On a Problem of Choosing Elements in a Family of Sequences

Alexander Kel'manov[1,2], Ludmila Mikhailova[1], and Semyon Romanchenko[1]

[1] Sobolev Institute of Mathematics,
4 Koptyug Ave., 630090 Novosibirsk, Russia
[2] Novosibirsk State University,
2 Pirogova St., 630090 Novosibirsk, Russia
kelm@math.nsc.ru, mikh@math.nsc.ru, rsm@math.nsc.ru

Abstract. In the problem considered, it is required to minimize the sum of elements chosen in a family of finite numerical sequences with some constraints on the choice of elements. Namely, given a family of L nonnegative N-element sequences and a positive integer J, we need to minimize the sum of J intra-sums each of which includes only one element in every input sequence with all possible L-permutations of these sequences and under some constraints on the choice of elements to be included in the general double sum. The problem is related, for example, to the distant noise-prove monitoring of several moving objects with possible arbitrary displacements (permutations) of these objects. For this problem we present an exact polynomial-time algorithm with $\mathcal{O}(N^5)$ running time.

Keywords: Optimal summing · Finite numerical sequences · Permutations · Exact polynomial-time algorithm

1 Introduction

In the paper we study one problem of optimal summing of elements chosen in a family of finite numerical sequences. The subject of this research is algorithmic approximability of the problem. Our goal is to construct an efficient algorithm with the guaranteed accuracy for this problem.

The research is motivated by the absence of available algorithmic results for the problem under consideration and the importance of the problem, in particular, in the distant monitoring of some objects (see next Section). Moreover, the problem considered is a generalization of the Earlier Studied Optimization Problem (see Next Section). This is another reason for this problem to be studied.

The paper has the following structure. Section 2 contains the problem formulation, its interpretation and known results for the particular case of the problem.

In: S. Belim et al. (eds.): OPTA-SCL 2018, Omsk, Russia, published at http://ceur-ws.org

In the Same Section, We Announce Our Result Obtained. Section 3 contains two auxiliary problems and exact polynomial-time algorithms for them, which are necessary to justify the proposed algorithm. Finally, in Section 4, we present and justify our algorithm.

2 Problem Formulation and Related Problems

Everywhere below, $\mathbb{R}$ denotes the set of real numbers, $\mathbb{R}_{\geq}$ denotes the set of non-negative real numbers, $\mathbb{N}$ denotes the set of positive integer numbers, π denotes the permutation on L elements, $\mathit{\Pi}$ denotes the set of all possible permutations on L elements, π_j denotes the j-th permutation, and $\pi_j(i)$, $j = 1, \ldots, J$, $i = 1, \ldots, L$, denotes the i-th element of the j-th permutation.

We consider the following

Problem 1. Given a family $\mathcal{S} = \Big\{ \{s_1(n), \ldots, s_L(n)\} \Big| s_i(n) \in \mathbb{R}_{\geq}, i = 1, \ldots, L, n = 1, \ldots, N \Big\}$ of sequences and a positive integer J such that $JL \leq N$. *Find* an index tuple $(n_1, \ldots, n_{JL})$, where $n_k \in \{1, \ldots, N\}$, $k = 1, \ldots, JL$, and a tuple $(\pi_1, \ldots, \pi_J)$ of J permutations on L elements such that

$$S(n_1, \ldots, n_{JL}, \pi_1, \ldots, \pi_J) = \sum_{j=1}^{J} \sum_{i=1}^{L} s_{\pi_j(i)}(n_{(j-1)L+i}) \longrightarrow \min \qquad (1)$$

subject to constraints

$$1 \leq n_1 < n_2 < \ldots < n_{JL} \leq N\,. \qquad (2)$$

In the particular case of Problem 1, when $\pi_j(i) = i$ for all $j = 1, \ldots, J$, $i = 1, \ldots, L$, we need to find an index tuple $(n_1, \ldots, n_{JL})$ that minimizes the value of

$$\sum_{j=1}^{J} \sum_{i=1}^{L} s_i(n_{(j-1)L+i})$$

under the same constraints (2). In [3], an exact polynomial algorithm with running time $\mathcal{O}(JLN^2)$ was presented for this case. This algorithm implements a dynamic programming scheme.

