C \in \mathbb{Z}^{n^2}$ — вектор входных данных,

$$B^+, B^- \in \{0,1\}^{n^2}, \quad \langle B^+, B^-\rangle = 0 \quad \text{и} \quad \langle B^+, \mathbf{1}\rangle = \langle B^-, \mathbf{1}\rangle > 0, \tag{5}$$

$\mathbf{1}$ — вектор из единиц. Иными словами, условие (5) означает, что множества единичных координат для B^+ и B^- равномощны и не пересекаются. Для каждого такого неравенства и для некоторого допустимого решения $\boldsymbol{y} \in X \subset \{0,1\}^{n^2}$ нам нужно будет проверить, что существует $C \in K(\boldsymbol{y})$, для которого это неравенство выполнено. Такой анализ существенно упрощается, если воспользоваться следующим критерием.

Лемма 1. *Пусть* $\boldsymbol{y} \in \{0,1\}^{n^2}$ *— характеристический вектор некоторого гамильтонова контура в полном орграфе* $G = ([n], A)$*. Если выполняются условия* (5) *и* $\langle B^+, \boldsymbol{y}\rangle \leqslant 2$*, то неравенство* (4) *и условие* $C \in K(\boldsymbol{y})$ *совместны.*

Доказательство. Пусть

$$S = \{(i,j) \in [n]^2 \mid y_{ij} = 1 \text{ и } B^+_{ij} = 0\}.$$

Из условия $\langle B^+, \boldsymbol{y}\rangle \leqslant 2$ следует, что $|S| \geqslant n-2$. Положим

$$C \coloneqq \mathbf{4} - B^-$$

и, после этого, $C_{ij} \coloneqq 0$ для $(i,j) \in S$. Тогда $\langle B^+, C\rangle = \langle B^+, \mathbf{4} - B^-\rangle = \langle B^+, \mathbf{4}\rangle$ и $\langle B^-, C\rangle \leqslant \langle B^-, \mathbf{4} - B^-\rangle = \langle B^+, \mathbf{4}\rangle - \langle B^-, B^-\rangle$ (так как B^+ и B^- удовлетворяют условиям (5)). Следовательно, неравенство (4) для такого C будет выполнено.

Покажем теперь, что $\langle \boldsymbol{y}, C\rangle < \langle \boldsymbol{x}, C\rangle$ для любого $\boldsymbol{x} \in X \setminus \boldsymbol{y}$.

Очевидно, $\langle \boldsymbol{y}, C\rangle = (n - |S|)4 \leqslant 8$.

Пусть $\boldsymbol{x} \in X$. Заметим, что если $\langle \boldsymbol{y}, \boldsymbol{x}\rangle \geqslant n-2$, то $\boldsymbol{x} = \boldsymbol{y}$, так как любой гамильтонов контур в орграфе на n вершинах однозначно определяется по любым своим $n-2$ дугам. Следовательно, $\langle \boldsymbol{x}, C\rangle \geqslant 3\cdot 3 = 9$ для любого $\boldsymbol{x} \in X \setminus \boldsymbol{y}$.

В частности, условия леммы выполнены, если в B^+ не более двух единиц.

Итак, положим $n = 4$ и рассмотрим следующий вектор входных данных (вместо бесконечности будем подставлять пробел):

$$C^* \coloneqq \begin{pmatrix} & 0 & 2 & 1 \\ 2 & & 0 & 2 \\ 1 & 2 & & 0 \\ 0 & 1 & 2 & \end{pmatrix} \tag{6}$$

Ясно, что единственным оптимальным решением будет вектор

$$\boldsymbol{x} \coloneqq \begin{pmatrix} & 1 & 0 & 0 \\ 0 & & 1 & 0 \\ 0 & 0 & & 1 \\ 1 & 0 & 0 & \end{pmatrix}$$

