Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 4 (2011) 1288–1295

Procedia Computer Science

International Conference on Computational Science, ICCS 2011

An efficient algorithm for computing the *K*-overlapping maximum convex sum problem

Mohammed Thaher[a], Tadao Takaoka[a,b,*]

[a,b]*Department of Computer Science and Software Engineering, University of Canterbury, Christchurch 8140, New Zealand*

Abstract

This research presents a new efficient algorithm for computing the *K*-Maximum Convex Sum Problem (*K*-MCSP). Previous research has investigated the *K*-MCSP in the case of disjoint regions. In this paper, an efficient algorithm with $O(Kn^3)$ time is derived for the overlapping case. This approach is implemented by utilizing the simplified (bidirectional) algorithm for the convex shape. This algorithm finds the first maximum sum, second maximum sum and up to the K^{th} maximum sum, based on dynamic programming. Findings from this paper show that using the *K*-Overlapping Maximum Convex Sum (*K*-OMCS), which is a novel approach, gives more precise results when compared with the results obtained from the rectangular shape. Additionally, this paper presets an example for prospective application of using the Maximum Convex Sum Problem (MCSP) to locate a cancer tumour inside a mammogram image.

Keywords: Maximum Convex Sum Problem; *K*-Disjoint Maximum Convex Sum Problem; *K*-Overlapping Maximum Convex Sum Problem.

1. Introduction

The Maximum Subarray Problem (MSP) is to find the most useful informative array portion that associates two parameters involved in data [1, 2]; previous research used the rectangular shape to find the most promising region [3,4,5,6,7,8,9,10]. MSP takes a new turn by using the convex shape [11]; Figure 1 shows an illustrative example of the MSP and the new method for computing the maximum sum, which is named the Maximum Convex Sum Problem (MCSP). MSP can be computed for one-dimensional and two-dimensional arrays with positive and negative elements. The one-dimensional case is also called the maximum-sum segment problem and is well known to be linear-time solvable using Kadane's algorithm [1]. For the two-dimensional array, the MSP and the MCSP involve a selection of a segment of consecutive array elements that has the largest possible sum compared with all other segments in presented data [1, 2]. In the two-dimensional case, the task is to find a subarray such that the sum of its elements is maximized [12].

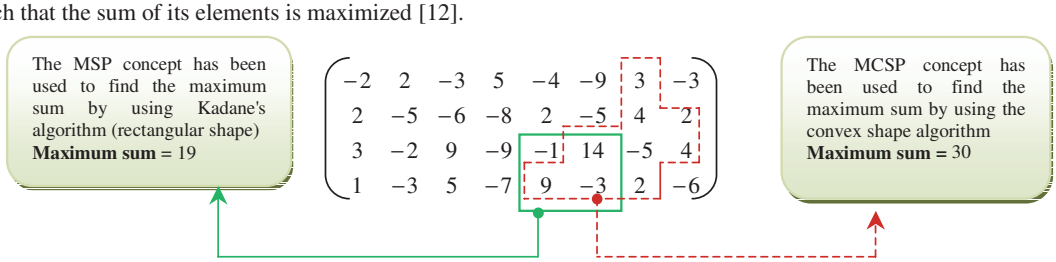

Fig. 1. maximum sum obtained from using the Kadane's algorithm and the convex shape algorithm.

* Corresponding author. Tel.: +64 3 364 2987; fax: +64 3 364 2569.
E-mail address: mohammed.thaher76@gmail.com, tad.takaoka@canterbury.ac.nz.

Selection and/or peer-review under responsibility of Prof. Mitsuhisa Sato and Prof. Satoshi Matsuoka
doi:10.1016/j.procs.2011.04.139

The MSP is widely used in pattern recognition [6, 13], data mining [11,14], biological sequence analysis [15,16,17,18,19,20,21], and recently in health sciences [22] and medical applications [11]. For example, an environmental sciences application has been discussed in Fukuda and Takaoka's paper [22].

In Fukuda and Takaoka's study [22], they applied the approach of the MSP in a health and environmental sciences application. Their study presented a new approach because epidemiological studies generally involve statistical/geographical approaches [22].

