algorithms

MDPI

Article

The Extraction of Maximal-Sum Principal Submatrix and Its Applications

Yizheng Zhang, Liuhong Luo * and Hongjun Li *

College of Science, Beijing Forestry University, Beijing 100083, China; yizhang082022@163.com
* Correspondence: llh7667@bjfu.edu.cn (L.L.); lihongjun69@bjfu.edu.cn (H.L.)

Abstract: Extracting k-order maximal-sum principal submatrix from an n-order real matrix is a typical combinatorial optimization problem and an NP-hard problem. To improve the computational efficiency of solving this problem, we, in this paper, propose an accelerated algorithm with row-by-row updates, called the fusion row update accelerated algorithm, which works by reducing the number of addition operations for submatrix elements. The new algorithm is applied to accelerate color combination selection and maximize color difference, which improves the readability of data visualization results; it is also applied to accelerate stock investment portfolio selection and minimize correlation degree, which decreases the investment risk in the view of daily return volatility.

Keywords: maximal-sum principal submatrix extraction; maximum subarrays; algorithm; principal submatrix; color combination selection; stock investment portfolio selection

Citation: Zhang, Y.; Luo, L.; Li, H. The Extraction of Maximal-Sum Principal Submatrix and Its Applications. *Algorithms* **2023**, *16*, 314. https://doi.org/10.3390/a16070314

Academic Editor: Frank Werner

Received: 26 May 2023
Revised: 23 June 2023
Accepted: 24 June 2023
Published: 26 June 2023

1. Introduction

The theory, application, and algorithm on maximal-sum submatrix (MSS) have received a lot of attention recently [1,2]. The MSS problem, also known as the maximum subarray problem [3,4], is to find the submatrix that maximizes the sum of matrix elements in it. The solutions to many application problems, such as solving the rectilinear picture compression [5] and finding a subset of genes that is relatively highly expressed among a subset of patients [1], can be converted to finding the maximal-sum submatrix from a matrix. The rows and columns in a matrix similar to the above two examples represent two different variables with different meanings.

However, there is also a special kind of real matrix from practice applications, whose rows and columns represent the same variable, such as correlation coefficient matrix, covariance matrix, and adjacency matrix. One of these specific cases is selecting the most informative principal submatrix of a pre-specified size from a covariance matrix [6] or the maximum edge-weighted subgraph from a weighted undirected graph [7]. It is necessary to choose the same k rows and k columns for the maximum sum submatrix to have practical significance. Therefore, finding the k-order MSS is transformed into finding the k-order maximal-sum principal submatrix (MSPS).

Although the issue of finding the MSPS is much closer to the MSS problem, there are still many differences between the two issues, including:

(1) The integer programming models describing the two problems are different;
(2) The formats of their solutions to the problems are different;
(3) The purposes of the application problems are different.

Therefore, the algorithm for solving the MSPS issue differs from that for solving the MSS problem.

In fact, extracting the MSPS from a real matrix is a typical combinatorial optimization problem and an NP-hard problem [1]. The challenge lies in the high computational complexity of finding an accurate solution. The total number of searches for the k-order MSPS

from an n-order real matrix A is $C_n^k = (n-k)!/(k!)$. When n and k are both relatively large, the computational cost is very large.

To improve the computational efficiency, there are two ways that can be taken into account. One is to reduce the number of comparisons of two submatrices; the other is to reduce the number of addition operations for submatrix elements. We, in this paper, construct a fast algorithm to extract the maximal-sum principal submatrix from a real matrix, called the fusion row update accelerated algorithm, and apply this algorithm to the color combination selection problem to maximize color difference and the stock investment portfolio selection to minimize correlation degree, which decreases the investment risk in the view of the volatility of daily returns.

The remainder of the paper is organized as follows. Section 2 briefly surveys the related works. Section 3 describes the problem and derives related properties. Section 4 develops the algorithm for finding the MSPS. Section 5 conducts several numerical experiments to demonstrate the efficiency and the solution quality of our proposed algorithm. Section 6 presents two applications. Section 7 concludes the paper and lists the limitations of the proposed algorithm.

2. Related Work

We review the relevant literature on three aspects: best submatrix extraction, color combination selection and stock investment portfolio construction.

2.1. Best Submatrix Extraction

There are many applications related to extracting the best $k \times k$ submatrix from a matrix A. Due to different purposes of application, the definition of the term "best" also varies. The best submatrix means that it has the maximum volume [8], the largest logarithm of the determinant or the minimal trace of the inverse of the selected principal submatrix [6], the max-algebraic permanent [9], etc.

In many cases, the best submatrix means to find the submatrix that has the maximum sum. The maximal-sum submatrix (MSS) can be a principal submatrix or not a principal submatrix [10,11]. The extracted submatrix can be a rectangle [5,12], which means that both the extracted rows and columns are continuous. The max-sum submatrix can also be defined as a rectangular submatrix, in which both the rows and the columns are non-necessarily contiguous [1,2].

The problems discussed in the existing literature mentioned above are both related and different from the problem we want to solve in this paper, but none of them is completely consistent with the problem we are solving. Similarities and differences of these problems between the mentioned literature and ours are listed in Table 1. Compared to the best $k \times k$ submatrix extraction, the submatrix to be extracted is a square matrix, and the calculated submatrix features are different. The characteristics of existing methods for calculation are the maximum volume [8], the largest logarithm of the determinant and the minimal trace of the inverse of the selected principal submatrix, but we focus on the sum of all elements of the submatrix.

Table 1. Similarities and differences of the problems between the mentioned literature and ours.

Ref.	Object	Connectivity	Principal Submatrix or Not	Submatrix Order	Accurate or Approximate Solution
[1]	The sum of the entries of submatrix	Discrete	N	Unfixed	Accurate
[2]	The sum of the entries of submatrix	Discrete	N	Unfixed	Approximate
[3–5]	The sum of the entries of submatrix	Consecutive	N	Unfixed	Accurate
[6]	The determinant of submatrix $^{+}$	Discrete	Y	Fixed	Approximate
[7]	The sum of the entries of submatrix $^{++}$	Discrete	Y	Fixed	Approximate
[8]	The maximum volume submatrix	Discrete	Y	Fixed	Approximate
Ours	The sum of the entries of submatrix	Discrete	Y	Fixed	Accurate

$^{+}$ This is equivalent to the largest logarithm of the determinant of submatrix in [6]; $^{++}$ The maximum edge-weighted subgraph problem in [7] is equivalent to the problem of maximal sum of the entries of submatrix.

Compared to the MSS extraction, the commonality of all methods is to find the optimal solution based on the sum of all elements of the submatrix, but most methods do not specify the size of the submatrix, and we, in this paper, focus on the k-order principal submatrix.

2.2. Color Combination Selection

There are many applications related to the k-order maximal-sum principal submatrix (MSPS). One of them is color combination selection. Constructing an optimal color combination is a critical factor for data visualization [13,14], and a lot of literature paid their attention to it [15–19]. A method to improve the readability of the data after clustering and visualizing is to consider the degree of color difference, which can be measured by the distance of two colors [20].

Our MSPS algorithm can be applied to automatically construct the optimal color combination to improve the readability of visualized data. The optimal color combination here is defined by the largest color difference among all selected colors. The issue that extracts k colors from n different colors to form a color combination with maximum color difference, corresponds to the problem that extracts the k-order maximal-sum principal submatrix from an n-order real symmetric matrix.

2.3. Stock Investment Portfolio Construction

Another application related to the k-order MSPS is the stock portfolio selection. Considering that the rise and fall of stocks are related to uncertain factors such as market behavior, company operations and national policies [21], diversified investment is often adopted to decrease the daily volatility [22]. The daily volatility as the investment risk can be measured by variance or standard deviation of the return; therefore, portfolio selection is proposed in order to maximize the expected return and minimize the risk [23,24]. For this goal, the stock portfolio can be constructed according to the minimum covariance matrix [21] and minimum correlation coefficient matrix of stock returns [25].

Most existing methods described above focus on the investment ratio of each stock in the selected k stocks as a portfolio. We, in this paper, pay attention to the construction of a portfolio that consists of k stocks from n specified stocks, which can be used as a preliminary step for investment ratio assignment issues.

3. Problem Description and Related Property

Different from those best principal submatrix selection problems reviewed in previous sections, our k-order maximal-sum principal submatrix (MSPS) is defined by the maximal sum of all elements in the principal submatrix. The relevant concepts and property related to the proposed MSPS extraction algorithm are summarized as follows.

3.1. Problem Description

The notations in this paper are defined as follows. Vectors are represented as bold lowercase characters $\boldsymbol{x}, \boldsymbol{y}$, etc. Matrices are represented as uppercase characters A, B, etc. Elements in vectors and matrices, as well as scalars in general, are represented as normal italic characters, x_i, m_{ij}, i, etc.

