UDC 519.6
Doi: 10.31772/2587-6066-2020-21-2-187-194

For citation: Kuznetsov A. A., Kishkan V. V. A routing algorithm for the Cayley graphs generated by permutation groups. *Siberian Journal of Science and Technology.* 2020, Vol. 21, No. 2, P. 187–194. Doi: 10.31772/2587-6066-2020-21-2-187-194

Для цитирования: Кузнецов А. А., Кишкан В. В. Об одном алгоритме маршрутизации на графах Кэли, порожденных группами подстановок // Сибирский журнал науки и технологий. 2020. Т. 21, № 2. С. 187–194. Doi: 10.31772/2587-6066-2020-21-2-187-194

A ROUTING ALGORITHM FOR THE CAYLEY GRAPHS GENERATED BY PERMUTATION GROUPS

A. A. Kuznetsov[*], V. V. Kishkan

Reshetnev Siberian State University of Science and Technology
31, Krasnoyarskii rabochii prospekt, Krasnoyarsk, 660037, Russian Federation
[*]E-mail: kuznetsov@sibsau.ru

The purpose of this work is to create an effective routing algorithm on the Cayley graphs of permutation groups, superior in its characteristics to an algorithm using an automatic group structure.

In the first section of the article we describe the auxiliary algorithm A–1 which allows numbering elements of a given permutation group.

In the second section we present the algorithm A–2 for calculating the routing table on the Cayley graph and algorithm A–3 for determination the optimal route between two arbitrary vertices of the graph. Estimates of time and space complexity are also obtained for these algorithms.

In the third section we describe the algorithm A–4 for calculation the minimal word of a group element. It is proved that the computational complexity of the algorithm will be proportional to the length of the input word.

The fourth section presents the results of computer experiments for some groups of permutation groups, which compare the time for calculating the minimum words using algorithm A – 4 and an algorithm based on the construction of an automatic group structure. It is shown that A – 4 is much faster than its competitor.

Keywords: Cayley graph, a permutation group.

ОБ ОДНОМ АЛГОРИТМЕ МАРШРУТИЗАЦИИ НА ГРАФАХ КЭЛИ, ПОРОЖДЕННЫХ ГРУППАМИ ПОДСТАНОВОК

А. А. Кузнецов[*], В. В. Кишкан

Сибирский государственный университет науки и технологий имени академика М. Ф. Решетнева
Российская Федерация, 660037, г. Красноярск, просп. им. газ. «Красноярский рабочий», 31
[*]E-mail: alex_kuznetsov80@mail.ru

Целью настоящей работы является создание эффективного алгоритма маршрутизации на графах Кэли групп подстановок, превосходящего по своим характеристикам алгоритм, использующий автоматическую структуру группы.

В первом разделе статьи описан вспомогательный алгоритм А–1, который позволяет нумеровать элементы заданной группы подстановок.

Во втором разделе представлен алгоритм А–2 для вычисления таблицы маршрутизации на графе Кэли и алгоритм А–3, который позволяет определить оптимальный маршрут между двумя произвольными вершинами графа. Для данных алгоритмов также получены оценки временной и пространственной сложности.

В третьем разделе описан алгоритм А–4, при помощи которого можно вычислить минимальное слово элемента группы. Доказано, что вычислительная сложность алгоритма будет пропорциональна длине входящего слова.

В четвертом разделе приведены результаты компьютерных экспериментов для некоторых групп подстановок, в которых сравнивается время вычисления минимальных слов по алгоритму А–4 и алгоритму, основанному на построении автоматической групповой структуры. Показано, что А–4 значительно быстрее своего конкурента.

Ключевые слова: граф Кэли, группа подстановок.

Introduction. Currently, the increasing demand for cloud computing is leading to the growth of large-scale data centers (DPCs).

Modern data centers contain hundreds of thousands of nodes interconnected by a network. The topology of such a network, i.e. the method of connecting nodes is a key link on which speed, fault-tolerance, reliability and other characteristics of the data center depend.

For this reason, network design is a very important task, including the search for graph models that have good topological properties and allow the use of efficient routing algorithms.

These are the qualities of Cayley graphs, which have such attractive topological properties as high symmetry, hierarchical structure, recursive construction, high connectivity, and fault-tolerance [1]. The definition of a Cayley graph implies that the vertices of the graph are elements of some algebraic group. The choice of the group and its generating elements allows us to obtain a graph [2] that meets the necessary requirements for diameter, degree of vertices, number of nodes, etc.

Suppose $G := \langle X \rangle$ is a finite group, generated by an ordered set $X := \{x_1 \prec x_2 \prec \ldots \prec x_m\}$, which is also called the alphabet. The set of all words (strings) over the alphabet X will be denoted by X^*. Let $w := x_1 x_2 \ldots x_l$ be a word over X and $|w| := l$ is its length. On the set X^* we also define the relation of order. Suppose v and w are two arbitrary words in the alphabet X. Then $v \prec w$, if $|v| < |w|$, a and in case of equal word lengths, the smaller word will be determined according to the introduced lexicographic order on generators. If it is necessary to emphasize that the string $v \in X^*$ matches the element $g \in G$, then we write v_g. The string v will be called the minimal word of the element g, if for all others $w \in X^*$, such that $v_g = w_g$, will be conducted $v \prec w$. It's obvious that every $g \in G$, then we will write v_g. It is evident that every $g \in G$ matches the unique minimal word. The length of an element of a group g is the length of its minimal word v, that is $|g| := \min\{|v_g| : v_g \in X^*\}$. It has been noted that in the general case, the task of determining the minimal word is NP-hard [3].