In the optimization model inducing this particular case [3], each element $s_i(n)$, $n = 1, \ldots, N$, $i = 1, \ldots, L$, is a squared distance between an element $y_n \in \mathbb{R}^d$ in a given sample sequence $(y_1, \ldots, y_N)$ and a point $x_i \in \mathbb{R}^d$ in a given collection $\{x_1, \ldots, x_L\}$ of L points (patterns). In other words, the i-th sequence in the family $\mathcal{F}$ is the result of comparisons of the i-th point in the collection $\{x_1, \ldots, x_L\}$ with each element in the sample sequence $(y_1, \ldots, y_N)$.

In the applied problem, the point x_i corresponds to the i-th object and the collection $\{x_1, \ldots, x_L\}$ of points (patterns) corresponds to the collection of L objects, while the sample sequence $(y_1, \ldots, y_N)$ corresponds to a time series obtained as a monitoring (or measurement) result. In the family $\mathcal{F}$, the i-th

sequence corresponds to the result of comparisons of the i-th object in the pattern collection $\{x_1, \ldots, x_L\}$ with each element of time ordered measurement result. The applied problem can be interpreted (see [3]) as a noise-proof detection of the given pattern collection $\{x_1, \ldots, x_L\}$ of "pulses" repeated J times in the signal $(y_1, \ldots, y_N)$. In the applied problem, all the "pulses" in the collection have the same "duration" d and different forms, whereas in each collection repetition, there are no "pulses" permutations. The interval $n_m - n_{m-1}$, $m = 2, \ldots, JL$, between two consecutive "pulses" is not constant, but it is bounded in accordance with the inequalities (2).

Problem 1 is induced by a similar but more complex applied problem in which the "pulses" (objects) in each repeated pattern collection can be arbitrary reordered. This reordering causes the presence of permutations in the objective function (1). The permutations correspond to some possible rearrangements of objects.

In this paper, we present a polynomial algorithm which finds an optimal solution of Problem 1 in $\mathcal{O}(N^5)$ time.

3 Algorithm Foundation

To solve Problem 1 we need two auxiliary problems with exact polynomial algorithms. In addition, we need two well-known problems: (1) *Minimum weight perfect matching in a complete bipartite graph*, (2) *Nearest neighbor search problem*.

The first auxiliary problem has the following formulation.

Problem 2. Given a family $\mathcal{F} = \left\{ \{f_1(n), \ldots, f_L(n)\} \,\middle|\, f_i(n) \in \mathbb{R}_{\geq}, i = 1, \ldots, L, n = a, \ldots, b-1;\ L \leq b-a,\ a, b \in \mathbb{N} \right\}$ of sequences. *Find* an index tuple $(n_1, \ldots, n_L)$, where $n_i \in \{a, \ldots, b-1\}$, $i = 1, \ldots, L$, and a permutation $\pi \in \mathit{\Pi}$ such that

$$F(n_1, \ldots, n_L, \pi) = \sum_{i=1}^{L} f_{\pi(i)}(n_i) \to \min\,, \tag{3}$$

while the elements of $(n_1, \ldots, n_L)$ satisfy the following constraints

$$a \leq n_1 < \ldots < n_L < b\,.$$

Let $\mathcal{G} = (\mathcal{V} \cup \mathcal{U}, \mathcal{E}, w)$ be a complete bipartite graph with the vertex parts $\mathcal{V} = \{v_a, \ldots, v_{b-1}\}$ and $\mathcal{U} = \{u_1, \ldots, u_{b-a}\}$, and let the edge weights be given as

$$w_{v_n u_j} = \begin{cases} f_j(n), & j = 1, \ldots, L\,, \\ 0, & j = L+1, \ldots, b-a\,, \end{cases} \tag{4}$$

for every $n = a, \ldots, b-1$. In this graph, each vertex v_n, $n = a, \ldots, b-1$, in $\mathcal{V}$ corresponds to the index n in $\{a, \ldots, b-1\}$. The vertices $u_1, \ldots, u_L$ in $\mathcal{U}$ correspond to the indices in $\{1, \ldots, L\}$ of the sequences in the input family $\mathcal{F}$. The vertices $u_{L+1}, \ldots, u_{b-a}$ are auxiliary ones. The weight of every edge

between these auxiliary vertices in $\mathcal{U}$ and the vertices in $\mathcal{V}$ is equal to zero, while the weight of the edge between the vertex u_l, $l = 1, \ldots, L$, and the vertex v_n, $n = a, \ldots, b-1$, is equal to the value of the n-th element in the l-th sequence in the input family $\mathcal{F}$.