Let an n-order real matrix $A = \left(a_{ij}\right)_{n\times n} (\forall i, j \in L)$, where $L = \{1, 2, 3, \cdots, n\}$ is a sequence number set and $|\cdot|$ is the cardinal number of a set.

Definition 1. *Let $I \subseteq L, |I| = k$; the submatrix $A_I^{(k)}$ consisting of the cross elements of I rows and I columns contained in the real matrix A is called the k-order principal submatrix of matrix A.*

Definition 2. *Among all k-order principal submatrices of A, the maximal-sum principal submatrix (MSPS) $A_{I^*}^{(k)}$ maximizes the sum of the elements.*

The problem of finding the MSPS of matrix A can be formulized as an optimization problem:

$$A_{I^*}^{(k)} = arg\max_{I \subseteq \mathrm{L}} \left\| A_I^{(k)} \right\|_S \tag{1}$$

where the sum of matrix elements is denoted by $\|\cdot\|_S$, i.e.,

$$\|A\|_S = \sum_{i,j \in L} a_{ij} \tag{2}$$

Let I^* be the optimal index set and $A_{\mathrm{I}^*}^{(k)}$ be the optimal solution. The optimal value is S^* as

$$S^* = \left\| A_{I^*}^{(k)} \right\|_S = \max_{I \subseteq L} \left\| A_I^{(k)} \right\|_S = \sum_{i,j \in I^*} a_{ij} \tag{3}$$

It is easy to know that matrix A and its transposed matrix A^T have the same MSPS and the optimal index set, as well as the optimal solution. If a positive integer k is specified, both the optimal index set and the optimal solution perhaps are not unique, but the optimal value is unique.

Example 1. Let A *be a real matrix,* $A = \begin{bmatrix} 2 & -1 & 3 & 0 \\ -1 & 0 & 4 & 2 \\ 3 & 4 & 2 & 2 \\ 1 & 2 & 2 & -1 \end{bmatrix}$. *When* $k = 2$, *there are two optimal solutions* $A_{\{1,3\}}^{(2)} = \begin{bmatrix} 2 & 3 \\ 3 & 2 \end{bmatrix}$ *and* $A_{\{2,3\}}^{(2)} = \begin{bmatrix} 0 & 4 \\ 4 & 2 \end{bmatrix}$, *the corresponding optimal index sets are respectively* $I^* = \{1,3\}$ *and* $I^* = \{2,3\}$, *and the optimal value is* $S^* = 10$.

Example 2. *Matrix* A *is a non-negative symmetric matrix of* m*-order,*

$$A = \begin{bmatrix} m & m-1 & m-2 & \cdots & 1 \\ m-1 & m-1 & m-2 & \cdots & 1 \\ m-2 & m-2 & m-2 & \cdots & 1 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & 1 & 1 & \cdots & 1 \end{bmatrix}.$$

For the matrix, if k *is specified, the MSPS of* k*-order is*

$$A_{I^*}^{(k)} = \begin{bmatrix} m & m-1 & \cdots & m-k+1 \\ m-1 & m-1 & \cdots & m-k+1 \\ \vdots & \vdots & \ddots & \vdots \\ m-k+1 & m-k+1 & m-k+1 & m-k+1 \end{bmatrix}.$$

The optimal index set $I^* = \{1, 2, \cdots, k\}$, *and the optimal value is*

$$S^* = k^2 m - \sum_{i=1}^{k} (k-i)[2(k-i)+1] = k^2 m - \frac{k(k-1)(4k+1)}{6}.$$

Remark. *The matrix* A *in this example is related to the optimal layout of the stacked graph for visualizing multi-dimensional time series data [13].*

3.2. MSPS Problem Optimization Model

The MSPS problem of a real matrix A can be expressed as a 0–1 integer programming problem as follows:

$$\begin{cases} \max S = x^T A x \\ s.t. x_1 + x_2 + \cdots + x_n = k, \\ x_i \in \{0,1\} \end{cases} \tag{4}$$

where $\boldsymbol{x} = (x_1, x_2, \cdots, x_n), x_i \in \{0,1\}$. The components with an element value of 1 correspond to the optimal indicator set I^*. The objective function value S is the sum of the elements of the MSPS, i.e., $\left\| A_{I^*}^{(k)} \right\|_S$.

If n and k are small, the branch and bound method can be employed to solve the integer programming problem (4). However, in the era of big data, n and k in practice are often very large. Therefore, it is necessary to design some efficient and accurate algorithms to solve this issue.

3.3. Property

We observed that when traversing each principal submatrix, two adjacent principal submatrices only differ by one row and one column. Using this feature, a formula for accelerating the calculation of the sum of matrix elements can be derived.

Property 1. *Let two k order matrices*

$$A[i_1, i_2, \cdots, i_k] = \begin{bmatrix} a_{i_1,i_1} & \cdots & a_{i_1,i_{k-1}} & a_{i_1,i_k} \\ \vdots & \ddots & \vdots & \vdots \\ a_{i_{k-1},i_1} & \cdots & a_{i_{k-1},i_{k-1}} & a_{i_{k-1},i_k} \\ a_{i_k,i_1} & \cdots & a_{i_k,i_{k-1}} & a_{i_k,i_k} \end{bmatrix},$$

and

$$A[i_1, i_2, \cdots, i_{k-1}, i_{k+1}] = \begin{bmatrix} a_{i_1,i_1} & \cdots & a_{i_1,i_{k-1}} & a_{i_1,i_{k+1}} \\ \vdots & \ddots & \vdots & \vdots \\ a_{i_{k-1},i_1} & \cdots & a_{i_{k-1},i_{k-1}} & a_{i_{k-1},i_{k+1}} \\ a_{i_{k+1},i_1} & \cdots & a_{i_{k+1},i_{k-1}} & a_{i_{k+1},i_{k+1}} \end{bmatrix}$$

have the same $k-1$ order principal submatrix in the upper left corner, then the sum of the elements of two matrices has the following relationship,

$$S_n = S_o + \Delta_{ru}. \tag{5}$$

In Equation (5), $S_n = \|A[i_1, i_2, \cdots, i_{k-1}, i_{k+1}]\|_S$ is the sum of elements in the next k-order matrix; $S_o = \|A[i_1, i_2, \cdots, i_k]\|_S$ is the sum of elements in the previous k-order matrix;

$$\Delta_{ru} = -\alpha_{i_k,\cdot} - \alpha_{\cdot,i_k,\cdot} + a_{i_k,i_k} + \alpha_{i_{k+1},\cdot} + \alpha_{\cdot,i_{k+1}} - a_{i_{k+1},i_{k+1}}, \tag{6}$$

where $\alpha_{i_k,\cdot} = \sum_{j=1}^{k} a_{i_k,i_j}$ is the sum of all elements in the row i_k in $A[i_1, i_2, \cdots, i_k]$ and $\alpha_{\cdot,i_k} = \sum_{j=1}^{k} a_{i_j,i_k}$ is the sum of all elements in the column i_k in $A[i_1, i_2, \cdots, i_k]$.

It can be seen that both $A[i_1, i_2, \cdots, i_{k-1}, i_k]$ and $A[i_1, i_2, \cdots, i_{k-1}, i_{k+1}]$ have the same $(k-1)$-order submatrix $A[i_1, i_2, \cdots, i_{k-1}]$. Then, applying $\|A[i_1, i_2, \cdots, i_{k-1}]\|_S$ to the calculation of the sum of elements in the next matrix $A[i_1, i_2, \cdots, i_{k-1}, i_{k+1}]$, we can avoid repeated addition operations and achieve the goal of improving efficiency.

On the right side of Equation (5), the crossing element a_{i_k,i_k} appears simultaneously in row i_k and column i_k, so it must be added back once. Due to the same reason, the crossing element $a_{i_{k+1},i_{k+1}}$ has been added twice, then it must be subtracted to obtain the final result S_n.

Obviously, summing all elements one by one to calculate S_n requires $\left(k^2-1\right)$ addition operations. However, if using the proposed Formula (5) to calculate S_n, the number of times to add or subtract is $4k-4$.

When $k \geq 4$,

$$k^2 - 1 > 4k - 4. \tag{7}$$

The inequality (7) means that when k is greater than 4, using Formula (5) to extract $\left\|A_{I^*}^{(k)}\right\|_S$ can reduce the number of additions.

4. The Proposed Algorithm

Integrating Property 1 as a key step, a new method to extract the maximal-sum principal submatrix using row update algorithm (MSPS-RU) is proposed. The algorithm flowchart is shown in Figure 1.

