ISSN 0146-4116, Automatic Control and Computer Sciences, 2021, Vol. 55, No. 7, pp. 816–826. © Allerton Press, Inc., 2021.
Russian Text © The Author(s), 2020, published in Modelirovanie i Analiz Informatsionnykh Sistem, 2020, No. 1, pp. 72–85.

The Branch-and-Bound Algorithm for the Traveling Salesman Problem is Not a Direct Algorithm

A. N. Maksimenko* (ORCID: 0000-0002-0887-1500)

Demidov Yaroslavl State University, Yaroslavl, 150003 Russia

**e-mail: maximenko.a.n@gmail.com*

Received December 3, 2019; revised January 5, 2020; accepted February 28, 2020

Abstract—This article considers the concept of a linear separation direct algorithm introduced by V.A. Bondarenko in 1983. The concept of a direct algorithm is defined using the solution graph of a combinatorial optimization problem. The vertices of this graph are all feasible solutions of the problem. Two solutions are called adjacent if there are input data for which these and only these solutions are optimal. A key feature of direct algorithms is that their complexity is bounded from below by the clique number of the solution graph. In 2015–2018, there were five articles published, the main results of which are estimates of the clique numbers of polyhedron graphs associated with various combinatorial optimization problems. The thesis that the class of direct algorithms is broad and includes many classical combinatorial algorithms, including the branch-and-bound algorithm for the traveling salesman problem proposed by J.D.C. Little, K.G. Murty, D.W. Sweeney, and C. Karel in 1963, was the main motivation for these articles. We show that this algorithm is not a direct algorithm. Earlier, in 2014, the author of this article showed that the Hungarian algorithm for the assignment problem is not a direct algorithm. Therefore, the class of direct algorithms is not as broad as previously assumed.

Keywords: branch-and-bound method, traveling salesman problem, linear decision tree, clique number, direct algorithm

DOI: 10.3103/S0146411621070269

INTRODUCTION

In 2015–2018, several articles [1–5] were published, the main results of which are estimates of clique numbers of polyhedron graphs associated with various combinatorial optimization problems. The main motivation for such estimates was the following thesis [5]: "It is known that this value characterizes the time complexity in a broad class of algorithms based on linear comparisons." Namely, we are talking about the class of direct algorithms first introduced in [6]. As a proof of this thesis, works [2, 3] stated that this class includes sorting algorithms, a greedy algorithm, dynamic programming, and the branch-and-bound method[1]. The proofs that these algorithms (as well as the Edmonds' algorithm for the matching problem) are direct algorithms were first published in thesis [7] (see also [8]). In 2014, it was shown in [9] that the Kuhn–Munkres algorithm for the assignment problem (and with it the Edmonds' algorithm) does not belong to this class. There we also described a method of modifying algorithms that is often used in practice, which takes them out of the class of direct algorithms. Below, we will prove that the classical branch-and-bound algorithm for the traveling salesman problem [10, 11] also does not belong to this class. It will thus be shown that Theorem 2.6.3 from [7] (Theorem 3.6.6 from [8]) cannot be proved in the initial formulation. This leads to the conclusion that the class of direct algorithms is not as broad as previously assumed.

The article is organized as follows. Section 1 presents a pseudocode of the classical branch-and-bound algorithm for the traveling salesman problem. Section 2 introduces the basic notions of the concept of direct algorithms and two key definitions: the direct algorithm and the "direct" algorithm. Section 3 shows that the classical branch-and-bound algorithm for the traveling salesman problem is not a direct algorithm and Section 4 shows that it is not a "direct" algorithm.

[1] But no source references with corresponding proofs were given.

1. BRANCH-AND-BOUND ALGORITHM FOR THE TRAVELING SALESMAN PROBLEM

Consider a complete orgraph $G = (V, A)$ with a set of vertices $V = [n] = \{1, 2, \ldots, n\}$ and arcs $A = \{(i, j) | i, j \in V, i \neq j\}$. Each arc $(i, j) \in A$ is assigned a number $c_{ij} \in \mathbb{Z}$, called an *arc length*. The *length of a subset* $H \subseteq A$ is the total length of its arcs: $\text{len}(H) = \sum_{(i,j)\in H} c_{ij}$. The traveling salesman problem consists in finding $H^* \subseteq A$, which is a Hamiltonian circuit in G and has a minimal length of $\text{len}(H^*)$.