Cayley graph $\Gamma := Cay(G, X)$ of group G with relation to X is called a directed unweighted labeled graph with many vertices $V(\Gamma) := \{g \mid g \in G\}$ and many edges $E(\Gamma) := \{(g, gx) \mid x \in X, g \in G\}$. Suppose $(g, gx) \in E(\Gamma)$, then generator x is called an edge token. If $X = X \cup X^{-1}$, then graph Γ will be non-oriented. We assume that the unit element $e \notin X$, that is in graph Γ there are no loops. As it is known [4], the shortest distance between two arbitrary vertices of the graph g and h, which we denote $d(g,h)$, is equal to the length of the minimum word of the element $g^{-1}h$, i. e. $d(g,h) := |g^{-1}h|$.

Let us dwell on the currently known routing algorithms on Cayley graphs. Traditional methods, such as Dijkstra or Bellman-Ford algorithms, can be used on graphs of any kind, but require significant spatial and temporal resources [5]. For some families of Cayley graphs, there are special routing algorithms that, unlike traditional methods, use the topological characteristics of the graph, while reducing time and/or spatial complexity. These include graph families such as hypercube [6], butterfly [7] and star graph [8], which are Cayley graphs. In work [9], a routing algorithm for creped and star graphs based on sorting permutations is presented. However, this approach does not provide the shortest routing. K. Teng and B. Arden prove [10] that all finite Cayley graphs can be represented by generalized chordal rings, and then offer an iterative routing algorithm based on the lookup table. The space complexity of such an algorithm is $O|G|^2)$, and temporary is $O(D)$, where $|G|$ and D are size and diameter of the network, respectively. To find the shortest paths on the Borel graph L. Wang and K. Tang offer an algorithm [11], which first calculates in an autonomous mode the routing table from one node to all the others; then, using the transitive property of the Cayley graph, creates a routing table for all nodes. The computational complexity of this algorithm is limited $O(\log_4 |G|)$, and space complexity is $O(D \cdot |G|^2)$. In [12] a distributed fault tolerant routing algorithm on a Borel graph is presented. This two-phase algorithm uses two types of routing tables: static and dynamic.

In the article [13], a routing algorithm for a special class of Cayley graphs used as the topology of a wireless data center network is presented. This algorithm is two-level: for sending messages between servers in the same rack and servers in different racks. Each server is identified by three values: the coordinates of the rack, the tier on which the server is located, and its number on the tier. In addition, each server uses three routing tables to forward packets from the source to the destination along the shortest route.

The monograph [14] is a fundamental work on the relationship of algebraic groups and finite state automaton. In this case, the automaton structure of a special kind is determined on the group $G = \langle X \rangle$, using which it is possible to calculate the minimal word for any element of the group.

According to [14], the group finite state machine reads an arbitrary word $w_g \in X^*$, processes it and produces the minimal word of the element g. Herewith the time T_0 of word processing w will be proportional to the square of its length, i. e. $T_0 = O(|w|^2)$.

Using this result, an algorithm for finding the shortest path on a Cayley graph (denoted by A – 0) was proposed in work [15], while its computational complexity is limited, and space complexity $M_0 = O(|X| \cdot |G| + |\mathcal{A}|)$, where $|\mathcal{A}|$ is the number of states of a group automaton.

It should be noted that all the above routing algorithms can be assigned to one of the following categories:

a) those that are designed for specific Cayley graphs; b) universal with high space and time complexity and c) with low complexity, which do not provide shortest paths.

The purpose of this work is to create an effective routing algorithm on Cayley graphs of permutation groups that surpasses in its characteristics the algorithm from [15].

The first section of the article describes the auxiliary algorithm A – 1, which allows you to number the elements of a given permutation group.

The second section presents the algorithm A – 2 for calculating the routing table on the Cayley graph. Algorithm A – 3 is described below, it allows to determine the optimal route between two arbitrary vertices of the graph. Estimates of time and space complexity are also obtained for these algorithms.

The third section describes the algorithm A – 4, with which you can calculate the minimum word of an element of the group. It is proved that the computational complexity of the algorithm will be proportional to the length of the incoming word.

The fourth section presents the results of computer experiments for some groups of permutations, which compare the time needed to calculate the minimal words using algorithm A – 4 and the procedure from [15].

In conclusion, the prospects for the development of the project are considered.

1. Algorithms on Cayley graphs of permutation groups. Suppose G is finite group of permutations on the set of points $\Omega = \{1,2,\dots,n\}$. We denote $\alpha^g := g[\alpha]$ element image $\alpha \in \Omega$ under the influence $g \in G$. Orbit point $\alpha \in \Omega$ is called set $\alpha^G := \{\alpha^g \mid g \in G\}$.