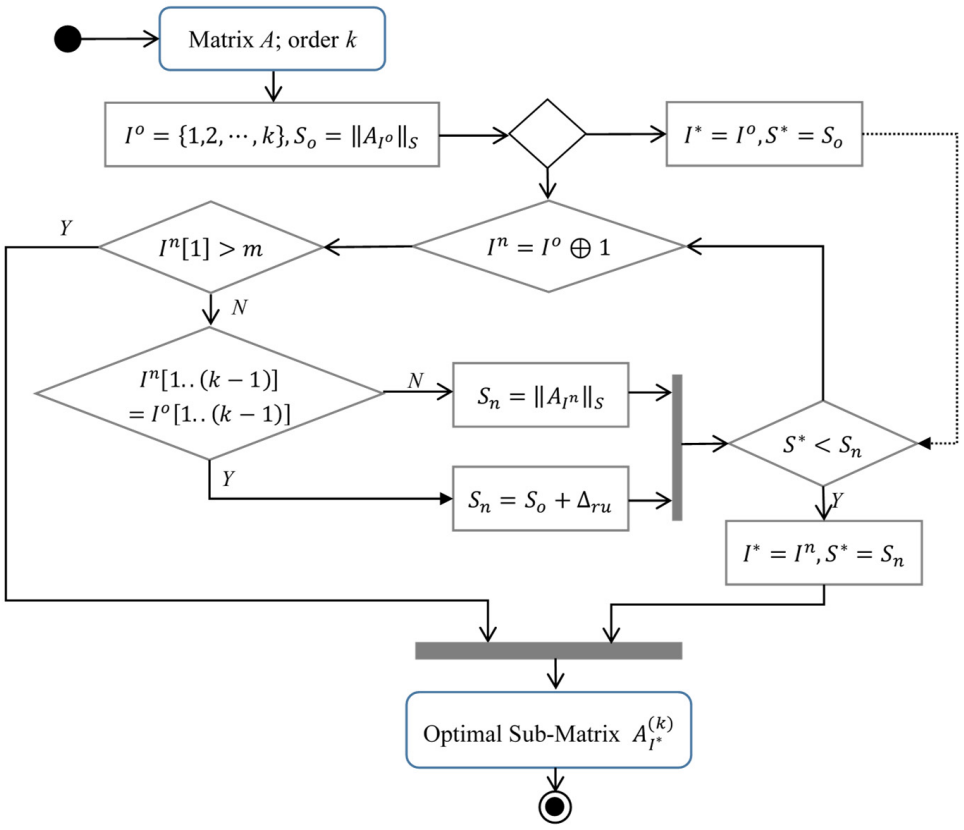


Figure 1. Flowchart of MSPS-RU algorithm.

In Figure 1, the input is an n-order real matrix and the order k of the maximal-sum principal submatrix we want to extract. According to the increasing order of columns in $I = \{i_1, i_2, \cdots, i_k\}$, all principal submatrices are traversed. Therefore, the condition for the algorithm to terminate is that all principal submatrices have been compared.

If the two matrices to be compared only have one row and one column different, Formula (5) is employed to accelerate the calculation of the sum of matrix elements. Otherwise, all elements should be added one by one using the naive enumeration method. The output is the optimal solution. In the entire algorithm, the reduction in addition is approximately $[k(n-k)]^2/2$.

The pseudocode of the algorithm (Algorithm 1) is described as follows.

Algorithm 1: MSPS-RU algorithm

Input: A; //an n-order real matrix A
k; //the order of the MSPS to be extracted
Output: I^*; // the optimal index set of the MSPS
1 Let $I^o = \{i_1, i_2, \cdots, i_k\} = \{1, 2, \cdots, k\}$;
2 build initial principal submatrix $A_o[i_1, i_2, \cdots, i_k]$, $I^* = \{i_1, i_2, \cdots, i_k\}$;
3 calculate the sum S_o of all elements in A_o, $S^* = S_o$;
4 while $(i_1 \leq n - k + 1)$
5 $i_k = i_k + 1; I^n = \{i_1, i_2, \cdots, i_{k-1}, i_k\}$;
6 build next principal submatrix $A_n[i_1, i_2, \cdots, i_k]$;
7 if $I^n[1..(k-1)] = I^o[1..(k-1)]$
8 $S_n = S_o + \Delta_{ru}$; // Formula (5)
9 else
10 $S_n = \|A_n\|_S = \sum_{i,j \in I^n} a_{ij}$;// Formula (2)
11 endif
12 if $S_n > S^*$
13 $S^* = S_n$; $I^* = \{i_1, i_2, \cdots, i_k\}$;
14 endif
15 $A_o = A_n$, $I^o = I^n$;
16 end while
17 **return** I^*, S^*;

5. Experiment

To evaluate performance of the proposed MSPS-RU algorithm, a naive enumeration algorithm (denoted as Enum) is used as the baseline. The difference between Enum and the MSPS-RU algorithm is that Enum does not have an accelerated summation step, and its pseudocode (Algorithm 2) has been described as follows.

Algorithm 2: Enum algorithm

Input: A; //an n-order real matrix A
k; //the order of the MSPS to be extracted
Output: I^*; // the optimal index set of the MSPS
1 Let $I^o = \{i_1, i_2, \cdots, i_k\} = \{1, 2, \cdots, k\}$;
2 build initial principal submatrix $A_o[i_1, i_2, \cdots, i_k]$, $I^* = \{i_1, i_2, \cdots, i_k\}$;
3 calculate the sum S_o of all elements in A_o, $S^* = S_o$;
4 while $(i_1 \leq n - k + 1)$
5 $i_k = i_k + 1$; $I^n = \{i_1, i_2, \cdots, i_{k-1}, i_k\}$;
6 build next principal submatrix $A_n[i_1, i_2, \cdots, i_k]$;
7 $S_n = \|A_n\|_S = \sum_{i,j \in I^n} a_{ij}$;// Formula (2)
8 if $S_n > S^*$
9 $S^* = S_n$; $I^* = \{i_1, i_2, \cdots, i_k\}$;
10 endif
11 $A_o = A_n$, $I^o = I^n$;
12 end while
13 **return** I^*, S^*;

Formula (8) gives the change in time complexity, that is, the number of addition operations is reduced from $O(k^2)$ of the Enum method to $O(k)$ of the proposed MSPS-RU method in each sum of submatrix elements.

We use several experiments to discuss the efficiency of our MSPS-RU algorithm and compare it with the Enum method. The degree of improvement in efficiency adopts a relative index η calculated by

$$\eta = \frac{T_{Enum} - T_{ours}}{T_{ours}} \times 100\%, \tag{8}$$

where T_{ours} and T_{Enum} are the average time taken to solve the issue using the MSPS-RU algorithm and the Enum method, respectively.

All algorithms have been implemented in the C++ programming language. Each run is executed with a single thread on a personal laptop, and the computer configuration is Intel (R) Core (TM) i5-6200U CPU @ 2.30 GHz 2.40 GHz processor and 4GB RAM per run. The runtime under each parameter is the average experiment time (unit: seconds) of 20 repeated experiments. Without loss of generality, considering the computer hardware conditions and time cost, the highest order of the real matrix is no more than 60 in the following experiments.

5.1. Parameter k Is Fixed and Parameter n Is Variable

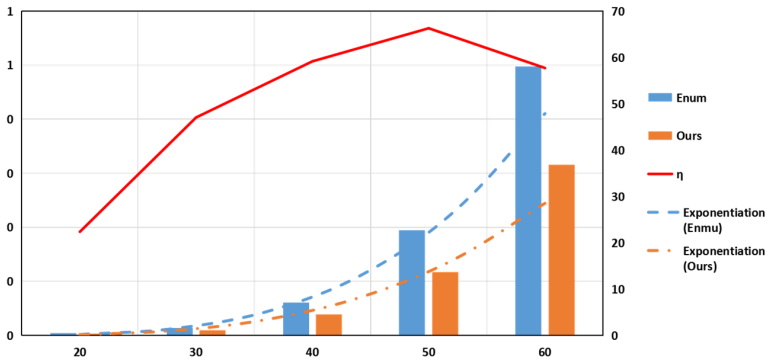
To explore the effect of parameter n on the calculation time of the MSPS-RU algorithm, the following comparative experiments were conducted.

The n-order real matrix is randomly generated using uniform distribution, and n varies from 20 to 60 with constant step size 10. Let $k = 5$ and 10, which means extracting the 5-order and 10-order MSPS of the n-order real matrix. The runtime (unit: second) of the Enum algorithm and the MSPS-RU algorithm are compared, and the average runtime of the two algorithms is recorded, as shown in Table 2. From the table, the average runtime of the MSPS-RU method is less than that of the Enum method, and the relative improvement degree η achieves over 20%.