For ease of further discussion, let us put the numbers c_{ij} in the matrix $C = (c_{ij})$. We assign to the diagonal elements c_{ii} the maximum possible lengths, $c_{ii} := \infty$, in order to exclude their effect on the operation of the algorithm, and we assume that $\infty - b = \infty$ for any number $b \in \mathbb{Z}$. We denote by $\text{I}(M)$ the set of matrix row indices M, and by $\text{J}(M)$ we denote the set of matrix column indices M. At the beginning of the algorithm operation, $\text{I}(C) = \text{J}(C) = V$. We denote by $M(S, T)$ the submatrix of the matrix M that lies at the intersection of rows $S \subseteq \text{I}(M)$ and columns $T \subseteq \text{J}(M)$.

The algorithm itself was described in detail in [11, Section 4.1.6] and [10]. We give only its pseudocode, Algorithm 1. Separately, Algorithm 2 describes the process of reducing matrix rows and columns, and Algorithm 3 describes the way of selecting such a zero element of a matrix that, when replaced by infinity, maximizes the sum of matrix reductions.

Algorithm 1. Branch-and-bound method for the traveling salesman problem

Global: Hamiltonian circuit `Hopt` of minimal length; its length is `lopt`. Before the algorithm was launched, `lopt` := ∞.

Input : distance matrix `M`; the set of arcs `Arcs` required to be included in the circuit; the current sum of all reductions `sum`. At the very beginning of the algorithm operation, `M` := C, `Arcs` := ∅, `sum` := 0

```
1  Procedure BranchBound (M, Arcs, sum)
       /* Reduce the matrix M                                          */
2      Reduction (M, sum)
3      if sum ≥ lopt then
4          terminate the current instance of the procedure
       /* Choose the optimal zero element of the matrix M              */
5      (i, j) := ChooseArc (M)
       /* Examine cases in which the circuit contains the arc (i, j)   */
6      if |I| = 3 then
           /* Find a single Hamiltonian circuit                        */
7          H := HamiltonCycle(Arcs ∪ {(i, j)})
8          if len(H) < lopt then
9              Hopt := H
10             lopt := len(H)
11     else
           /* Remove the ith row and jth column                        */
12         Mnew := M(I(M) \ {i}, J(M) \ {j})
           /* Find the forbidden arc                                   */
13         (l, k) := ForbiddenArc (Arcs, (i,j))
14         Mnew[l,k] := ∞
15         BranchBound (Mnew, Arcs ∪ {(i, j)}, sum)
       /* Examine cases in which the circuit does not have the arc (i, j) */
16     M[i,j] := ∞
17     BranchBound (M, Arcs, sum)

18 Function HamiltonCycle (Arcs)
19     Find a Hamiltonian circuit which contains all arcs from Arcs.
```

```
20 Function ForbiddenArc (Arcs, (i, j))
21     Find a pair of vertices l and k which are the end and the beginning of the largest (by inclusion)
       path in Arcs containing (i, j).
```

Algorithm 2. Reducing the rows and columns of a matrix

```
Input  : matrix M; the current sum of all reductions sum.
Output : reduced matrix M; modified sum.
1  Procedure Reduction (M, sum)
       /* Reduce the rows of the matrix M                                   */
2      for i ∈ I(M) do
3          m := ∞
           /* Find m = m(i) = min_{j ∈ J(M)} M[i,j]                         */
4          for i ∈ 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
       /* Reduce columns of the matrix 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
```

Algorithm 3. Arc selection

```
Input  : matrix M.
Output : the arc (i*, j*) whose lower Hamiltonian circuit length estimate is maximum when forbidden.
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 := ∞
                   /* Find 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 := ∞
                   /* Find k = min_t M[t,j]                                 */
10                 for t ∈ I(M) \ {i} do
11                     if k > M[t,j] then k := M[t,j]
                   /* Compare m + k with the current record w               */
12                 if m + k > w then
13                     w := m + k
14                     (i*, j*) := (i, j)
```

2. DIRECT ALGORITHMS

In presenting the basics of the theory of direct algorithms, we follow [7] (see also [8]).

For the purpose of unifying the description, the arc-length matrix C will hereafter be called an *input data vector*[2] or simply an *input*. The solution of the traveling salesman problem, i.e., the Hamiltonian circuit $H \subseteq A$, will be represented as a 0/1-vector $\mathbf{x} = (x_{ij})$ with the same dimensionality as C. The coordinates of this vector $x_{ij} = 1$, for $(i, j) \in H$, and $x_{ij} = 0$ otherwise. We denote by X the set of all 0/1"-vectors $\mathbf{x}$ corresponding to the Hamiltonian circuits in the orgraph G under consideration. Therefore, given a fixed input C, the traveling salesman problem is to find a solution $\mathbf{x}^* \in X$ such that $\langle \mathbf{x}^*, C\rangle \le \langle \mathbf{x}, C\rangle$ $\forall \mathbf{x} \in X$. Hereafter, we call such a solution $\mathbf{x}^*$ *optimal with respect to input C*. Following [7, Definition 1.1.2], the set of all such optimization problems formed by a fixed set of feasible solutions X (in the case of the traveling salesman problem, X is uniquely defined by the number of vertices of the orgraph G) and all possible input vectors C will be called a *problem* X. Two feasible solutions $\mathbf{x}, \mathbf{y} \in X$ of the problem X are called *adjacent* if the vector C is found such that they and only they are optimal with respect to C. The subset $Y \subseteq X$ is called a *clique* if any pair $\mathbf{x}, \mathbf{y} \in Y$ is adjacent.