Point stabilizer $\alpha \in \Omega$ we will call the set $G_\alpha := \{g \in G \mid \alpha^g = \alpha\}$. For given $\beta_1,\beta_2,\dots,\beta_i \in \Omega$ we inductively define

$$G_{\beta_1,\beta_2,\dots,\beta_i} := (G_{\beta_1,\beta_2,\dots,\beta_{i-1}})_{\beta_i} =$$
$$= \{g \in G \mid \beta_j^g = \beta_j,\ j = 0,1,\dots,i\}.$$

Sequence of different elements $B := (\beta_1,\beta_2,\dots,\beta_m)$ we will call the base of the group G, if $G_{\beta_1,\beta_2,\dots,\beta_m} = e$. Thus, only a single element of the group leaves all the points of the base stationary.

Suppose $G^{(i)} := G_{\beta_1,\beta_2,\dots,\beta_{i-1}}$. Next, we determine the chain of stabilizers

$$G := G^{(1)} \geq G^{(2)} \geq \ldots \geq G^{(m)} \geq G^{(m+1)} = e.$$

If $G^{(i+1)}$ is its own subgroup $G^{(i)}$ for $i \in [1,m]$, then the base B is called irreducible.

Adjacent group classes $G^{(i)}$ by subgroup $G^{(i+1)}$ have a one-to-one correspondence with the elements of the orbit $\Delta^{(i)} := \beta_i^{G^{(i)}}$. If $a,b \in G^{(i)}$ and $G^{(i+1)}a = G^{(i+1)}b$, then for some $h \in G^{(i+1)}$ will be conducted $a = hb$. Therefore, $\beta_i^a = \beta_i^{hb} = \beta_i^b$.

The above fact makes it possible to calculate a family of representatives of cosets (transversal) $U^{(i)}$ groups $G^{(i)} \bmod G^{(i+1)}$. For every $\gamma \in \Delta^{(i)}$ we define $u_i(\gamma) \in G^{(i)}$, which displays β_i in γ, other words $\beta_i^{u_i(\gamma)} = \gamma$. In the particular case if $\beta_i \to \beta_i$, then $u_i(\beta_i) := e$.

According to the notation introduced, we obtain an ordered sequence $U^{(i)} := (u_i(\gamma) \mid \gamma \in \Delta^{(i)})$. It's obvious that $|U^{(i)}| = |\Delta^{(i)}|$.

Putting together all $U^{(i)}$, we get a transversal:

$$U = \bigcup_{i=1}^{m} U^{(i)}.$$

By the Lagrange theorem $|G| = |G^{(1)} : G^{(2)}| \cdot |G^{(2)}| = |U^{(1)}| \cdot |G^{(2)}|$. Similarly, $|G^{(2)}| = |G^{(2)} : G^{(3)}| \cdot |G^{(3)}| = |U^{(2)}| \cdot |G^{(3)}|$. Continuing this process, we get $|G| = |U^{(1)}| \cdot |U^{(2)}| \cdots |U^{(m)}|$.

Suppose $B = (\beta_1,\beta_2,\dots,\beta_m)$ is the base G, then for any element of the group we can determine its base image $B^g := (\beta_1^g,\beta_2^g,\dots,\beta_m^g)$.

Lemma 1. *For* $\forall g \in G$ *base image* B^g *has a unique view.*

Proof. Suppose $B^g = B^h$, then $B^{gh^{-1}} = B$. Therefore, according to the definition of the base, $gh^{-1} = e$. Consequently, $g = h$.

Lemma 2. *Any element* $g \in G$ *can be unambiguously written in canonical form*

$$g := (u_m, u_{m-1},\dots,u_1) = u_m u_{m-1} \dots u_1,$$

where

$$u_i \in U^{(i)} \ \ u \ \ i \in [1,m]. \qquad (1)$$

Proof. As $g \in G$, then it is contained in some coset of the group G by group, th $G^{(2)}$ erefore, we write it in the form $g = h_2 u_1$, where $h_2 \in G^{(2)}$ and $u_1 \in U^{(1)}$. Similarly, $h_2 = h_3 u_2$, where $h_2 \in G^{(2)}$ and $u_2 \in U^{(2)}$.

Continuing this process, we obtain the desired formula.

In the future, we will also need many inverse elements of the transversal

$$\hat{U} = \bigcup_{i=1}^{m} \hat{U}^{(i)},$$

где $\hat{U}^{(i)} := (u_i^{-1}(\gamma) \mid \gamma \in \Delta^{(i)})$

To find items in $U^{(i)}$ and $\hat{U}^{(i)}$ we need an auxiliary array of indices $A_{m \times n}$, whose elements are defined as follows:

$$A_{ij} := \begin{cases} a_{ij} \in [0, |U^{(i)}|-1] - \\ \text{the element number of } u_i(j) \text{ in } U^{(i)}, \text{ if } j \in \Delta^{(i)}; \\ -1 \text{ if } j \notin \Delta^{(i)}. \end{cases}$$

Below is an algorithm that received a word at the input consisting of a product of elements $g_1g_2\ldots g_l$ of the group *G*, its base *B*, the transversal *U* and its inverse $\hat{U}$, as well as an auxiliary array *A*, returns canonical representation $u_mu_{m-1}\ldots u_1$ of this product.