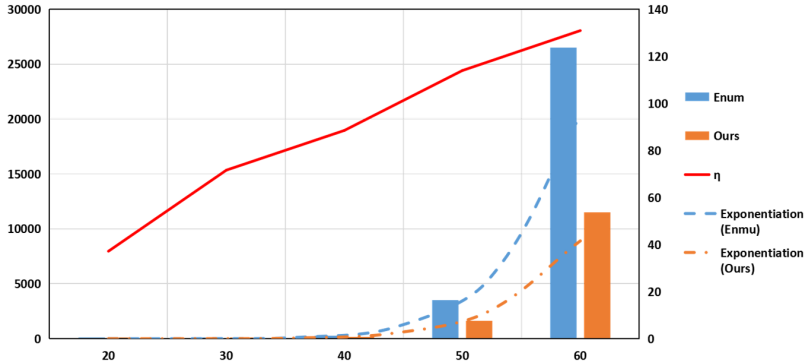
Table 2. The average runtime of extracting 5-order and 10-order MSPS.

n	k	T_{Enum}	T_{ours}	η (%)	k	T_{Enum}	T_{ours}	η (%)
20	5	0.0020	0.0016	22.36	10	0.0746	0.0544	37.15
30	5	0.0138	0.0094	47.04	10	10.5342	6.1387	71.60
40	5	0.0610	0.0384	59.15	10	289.4542	153.5836	88.47
50	5	0.1946	0.1170	66.29	10	3485.9831	1629.6164	113.91
60	5	0.4980	0.3157	57.71	10	26,507.8060	11,480.9314	130.89

Figure 2 illustrates the comparison of runtime growth trends of extracting 5-order and 10-order MSPS using different algorithms. It also shows that with the increase in n, the time growth rate of our algorithm is lower than that of the Enum algorithm according to the trend lines. Therefore, it indicates that MSPS-RU performs well in solving the MSPS issue of larger-order matrices.



(**a**) n varies from 20 to 60, k = 5.



(**b**) n varies from 20 to 60, k = 10.

Figure 2. The comparison of runtime growth trends of extracting 5-order and 10-order MSPS using different algorithms.

5.2. Parameter n Is Fixed and Parameter k Is Variable

To explore the impact of parameter k on the time efficiency of the MSPS-RU algorithm, a 30-order ($n = 30$) real matrix is generated randomly according to the uniform distribution, and the $k(k \leq n)$ order MSPS is extracted using the Enum method and MSPS-RU algorithm, respectively. The runtime of the two algorithms is listed in Table 3.

Table 3. The runtime of extracting k-order MSPS of a 30-order real matrix.

k	T_{Enum}	T_{ours}	η (%)
2	0.0015	0.0009	62.76
5	0.0155	0.0099	57.02
8	1.3105	0.7269	80.29
11	23.2704	13.2104	76.15
14	102.9598	64.4397	59.78
17	143.0534	104.1774	37.32
20	89.3660	77.5431	15.25
23	62.8989	61.8635	1.67
26	64.7635	64.9295	−0.26
29	57.2279	57.1216	0.19

From Table 3, we can find that the time consumption of both algorithms first increases and then decreases with the increasement of k. The maximum value of the time consumption is reached when k is 15 or so. The reason, perhaps, is that the number of k-order principal submatrices in a 30-order real matrix is C_{30}^{k}, and when $k = 15$, $C_{30}^{15} = \underset{k<n}{max}\left\{C_n^k\right\}$. In addition, the time efficiency of the two methods is not much different when the value k is relatively close to n; resulting from that, the number of running code Line 8 in Algorithm 1 is small.

Table 3 also tells us that our method is significantly more time-efficient than the Enum method when $k \leq 20$. We conducted a significance hypothesis test. Assuming: $H_0{:}\mu_1 = \mu_2$, $H_1{:}\mu_1 \neq \mu_2$, (μ_1, μ_2) is the average time consumption of the two methods. The statistics $Z = 2.014 > 1.96 = Z_{0.05}$, $p < 0.05$, then H_1 is accepted at a significance level $\alpha = 0.05$.

Figure 3 illustrates the comparison of runtime growth trends of extracting k-order MSPS from a 30-order matrix using different algorithms. The trend lines are generated using the moving average method with 2 period. It also shows that with the increase in k, the time growth rate of our algorithm is lower than that of the Enum algorithm according to the trend lines. Therefore, it indicates that MSPS-RU performs better than that of the Enum algorithm.

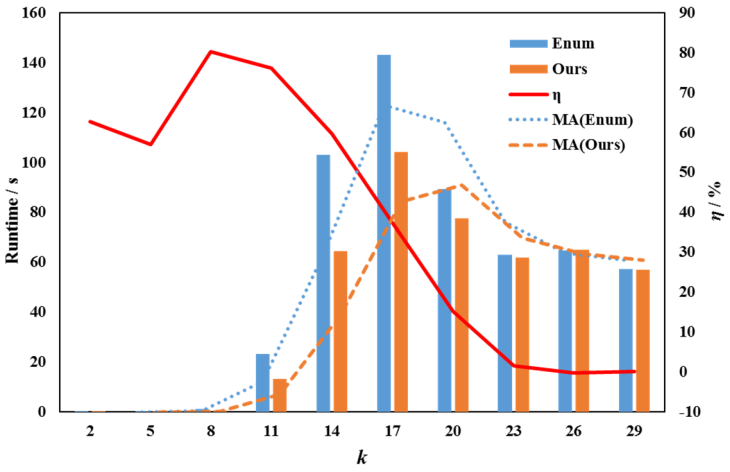


Figure 3. Runtime trends of extracting k-order MSPS in a 30-order real matrix.

5.3. Parameters n and k Are Linear Function

In this subsection, we test the time efficiency of different algorithms when both n and k are variables. For convenience, let $n = 3k$ in this experiment. When $n = 9, 15, 21, 27, 33, 39$, the corresponding $k = 3, 5, 7, 9, 11, 13$. Both n and k are arithmetic series. The common difference is 6 and 2, respectively. Using the Enum method and MSPS-RU algorithm to extract the k-order MSPS, the average runtime is listed in Table 4. From the table, we can find that the average runtime of the Enum method is more than that of the MSPS-RU algorithm; when $n = 39$, the runtime of the Enum method is nearly 2 times that of the MSPS-RU algorithm, which means that the computational efficiency of the MSPS-RU algorithm is significantly improved compared to the Enum method when both n and k are large numbers.

Table 4. The average runtime of extracting k-order MSPS from $3k$-order matrix.

n	k	T_{Enum}	T_{ours}	η
9	3	0.001115	0.000776	43.76
15	5	0.000785	0.000775	1.23
21	7	0.022176	0.015587	42.27
27	9	1.362391	0.810531	68.09
33	11	81.17768	44.27129	83.36
39	13	4693.559	2424.06	93.62

Figure 4 illustrates the comparison of runtime growth trends of extracting k-order MSPS from a $3k$-order matrix using different algorithms. The trend lines are generated using the moving average method with 2 period. From the MA lines, it can be found that as k and n increase, the growth trend of our method's moving average curve is significantly slower than that of the Enum method, and the efficiency improvement also shows an increasing trend.

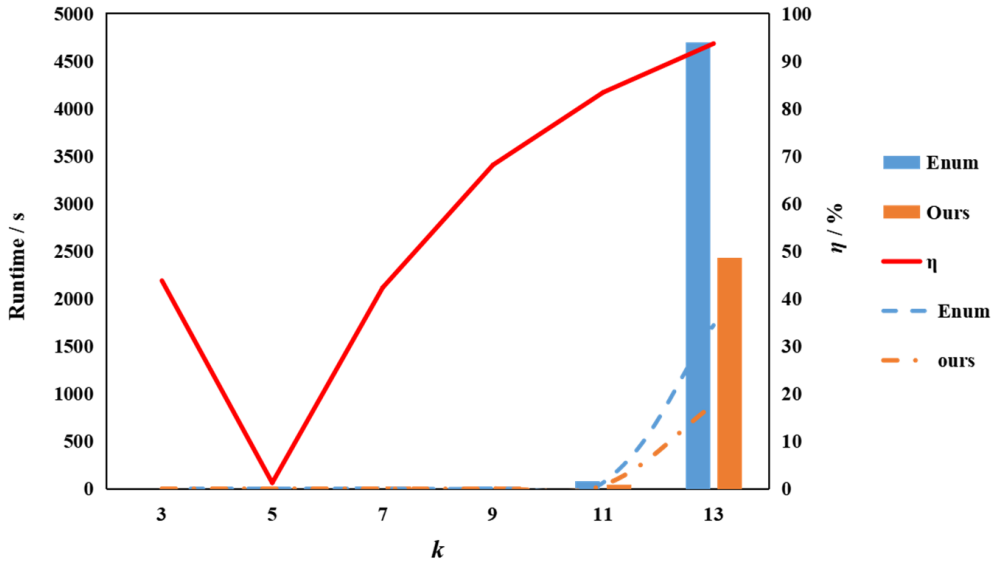


Figure 4. Runtime trend curves of extracting k-order MSPS in a $3k$-order matrix.

In summary, the above three algorithmic efficiency experiments show that the proposed MSPS-RU algorithm works well and has significantly improved efficiency compared to the Enum method.

6. Applications

The extracted MSPS has a lot of applications. Our MSPS algorithm can be used to improve time efficiency of extracting MSPS. This section introduces two applications:

(1) automatically constructing an optimal color combination to improve the readability of visualized data; (2) automatically constructing an optimal stock portfolio selection to minimize correlation degree and reduce the investment risk in the view of daily return volatility.