The convex hull $\mathrm{conv}(X)$ is called a *polyhedron of the problem* X. Since X is a subset of the vertices of the unit cube in the traveling salesman problem, X coincides with the set of vertices of the polyhedron $\mathrm{conv}(X)$. In this terminology, two solutions $\mathbf{x}, \mathbf{y} \in X$ are adjacent if and only if the corresponding vertices of the polyhedron $\mathrm{conv}(X)$ are adjacent [7]. It is known [12] that all vertices of the traveling salesman polyhedron are pairwise adjacent for $n < 6$, where n is the number of vertices of the orgraph G in which the optimal Hamiltonian circuit needs to be found.

Direct algorithms belong to the class of linear separation algorithms, which can be conveniently represented as linear decision trees.

Definition 1 ([7, Definition 1.3.1]). A linear decision tree of the problem $X \subset \mathbb{Z}^m$ is an oriented tree that has the following properties:

(a) Every node except one, called the root, has exactly one arc; there are no arcs entering the root.

(b) For every node, there are either two arcs coming out of it, or there are no such arcs at all; in the first case, the node is called an inner node and in the second one, an outer node, or a leaf.

(c) Some vector $B \in \mathbb{Z}^m$ is associated with some inner nodes.

(d) Each leaf corresponds to an element from X (several leaves can correspond to the same element of the set X).

(e) Each arc d corresponds to a number $\mathrm{sgn}d$, either 1 or -1; two arcs coming from the same node have different values.

(f) For each circuit $W = B_1 d_1 B_2 d_2 \ldots B_k d_k \mathbf{x}$ connecting the root and the leaf (the notation of the circuit lists vectors B_i corresponding to its nodes; the arc d_i exits a node B_i, $i \in [k]$), and for any input C from the inequalities $\langle B_i, C\rangle \mathrm{sgn} d_i \ge 0$, $i \in [k]$, it follows that the solution $\mathbf{x}$ is optimal with respect to C.

Therefore, within the theory of linear separation algorithms, attention is paid only to those operations in which conditions $\langle B, C\rangle \ge 0$ are tested, where C is the vector of input data. For example, in line 5 of Algorithm 2, the inequality $\infty > C_{11}$ is tested in the very first step of the loop; in the second step, the condition $C_{11} > C_{12}$ is tested, etc. In functions `HamiltonCycle` and `ForbiddenArc` of Algorithm 1, from the point of view of linear separation algorithms, nothing interesting happens, because no comparisons with elements of the input data vector are performed.

The operation of the linear separation algorithm for a fixed input data vector C is some circuit $B_1 d_1 B_2 d_2 \ldots B_m d_m \mathbf{x}$ connecting the root B_1 and some leaf $\mathbf{x}$ of the corresponding linear decision tree. The leaf in our case is a Hamiltonian circuit (more precisely, its characteristic vector) which it is optimal with respect to C.

Let B be some internal node in the linear decision tree of the algorithm, and X be the set of all feasible solutions (the set of labels of all leaves). Denote by X_B, $X_B \subseteq X$, the set of labels of all leaves of this tree preceded by the node B, and by X_B^+ and X_B^- we denote subsets of the set X_B corresponding to two arcs

[2] Elements of the matrix can always be written out in a row or a column.

exiting from B. Obviously, $X_B = X_B^+ \cup X_B^-$. Denote by $R_B^- = X_B^+ \backslash X_B^-$ the set of labels that are discarded in the case of transition along the negative arc. By analogy, let us define the set of labels $R_B^+ = X_B^- \backslash X_B^+$ that are discarded when passing through a "positive" arc.

Definition 2 ([7, Definition 1.4.2]). A linear decision tree is called a direct tree if for any internal node B and for any clique $Y \subseteq X$ the inequality

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

It follows directly from the definition that the height of a direct tree (i.e., the number of comparisons used by the algorithm in the worst case) for the problem X cannot be less than $\omega(X) - 1$, where $\omega(X)$ is the clique number of the set X [7, Theorem 1.4.3].