Algorithm A–1. $u_mu_{m-1}\ldots u_1 = \text{Factor}(g_1g_2\ldots g_l, U, \hat{U}, A, B)$

Input: $g_1g_2\ldots g_l$, $\hat{U}$, *A* and *B*

Output: $u_mu_{m-1}\ldots u_1$

```
1. $h := B$
2. For all $i = 1,2,\ldots,l$
3. $h := (h_1^{g_i}, h_2^{g_i}, \ldots, h_m^{g_i})$
4. For all $i = 1,2,\ldots,m$
5. $j := h[i]$
6. $u_i := U^{(i)}[a_{ij}]$
7. $u_i^{-1} := \hat{U}^{(i)}[a_{ij}]$
8. $h := (h_1^{u_i^{-1}}, h_2^{u_i^{-1}}, \ldots, h_m^{u_i^{-1}})$
9. Return $u_mu_{m-1}\ldots u_1$
```

Suppose $T_i = O(f)$ and $M_i = O(f)$ is upper asymptotic estimates of the computational and space complexity of the i algorithm, respectively.

Lemma 3. *Algorithm A–1 is correct and* $T_1 = O(l \cdot m + m^2)$.

Proof. The correctness of the algorithm follows from Lemmas 1 and 2. First we get the base image of the element $g := g_1g_2\ldots g_l$, and then at each $i \in [1,m]$ stabilize the point $\beta_i \in B$. As a result, we get $B^{gu_1^{-1}\ldots u_m^{-1}} = B$. Consequently, $g = u_mu_{m-1}\ldots u_1$.

To assess the computational complexity, we note that the number of operations in a cycle of 2–3 is limited $O(l \cdot m)$, and cycle 4–8 – $O(m^2)$. Therefore, $T_1 = O(l \cdot m + m^2)$.

Decomposition (1) makes it possible to effectively number all $g \in G$, using the method of listing tuple elements in a mixed number system [16]. Suppose $c := (c_m, c_{m-1}, \ldots, c_1)$ is the basis of a mixed base notation in which $c_1 := 1$ and $c_i = c_{i-1} \cdot |U^{(i-1)}|$ for $i \geq 2$.

Suppose $g := u_mu_{m-1}\ldots u_1$. We define a bijective mapping $\mathcal{N} : u_mu_{m-1}\ldots u_1 \leftrightarrow (a_m, \ldots, a_1)$, where $a_i \in [0, |U^{(i)}|-1]$ is a number of the element u_i in $U^{(i)}$.

We note that the vector $(a_m, \ldots, a_1)$ is a number $\mathcal{N}(g) \in [0, |G|-1]$ in the scale of notation with mixed base $(c_m, \ldots, c_1)$.

Suppose $k := \mathcal{N}(g)$ is the element number g.

Lemma 4. *Computational complexity* $\mathcal{N}(g)$ *u* $\mathcal{N}^{-1}(k)$ *does not exceed* $O(m)$.

Proof. Suppose $g = u_mu_{m-1}\ldots u_1$ and a_i is number u_i in $U^{(i)}$. Then $k = \mathcal{N}(g) = a_mc_m + \ldots + a_1c_1$.

In backward case $g = \mathcal{N}^{-1}(k) = u_mu_{m-1}\ldots u_1$, where factors u_i are calculated as follows:

```
1. For all $i = 1,2,\ldots,m$
2. $a_i := k \mod |U^{(i)}|$
3. $k := \lfloor k / |U^{(i)}| \rfloor$
4. $u_i := U^{(i)}[a_i]$.
```

Obviously, the number of operations in these procedures does not exceed $O(m)$.

Comment 1. *For definiteness, we assume that all sequences* $U^{(i)}$ *start with a unity element e. In this case* $\mathcal{N}(e) := 0$.

For numbering the elements of a group, we need to know its base *B* and the complete family of representatives of adjacent classes *U*. To calculate them, we will use the well-known Schreyer – Sims algorithm proposed by C. Sims in 1970 [17]. Currently, there is a variety of its modifications [18]. The most efficient versions of the algorithm have low computational complexity and are implemented in computer algebra systems such as GAP, Magma, and Mathematica, as well as in the SymPy library for Python.

2. Cayley graph routing algorithms. To find the shortest paths on Cayley's graph $Cay(G,X)$ we need a routing table $P_{2\times|G|}$, which is also called the parent tree [19]. The following is an algorithm that, having received at the input a generating set of a group X, its base *B*, the transversal U and its inverse $\hat{U}$, as well as a backing array *A*, returns the specified table.

Hereinafter, we will be interested in the case $|X| \ll |G|$ and $n \ll |G|$.