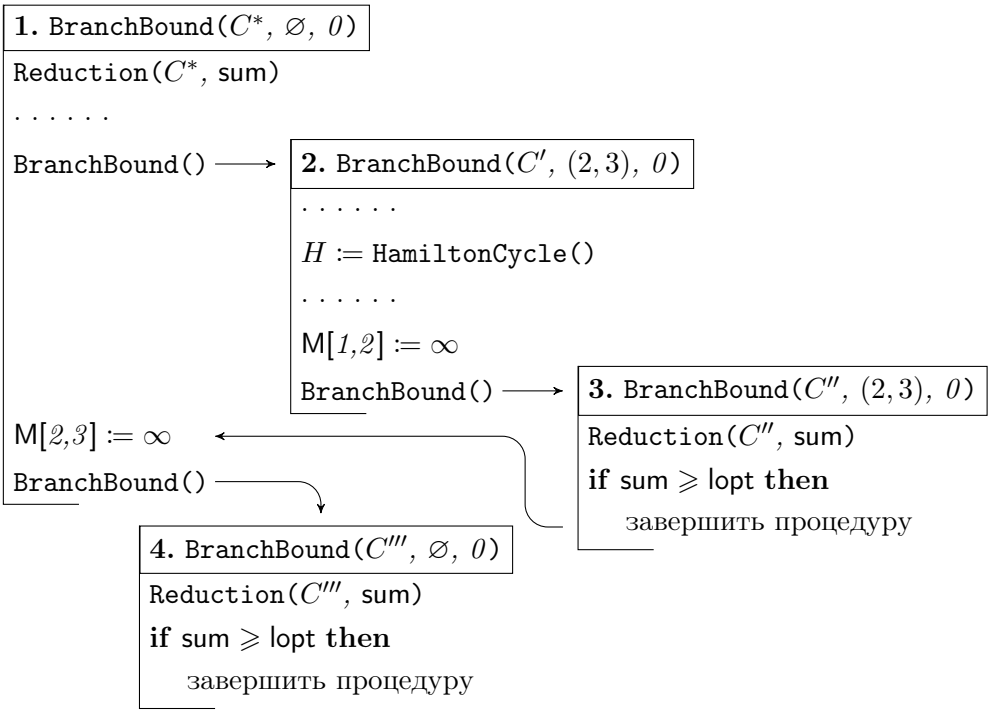

Рис. 1. Общая схема работы алгоритма 1 для входа, задаваемого формулой (6)

и соответствующий ему контур $\{(1,2),(2,3),(3,4),(4,1)\}$. Нетрудно проверяется, что множество всех допустимых решений X состоит из 6 попарно смежных векторов. Положим

$$\boldsymbol{y} \coloneqq \begin{pmatrix} & 0 & 0 & 1 \\ 0 & & 1 & 0 \\ 1 & 0 & & 0 \\ 0 & 1 & 0 & \end{pmatrix}$$

Обратим внимание, что $\boldsymbol{y}$ является вторым (после $\boldsymbol{x}$) по оптимальности относительно C^*. Именно это обстоятельство во многом упрощает дальнейшую проверку соответствующих сравнений.

В целом схема работы алгоритма при заданном входе C^* изображена на рис. 1.

Рассмотрим, прежде всего, какие неравенства проверяются при первом входе в процедуру `BranchBound` с входом C^*. При редуцировании первой строки матрицы C^* (строка 5 алгоритма 2) проверяются (и выполняются) неравенства $\infty > C_{12}$, $C_{13} > C_{12}$ и $C_{14} > C_{12}$. Далее мы не

будем рассматривать неравенства, в которых сумма (либо разность) элементов исходной матрицы сравнивается с бесконечностью, так как они всегда выполняются и совместны с любым допустимым решением. Заметим, что только что перечисленные неравенства удовлетворяют условиям леммы 1, так как $\langle B^+, \mathbf{1}\rangle = 1$. А значит, они совместны с условием $C \in K(\boldsymbol{y})$.

После редуцирования первой строки в её ячейках $\mathsf{M}[1, j]$, $j \in [4]$, содержатся разности $C_{1j} - C_{12}$, а переменная sum принимает значение C_{12}.

При редуцировании второй строки проверяются неравенства $C_{21} > C_{23}$ и $C_{24} > C_{23}$. Согласно лемме 1, они совместны с условием $C \in K(\boldsymbol{y})$.

После редуцирования второй строки в её ячейках $\mathsf{M}[2, j]$, $j \in [4]$, содержатся разности $C_{2j} - C_{23}$, а переменная sum принимает значение $C_{12} + C_{23}$.