If we want to prove that some algorithm is not a direct algorithm, it is sufficient to specify a clique Y consisting of four solutions and a node B such that $\left|R_B^+ \cap Y\right| = \left|R_B^- \cap Y\right| = 2$.

For each $\mathbf{x} \in X$, let us define the *initial data cone*

$$K(\mathbf{x}) = \left\{C \,\middle|\, \langle \mathbf{x}, C\rangle \le \langle \mathbf{y}, C\rangle,\ \forall \mathbf{y} \in X\right\}.$$

That is, $K(\mathbf{x})$ consists of all vectors C such that $\mathbf{x}$ is optimal with respect to C.

Definition 3 ([7, Definition 1.4.4]). A linear decision tree is called a "direct" tree if every circuit $B_1 d_1 B_2 d_2 \ldots B_k d_k \mathbf{x}$ connecting the root and the leaf satisfies the following conditions:

(*) For any $\mathbf{y} \in X$ adjacent to $\mathbf{x}$, there is a number $i \in [k]$ such that the conditions $\langle B_i, C\rangle \operatorname{sgn} d_i > 0$ and $C \in K(\mathbf{y})$ are incompatible.

(**) For any $i \in [k]$, from the incompatibility of the conditions

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

for $\mathbf{y}$ adjacent to $\mathbf{x}$, and from the solidity of the cone

$$K(\mathbf{x}) \cap \left\{C \,\middle|\, \langle B_i, C\rangle \operatorname{sgn} d_i \le 0\right\}$$

it follows that the branch starting at the node B_i with the arc $-d_i$ has at least one leaf labeled $\mathbf{x}$.

"Direct" trees are united with direct trees by the fact that their height is also bounded from below by the quantity $\omega(X) - 1$ [7, Theorem 1.4.5].

In order to prove that Algorithm 1 is not a "direct" algorithm, we restrict ourselves to testing condition (*) of this definition. Namely, we specify a very particular input vector C^* that uniquely defines some circuit $B_1 d_1 B_2 d_2 \ldots B_k d_k \mathbf{x}$. Next, we choose $\mathbf{y} \in X$ adjacent to $\mathbf{x}$, for which conditions $\langle B_i, C\rangle \operatorname{sgn} d_i > 0$ and $C \in K(\mathbf{y})$ are compatible for any $i \in [k]$. Let us emphasize that we need to test the compatibility of conditions $\langle B_i, C\rangle \operatorname{sgn} d_i > 0$ and $C \in K(\mathbf{y})$ separately for each $i \in [k]$, regardless of the results of other comparisons. That is, for each $i \in [k]$, it is sufficient to specify C_i such that $\langle B_i, C_i\rangle \operatorname{sgn} d_i > 0$ and $C_i \in K(\mathbf{y})$.

3. ALGORITHM 1 IS NOT "DIRECT"

Consider the traveling salesman problem in a complete orgraph on five vertices. The set of feasible solutions X of such a problem consists of 24 0/1-vectors corresponding to the Hamiltonian circuits in that orgraph. All 24 solutions are pairwise adjacent [12].

Assume that the elements of the arc length matrix $C \in \mathbb{Z}^{5\times 5}$ satisfy the following conditions:

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

At the very beginning of the operation of the algorithm under consideration, this matrix is reduced (Algorithm 2). We limit our discussion to the row reduction stage. Successive comparisons result in selecting the smallest element in the first row (in this case, c_{12}) and subtracting it from all its elements. Then the minimum element in the second line is selected, which is c_{21}, and the minimum element in the third line is c_{35}. Then the algorithm proceeds to test the following inequality

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

(the comparison $\infty > c_{41}$ is used in the algorithm only for brevity and does not contain any information). The corresponding node of the linear decision tree of the algorithm is denoted by B. It is clear that the algorithm enters this tree node if and only if conditions (2) are satisfied for the input vector C.

Consider the characteristic vectors of the four Hamiltonian circuits:

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

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

It is easy to test that the input vectors

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

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

satisfy conditions (2), and for each $\mathbf{t} \in \{\mathbf{x}, \mathbf{y}, \mathbf{z}, \mathbf{w}\}$ and for any $\mathbf{s} \in X \setminus \{\mathbf{t}\}$ the inequality $\langle \mathbf{t}, C_t \rangle = 5 < \langle \mathbf{s}, C_t \rangle$ is satisfied. Hence, all four vectors are included in the set of labels X_B of all the leaves of the tree of the algorithm preceded by the node B.

Let us show that $\mathbf{z}$ and $\mathbf{w}$ are part of the set of labels R_B^+ discarded if inequality (3) is satisfied, and $\mathbf{x}$ and $\mathbf{y}$ are part of the set of labels R_B^- discarded if inequality (3) is not satisfied.