6.1. Construct a Optimal Color Combination

The optimal color combination here is defined by the largest color difference among all selected colors. The issue of extracting k colors from n different colors to form a color combination with maximum color difference corresponds to the problem of extracting a k-order maximal-sum principal submatrix from an n-order real matrix.

The first step is to build a distance matrix of which the element is the distance between two colors. Then, based on this distance matrix, the k-order MSPS is extracted to determine the combination of k colors with maximum difference, resulting in the enhancement of the readability of data visualization naturally.

6.1.1. Construction of Color Distance Matrix

The most commonly used color space in computer systems is RGB color space, whose pixel values are generally obtained through hardware devices such as image acquisition cards and CCD cameras. Throughout the entire color space Ω, all colors can be represented by three channels:

$$\Omega = \{(r, g, b) | r, g, b \epsilon \{0, 1, \cdots, 255\}\}$$

Traditionally, red can be represented as $(255, 0, 0)$, green can be represented as $(0, 255, 0)$, and blue can be represented as $(0, 0, 255)$.

Definition of standard color difference is based on RGB color space. In Ω, let any two colors $x_i = (r_i, g_i, b_i)$ and $x_j = (r_j, g_j, b_j)$; the spatial distance $d(x_i, x_j)$ between colors x_i and x_j is defined by the Euclidean distance.

$$d(x_i, x_j) = \sqrt{(r_i - r_j)^2 + (g_i - g_j)^2 + (b_i - b_j)^2}$$

Obviously, $d(x_i, x_j) = d(x_j, x_i)$, and we abbreviate $d(x_i, x_j)$ as d_{ij}.

Define the distance matrix D between any two colors. Suppose $|\Omega| = n$, which means that there are n colors used as color candidates. Let $\forall i, j \in L = \{1, 2, \cdots, n\}$, and the distance matrix between different colors can be constructed as:

$$D = \begin{bmatrix} 0 & d_{12} & \cdots & d_{1n} \\ d_{21} & 0 & \cdots & d_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ d_{n1} & d_{n2} & \cdots & 0 \end{bmatrix}$$

Matrix D is an n-order real symmetric non-negative matrix.

The MSPS-RU algorithm is applied to the n-order distance matrix D to extract the optimal solution $A_{I^*}^{(k)}$ and the optimal indicator set I^*. I^* corresponds to the k selected colors. By this way, the selected colors have the maximal color distance d_{ij}. Therefore, these k selected colors can form a color combination with maximal difference and readability.

6.1.2. Application on Stacked Graph Coloring

The experimental data are the precipitation information (unit: mm) of each month in the year (1955) provided by ten urban meteorological stations, namely Huma, Hailar, Nenjiang, Xiguitu Banner Boketu, Keshan, Qiqihar, Hailun, Fujin, Horqin Right Front Banner, and Anda. These are time series data (Figure 5) that can be downloaded from the Institute of Geographic Sciences and Resources, Chinese Academy of Sciences [26].

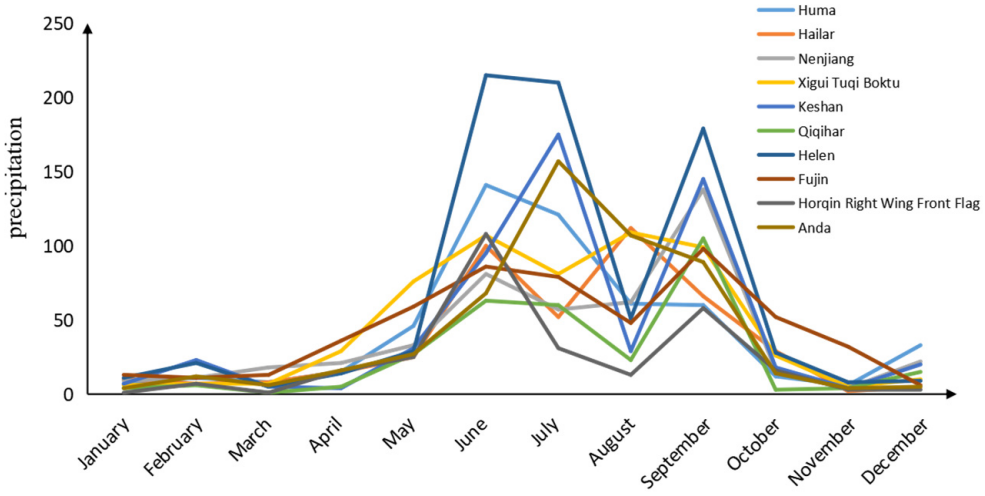


Figure 5. Monthly rainfall of 10 cities in a year.

The 10-line strips with different colors in Figure 5 represent monthly rainfall of 10 cities in a year. In order to observe the accumulated rainfall of all cities and the trend of rainfall in each city more conveniently, the monthly rainfall of 10 cities can be visualized using a stacked graph (also called streamgraphs or ThemeRiver), as shown in Figure 6. In the stacked graph, the color of each strip is a key factor related to the readability of the figure.

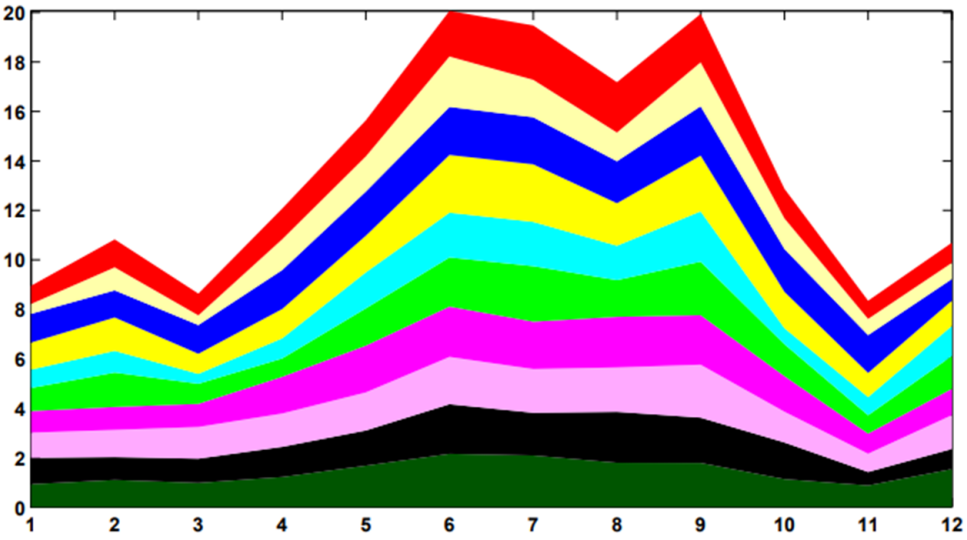


Figure 6. Color combination with maximal difference.

Figure 6 shows the stacked graph with the color combination built by 10 colors with maximal difference. These colors are selected from standard RGB color space Ω based on the 10-order MSPS obtained by the MSPS-RU algorithm.

Besides RGB color space Ω, there are many other color spaces, for example, ΔE^* color space. In this dissertation [20], the distance between colors has been expounded, and the distance matrix is built using the distances among 18 candidate colors. We extracted 10 colors using our MSPS-RU algorithm from 17 candidate colors, where the white color is removed because it is used for the background color, to visualize the monthly rainfall of 10 cities. The colored stacked graph can be seen in Figure 7.

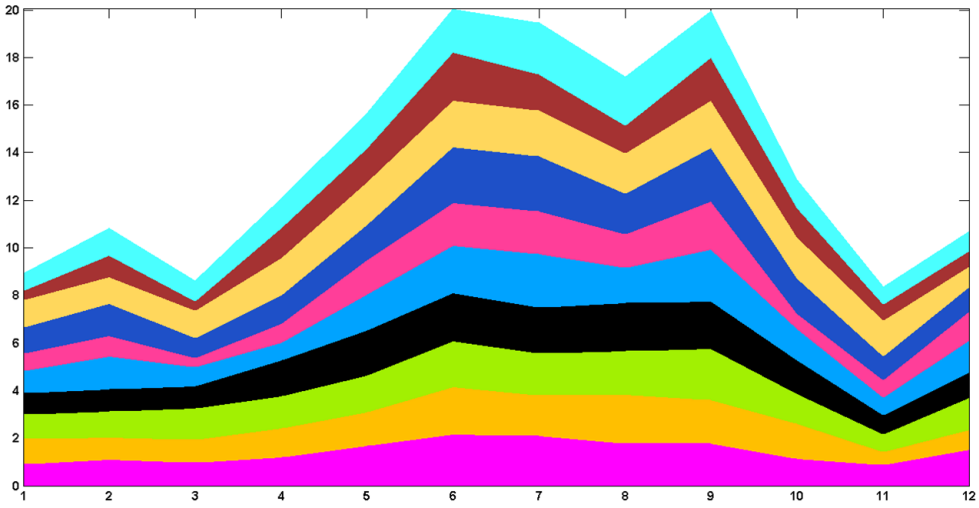


Figure 7. The stacked graph colored with maximal difference 10 colors extracted from the ΔE* color distance matrix [20].