При редуцировании последних двух строк ситуация полностью аналогична. После завершения редуцирования строк

$$\mathsf{sum} = C_{12} + C_{23} + C_{34} + C_{41},$$

$$\mathsf{M} = \begin{pmatrix} & 0 & C_{13} - C_{12} & C_{14} - C_{12} \\ C_{21} - C_{23} & & 0 & C_{24} - C_{23} \\ C_{31} - C_{34} & C_{32} - C_{34} & & 0 \\ 0 & C_{42} - C_{41} & C_{43} - C_{41} & \end{pmatrix}$$

Далее, при редуцировании первого столбца проверяются неравенства $\mathsf{M}[2, 1] > \mathsf{M}[3, 1]$ и $\mathsf{M}[3, 1] > \mathsf{M}[4, 1]$. Нам известно, что $\mathsf{M}[2, 1] = C_{21} - C_{23}$, $\mathsf{M}[3, 1] = C_{31} - C_{34}$, $\mathsf{M}[4, 1] = C_{41} - C_{41} = 0$. Следовательно, проверяются неравенства $C_{21} - C_{23} > C_{31} - C_{34}$ и $C_{31} - C_{34} > 0$. Каждое из них удовлетворяет условиям леммы 1.

При редуцировании оставшихся трех столбцов ситуация повторяется. Значение sum при редуцировании столбцов не меняется, так как каждый столбец уже содержит нули.

После этого в алгоритме 1 выполняется проверка условия $\mathsf{sum} \geqslant \mathsf{lopt}$. Но $\mathsf{lopt} = \infty$. Поэтому алгоритм переходит к вычислению функции `ChooseArc`.

Первым нулевым элементом является $\mathsf{M}[1, 2]$. После этого в строке 8 алгоритма 3 выполняются сравнения $\infty > \mathsf{M}[1, 3]$ и $\mathsf{M}[1, 3] > \mathsf{M}[1, 4]$. При этом, после предыдущего этапа редукции, имеем $\mathsf{M}[1, 3] = C_{13} - C_{12}$ и $\mathsf{M}[1, 4] = C_{14} - C_{12}$. Очевидно, неравенство $C_{13} - C_{12} > C_{14} - C_{12}$ удовлетворяет условиям леммы 1. На этом шаге выполняется присвоение $m \coloneqq C_{14} - C_{12}$. Далее, в строке 11 алгоритма 3 выполняются сравнения $\infty > \mathsf{M}[3, 2]$ и $\mathsf{M}[3, 2] > \mathsf{M}[4, 2]$. При этом $\mathsf{M}[3, 2] = C_{32} - C_{34}$ и $\mathsf{M}[4, 2] = C_{42} - C_{41}$. Условия леммы 1 снова выполнены. На этом шаге выполняется присвоение $k \coloneqq C_{42} - C_{41}$. Далее выполняется сравнение $m + k > -1$ или, что то же самое, $C_{14} - C_{12} + C_{42} - C_{41} > -1$. Очевидно, это неравенство

совместимо с условием $C \in K(\boldsymbol{y})$. В переменную w заносится значение выражения $C_{14} - C_{12} + C_{42} - C_{41}$.

Второй нулевой элемент — $\mathsf{M}[2,3]$. Действуя по аналогии, перечислим только нетривиальные сравнения. Неравенство $\mathsf{M}[2,1] \leqslant \mathsf{M}[2,4]$ или $C_{21} - C_{23} \leqslant C_{24} - C_{23}$, очевидно, совместимо с условием $C \in K(\boldsymbol{y})$. Неравенство $\mathsf{M}[1,3] \leqslant \mathsf{M}[4,3]$ тоже совместимо. Далее, в строке 12 проверяется неравенство $m + k > w$ или, с учетом предыдущих действий,

$$C_{21} - C_{23} + C_{13} - C_{12} > C_{14} - C_{12} + C_{42} - C_{41}.$$

Очевидно, оно удовлетворяет условиям леммы 1. После этого шага

$$w = C_{21} - C_{23} + C_{13} - C_{12}.$$

Третий нулевой элемент — $\mathsf{M}[3,4]$. Неравенство $\mathsf{M}[3,1] < \mathsf{M}[3,2]$ или $C_{31} - C_{34} < C_{32} - C_{34}$, очевидно, совместимо с условием $C \in K(\boldsymbol{y})$. Неравенство $\mathsf{M}[1,4] < \mathsf{M}[2,4]$ тоже совместимо. Условие $m + k < w$ имеет вид

$$C_{14} - C_{12} + C_{31} - C_{34} < C_{21} - C_{23} + C_{13} - C_{12}$$

и тоже совместимо с условием $C \in K(\boldsymbol{y})$.