Assume that conditions (2) and inequality (3) are satisfied for the input matrix C. Then $\langle \mathbf{z}, C \rangle > \langle \mathbf{z}', C \rangle$ for

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

Similarly, $\langle \mathbf{w}, C \rangle > \langle \mathbf{w}', C \rangle$ for

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

Therefore, $\mathbf{z}, \mathbf{w} \in R_B^+$.

Assume that conditions (2) are satisfied for C, but inequality (3) is not satisfied. Then $\langle \mathbf{x}, C \rangle > \langle \mathbf{x}', C \rangle$ for

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

and $\langle \mathbf{y}, C \rangle > \langle \mathbf{y}', C \rangle$ for

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

Hence, $\mathbf{z}, \mathbf{w} \in R_B^+$.

Therefore, condition (1) for a given node B is not satisfied, and Algorithm 1 is not a direct algorithm.

4. ALGORITHM 1 IS NOT "DIRECT"

When analyzing Algorithm 1 as a linear decision tree, we only encounter inequalities of the following kind:

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

where $C \in \mathbb{Z}^{n^2}$ is the input vector,

$$\begin{gathered} B^+, B^- \in \{0,1\}^{n^2}, \ \ \langle B^+, B^- \rangle = 0 \\ \text{and } \langle B^+, \mathbf{1} \rangle = \langle B^-, \mathbf{1} \rangle > 0, \end{gathered} \tag{5}$$

$\mathbf{1}$ is the all-ones vector. In other words, condition (5) means that the sets of unit coordinates for B^+ and B^- are equal and do not overlap. For each such inequality and for some feasible solution of $\mathbf{y} \in X \subset \{0,1\}^{n^2}$, we need to check that there exists $C \in K(\mathbf{y})$ for which this inequality holds. This analysis is greatly simplified if we use the following criterion.

Lemma 1. *Let $\mathbf{y} \in \{0,1\}^{n^2}$ be the characteristic vector of some Hamiltonian circuit in a complete orgraph $G = ([n], A)$. If conditions (5) and $\langle B^+, \mathbf{y} \rangle \le 2$ are satisfied, then inequality (4) and condition $C \in K(\mathbf{y})$ are compatible.*

Proof. Let

$$S = \left\{ (i,j) \in [n]^2 \,\middle|\, y_{ij} = 1 \text{ and } B^+_{ij} = 0 \right\}.$$

It follows from condition $\langle B^+, \mathbf{y} \rangle \le 2$ that $|S| \ge n - 2$. Assume

$$C := \mathbf{4} - B^-$$

and, after that, $C_{ij} := 0$ for $(i, j) \in S$.

Then $\langle B^+, C\rangle = \langle B^+, \mathbf{4} - B^-\rangle = \langle B^+, \mathbf{4}\rangle$ and $\langle B^-, C\rangle \le \langle B^-, \mathbf{4} - B^-\rangle = \langle B^+, \mathbf{4}\rangle - \langle B^-, B^-\rangle$ (since B^+ and B^- satisfy condition (5)). Hence, inequality (4) for such C will be satisfied.

Let us now show that $\langle \mathbf{y}, C\rangle < \langle \mathbf{x}, C\rangle$ for any $\mathbf{x} \in X \setminus \mathbf{y}$.

Obviously, $\langle \mathbf{y}, C\rangle = (n - |S|) \times 4 \le 8$.

Let $\mathbf{x} \in X$. Note that if $\langle \mathbf{y}, \mathbf{x}\rangle \ge n - 2$, then $\mathbf{x} = \mathbf{y}$, because any Hamiltonian circuit in an orgraph on n vertices is uniquely defined by any of its $n - 2$ arcs. Hence, $\langle \mathbf{x}, C\rangle \ge 3 \times 3 = 9$ for any $\mathbf{x} \in X \setminus \mathbf{y}$.

In particular, the conditions of the lemma are satisfied if B^+ has no more than two units.

Therefore, assume $n = 4$ and consider the following input vector (we substitute infinity with a space):

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

It is clear that the only optimal solution is the following vector

$$\mathbf{x} := \begin{pmatrix} & 1 & 0 & 0 \\ 0 & & 1 & 0 \\ 0 & 0 & & 1 \\ 1 & 0 & 0 & \end{pmatrix}$$

and its corresponding circuit $\{(1,2),(2,3),(3,4),(4,1)\}$. It is easy to test that the set of all feasible solutions X consists of six pairwise adjacent vectors. Assume

$$\mathbf{y} := \begin{pmatrix} & 0 & 0 & 1 \\ 0 & & 1 & 0 \\ 1 & 0 & & 0 \\ 0 & 1 & 0 & \end{pmatrix}.$$

Note that $\mathbf{y}$ is the second (after $\mathbf{x}$) by optimality with respect to C^*. This fact greatly simplifies further testing of corresponding comparisons.