Research on maximum sums has been recently extended to incorporate novel approaches. In 2010, Thaher and Takaoka [11] used concepts of the Maximum Convex Sum to optimize results obtained from the MSP. Thaher and Takaoka departed from using the rectangular region in the MSP by using the convex shape. They developed a new algorithm to compute the maximum sum using concepts of the convex shape approach. Additionally, their work was extended to compute the *K*-disjoint maximum convex sums in $O(Kn^3)$ time. However, this result can be further extended to another problem of the overlapping case.

In this paper, the aim is to compute the *K*-overlapping maximum convex sums in $O(Kn^3)$ time. The *K*-overlapping convex maximum sums result in ranked maximum convex sums, which are the first, the second and up to the K^{th} maximum sums. The new implementation of the overlapping case in MCSP is expected to efficiently optimize the results when compared to results obtained from previous work that used the rectangular shape to find the overlapping regions [8, 12].

This study covers the following sections: background; mathematics of the *K*-OMSP computations; implementation of the convex shape; the *K*-Convex sum problem which includes *K*-OMCSP algorithm; results and analysis; conclusions and future work.

2. Background

The history of the MSP dates back to 1977. This is when problems were encountered in pattern recognition [6,13]. These problems led to realization of the maximum subarray problem [12]. The discovery of this problem and its associated challenges guided scientists to develop many algorithms and enhanced aspects for the MSP. These are covered elsewhere in [23,24,25].

In this paper, studies related to finding the *K* maximum sums are presented. One of the studies classified the *K* maximum subarray problem into the *K*-Overlapping Maximum Subarray Problem (*K*-OMSP) and the *K*-Disjoint Maximum Subarray Problem (*K*-DMSP) [12]. Bae in his study presented methodologies and techniques to speed up computation for these two categories [12]. Moreover, Bae adopted a general framework based on subtraction of the minimum prefix sum from each prefix sum to produce candidates for the final solution. He investigated various methods to compute the *K*-OMSP efficiently; his approach was based on using the rectangular shape region. In 2007, Bashar and Takaoka developed an algorithm to generalize the MSP by using average case analysis; this was also based on using the rectangular region [26].

In addition to the aforementioned research, Fukuda and his colleagues discussed data mining based on association rules for two numeric attributes and one Boolean [14]. They proposed an efficient algorithm for computing the regions that give optimal association rules for gain, support and confidence. The main aim of their algorithm was to generate two-dimensional association rules that represent the dependence on a pair of numeric attributes.

Although the above research achieved successful results, these were all based on using the rectangular region as part of the MSP, except for Fukuda's work [14] for a rectilinear region. A more optimized solution for K sums compared to the solution obtained from using the rectangular shape in the MSP was achieved using the convex shape [11].

3. Mathematics of the *K*-overlapping maximum sum problem computation

3.1. Overview

The problem of the *K*-OMSP was first presented in 2004 [8]. Bengttson and Chen also studied the problem independently around the same time [24].

Since then, a rich collection of publications addressing the problem has been accumulated [8, 23,24, 25], and the time complexity of the problem has been increasingly improved.

3.2. Problem definition

We have started from the basic case of one-dimensional array to provide background information in relation to the topic. Our approach is based on a two-dimensional array.

For a given array a[1..n] containing positive and negative real numbers and 0, the maximum subarray is the consecutive array elements of the greatest sum. Let MAX(*K*, *L*) be the operation that selects the *K* largest elements in a list *L* in non-increasing order. The definition of *K* overlapping maximum subarrays is given as follows.

$$R = \text{MAX}(K, L), \text{ where } L = \left\{ \sum_{x=i}^{j} a[x] \mid 1 \le i \le j \le n \right\} \quad (1)$$

Here, the K maximum sums are stored in $R[1..K]$. Note that the solution set R is in sorted order. This sorting was not required, but all other literature on this problem unanimously assumed the order.

Example (1):

Let a = {3, 51, -41, -57, 52, 59, -11, 93, -55, -71, 21, 21}. For this array of size 12, a total of 78(= 12(12+1)/2) subarrays exist. Among them, the first maximum subarray is 193, a[5]+a[6]+a[7]+a[8] if the first element is indexed 1. We denote this by 193(5, 8). When overlapping is allowed, the second and third maximum subarrays are 149(1, 8) and 146(2, 8). The 78-th maximum subarray, (or the minimum subarray) is −126 (9, 10).