Algorithm A–2. $P = \text{BFS}(X, U, \hat{U}A, B)$

Input: X, U, $\hat{U}$, *A* and *B*

Output: Routing table *P* on Cayley's graph $Cay(G,X)$

```
1. $P_{2\times|G|} := [\infty \; \ldots \; \infty]$
2. $k := \mathcal{N}(e) := 0$
3. $P[1][k] := -1$
4. $P[2][k] := -1$
5. $Q := \{k\}$
6. Until $Q \neq \varnothing$
7.    pop $q \in Q$ from the queue
8.    For all $x \in X$
9.        $s := \text{Factor}(\mathcal{N}^{-1}(q)x, U, \hat{U}, A, B)$
10.       $k := \mathcal{N}(s)$
```

```
11.    If $P[1][k] = \infty$
12.        add $k$ to the queue $Q$
13.        $P[1][k] := q$
14.        $P[2][k] := x$
15. Return $P$
```

Theorem 1. *Algorithm A–2 is correct and* $T_2 = O(m^2 \cdot |X| \cdot |G|)$.

Proof. This algorithm is a classical method in breadth-first search on a graph [19]. In this case, the vertex with the unit element number *e* (according to comment 1, we have $\mathcal{N}(e) := 0$) will be the root of the parent tree *P*. Suppose $gx = h$, where $\mathcal{N}(g) = k$ and $\mathcal{N}(h) = l$, then $P[1][l] := k$, $P[2][l] = x$. That means the vertex k is the parent of the vertex *l* and *x* is edge tokens (k, l).

We need to check altogether $|X| \cdot |G|$ of elements. The verification time for each element according to Lemmas 3 and 4 is limited $O(m^2)$. Accordingly, $T_2 = O(m^2 \cdot |X| \cdot |G|)$.

The following algorithm using the routing table *P* calculates the shortest route $w := x_1 x_2 \ldots x_s$ between vertexes $a, b \in Cay(G, X)$, where $x_i \in X$. It's obvious that $s \le D$, where *D* is diameter of the graph.

Algorithm A–3. $w = \text{Route}(a, b, U, \hat{U} A, B, P)$

Input: *a*, *b*, U, $\hat{U}$, *A*, *B* and *P*

Output: $w := x_1 x_2 \ldots x_s$ the shortest route from the vertex *a* to the vertex *b*

```
1. $w := [\ ]$ – empty word
2. $g_1 := \mathcal{N}^{-1}(a)$
3. $g_2 := \mathcal{N}^{-1}(b)$
4. $g := \text{Factor}(g_1^{-1} g_2, U, \hat{U}, A, B)$
5. $k := \mathcal{N}(g)$
6. $l := k$
7. Until $P[1][k] \ne -1$
8.     $k := P[1][l]$
9.     $w := P[2][l] \oplus w$ – string concatenation
10.    $l := k$
11. Return $w$
```

Theorem 2. *Algorithm A–3 is correct,* $T_3 = O(m^2 + D)$ *and* $M_3 = O(m \cdot n + |G|)$.

Proof. Cayley's shortest route from the vertex *a* to the vertex *b* will be a minimal word $w := x_1 x_2 \ldots x_s$ of the element $g := g_1^{-1} g_2$, where $g_1 := \mathcal{N}^{-1}(a)$, $g_2 := \mathcal{N}^{-1}(b)$ and $x_i \in X$ [4]. Suppose k is an element vertex number g. Moving through the parent tree *P* from the vertex k to its root, we will get the desired route.

Items 1–6 are limited in time $O(m^2)$, and cycle 7–10 is $O(D)$. As a result, we get $T_3 = O(m^2 + D)$.

Space complexity of variables U, $\hat{U}$, *A*, and *B* is limited $O(m \cdot n)$, and tables P – $O(|G|)$. So, $M_3 = O(m \cdot n + |G|)$.

Comment 2. *In many applied problems in the study of Cayley's graph* $\Gamma := Cay(G, X)$ *the order of the generating group* G *significantly exceeds its degree, i. e.* $m \le n \ll |G|$. *In this case* $M_3 = O(|G|)$.

Example. Let us consider the group $G = \langle X \rangle$, generated by two cycles $x = (1,5,4)$ and $y = (3,4)$. Let us calculate in GAP a base of G and a transversal:

$$B = (1,3,4),$$

$$U^{(1)} = (e, (1,4)(3,5), (1,5)(3,4), (1,3)(4,5)),$$

$$\Delta^{(1)} = (1,4,5,3),$$

$$U^{(2)} = (e, (3,4,5), (3,5,4)),\ \Delta^{(2)} = (3,4,5),$$

$$U^{(3)} = (e, (4,5)),\ \Delta^{(3)} = (4,5).$$

Using Algorithm 2, we get the Cayley graph $\Gamma = Cay(G, X)$ and its parent tree P (fig.1).