As a comparison, a color combination based on the Rcolorbrewer function package and user study [13] is used to color the stacked graph, as shown in Figure 8.

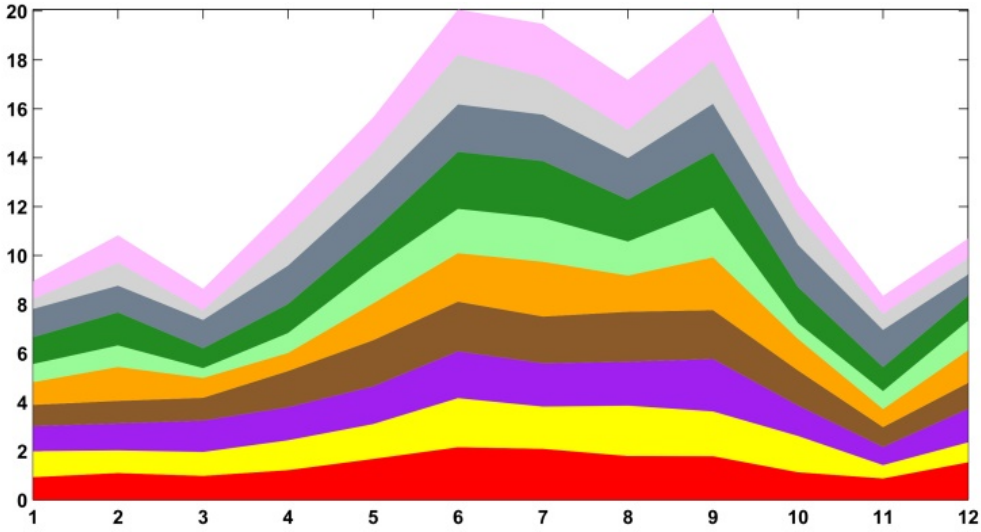


Figure 8. Color combination built by the rule from reference [13].

Comparing Figure 8 to Figures 6 and 7, the former shows that the color combination built by the rule from reference [13] contains certain aesthetic preferences, and the co-occurrence of colors is relatively suitable and visually comfortable. However, the stacked graphs in Figures 6 and 7, colored using our method, display the best contrast of coloring among the three figures and avoid similar colors appearing in the stacked graph. This observation can also be supported by quantitative data. The average value of the sum of elements of all k-order principal submatrices is 11.2273, but the corresponding values in Figures 6–8 are 56.6027, 43.7446 and 38.7163, respectively. These figures indicate that using our method or the rule from [13] can obtain better results, and our method is even more advantageous. Therefore, from the perspective of increasing contrast, our method provides an alternative solution.

6.2. Construct a Optimal Stock Portfolio

Another application related to the k-order MSPS is the stock portfolio selection. An optimal stock portfolio is here defined as the minimum correlation degree of stock returns, which is regarded as a kind of investment risk [25]. The minimum correlation degree means increased diversification of investment items. Our goal is to select a portfolio with the minimum correlation degree from the candidate stocks. The algorithm, named find a portfolio with the minimum correlation degree (FP-MCD), mainly includes the following steps to achieve this goal:

(1) Building a negative correlation coefficient matrix: Since MSPS-RU deals with the maximal problem, the minimum correlation degree can be converted to the maximal problem of the negative correlation coefficient matrix.
(2) Using MSPS-RU algorithm to extract k stocks which have the maximal negative correlation degree.
(3) Proposing investment suggestions.

The pseudocode of the algorithm (Algorithm 3), FP-MCD, has been described as follows.

Algorithm 3: FP-MCD algorithm

Input: $R_{t\times n}$; //The *return rates of n candidate stocks in continuous t days*;
k; //the number of stocks in the portfolio
t_1; // t_1 days used for learning, constructing the correlation coefficient matrix; $(t - t_1)$ days used to test portfolio returns
Output: I^*; // the optimal index set of the n candidate stocks

Let $R_{t\times n} = \begin{bmatrix} P_{t_1\times n} \\ Q_{(t-t_1)\times n} \end{bmatrix}$;
Construct the correlation coefficient matrix $\mathrm{T}_{n\times n}$ using $P_{t_1\times n}$;
Build the negative correlation coefficient matrix $\mathrm{M}_{n\times n} = -\mathrm{T}_{n\times n}$;
Use MSPS-RU to obtain the optimal portfolio I^*
Test portfolio returns with I^* and data $Q_{(t-t_1)\times n}$;
If the I^* is accepted
  return I^*;
else
  return "Not recommended";
endif

6.2.1. Data

We recorded the weekly return rates of 50 stocks in the Shanghai Stock Exchange in the past 3 years from 3 April 2020 to 24 March 2023, a total of 36 months and 153 weeks on the Sohu website (https://q.stock.sohu.com/cn/bk_4507.shtml, accessed on 4 April 2023). During this period, the return data of four companies, China Yangtze Power, Three Gorges Energy, CICC, and Trina Solar, are incomplete, so these four stocks are removed from the candidate stock set. Then, the candidate stock set consists of 46 companies, including Baotou Steel, Sinopec, CITIC Securities, Sany Heavy Industry, China Merchants Bank, Poly Real Estate, SAIC Motor, China Northern Rare Earth, Fosun Pharma, Jiangsu Hengrui Medicine, Wanhua Chemical Group, Hengli Petrochemical, Guodian NARI, Pien tze huang, Tongwei, Kweichow Moutai, Hailuo Cement, Haier Zhijia, Wingtech, Shanxi Fenjiu, Yili, Hangfa Power, Longji Green Energy CSCI, China Shenhua, Industrial Bank, Shaanxi Coal Industry, Agricultural Bank of China, Ping An Insurance, Industrial and Commercial Bank of China, China Pacific Insurance, China Life Insurance, Great Wall Motor, CSCEC, Power Construction Corporation of China, Huatai Securities, PetroChina, China Exemption, Zijin Mining, COSCO Marine Holdings, WuXi AppTec, Hesheng Silicon, Foshan Haitian Flavouring&Food Co, Weier, Huayou Cobalt, Zhaoyi Innovation. A total of 7038 sample data were obtained, as shown in Figure 9.

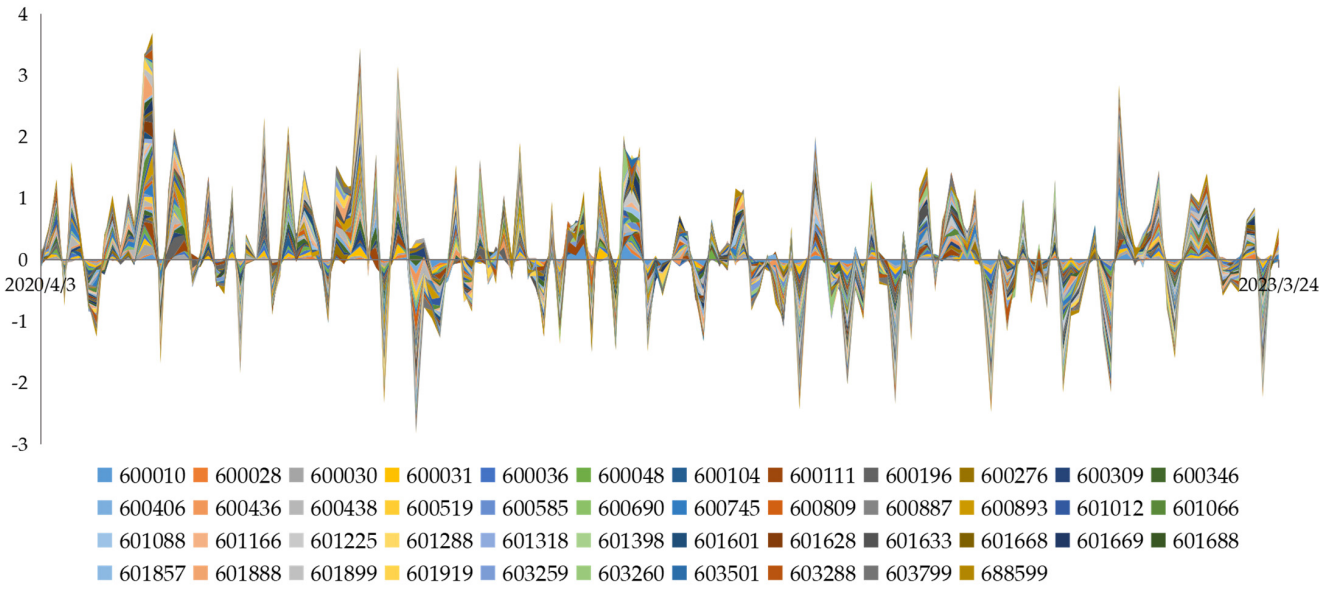


Figure 9. Stacked chart of weekly return rates of 46 stocks.