Четвертый нулевой элемент — $\mathsf{M}[4,1]$. Легко проверить, что $\mathsf{M}[4,2] < \mathsf{M}[4,3]$ и $\mathsf{M}[3,1] < \mathsf{M}[2,1]$ совместимы с условием $C \in K(\boldsymbol{y})$. Условие $m + k < w$ имеет вид

$$C_{31} - C_{34} + C_{42} - C_{41} < C_{21} - C_{23} + C_{13} - C_{12}$$

и тоже совместимо.

В данный момент мы все еще находимся в первом экземпляре процедуры `BranchBound`. После описанного выше выполнения функции `ChooseArc` выбирается дуга $(i,j) = (2,3)$ (сумма $m + k$ для нее оказалась наибольшей), из матрицы M вычеркиваются 2-я строка и 3-й столбец, а дуга $(3,2)$ становится запрещенной. На вход второго экземпляра процедуры `BranchBound` подается матрица

$$C' \coloneqq \begin{pmatrix} & 0 & 1 \\ & & \\ 1 & & 0 \\ 0 & 1 & \end{pmatrix}$$

(Пустая строка и пустой столбец оставлены для удобства чтения.) Ясно, что при её редуцировании ничего нового не происходит, так как каждая строка и каждый столбец содержат нули. При вызове функции `ChooseArc` в строке 12 выполняются следующие сравнения типа $m + k > w$.

$$C_{14} - C_{12} + C_{42} - C_{41} > -1.$$

Очевидно, это неравенство совместимо с условием $C \in K(\boldsymbol{y})$. Далее, выполняется неравенство

$$C_{31} - C_{34} + C_{14} - C_{12} \leqslant C_{14} - C_{12} + C_{42} - C_{41},$$

которое удовлетворяет условиям леммы 1. Следующее сравнение

$$C_{31} - C_{34} + C_{42} - C_{41} \leqslant C_{14} - C_{12} + C_{42} - C_{41}$$

тоже совместимо с $C \in K(\boldsymbol{y})$.

Итак, после вызова функции `ChooseArc` во втором экземпляре `BranchBound`, выбирается дуга $(1, 2)$. Гамильтонов цикл с дугами $(2, 3)$ и $(1, 2)$ определяется однозначно. Выполняется присвоение

$$\mathsf{lopt} \coloneqq C_{12} + C_{23} + C_{34} + C_{41}.$$

После этого алгоритм переходит к рассмотрению случаев, когда контур содержит дугу $(2, 3)$, но не содержит $(1, 2)$. Запускается третий экземпляр `BranchBound` с матрицей

$$C'' \coloneqq \begin{pmatrix} & & 1 \\ & & \\ 1 & & 0 \\ 0 & 1 & \end{pmatrix}$$

При редуцировании две единицы заменяются нулями. Никакие «отбрасывающие» сравнения не выполняются. Значение переменной sum увеличивается на $\mathsf{M}[1, 4] = C_{14} - C_{12}$ и на $\mathsf{M}[4, 2] = C_{42} - C_{41}$. Текущий экземпляр процедуры завершается в строке 3 после проверки неравенства $\mathsf{sum} \geqslant \mathsf{lopt}$:

$$(C_{14} - C_{12}) + (C_{42} - C_{41}) > 0.$$

Заметим, что допустимое решение $\boldsymbol{y}$ полностью отбраковывается алгоритмом именно на этом шаге (с учетом ранее проверенного неравенства $C_{31} > C_{34}$). Тем не менее, это неравенство удовлетворяет условиям леммы 1 и, следовательно, совместно с условием $C \in K(\boldsymbol{y})$.