To illustrate Algorithm 3, we find the distance from the vertex $a := 15$ to $b := 22$.

$$g_1 := \mathcal{N}^{-1}(a) = U^{(3)}[1] U^{(2)}[0] U^{(1)}[3],$$

$$g_2 := \mathcal{N}^{-1}(b) = U^{(3)}[1] U^{(2)}[2] U^{(1)}[2],$$

$$g := \text{Factor}(g_1^{-1} g_2) = U^{(3)}[0] U^{(2)}[2] U^{(1)}[0],$$

$$k := \mathcal{N}(g) = 8.$$

Moving along the parent tree P from the vertex with the number to its root, we get

$$8 \xrightarrow{y} 20 \xrightarrow{x} 17 \xrightarrow{x} 14 \xrightarrow{y} 6 \xrightarrow{x} 0.$$

Consequently, $w := xyxxy$. On the graph Γ, the route will have the following form:

$$15 \xrightarrow{x} 19 \xrightarrow{y} 1 \xrightarrow{x} 4 \xrightarrow{x} 10 \xrightarrow{y} 22.$$

3. Problem of a minimal word. A small modification of algorithm A – 3 allows us to calculate the minimal word *w* from an arbitrary string $v := x_1 x_2 \ldots x_r$ in the alphabet of generators *X*.

Algorithm A–4. $w = \text{MinimalWord}(v, U, \hat{U} A, B, P)$

Input: *v*, *U*, $\hat{U}$, *A*, *B* и *P*

Output: $w := x_1 x_2 \ldots x_s$ – the minimal word

```
1. $w := [\ ]$ – the empty word
2. $g := \text{Factor}(v, U, \hat{U}, A, B)$
3. $k := \mathcal{N}(g)$
4. $l := k$
5. Until $P[1][k] \ne -1$
6.     $k := P[1][l]$
```

7. $w := P[2][l] \oplus w$ – string concatenation

8. $l := k$

9. **Return** w

Theorem 3. $T_4 = O(|v|)$ and $M_4 = O(m \cdot n + |G|)$.

Proof. According to Lemmas 3 and 4, the execution time of items 1–4 is limited $O(|v| \cdot m + m^2)$. Paragraphs 5–9 of the algorithm represent an upward movement in the parent tree *P*, therefore, the complexity of this section does not exceed $O(D)$. So we get, $T_4 = O(|v| \cdot m + m^2 + D)$. If the word is long, then $T_4 = O(|v|)$.

Space complexity of variables *U*, $\hat{U}$, *A* and *B* is limited $O(m \cdot n)$, and tables *P* – $O(|G|)$. So, $M_4 = O(m \cdot n + |G|)$.

4. Comparative analysis of algorithms. The following table provides estimates of the time and space complexity of algorithms A – 0 and A – 4.

Algorithm	T_i	M_i
A–0	$O(\|v\|^2)$	$O(\|X\| \cdot \|G\| + \|\mathcal{A}\|)$
A–4	$O(\|v\|)$	$O(m \cdot n + \|G\|)$

The table shows that the algorithm A – 4 has lower computational complexity in comparison with the algorithm A – 0.

As for space complexity, according to comment 2, for many interesting topologies $M_0 \sim M_4 \sim |G|$ will be conducted.

A – 4 was implemented by the authors in C ++; in turn, A – 0 was written in C and is a part of the freely distributed KBMAG package. In order to ensure the purity of the experiment, these algorithms were broadcast by the GCC compiler with the same parameters. At the initial stage, two groups were tested.

a) symmetrical group $S_9 = \langle Y \rangle$, where $Y = \{(i, i+1) \mid i \in [1,8]\}$. Caley graph $Cay(S_9, Y)$ is also called a bubble sort graph. It is well known that $S_n = n!$, $m = n-1$ and $D_Y(S_n) = \frac{n(n-1)}{2}$.

б) Matthew's sporadic simple group $M_{22} = \langle X \rangle$, where $X = \{x_1, x_2, x_2^{-1}\}$ and

$$x_1 = (1,13)(2,8)(3,16)(4,12)(6,22)(7,17)(9,10)(11,14);$$

$$x_2 = (1,22,3,21)(2,18,4,13)(5,12)(6,11,7,15)(8,14,20,10)(17,19).$$

Therein $|M_{22}| = 443520$, $m = 5$ and $D_X(M_{22}) = 34$.

Fig. 2 shows graphs of time dependence $T_4(l)$ execution of algorithm A – 4 for groups $S_9 = \langle Y \rangle$ and $M_{22} = \langle X \rangle$ depending on the length of the incoming word.

Fig. 3 shows graphs of the dependence of time $T_0(l)$ and $T_4(l)$ on the length of the input word for the specified groups.

These graphs clearly show that algorithm A – 4 is much faster than A – 0.