The correlation coefficient matrix T of the weekly increase in 46 companies is shown in Figure 10.

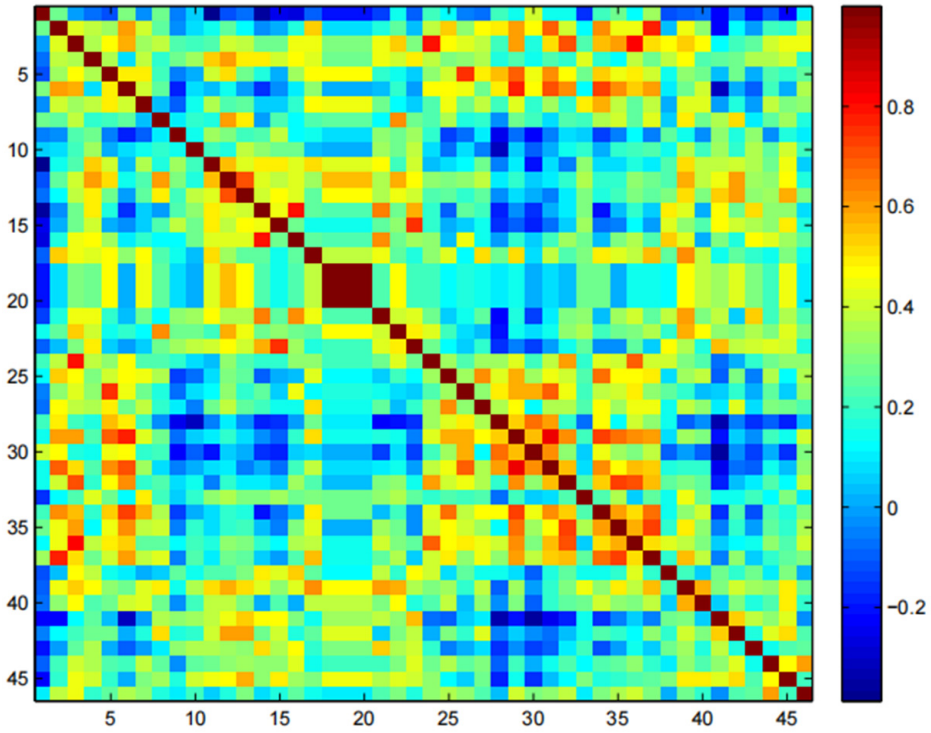


Figure 10. The correlation matrix for the weekly increase in 46 stocks.

6.2.2. Construct the Optimal Investment Portfolio

Using the MSPS-RU algorithm, the k-order maximal-sum principal submatrices from the negative correlation coefficient matrix M can be extracted. Table 5 lists different investment portfolios corresponding to different k values.

Table 5. Different investment portfolios corresponding to different k values.

k	Portfolios
3	1 14 40
4	1 14 28 41
5	1 9 14 28 40
6	1 9 16 28 40 41
7	1 9 14 28 30 40 41
8	1 9 14 28 30 33 40 41
9	1 9 14 28 30 33 40 41 44
10	1 9 10 28 30 33 38 40 41 44

6.2.3. Test the Optimal Investment Portfolio

The test experiment is conducted using a simple portfolio, $k = 3$, without losing generality. When $k = 3$, the optimal portfolio consists of the 1st stock Baogang Shares, the 14th stock Pien tze huang, and the 40th stock China Exempt, according to Table 5.

We use the first one-third of the 153 weekly return rates ($t_1 = 51$) to calculate the correlation coefficient matrix and proceed with investment from the next working week. For convenient comparison, 30 random investment portfolios are also simulated, and each random investment portfolio consists of three stocks randomly selected from 46 stocks. Let the initial investment be 1 unit, and there is no buying or selling transactions during the period. The changes in assets during the investment period of all 30 random investment portfolios and ours are drawn as line charts (Figure 11).

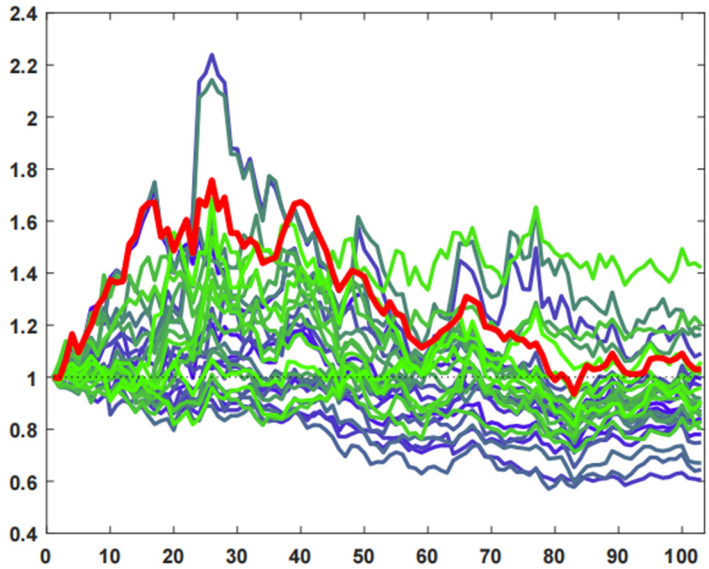


Figure 11. The changes in assets during the investment period according to the optimal investment portfolio and the 30 random portfolios.

In Figure 11, the red line shows that after 102 weeks, its result is greater than the initial fund 1. This means successfully investing with the optimal investment portfolio recommended by our method. After 102 weeks, the results of the green or blue lines are diversified, which reflects the instability of random investments.

Two indicators are used for quantitative analysis. The first is the overall standard deviation σ defined by Formula (9), which measures the volatility of investment returns [27].

$$\sigma = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(x_i - \overline{x})^2} \tag{9}$$

where $\overline{x}$ is the arithmetic mean, and x_i is the daily increase.

Another indicator for quantitative analysis is the final asset quantity, denoted by *Ret*.

Using "Rnd k" to represent 30 random portfolios, "Ours" to represent the portfolio obtained by the MSPS-UR algorithm, the result after 102 weeks in Figure 11 is listed in Table 6.

Table 6. Return and σ of simulated investments for each portfolio after 102 weeks.

Indicator	Ours	Rnd1	Rnd2	Rnd3	Rnd4	Rnd5	Rnd6	Rnd7	Rnd8	Rnd9	Rnd10
σ	0.0278	0.0307	0.0260	0.0133	0.0197	0.0343	0.0172	0.0291	0.0293	0.0220	0.0316
Ret	1.0332	1.0197	1.0300	0.9862	0.9584	1.0240	1.0260	1.0024	1.0029	1.0522	0.9509
Indicator	Rnd11	Rnd12	Rnd13	Rnd14	Rnd15	Rnd16	Rnd17	Rnd18	Rnd19	Rnd20	
σ	0.0243	0.0229	0.0289	0.0189	0.0251	0.0149	0.0312	0.0302	0.0237	0.0135	
Ret	0.9866	1.0366	1.0564	1.0164	1.0185	1.0353	1.0451	0.9961	0.9950	1.0168	
Indicator	Rnd21	Rnd22	Rnd23	Rnd24	Rnd25	Rnd26	Rnd27	Rnd28	Rnd29	Rnd30	
σ	0.0260	0.0206	0.0410	0.0287	0.0250	0.0242	0.0206	0.0264	0.0204	0.0169	
Ret	1.0243	1.0055	1.0170	1.0244	1.0165	1.0361	1.0388	1.0386	1.0122	0.9902	

For a more direct comparison, we subtract the corresponding indicators, i.e.,

$$\Delta\sigma = \sigma(Rnd\ k) - \sigma(Ours) \tag{10}$$

$$\Delta Ret = Ret(Rnd\ k) - Ret(Ours) \tag{11}$$

The comparison of 30 random portfolios and our portfolio is shown in Table 7.

Table 7. The difference in return and σ between Rnd k and Ours.

Diff.	Rnd1	Rnd2	Rnd3	Rnd4	Rnd5	Rnd6	Rnd7	Rnd8	Rnd9	Rnd10
$\Delta\sigma$	0.0028	−0.0019	−0.0146	−0.0081	0.0064	−0.0106	0.0013	0.0015	−0.0058	0.0038
ΔRet	−0.0134	−0.0031	−0.0470	−0.0747	−0.0092	−0.0072	−0.0308	−0.0302	−0.0190	−0.0823
Diff.	Rnd11	Rnd12	Rnd13	Rnd14	Rnd15	Rnd16	Rnd17	Rnd18	Rnd19	Rnd20
$\Delta\sigma$	−0.0035	−0.0049	0.0011	−0.0089	−0.0028	−0.0129	0.0034	0.0023	−0.0042	−0.0143
ΔRet	−0.0466	0.0034	0.0232	−0.0167	−0.0146	0.00215	0.0119	−0.0371	−0.0382	−0.0163
Diff.	Rnd21	Rnd22	Rnd23	Rnd24	Rnd25	Rnd26	Rnd27	Rnd28	Rnd29	Rnd30
$\Delta\sigma$	−0.0018	−0.0073	0.0132	0.0008	−0.0028	−0.0036	−0.0072	−0.0015	−0.0074	−0.0109
ΔRet	−0.0089	−0.0277	−0.0162	−0.0088	−0.0167	0.0029	0.0057	0.0055	−0.0210	−0.0430

In Table 7, there are eight portfolios that satisfied $\Delta Ret < 0$ and $\Delta\sigma > 0$, which means that these portfolios not only have lower final returns than those obtained by the MSPS-RU algorithm, but also have greater asset volatility. There are 15 combinations that satisfied $\Delta\sigma < 0$ and $\Delta Ret < 0$, which means that although these portfolios have lower asset volatility, their returns are lower than ours too.

