ISSN 1064-5624, Doklady Mathematics, 2017, Vol. 96, No. 2, pp. 503–505. © Pleiades Publishing, Ltd., 2017.
Original Russian Text © K.Yu. Gorbunov, V.A. Lyubetsky, 2017, published in Doklady Akademii Nauk, 2017, Vol. 476, No. 6, pp. 614–616.

MATHEMATICS

The Minimum-Cost Transformation of Graphs

K. Yu. Gorbunov[a]* and V. A. Lyubetsky[a,b]**

Presented by Academician of the Russian Academy of Sciences A. L. Semenov May 13, 2017

Received May 17, 2017

Abstract—A complete proof that algorithms proposed by the authors solve the problem of minimum-cost transformation of a graph into another graph is given. The problem is solved both by a direct algorithm of linear complexity and by a reduction to quadratic integer linear programming.

DOI: 10.1134/S1064562417050313

1. STATEMENT OF TWO PROBLEMS AND RESULTS

The first problem. Suppose given directed graphs a and b and a list of operations over graphs. The operations are fixed and coincide with those proposed in [1] (their description is also given in [2, p. 162]): these are double, sesqui-fold, and single (cut and join) regluings, deletion, and insertion. Each operation is assigned a cost, which is a strictly positive rational number. It is required to find a sequence of operations of minimum total cost which transforms a into b. In this problem and the problem stated below, graphs are understood as finite sets of disjoint chains and cycles; each edge is assigned a name, which is positive integer. In [1, 2], such graphs were called chromosomal structures. In the first problem, the names of edges in each graph do not repeat.

We refer to the minimum total cost as the distance between a and b, although it is not a usual metric. A solution of the problem is called a shortest sequence. The directions of edges in chains and cycles are not assumed to be consistent.

The second problem. The data of this problem is the same as in the first problem, except that the names of edges in graphs are allowed to repeat, but, in return, the costs of all operations are equal. It is required to reduce this problem to quadratic integer linear programming (ILP). "Quadratic" means that the number of variables and constraints depends quadratically on the total size of the initial graphs a and b, say on the number of edges in these graphs. We denote the sets of edges with name n in a and b by $X(n, a)$ and $X(n, b)$, respectively. "Reduce" means that, for each n, it is required to find a partial injective mapping of a smaller (in size) set of elements of $X(n, a)$ and $X(n, b)$ to a larger set such that the corresponding graphs a' and b', in which the names of edges no longer repeat, have minimum distance. To be more precise, those edges which have coinciding names in a acquire unique names in a' (for b and b', a similar condition holds), and unique names are preserved by the mappings under consideration. Unique names can be obtained, e.g., simply by extending initial names by second positions invariant under the mappings; then a unique name has the form $n.k$ both in a and b, so that the new names do not repeat in a (and, similarly, in b).

We do not assume that the sets of names in a and b coincide, which fundamentally complicates both problems.

In addition to Theorem 1 stated below, the authors have obtained a rigorous solution of the first problem for the case where the cost d of insertion lies in the interval between c (equal to the costs of all other operations) and $2c$ (the solution is minimal up to the additive constant $d - c$) [2]. This case can be called nonstationary, and the case of Theorem 1a, stationary. This immediately implies a solution of the problem for the case of equal costs of operations.

The first problem was stated in [3, 4] and the second (in simplified form), in [5, 6]. The first problem has been extensively studied in solving various applied problems; its setting was augmented by strong assumptions, and under these assumptions, heuristic solution algorithms were proposed; detailed references can be found, e.g., in [3, 4]. To the best knowledge of the authors, mathematical results related to the first problem in the setting given above are contained only in [7, 8]. In [7], a solution of this problem for the special case

[a] *Kharkevich Institute for Information Transmission Problems, Russian Academy of Sciences, Moscow, 127051 Russia*
[b] *Mechanics and Mathematics Faculty, Moscow State University, Moscow, 119991 Russia*
**e-mail: gorbunov@iitp.ru*
***e-mail: lyubetsk@iitp.ru*

of equal costs of operations was suggested. In [8], a plan for solving this problem under assumptions specified below in Theorem 1 was proposed. As far as we know, the corresponding proof has not been published so far; apparently, the proposed plan cannot be implemented; the solution presented here is obtained in a different way (see below). The algorithms of [7, 8] fundamentally differ from those suggested here. For the second problem, mathematical results were obtained only in [5, 6]. In [5], the special case of coinciding sets of names in the graphs was considered; in [6], the problem was solved for another peculiar special case, where the graphs may have noncoinciding sets of edge names, but after a bijection between $X(n, a)$ and $X(n, b)$ is established, those edges which are encountered in only one of the graphs are ignored under the reduction of a to b.

For an equiaccessible address machine with uniform weight criterion, the first problem is to be solved by a linear algorithm and the second, by a quadratic algorithm.