This paper presents a new approach of using the overlapping case in finding the maximum sum in two-dimensions, whilst departing from using the rectangular shape. Using the rectangular shape to find the K overlapping maximum sums computed in [8] is depicted in Figure 2.

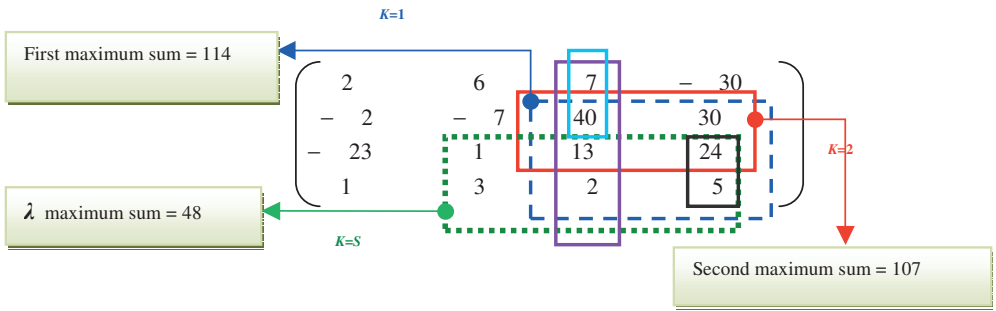

Fig. 2. finding K-OMSP by using the rectangular shape. λ represents one of the maxima before finding the K^{th} overlapping maximum sums.

Using the convex shape is expected to be applicable to wider sets of data distributions, which results in increased efficiency and optimisation in the form of obtaining a maximized gain. Figure 3 is an illustrative figure for the K overlapping case in two-dimensions.

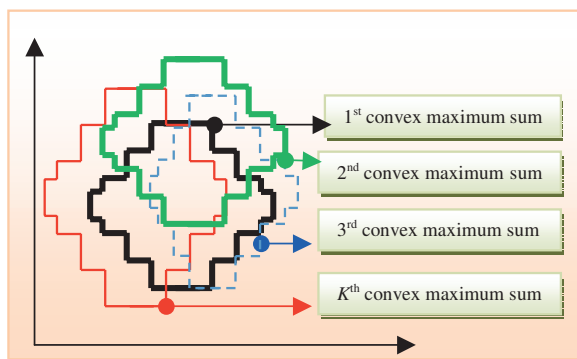

Fig. 3. K-overlapping convex maximum sums in two-dimensional array.

4. Implementation of the convex shape

Previous work discuses the convex shape and its development in the MSP approach [11]; the studies are based on dynamic programming. The following is a brief summary of the convex shape in MSP [11]:

Definitions of W and N shapes (1): A region is called a W region [14] when its top contour inclines or remains horizontal and the bottom contour declines or remains horizontal from left to right. The name W comes from the fact that the shape widens. A mirror image of the W shape is called an N shape, because it narrows from left to right. A WN shape is obtained by W and N together. In this paper, the WN shape will be called the convex shape, for simplicity.

Definition of WN Convex Shape (2): A region is called WN convex shape [14], if it is comprised of x-monotone and y-monotone shapes (Figure 4) [14]. X-monotone means the region is continuously occupied in x-axis. Y-monotone is similar. The WN shape is a sub-family of rectilinear regions [14]; however it is not as per the convex shape in the strict sense of geometry.

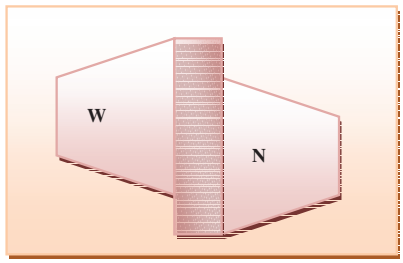

Fig. 4. convex shape.

Theorem (1): We compute *W* sums ending at column k from position s to position t. This ending portion is called the anchor column. Optimised-gain convex shape can be computed for all anchor columns in O (n^3) time.

To find the best W shape, there are three cases:

- ***Case (1):*** The first scenario is based on the previous solution up to k-1. We need to check if the extension to the k^{th} column is appropriate to maximise the gain or sum (Figure 5.a).
- ***Case (2):*** The second scenario is the *W* shape in Figure 5.b, based on the previous solution in the k^{th} column with a narrower interval. We add the value at (s, k) to the best solution to get more gain or sum.
- ***Case (3):*** The third scenario is the *W* shape in Figure 5.c with a narrower interval; we add the value at (t, k) to the best solution in the k^{th} column to get more gain or sum.