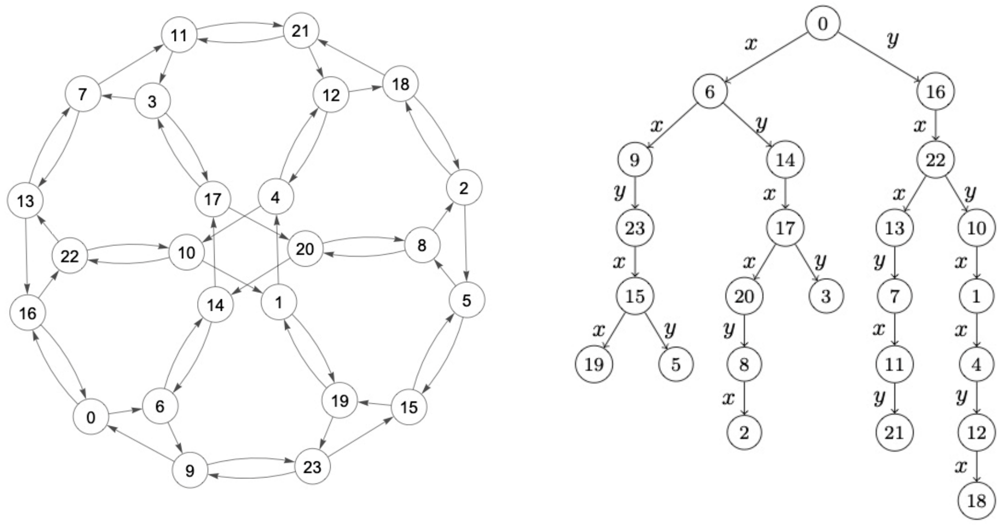

Fig. 1. The Cayley graph Γ = *Cay*(*G*, *X*) and its parent tree.
The edges of the graph Γ with the label *x* are represented as a straight line and *y* as an arc

Рис. 1. Граф Кэли Γ = *Cay*(*G*, *X*) и его родительское дерево *P*.
Ребра графа Γ с меткой *x* представлены в виде прямой линии, а *y* в форме дуги

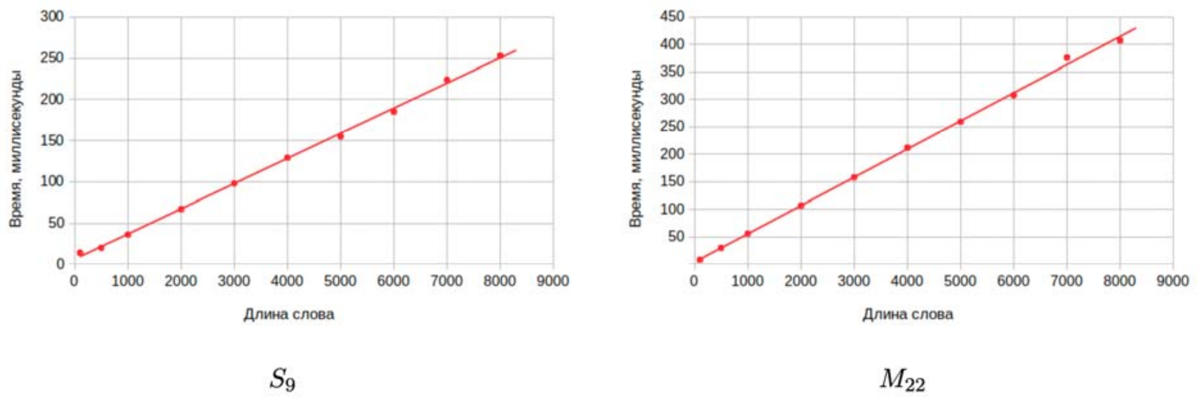

Fig. 2. Graphs of $T_4(l)$ for $S_9 = \langle Y \rangle$ and $M_{22} = \langle X \rangle$

Рис. 2. Графики $T_4(l)$ для групп $S_9 = \langle Y \rangle$ и $M_{22} = \langle X \rangle$

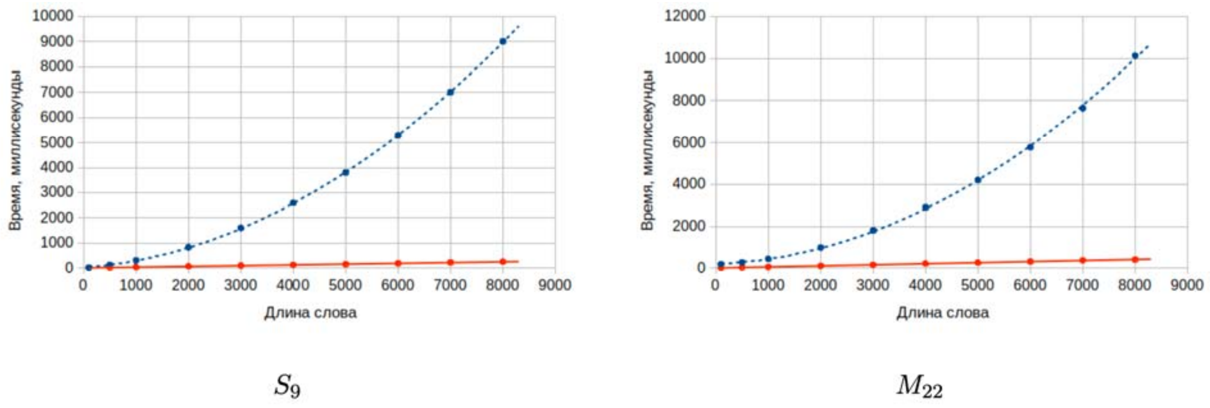

Fig. 3. Graphs of $T_0(l)$ (dotted line) and $T_4(l)$ for groups $S_9 = \langle Y \rangle$ and $M_{22} = \langle X \rangle$

Рис. 3. Графики $T_0(l)$ (пунктир) и $T_4(l)$ для групп $S_9 = \langle Y \rangle$ и $M_{22} = \langle X \rangle$

Conclusion. The algorithms presented in this paper will serve as a starting point for creating new resource-efficient routing algorithms. Two directions can be distinguished. Firstly, it is the creation of algorithms that take into account the network topology. In this case, algorithms will be designed for specific classes of Cayley graphs. Secondly, it is the development of hybrid algorithms that include both static and dynamic routing tables. This will allow to calculate the optimal routes depending on the current state of the network.