In general, the working scheme of the algorithm for a given input C^* is shown in Fig. 1.

First of all, let us consider which inequalities are tested at the first run of the `BranchBound` procedure with input C^*. When reducing the first row of the matrix C^* (row 5 of Algorithm 2), the inequalities $\infty > C_{12}$, $C_{13} > C_{12}$, and $C_{14} > C_{12}$ are tested (and satisfied). Further on, we will not consider inequalities in which the sum (or difference) of elements of the initial matrix is compared to infinity, because they are always satisfied and compatible with any feasible solution. Note that the listed inequalities satisfy the conditions of Lemma 1, because $\langle B^+, \mathbf{1}\rangle = 1$. Hence, they are compatible with the condition $C \in K(\mathbf{y})$.

When the first row is reduced, its cells $\mathtt{M}[1, j]$, $j \in [4]$, contain differences $C_{1j} - C_{12}$, and the variable `sum` takes the value of C_{12}.

When the second row is reduced, the inequalities $C_{21} > C_{23}$ and $C_{24} > C_{23}$ are tested. According to Lemma 1, they are consistent with the condition $C \in K(\mathbf{y})$.

When the second row is reduced, in its cells $\mathtt{M}[2, j]$, $j \in [4]$, there are differences $C_{2j} - C_{23}$, and the variable `sum` takes the value $C_{12} + C_{23}$.

When the last two rows are reduced, the situation is exactly the same. When the reduction of the rows is finished

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

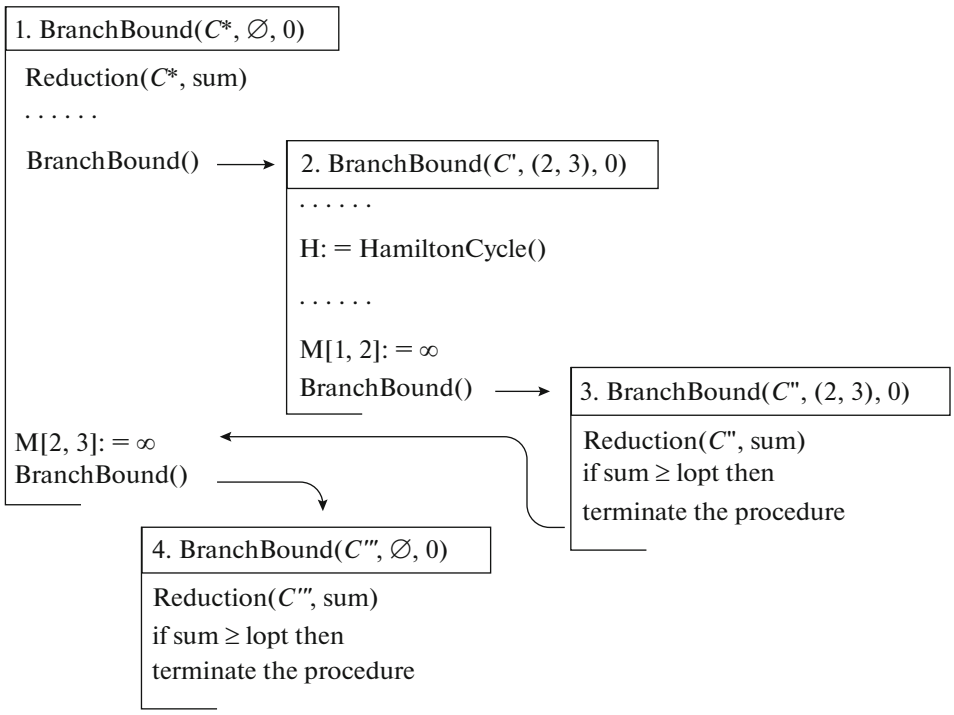

Fig. 1. General scheme of Algorithm 1 for the input given by formula (6).

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

Next, when the first column is reduced, the inequalities $\mathtt{M}[2,1] > \mathtt{M}[3,1]$ and $\mathtt{M}[3,1] > \mathtt{M}[4,1]$ are tested. We know that $\mathtt{M}[2,1] = C_{21} - C_{23}$, $\mathtt{M}[3,1] = C_{31} - C_{34}$, $\mathtt{M}[4,1] = C_{41} - C_{41} = 0$. Hence, the inequalities $C_{21} - C_{23} > C_{31} - C_{34}$ and $C_{31} - C_{34} > 0$ are tested. Each of them satisfies the conditions of Lemma 1.