Вместе с третьим экземпляром процедуры `BranchBound` завершается и второй её экземпляр. Алгоритм переходит к выполнению предпоследней строки в первом экземпляре. В этом экземпляре

$$\mathsf{sum} = C_{12} + C_{23} + C_{34} + C_{41}.$$

Для разбора случаев, когда контур не содержит дугу $(2, 3)$ вызывается четвертый экземпляр процедуры с матрицей

$$C''' \coloneqq \begin{pmatrix} & 0 & 2 & 1 \\ 2 & & & 2 \\ 1 & 2 & & 0 \\ 0 & 1 & 2 & \end{pmatrix}$$

При редуцировании второй строки выполняется сравнение $\mathsf{M}[2,1] \leqslant \mathsf{M}[2,4]$. При редуцировании третьего столбца — $\mathsf{M}[1,3] \leqslant \mathsf{M}[4,3]$. Очевидно, ни то ни другое не отбрасывают целиком конус $K(\boldsymbol{y})$. Значение sum увеличивается на $(C_{21} - C_{23}) + (C_{13} - C_{12})$.

И, наконец, сравнение $\mathsf{sum} \geqslant \mathsf{lopt}$ завершает этот четвертый экземпляр процедуры и вообще весь алгоритм. Это сравнение имеет вид

$$(C_{21} - C_{23}) + (C_{13} - C_{12}) \geqslant 0$$

и тоже совместимо с условием $C \in K(\boldsymbol{y})$.

Итак, условие (*) из определения 3 не выполнено для этого алгоритма.

ЛИТЕРАТУРА

1. **Бондаренко В. А., Николаев А. В., Шовгенов Д. А.** Полиэдральные графы задач об остовных деревьях при дополнительных ограничениях // Моделирование и анализ информационных систем. 2015. Т. 22, № 4. С. 453–463. Англ. пер.: **Bondarenko V. A., Nikolaev A. V., Shovgenov D. A.** 1-skeletons of the spanning tree problems with additional constraints // Automatic Control and Computer Sciences. 2017. Vol. 51, No. 7. P. 682–688.
2. **Bondarenko V., Nikolaev A.** On graphs of the cone decompositions for the min-cut and max-cut problems // International Journal of Mathematics and Mathematical Sciences. 2016. 7863650.
3. **Bondarenko V., Nikolaev A.** Some properties of the skeleton of the pyramidal tours polytope // Electronic Notes in Discrete Mathematics. 2017. Vol. 61. P. 131–137.
4. **Бондаренко В. А., Николаев А. В., Шовгенов Д. А.** Полиэдральные характеристики задач о сбалансированном и несбалансированном двудольных подграфах // Моделирование и анализ информационных систем. 2017. Т. 24, № 2. С. 141–154. Англ. пер.: **Bondarenko V. A., Nikolaev A. V., Shovgenov D. A.** Polyhedral characteristics of balanced and unbalanced bipartite subgraph problems // Automatic Control and Computer Sciences. 2017. Vol. 51, No. 7. P. 576–585.
5. **Bondarenko V. A., Nikolaev A. V.** On the skeleton of the polytope of pyramidal tours // Journal of Applied and Industrial Mathematics. 2018. Vol. 12, No. 1. P. 9–18.
6. **Бондаренко В. А.** Неполиномиальная нижняя оценка сложности задачи коммивояжера в одном классе алгоритмов // Автоматика и телемеханика. 1983. Т. 9. С. 45–50. Англ. пер.: **Bondarenko V. A.** Nonpolynomial lowerbound of the traveling salesman problem complexity in one class of algorithms // Automation and Remote Control. 1983. Т. 44, № 9. С. 1137–1142.
7. **Бондаренко В. А.** Геометрические методы системного анализа в комбинаторной оптимизации // дисс. ... докт. физ.-мат. наук. Ярославль, 1993. 148 с.

8. **Бондаренко В. А., Максименко А. Н.** Геометрические конструкции и сложность в комбинаторной оптимизации. М: URSS, 2008. 182 с.
9. **Maksimenko A. N.** Характеристики сложности: кликовое число графа многогранника и число прямоугольного покрытия // Моделирование и анализ информационных систем. 2014. Т. 21, № 5. С. 116–130.
10. **Little J. D. C., Murty K. G., Sweeney D. W., Karel C.** An algorithm for the traveling salesman problem // Operations research. 1963. Vol. 11, No. 6. P. 972–989.
11. **Рейнгольд Э. М., Нивергельт Ю., Део Н.** Комбинаторные алгоритмы. Теория и практика. Пер. с англ. М: Мир, 1980. 476 с.
12. **Padberg M. W., Rao M. R.** The travelling salesman problem and a class of polyhedra of diameter two // Mathematical Programming. 1974. Vol. 7, No. 1. P. 32–45.