The correctness can be seen from the fact that the optimal *W* shape ending at column k from s to t is composed of one of the above three cases.

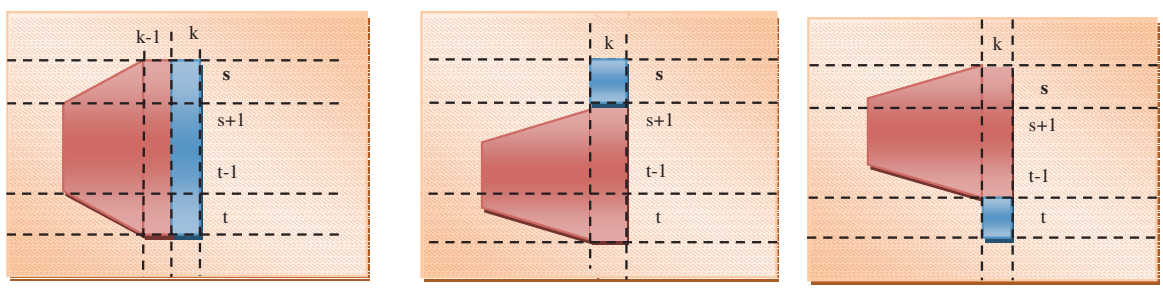

Fig. 5. (a) first case of *W* shape; (b) second case of *W* shape; (c) third case of *W* shape

Let *v* be the array containing the original data. In algorithm terms, f_W is computed as follows:

Algorithm (1)

f_W (0, [s,t])=0 for all s ≤ t
f_W (k, [s,t])=0 for all s > t and for all k
for k=1 to n do
  for all intervals of [s, t] in increasing
   order of t-s where s ≤ t do
  f_W (k, [s,t]) =
  max{ f_W (k-1, [s,t])+sum[k,s,t] (case 1)
    f_W (k, [s+1,t])+a[s, k], (case 2)
    f_W (k, [s, t-1])+a[t, k]} (case 3)

Where f_W is the function to find W shape. f_W (k, [s, t]) is the optimal value of the sum of W shape ending from position s to position t in column k. The sum, sum[k, s, t], is the sum of the k^{th} column from position s to position t used in scenario one. The elements, a[s,k] and a[t,k], are the values added in scenarios two and three respectively. The number of possible intervals is O (n^2), which is placed inside the loop by k. Thus, this algorithm takes O (n^3) time.

4.1. Convex shape algorithm

The following algorithm is based on the results obtained from Algorithm 1:

Algorithm (2):

Let sum[k, s, t] be the column sum from s to t of the k^{th} column.

1. Compute W shape from left to right for each k, s and t in $f_{w,}$ using Algorithm 1.
2. Compute N shape from right to left for each k, s and t, resulting in f_N, using Algorithm 1;
   /** Finalization by combination **/
3. For k=1 to n do
   For s=1 to n do
   For t=s to n do
   f_{WN} (k, [s, t])= f_W (k, [s, t]+ f_N (k, [s, t] – sum[k, s, t]
4. Take the maximum of f_{WN} (k, [s, t]) for all k, s, t.

We call the array portion from index s to t in the k^{th} column the anchor column of f_{WN} (k,[s,t]), denoted by (k, s, t). The finalization part takes O(n^3) time since we have a triply nested structure.

4.2 Application of convex shape algorithm

Applications of MCSP include science branches, medical and computer applications. For example, computer vision, data mining and medical research related to Oncology. In this section, an example on Oncology is explained in the context of MCSP

4.2.1. Medical application: Breast cancer example

Breast cancer is a cancerous growth that begins in the tissues of the breast [27]. Cancer is a disease in which abnormal cells grow in an uncontrolled way [28]. Breast cancer is the most common cancer in women, but it can also appear in men [28].

Effective treatment to remove the cancerous lump(s) completely relies on accurately locating the tumour. In many cases this is not a straightforward task. MSP can be of great use in this task, where cancer tumours can be located via applying a specifically designed algorithm on digitised images. The significance of the convex shape algorithm is that it can be potentially used in medical applications to assist doctors interpret aspects of medical images. For example, algorithms help scan digital images, such as those from computed tomography.