When we reduce the remaining three columns, the situation repeats. The value of `sum` does not change when the columns are reduced, because each column already contains zeros.

After that, the `sum` ≥ `lopt` condition is tested in Algorithm 1. However, `lopt` = ∞. Therefore, the algorithm proceeds to calculate the function `ChooseArc`.

The first zero element is $\mathtt{M}[1,2]$. After that, the comparisons are performed $\infty > \mathtt{M}[1,3]$ and $\mathtt{M}[1,3] > \mathtt{M}[1,4]$ in row 8 of Algorithm 3. In this case, after the previous reduction step, we have $\mathtt{M}[1,3] = C_{13} - C_{12}$ and $\mathtt{M}[1,4] = C_{14} - C_{12}$. Obviously, the inequality $C_{13} - C_{12} > C_{14} - C_{12}$ satisfies the conditions of Lemma 1. In this step, $m := C_{14} - C_{12}$ is assigned. Next, in line 11 of Algorithm 3, comparisons $\infty > \mathtt{M}[3,2]$ and $\mathtt{M}[3,2] > \mathtt{M}[4,2]$ are performed. In this step, $\mathtt{M}[3,2] = C_{32} - C_{34}$ and $\mathtt{M}[4,2] = C_{42} - C_{41}$. The conditions of Lemma 1 are satisfied again. In this step, $k := C_{42} - C_{41}$ is assigned. Next, the comparison $m + k > -1$ or, what is the same, $C_{14} - C_{12} + C_{42} - C_{41} > -1$ is performed. Obviously, this inequality is compatible with the condition $C \in K(\mathbf{y})$. The variable w is populated with the value of the expression $C_{14} - C_{12} + C_{42} - C_{41}$.

The second zero element is $\mathtt{M}[2,3]$. By analogy, let us list only nontrivial comparisons. The inequality $\mathtt{M}[2,1] \le \mathtt{M}[2,4]$ or $C_{21} - C_{23} \le C_{24} - C_{23}$ is obviously compatible with the condition $C \in K(\mathbf{y})$. The inequality $\mathtt{M}[1,3] \le \mathtt{M}[4,3]$ is also compatible. Next, in row 12 the inequality $m + k > w$ is tested or, given the previous steps,

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

Obviously, it satisfies the conditions of Lemma 1. After this step

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

The third zero element is M[3,4]. The inequality M[3,1] < M[3,2] or $C_{31} - C_{34} < C_{32} - C_{34}$ is obviously compatible with the condition $C \in K(\mathbf{y})$. The inequality M[1,4] < M[2,4] is also compatible. The condition $m + k < w$ is as follows

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

and is also compatible with the condition $C \in K(\mathbf{y})$.

The fourth zero element is M[4,1]. It is easy to test that M[4,2] < M[4,3] and M[3,1] < M[2,1] are compatible with the condition $C \in K(\mathbf{y})$. The condition $m + k < w$ is as follows

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

and is also compatible.

At this point, we are still in the first instance of the BranchBound procedure. After the implementation of the function ChooseArc described above, the arc $(i, j) = (2,3)$ is selected (the sum $m + k$ turned out to be the largest for it), the second row and the third column are removed from the matrix M, and the arc (3,2) becomes forbidden. The following matrix is fed to the input of the second instance of the procedure BranchBound

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

(the empty row and the empty column are left for the ease of reading). It is clear that nothing new happens when it is reduced, because each row and each column contains zeros. When the function ChooseArc is called in line 12, the following comparisons $m + k > w$ are carried out.

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

Obviously, this inequality is compatible with the condition $C \in K(\mathbf{y})$. Next, the following inequality holds

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

which satisfies the conditions of Lemma 1. The following comparison

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

is also compatible with $C \in K(\mathbf{y})$.

Thus, after calling the function (1,2) in the second instance of BranchBound, the arc (1,2) is selected. A Hamiltonian circuit with the arcs (2,3) and (1,2) is uniquely defined. The following assignment is performed

$$\texttt{lopt} := C_{12} + C_{23} + C_{34} + C_{41}.$$

The algorithm then proceeds to consider cases in which the circuit contains the arc (2,3) but does not contain (1,2). The third instance of BranchBound is started with the matrix

$$C'' := \begin{pmatrix} & & 1 \\ & & \\ 1 & & 0 \\ 0 & 1 & \end{pmatrix}.$$

In the reduction, the two ones are replaced by zeros. No "discarding" comparisons are carried out. The value of the variable sum is incremented by $\texttt{M}[1,4] = C_{14} - C_{12}$ and by $\texttt{M}[4,2] = C_{42} - C_{41}$. The current instance of the procedure terminates on row 3 after testing the inequality $\texttt{sum} \ge \texttt{lopt}$:

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

Note that a valid solution $\mathbf{y}$ is completely discarded by the algorithm at this very step (taking into account the previously tested inequality $C_{31} > C_{34}$). Nevertheless, this inequality satisfies the conditions of Lemma 1 and, hence, together with the condition $C \in K(\mathbf{y})$.