References

1. Heydemann M. Cayley graphs and interconnection networks, in Graph symmetry: algebraic methods and applications (Editors: Hahnand Sabidussi). *Dordrecht: Kluwer Academic Publishers*. 1997, P. 167–226.

2. Loz E. New record graphs in the degree-diameter problem. *Australasian Journal of Combinatorics*. 2008, Vol. 41, P. 63–80.

3. Even S., Goldreich O. The Minimum Length Generator Sequence is NP–Hard. *Journal of Algorithms*. 1981, Vol. 2, P. 311–313.

4. Holt D., Eick B., O'Brien E. *Handbook of computational group theory*. Boca Raton: Chapman & Hall/CRC Press, 2005, 514 p.

5. Schibell S., Stafford R. Processor interconnection networks and Cayley graphs. *Discrete Applied Mathematics*. 1992, Vol. 40, P. 337–357.

6. Stamoulis G., Tsitsiklis J. Effcient routing Scheme for Multiple Broadcasts in Hypercubes. *IEEE Trans. on Parallel and Distributed Systems*. 1993, Vol. 4(7), P. 725–739.

7. Stamoulis G., Tsitsiklis J. The Effciency of Greedy Routing in Hypercubes and Butteries. *IEEE Transaction on Communication*. 1994, Vol. 42(11), P. 3051–3061.

8. Kiasari A., Sarbazi-Azad H. Analytic performance comparison of hypercubes and star graphs with implementation constraints. *Journal of Computer and System Sciences*. 2008, No. 6, P. 1000–1012.

9. Akers S., Krishnamurthy B. A group theoretic model for symmetric interconnection networks. *Proceedings of the International Conference on Parallel Processing*. 1986, P. 216–223.

10. Tang K., Arden B. Vertex-transitivity and routing for Cayley graphs in GCR representations. *Proceedings of ACM Symposium on Applied Computing SAC*. 1992, P. 1180–1187.

11. Wang L., Tang K. Topology-Based Routing for Xmesh in Wireless Sensor Networks. *Lecture Notes in Electrical Engineering*. 2009, Vol. 44, P. 229–239.

12. Ryu J., Noel E., and Tang K. Fault-tolerant Routing on Borel Cayley Graph. *IEEE ICC Next Generation Networking Symposium*. 2012, P. 2872–2877.

13. Shin J., Sirer E., Weatherspoon H., Kirovski D. On the feasibility of completely wireless datacenters. *EEE/ACM Transaction On Networking*. 2013, Vol. 21(5), P. 1666–1679.

14. Epstein D., Paterson M., Cannon J., Holt D., Levy S. and Thurston W. *Word Processing in Groups*. Boston: Jones and Barlett Publishers, 1992, 330 p.

15. Camelo M., Papadimitriou D., Fabrega L., Vila P. Efficient Routing in Data Center with Underlying Cayley Graph. *Proceedings of the 5th Workshop on Complex Networks CompleNet*. 2014, P. 189–197.

16. Knuth D. *The Art of Computer Programming, Volume 4A: Combinatorial Algorithms, Part 1*. Boston: Addison-Wesley Professional, 2011, 912 p.

17. Sims C. Computational methods in the study of permutation groups, *Computational problems in abstract algebra (Pergamon Press, Oxford)*. 1970, P.169–183.

18. Seress A. *Permutation Group Algorithms*. Cambridge: Cambridge University Press, 2003, 274 p.

19. Skiena S. *The Algorithm Design Manual*. London: Springer Science+Business Media, 2008, 730 p.

Kuznetsov Alexander Alexeevich – D. Sc., Professor, Head of Institute of Space Research and High Technologies; Reshetnev Siberian State University of Science and Technology. E-mail: alex_kuznetsov80@mail.ru.

Kishkan Vladimir Vladimirovich – Postgraduate student; Reshetnev Siberian State University of Science and Technology. E-mail: kishkan@mail.ru.

Кузнецов Александр Алексеевич – доктор физико-математических наук, профессор, директор НОЦ «Институт космических исследований и высоких технологий»; Сибирский государственный университет науки и технологий имени академика М. Ф. Решетнева. E-mail: alex_kuznetsov80@mail.ru.

Кишкан Владимир Владимирович – аспирант; Сибирский государственный университет науки и технологий имени академика М. Ф. Решетнева. E-mail: kishkan@mail.ru.