For every perfect matching $\mathcal{P}$ in $\mathcal{G}$ define a bijection $r : \{a, \ldots, b-1\} \longrightarrow \{1, \ldots, b-a\}$, where $\{a, \ldots, b-1\}$ and $\{1, \ldots, b-a\}$ are the sets of indices of $\mathcal{U}$ and $\mathcal{V}$ respectively. Let $W(\mathcal{P})$ be the weight of the perfect matching $\mathcal{P} = \{(v_n, u_{r(n)}),\ n = a, \ldots, b-1\}$. The following lemma is true.

Lemma 1. *Let $\mathcal{P}^* = \{(v_n, u_{r^*(n)}), n = a, \ldots, b-1\}$ be the minimum weight perfect matching in $\mathcal{G}$, where r^* is the optimal bijection. Then in Problem 2, the optimal tuple $\eta^*(a,b) = (n_1^*, \ldots, n_L^*)$, and the optimal permutation $\pi^*(a,b)$ can be found by the following formulas:*

$$n_1^* = \min\{n \mid r^*(n) \in \{1, \ldots, L\}, n \in \{a, \ldots, b-1\}\}\,, \tag{5}$$

$$n_i^* = \min\{n \mid r^*(n) \in \{1, \ldots, L\},\ n \in \{n_{i-1}^* + 1, \ldots, b-1\}\},\ i = 2, \ldots, L\,, \tag{6}$$

and

$$\pi^*(a,b) = (r^*(n_1^*), \ldots, r^*(n_L^*))\,. \tag{7}$$

The optimal value $F^*(a,b)$ of the objective function (3) *can be found as*

$$F^*(a,b) = W(\mathcal{P}^*)\,. \tag{8}$$

To prove this lemma we establish the correspondence between an admissible tuple and a permutation in Problem 2 and the perfect matching in the graph $\mathcal{G}$. Then we prove that the optimal value of the objective function (3) is equal to the weight of the minimum weight perfect matching in $\mathcal{G}$.

Note that the minimum weight perfect matching in $\mathcal{G}$ can be found by a well-known algorithm (see, for example [4]) in the polynomial time.

Thus, we have the following algorithm for auxiliary Problem 2 in a step-by-step form.

Algorithm $\mathcal{A}_1$.

INPUT: a family $\mathcal{F}$ of sequences.

Step 1. Construct $\mathcal{G}$ using (4), and find the minimum weight perfect matching $\mathcal{P}^*$ in $\mathcal{G}$ using the well-known algorithm [4].

Step 2. Find the optimal permutation $\pi^*(a,b)$ and the optimal tuple $\eta^*(a,b)$ by (5), (6), and (7).

Step 3. Compute $W(\mathcal{P}^*)$ and $F^*(a,b)$ by (8).

OUTPUT: the permutation $\pi^*(a,b)$, the tuple $\eta^*(a,b) = (n_1^*, \ldots, n_L^*)$, and the value $F^*(a,b)$.

The following proposition is true.

Proposition 1. *Algorithm $\mathcal{A}_1$ finds the optimal solution of Problem 2 in $\mathcal{O}((b-a)^3)$ time.*

The proof follows immediately from Lemma 1 and well-known time complexity of the algorithm for the problem of searching the minimum weight perfect matching [4].

The second auxiliary problem has the following formulation.

Problem 3. Given positive integers L, J, and N such that $JL \le N$, a family $\mathcal{Q} = \{q(n,m) \in \mathbb{R}_{\ge} \mid 1 \le n < m \le N+1;\ L \le m-n \le N-(J-1)L,\ n,m \in \mathbb{N}\}$. *Find* an index tuple $(\mu_1, \ldots, \mu_{J+1})$, where $\mu_j \in \{1, \ldots, N+1\}$, $j = 1, \ldots, J+1$, such that

$$Q(\mu_1, \ldots, \mu_{J+1}) = \sum_{j=1}^{J} q(\mu_j, \mu_{j+1}) \to \min, \tag{9}$$