7. Conclusions

We, in this paper, propose a fusion row update accelerated algorithm (MSPS-RU) and present two different applications. The experimental results show that the proposed algorithm avoids repeatedly calculating the sum of the same submatrix and significantly improves computing efficiency compared to the Enum algorithm; the color combination based on MSPS-RU enhances the contrast of the data visualization results; the stock investment portfolio constructed by MSPS-RU can bring higher investment returns but lower risks, avoiding the uncertainty of random investments.

Our method has limitations, and solving these problems can become future research topics. The improvement of computational efficiency in this paper mainly depends on reducing the number of matrix elements to be summed. However, it can also be achieved

by reducing the number of principal submatrices to be calculated, for example, using pruning. The selection of color combinations should consider not only the maximizing color difference but also aesthetic habits and preferences. In addition, more application problems can be modeled by extracting the maximum sum principal submatrix from a raw data matrix.

Author Contributions: Conceptualization, L.L. and H.L.; methodology, Y.Z. and H.L.; software, Y.Z. and H.L.; validation, Y.Z., L.L. and H.L.; formal analysis, H.L.; investigation, Y.Z.; resources, Y.Z.; data curation, Y.Z.; writing—original draft preparation, Y.Z., L.L. and H.L.; writing—review and editing, L.L.; visualization, Y.Z. and H.L.; supervision, L.L. and H.L.; project administration, H.L. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: Data can be download from http://www.data.ac.cn/table/tbc40, accessed on 21 May 2023.

Acknowledgments: The author thanks the anonymous referees for their valuable suggestions.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Branders, V.; Schaus, P.; Dupont, P. Combinatorial optimization algorithms to mine a sub-matrix of maximal sum. In Proceedings of the 6th International Workshop on New Frontiers in Mining Complex Patterns in Conjunction with ECML-PKDD 2017, Skopje, Macedonia, 18–22 September 2017; Lecture Notes in Computer Science. Springer International Publishing: Berlin/Heidelberg, Germany, 2018; Volume 10785, pp. 65–79. [CrossRef]
2. Derval, G.; Schaus, P. Maximal-Sum submatrix search using a hybrid contraint programming/linear programming approach. *Eur. J. Oper. Res.* **2022**, *297*, 853–865. [CrossRef]
3. Ferreira, C.S.; Camargo, R.Y.; Song, S.W. A Parallel Maximum Subarray Algorithm on GPUs. In Proceedings of the 2014 International Symposium on Computer Architecture and High Performance Computing Workshop, Paris, France, 22–24 October 2014; pp. 12–17. [CrossRef]
4. Weddell, S.J.; Read, T.R.; Thaher, M.; Takaoka, T. Maximum subarray algorithms for use in astronomical imaging. *J. Electron. Imaging* **2013**, *22*, 043011. [CrossRef]
5. Koch, I.; Marenco, J. The maximum 2D subarray polytope: Facet-inducing inequalities and polyhedral computations. *Discret. Appl. Math.* **2022**, *323*, 286–301. [CrossRef]
6. Li, Y.; Xie, W. Best Principal Submatrix Selection for the Maximum Entropy Sampling Problem: Scalable Algorithms and Performance Guarantees. *arXiv* **2023**, arXiv:2001.08537. [CrossRef]
7. Macambira, E.M. An Application of Tabu Search Heuristic for the Maximum Edge-Weighted Subgraph Problem. *Ann. Oper. Res.* **2002**, *117*, 175–190. [CrossRef]
8. Massei, S. Some algorithms for maximum volume and cross approximation of symmetric semidefinite matrices. *BIT Numer. Math.* **2022**, *62*, 195–220. [CrossRef]
9. Lewis, S.C. On the Best Principal Submatrix Problem. Ph.D. Thesis, University of Birmingham, Birmingham, UK, 2006.
10. Wen, Z. Fast parallel algorithms for the maximum sum problem. *Parallel Comput.* **1995**, *21*, 461–466. [CrossRef]
11. Tamaki, H.; Tokuyama, T. Algorithms for the Maximum Subarray Problem Based on Matrix Multiplication. *Interdiscip. Inf. Sci.* **2000**, *6*, 99–104. [CrossRef]
12. Takaoka, T. Efficient Algorithms for the Maximum Subarray Problem by Distance Matrix Multiplication. *Electron. Notes Theor. Comput. Sci.* **2002**, *61*, 191–200. [CrossRef]
13. He, Y.; Li, H. Optimal layout of stacked graph for visualizing multidimensional financial timeseries data. *Inf. Vis.* **2022**, *21*, 63–73. [CrossRef]
14. Healey, C.G. Choosing effective colours for data visualization. In Proceedings of the Seventh Annual IEEE Visualization '96, San Francisco, CA, USA, 27 October–1 November 1996; pp. 263–270. [CrossRef]
15. Zhou, L.; Hansen, C.D. A Survey of Colormaps in Visualization. *IEEE Trans. Vis. Comput. Graph.* **2016**, *22*, 2051–2069. [CrossRef] [PubMed]
16. El-Assady, M.; Kehlbeck, R.; Metz, Y.; Schlegel, U.; Sevastjanova, R.; Sperrle, F.; Spinner, T. Semantic Color Mapping: A Pipeline for Assigning Meaningful Colors to Text. In Proceedings of the 2022 IEEE 4th Workshop on Visualization Guidelines in Research, Design, and Education (VisGuides), Oklahoma City, OK, USA, 17 October 2022; pp. 16–22.
17. Anderson, C.L.; Robinson, A.C. Affective Congruence in Visualization Design: Influences on Reading Categorical Maps. *IEEE Trans. Vis. Comput. Graph.* **2022**, *28*, 2867–2878. [CrossRef] [PubMed]
18. Samsel, F.; Bartram, L.; Bares, A. Art, Affect and Color: Creating Engaging Expressive Scientific Visualization. In Proceedings of the IEEE VISAP, Berlin, Germany, 23–26 October 2018.

19. Wang, Y.; Chen, X.; Ge, T.; Bao, C.; Sedlmair, M.; Fu, C.-W.; Deussen, O.; Chen, B. Optimizing Color Assignment for Perception of Class Separability in Multiclass Scatterplots. *IEEE Trans. Vis. Comput. Graph.* **2019**, *25*, 820–829. [CrossRef] [PubMed]
20. Abeyta, R.N. The Distance between Colors; Using DeltaE* to Determine Which Colors Are Compatible. Ph.D. Thesis, Embry-Riddle Aeronautical University, Daytona Beach, FL, USA, 2011. Available online: https://commons.erau.edu/edt/9 (accessed on 23 June 2023).
21. Ledoit, O.; Wolf, M. Improved estimation of the covariance matrix of stock returns with an application to portfolio selection. *J. Empir. Financ.* **2003**, *10*, 603–621. [CrossRef]
22. Onali, E.; Mascia, D.V. Corporate diversification and stock risk: Evidence from a global shock Author links open overlay panel. *J. Corp. Financ.* **2022**, *72*, 102150. [CrossRef]
23. Markowitz, H. Portfolio selection. *J. Financ.* **1952**, *7*, 77–91.
24. Ruppert, D.; Matteson, D.S. Portfolio Selection. In *Springer Texts in Statistics*; Springer: Berlin/Heidelberg, Germany, 2015; pp. 465–493. [CrossRef]
25. Tiwari, A.K.; Abakah, E.J.A.; Karikari, N.K.; Hammoudeh, S. Time-varying dependence dynamics between international commodity prices and Australian industry stock returns: A Perspective for portfolio diversification. *Energy Econ.* **2022**, *108*, 105891. [CrossRef]
26. Institute of Geographic Sciences and Resources, Chinese Academy of Sciences. Monthly Precipitation over the Years (by Station), Dataset. Available online: http://www.data.ac.cn/table/tbc40 (accessed on 21 May 2023).
27. Moledina, A.A.; Roe, T.L.; Shane, M. Measuring commodity price volatility and the welfare consequences of eliminating volatility. In Proceedings of the American Agricultural Economics Association Annual Meeting, Denver, CO, USA, 1–4 August 2004.