By a component we mean a chain or a cycle. We refer to a graph consisting of cycles as a cyclic graph. In [9], for the case of coinciding sets of names in cyclic graphs a and b and equal costs of operations, a passage to the dual problem was proposed, namely, to the problem of reducing a breakpoint graph defined in [9] to the final form by means of dual operations. We define such a graph for any graphs a and b with noncoinciding sets of names; we denote it by $a + b$ in what follows. The definitions of these notions are given [2, p. 163]. We also introduce the notion of a chain interaction, which includes the notions of the type and the size of a component ([2, pp. 163–167]).

Theorem 1. *There is a linear algorithm solving the first problem for case where* (a) *the equal costs of insertion and deletion do not exceed half the equal costs of the other operations*; (b) *the initial graphs are cyclic*, *the costs of insertion and deletion are equal*, *and the costs of the other operations are equal to each other.*

Theorem 2. *There is a quadratic algorithm reducing the second problem to quadratic ILP.*

Solution algorithm for the first problem. For $a + b$, we perform the following steps:

(i) delete all loops;

(ii) excise ([2, p. 164, Step 2]) all ordinary edges except 2-cycles;

(iii) apply interactions 3.1, 3.2, 3.12, and 3.13, beginning with the first possible one;

(iv) close the chains of strictly positive size ([2, p. 170, item 4.21]) by edges so as to obtain cycles;

(v) divide ([2, p. 171, Step 5]) the cycles of size strictly larger than two into 2-cycles;

(vi) delete ([2, Step 5]) the remaining singular vertices.

Sketch of the proof of Theorem 1a. Let $c(G)$ be the least possible total cost of a sequence of operations reducing the graph $G = a + b$ to the final form, and let $T(G)$ be the total cost of operations in the reduction algorithm. We show that, for any G, we have

$$T(G) \le c(G). \tag{1}$$

The following important relation is also valid:

$$T(G) = (1 - w)(0.5d + 0.5f - c) + w(B + S + D) - P(G), \tag{2}$$

where w is the cost of the deletion and insertion operations; d, f, and c are, respectively, the total size of components in the graph $a + b$ and the numbers of odd chains and cycles in this graph; B is the total number of singular vertices in $a + b$; S is the sum of the integer parts of half-lengths of maximal connected pieces (we call them segments) composed of ordinary edges plus the number of extreme odd segments minus the number of cyclic segments; D equals 1 in the case of a chain of one of the types $1a$, $1b$, $3a$, $3b$, and 3 (and 0 otherwise) [2]; and $P(G)$ is difference of the total costs of operations in the above algorithm including or not including step (iii).

Inequality (1) follows directly from the inequality

$$c(o) \ \ge \ T(G) - T(o(G)), \tag{3}$$

which holds for any operation o and any G. Inequality (3) is verified by searching through all operations. Let o be the deletion of a singular vertex. Under the passage from G to $o(G)$, B decreases by 1. Let us consider the possible cases.

Suppose that an isolated singular vertex is deleted. Then, under the same passage, S, D, and c do not change, d increases by 1, and f decreases by 1. The value of P does not change or decreases by w; indeed, this value cannot increase under the deletion of a chain, and it cannot decrease by more than w either, because all interactions containing arguments of types $2a$ and $2b$ preserve the cost w. As a result, the value of T either does not change or decreases by w.

Suppose that a singular vertex is deleted from a cycle or a loop. Then, under the same passage, S does not change (if both segments adjacent to the vertex being deleted are even or the vertex is unique in the cycle) or increase by 1; the other values do not change.

If an interior singular vertex is deleted from a chain, i.e., there are other singular vertices on both sides of the vertex being removed, then the type of the chain does not change: it is determined by the size of the chain, by the type of the boundary (pendant or nonpendant), and (in the case of nonpendant boundary) by the length of the extreme segment. In this case, we repeat the argument from the preceding paragraph.

The other operations are treated in a similar way.

2. ALGORITHMS AND SKETCH OF THE PROOF OF THEOREM 1b

In $a + b$, all components are cycles or loops; the algorithms presented below are applied to $a + b$. We can discard all regluings except the double one. Suppose that the costs w of insertion and deletion are the same and the costs of the other operations equal 1. By a singular operation, together with its arguments, we mean an operation of merging two singular vertices into a single vertex. We denote the special double regluing by SDR and refer to it as the SDR-operation.

Suppose that $0 < w \le 1$. In this case, the algorithm and the proof are the same as above, and the cost $T(G)$ equals

$$(1 - w) \cdot (0.5d - c) + w \cdot (B + S).$$

Suppose that $1 < w \le 2$. By an a-cycle in the breakpoint graph we mean a cycle containing a-vertices and not containing b-vertices; the definition of a b-cycle is similar; an (a, b)-cycle contains both a- and b-vertices.