under the following constraints

$$1 = \mu_1 < \ldots < \mu_{J+1} = N+1\,,$$

$$\mu_{j+1} - \mu_j \ge L, j = 1, \ldots, J\,.$$

To solve this auxiliary problem we use an algorithm for the well-known

Nearest neighbor search problem(see, for example, [2] and [1]). *Given* a family $\mathcal{H} = \{h(n,m) \in \mathbb{R}_{\ge} | \, 1 \le n \le m \le N+1;\ n,m \in \mathbb{N}\}$ and a positive integer J. *Find* an index tuple $(\nu_1, \ldots, \nu_{J+1})$, where $\nu_j \in \{1, \ldots, N+1\}$, $j = 1, \ldots, J+1$, such that

$$H(\nu_1, \ldots, \nu_{J+1}) = \sum_{j=1}^{J} h(\nu_j, \nu_{j+1}) \to \min\,, \tag{10}$$

subject to constraints

$$1 = \nu_1 \le \nu_2 \le \ldots \le \nu_{J+1} = N+1\,.$$

As it is known [1], the *Nearest neighbor search* problem can be solved in $\mathcal{O}(JN^2)$ time.

The following lemma is true.

Lemma 2. *Let* $(\nu_1^*, \ldots, \nu_{J+1}^*)$ *be the optimal solution of Nearest neighbor search problem with*

$$h(n,m) = \begin{cases} q(n,m), & n = 1, \ldots, N-L+1, \\ & \quad m = n+L, \ldots, \min\{N+1, n+N-(J-1)L\}\,, \\ +\infty, & n = 1, \ldots, N+1, m = n, \ldots, \min\{n+L-1, N+1\}\,, \\ +\infty, & n = 1, \ldots, 1+(J-1)L, \\ & \quad m = n+N-(J-1)L+1, \ldots, N+1\,, \end{cases} \tag{11}$$

and H^* *be the optimal value of objective function* (10). *Then the components of the optimal tuple* $(\mu_1^*, \ldots, \mu_{J+1}^*)$ *and the optimal value* Q^* *of the function* (9) *can be found by the following formulas*

$$\mu_i^* = \nu_i^*,\ \ i = 1, \ldots, J+1\,, \tag{12}$$

$$Q^* = H^*\,. \tag{13}$$

To prove this lemma we first verify that the optimal tuple in the *Nearest neighbor search* problem is an admissible one in Problem 3. Then we show that this tuple is the optimal solution of Problem 3.

Thus, we have the following algorithm for auxiliary Problem 3.

Algorithm $\mathcal{A}_2$.

INPUT: positive integers L, J, and a family $\mathcal{Q}$.

STEP 1. Find the family $\mathcal{H}$ by (11). Compute the optimal value H^* and find the optimal tuple $(\nu_1^*, \ldots, \nu_{J+1}^*)$ of the *Nearest neighbor search* problem.

STEP 2. Compute the optimal value Q^* of the objective function of Problem 3 using (13), and find the optimal tuple $(\mu_1^*, \ldots, \mu_{J+1}^*)$ of indices using (12).

OUTPUT: the tuple $(\mu_1^*, \ldots, \mu_{J+1}^*)$ and the value Q^*.

The following proposition is true.

Proposition 2. *Algorithm $\mathcal{A}_2$ finds an optimal solution of Problem 3 in $\mathcal{O}(JN^2)$ time.*

The validity of Proposition 2 follows from Lemma 2 and time complexity of the algorithm for the *Nearest neighbor search* problem [2].

4 Exact Algorithm

Before presenting our algorithm, let us formulate the following lemma.

Lemma 3. *Let N, J, and L be the instances of Problem 1. Let $\eta^*(a,b)$, $\pi^*(a,b)$ be the optimal solution of Problem 2 for each $a = 1, \ldots, N-L+1$, and $b = a+L, \ldots, \min\{N+1,\, a+N-(J-1)L\}$ with*

$$f_i(j) = s_i(j), \quad j = a, \ldots, b-1, \quad i = 1, \ldots, L\,,$$

where $s_i(j)$ is the sequence in the input family $\mathcal{F}$ of Problem 1, and $F^*(a,b)$ be the optimal value of the objective function (3).

Let $(\mu_1^*, \ldots, \mu_{J+1}^*)$ be the optimal solution of Problem 3 with

$$q(n,m) = F^*(n,m),\ n = 1, \ldots, N-L+1,\quad m = n+L, \ldots, \min\{N+1, n+N-(J-1)L\}\,, \quad (14)$$

in the family $\mathcal{Q}$, and Q^* be the optimal value of the function (9).