Along with the third instance of the procedure `BranchBound`, the second instance of this procedure is also terminated. The algorithm proceeds to execute the next to last row in the first instance. In this instance

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

In order to analyze cases, in which the circuit does not have the arc (2,3), we call a fourth instance of the procedure with the following matrix

$$C''' := \begin{pmatrix} & 0 & 2 & 1 \\ 2 & & & 2 \\ 1 & 2 & & 0 \\ 0 & 1 & 2 & \end{pmatrix}.$$

When reducing the second row, the comparison $\texttt{M}[2,1] \le \texttt{M}[2,4]$ is performed, and when reducing the third column, $\texttt{M}[1,3] \le \texttt{M}[4,3]$. Obviously, neither of them discards the entire cone $K(\mathbf{y})$. The value is increased by $(C_{21} - C_{23}) + (C_{13} - C_{12})$.

Finally, the comparison $\texttt{sum} \ge \texttt{lopt}$ completes this fourth instance of the procedure and the entire algorithm in general. This comparison is as follows

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

and is also compatible with the condition $C \in K(\mathbf{y})$.

Thus, condition (*) from Definition 3 is not satisfied for this algorithm.

CONFLICT OF INTEREST

The author declares that he has no conflicts of interest.

REFERENCES

1. Bondarenko, V., Nikolaev, A., and Shovgenov, D., 1-skeletons of the spanning tree problems with additional constraints, *Autom. Control Comput. Sci.*, 2017, vol. 51, no. 7, pp. 682–688. https://doi.org/10.3103/S0146411617070033
2. Bondarenko, V. and Nikolaev, A., On graphs of the cone decompositions for the min-cut and max-cut problems, *Int. J. Math. Math. Sci.*, 2016, vol. 2016, p. 7863650. https://doi.org/10.1155/2016/7863650
3. Bondarenko, V. and Nikolaev, A., Some properties of the skeleton of the pyramidal tours polytope, *Electron. Notes Discrete Math.*, 2017, vol. 61, pp. 131–137. https://doi.org/10.1016/j.endm.2017.06.030
4. Bondarenko, V.A., Nikolaev, A.V., and Shovgenov, D.A., Polyhedral characteristics of balanced and unbalanced bipartite subgraph problems, *Autom. Control Comput. Sci.*, 2017, vol. 51, no. 7, pp. 576–585. https://doi.org/10.3103/S0146411617070276
5. Bondarenko, V.A. and Nikolaev, A.V., On the skeleton of the polytope of pyramidal tours, *J. Appl. Ind. Math.*, 2018, vol. 12, no. 1, pp. 9–18. https://doi.org/10.1134/S1990478918010027
6. Bondarenko, V.A., Nonpolynomial lowerbound of the traveling salesman problem complexity in one class of algorithms, *Autom. Remote Control*, 1983, vol. 44, no. 9, pp. 1137–1142.
7. Bondarenko, V., Geometrical methods of systems analysis in combinatorial optimization, *Doctor Sci. (Phys.-Math.) Dissertation*, Yaroslavl': Demidov Yaroslavl State Univ., 1993.
8. Bondarenko, V. and Maksimenko, A., *Geometricheskie konstruktsii i slozhnost' v kombinatornoi optimizatsii* (Geometric Structures and Complexity in Combinatorial Optimization), Moscow: URSS, 2008.
9. Maksimenko, A.N., Characteristics of complexity: clique number of a polytope graph and rectangle covering number, *Mod. Anal. Inf. Sist.*, 2014, vol. 21, no. 5, pp. 116–130.
10. Little, J.D.C., Murty, K.G., Sweeney, D.W., and Karel, C., An algorithm for the traveling salesman problem, *Oper. Res.*, 1963, vol. 11, no. 6, pp. 972–989.
11. Reingold, E., Nievergelt, J., and Deo, N., *Combinatorial Algorithms: Theory and Practice*, Prentice Hall College Div., 1977.
12. Padberg, M.W. and Rao, M.R., The travelling salesman problem and a class of polyhedra of diameter two, *Math. Program.*, 1974, vol. 7, no. 1, pp. 32–45. https://doi.org/10.1007/BF01585502

Translated by O. Pismenov