The algorithm is essentially close to that described above:

(i) excise ordinary edges;

(ii) SDR-unite the vertex of a loop with a singular vertex of any component having the same name;

(iii) delete loops;

(iv) SDR-divide an (a, b)-cycle of size strictly larger than 4 into a 2-cycle and a smaller cycle; in the latter, SDR-excise an ordinary edge;

(v) transform an (a, b)-cycle of size 4 into a cycle with one ordinary edge by applying an SDR-inversion and excise this edge;

(vi) SDR-unite two (a, b)-cycles in one cycle and SDR-cut out an ordinary edge;

(vii) delete singular vertices.

Let C_a and C_b be the numbers of a- and b-cycles, respectively, in the graph $a + b$.

The number of deletions of singular vertices in the algorithm equals $C_a + C_b + 2I_{ab} + I_{pa} + I_{pb}$; here, $I_{ab} = 1$, if $a + b$ contains an (a, b)-cycle (and $I_{ab} = 0$ otherwise), $I_{pa} = 1$ if $a + b$ contains an a-loop but does not contain a cycle with an a-vertex (and $I_{pa} = 0$ otherwise), and I_{pb} is a similar quantity for a b-loop. All operations, except those performed at step (i), are singular. Suppose that the number of singular operations in the algorithm equals B and the number of nonsingular operations equals S; then

$$\begin{aligned} T(G) &= w \cdot (C_a + C_b + 2I_{ab} + I_{pb}) \\ &+ (B + S - C_a - C_b - 2I_{ab} - I_{pa} - I_{pb}) \\ &= (w - 1) \cdot (C_a + C_b + 2I_{ab} + I_{pa} + I_{pb}) + B + S. \end{aligned}$$

We must search through all operations o applied to G.

Suppose that $w > 2$. Then we supplement the first six steps of the algorithm described above by the following steps:

(vii) SDR-unite an (a, b)-cycle and an a-cycle and excise any of the two ordinary edges; unite an (a, b)-cycle and a b-cycle in a similar way;

(viii) SDR-unite two a-cycle and excise any of the three ordinary edges; unite two b-cycles in a similar way;

(ix) delete the singular vertices.

The total number of operations in the algorithm equals $B + S + C_a - I_{ca} \cdot (1 - I_{ab}) + C_b - I_{cb} \cdot (1 - I_{ab}) = B + S + C_a + C_b - (I_{ca} + I_{cb}) \cdot (1 - I_{ab})$; here, $I_{ca} = 1$ ($I_{cb} = 1$) if $a + b$ contains an a- cycle (respectively, a b-cycle); otherwise, these quantities equal 0. These operations include $I_a + I_b$ deletion operations; here, $I_a = 1$ ($I_b = 1$) if $a + b$ contains an a-vertex (respectively, b-vertex); otherwise, these quantities equal 0. We have

$$\begin{aligned} T(G) &= w \cdot (I_a + I_b) + (B + S + C_a + C_b \\ &- (I_{ca} + I_{cb}) \cdot (1 - I_{ab}) - I_a - I_b) \\ &= (w - 1) \cdot (I_a + I_b) + B + S + C_a + C_b \\ &- (I_{ca} + I_{cb}) \cdot (1 - I_{ab}). \end{aligned}$$

Search through all operations.

REFERENCES

1. S. Yancopoulos, O. Attie, and R. Friedberg, Bioinformatics, No. 21, 3340–3346 (2005).
2. K. Yu. Gorbunov and V. A. Lyubetsky, in *CEUR Workshop Proceedings (CEUR-WS.Org): Selected Papers of the First International Scientific Conference Convergent Cognitive Information Technologies (Convergent 2016), Moscow, Russia, 2016* (Moscow, 2016), Vol. 1763, pp. 162–172.
3. M. D. V. Braga, E. Willing, and J. Stoye, J. Comput. Biol. **18**, 1167–1184 (2011).
4. P. H. da Silva, R. Machado, S. Dantas, and M. D. V. Braga, Algorithms Molec. Biol. **8**, 21.1–21.15 (2013).
5. M. Shao, Y. Lin, and B. Moret, in *Research in Computational Molecular Biology: 18th Annual International Conference RECOMB 2014, Pittsburgh, PA, USA, 2014* (Springer, New York, 2014), pp. 280–292.
6. F. V. Martinez, P. Feijão, M. D. V. Braga, and J. Stoye, Algorithms Molec. Biol. **10**, 21.1–21.15 (2015).
7. P. E. C. Compeau, Algorithms Molec. Biol. **8**, 6.1–6.9 (2013).
8. P. E. C. Compeau, in *Proceedings of 14th International Workshop "Algorithms in Bioinformatics," Wroclaw, Poland, 2014* (Wroclaw, 2014), Vol. 8701, pp. 38–51.
9. M. A. Alekseyev and P. A. Pevzner, Theor. Comput. Sci. **395** (2–3), 193–202 (2008).

Translated by O. Sipacheva