Then the optimal tuples $(n_1^*, \ldots, n_{JL}^*)$ and $(\pi_1^*, \ldots, \pi_L^*)$ of Problem 1 can be found by the following formulas:

$$(n_{(j-1)L+1}^*, \ldots, n_{jL}^*) = \eta^*(\mu_j^*, \mu_{j+1}^*),\ \ j = 1, \ldots, J\,, \quad (15)$$

$$\pi_j^* = \pi^*(\mu_j^*, \mu_{j+1}^*),\ \ j = 1, \ldots, J\,, \quad (16)$$

and the optimal value S^* of the function (1) *can be found as*

$$S^* = Q^*\,. \quad (17)$$

We prove Lemma 3 using the results of Section 3.

Finally, we can present the algorithm for Problem 1.

Algorithm $\mathcal{A}$.

INPUT: a family $\mathcal{S}$ of sequences and a positive integer J.

Step 1 (solving the family of Problems 2). For each $a = 1, \ldots, N-L+1$, and $b = a+L, \ldots, \min\{N+1,\, a+N-(J-1)L\}$, put $f_i(j) = s_i(j)$, $j = a, \ldots, b-1$, $i = 1, \ldots, L$. Using Algorithm $\mathcal{A}_1$, find the optimal solution $\pi^*(a,b)$, $\eta^*(a,b)$ and compute the optimal value $F^*(a,b)$ of the objective function (3).

Step 2 (solving Problem 3). Find the family $\mathcal{Q}$ in accordance with (14). Find the optimal solution $(\mu_1^*, \ldots, \mu_{J+1}^*)$ and compute the optimal value Q^* of the objective function (9) using Algorithm $\mathcal{A}_2$.

Step 3. Compute the value S^* of the objective function (1) by (17). Construct $(n_1^*, \ldots, n_{JL}^*)$ and $(\pi_1^*, \ldots, \pi_L^*)$ using (15) and (16).

OUTPUT: the tuples $(n_1^*, \ldots, n_{JL}^*)$, $(\pi_1^*, \ldots, \pi_L^*)$ and the value S^*.

The following theorem is true.

Theorem 1. *Algorithm $\mathcal{A}$ finds an optimal solution of Problem 1 in $\mathcal{O}(N^5)$ time.*

The optimality of the solution follows from Proposition 1, Proposition 2, and Lemma 3. The total estimate of the running time follows from the inequality $LJ \leq N$ an from the time complexity of the three algorithmic steps. Namely, Step 1, Step 2, and Step 3 require $\mathcal{O}((N-L+1)^2(N-(J-1)L)^3)$, $\mathcal{O}(JN^2)$, and $\mathcal{O}(JL)$ running times respectively.

Remark 1. If J and L are bounded by some constants, then the time complexity of Algorithm $\mathcal{A}$ is $\mathcal{O}(N^5)$. In a particular case, when $L = N$ we have $J = 1$ and the running rime the algorithm is $\mathcal{O}(N^3)$.

5 Conclusion

In the paper we show that one optimization problem arising in applications connected with the analysis of time series is solvable in a polynomial time. In our opinion, the main line of the further research on the problem is to construct faster approximation polynomial-time algorithm with a guaranteed accuracy. In connection with the known Big data problem, a linear-time or a sublinear-time approximation algorithm for the considered problem is of specific interest.

Acknowledgement. The study presented in Sections 3 was supported by the Russian Science Foundation, project 16-11-10041. The study presented in Sections 2, 4 was supported by the Russian Foundation for Basic Research, projects 16-07-00168, and by the Russian Academy of Science (the Program of Basic Research), project 0314-2016-0015, and by the Russian Ministry of Science and Education under the 5-100 Excellence Programme.

References

1. Bellman, R.: Dynamic Programming. Princeton University Press (1957)
2. Beresnev, V.L., Gimadi, E.Kh., Dement'ev, V.T.: Extremal Standartization Problems. Nauka, Novosibirsk (1978), (in Russian)
3. Kel'manov, A.V., Mikhailova, L.V., Khamidullin, S.A.: A posteriori joint detection of a recurring tuple of reference fragments in a quasi-periodic sequence. Comp. Math. and Math. Phys. 48(12), 2276–2288 (2008)
4. Papadimitrou, C., Steiglitz, K.: Combinatorial Optimization: Algorithms and Complexity. Prentice-Hall (1982)