It is found that the shape of the developed tumour in breast cancer has similar characteristics to a shape known as a convex shape (*WN* shape). This potentially implies that applying the concept of the maximum convex sum problem (MCSP) can result in accurately locating the lump.

To experiment the MCSP approach in the breast cancer application, we used Algorithms one and two, which were designed and implemented to locate the maximum convex sum, for example in a breast tumour image. We also developed software that reads the digital breast cancer image, whilst normalizing the values into positive and negative numbers. The normalization is carried out by subtracting a positive value from the original matrix; this vale is the overall mean of the matrix. The output from Algorithms one and two give the maximum convex sum and the corresponding matrix portion. After this step, a reassembly of the digital image occurs using the original normalized matrix that we calculated at the initial step. The Maximum convex sum output gives the position of the tumour.

The above process was applied to a mammogram digital image. The input to the algorithm is the image in Figure 6.a, which is the original image. The image in Figure 6.b is the result from the convex sum algorithm implementation. It can be clearly seen that the tumour was located using the MCSP approach.

K-OMCSP can be applied in monitoring the development of the cancer at its early stages. This is important because it gives oncologists an idea about the cancer behaviour, which can inform the patient of the best treatment therapy, such as surgery, chemotherapy, radiation therapy and immunotherapy/biologic therapy [28]. Oncologists would be able to recommend the treatment plane based on the patient's situation, how far cancer is spread, and other important factors like age and general health.

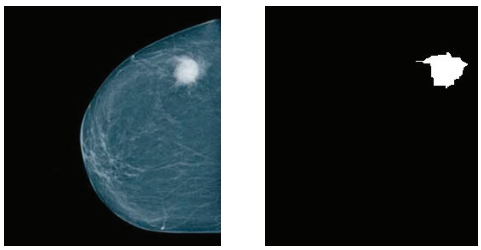

Fig.6: (a) a breast cancer tumour; (b) result from MCSP

5. *K* Convex sum problem

The *K* maximum sum can be classified into two types: the disjoint sum and the overlapping sum. Research has previously covered the disjoint techniques in finding the *K* convex sums. This paper covers a new approach for finding the *K* overlapping convex sums.

5.1 K-disjoint convex maximum sum algorithm

This section highlights work from previous research in relation to finding the maximum sum by using the first type of the *K* maximum sum problem, which is using the disjoint technique. Implementation of this algorithm can be found in [11].

The concept of the *K* disjoint technique is to consecutively find convex portions with maximum sums from the remaining portions of the given array. This means after discarding the first maximum convex sum portion, the remaining array gets processed in the algorithm. This procedure of finding the *K* maximum convex sums continues for the first, second, third,…, K^{th} maximum convex sum.

The implementation of an algorithm to compute the *K* disjoint maximum convex sum involves the following steps [11]: (a) The first maximum convex sum shape has to be found based on Algorithm (2). (b) After finding this convex shape from the first step, elements from the found portion of the maximum convex sum are given a value of minus infinity. (c) Algorithm (2) is used again to find the second maximum convex sum, and the elements of the array portion corresponding to this are replaced by minus infinity. (d) This process continues for the third and fourth until we reach the K^{th} maximum convex sum. This approach takes $O(Kn^3)$ time in a 2-D (n, n) array.

5.2 K-overlapping maximum convex sum algorithm

In [8], the tentative maximum sum and the currently accumulated sum are extended to *K*-tuples. In this algorithm we similarly extend single values of f_W and f_N to *K*-tuples expressed by F_W *and* F_N. Algorithm 1 is extended to *K*-tuples as follows. Suppose L is a *K*-tuple $(a_1, a_2, \ldots, a_K)$. For a single value x, L+x is defined by $L+x = (a_1+x, a_2+x, \ldots, a_K+x)$. L-x is similarly defined. For sorted *K*-tuples $L_1, L_2, \ldots, L_m$,, $\max\{L_1, L_2, \ldots, L_m\}$ are the largest *K* numbers in the merged list of $L_1, L_2, \ldots, L_m$ in sorted order. For two *K*-tuples L_1 and L_2, L_1+L_2 are the largest *K* numbers in sorted order from the set, Cartesian sum, $\{x+y \mid x \text{ is in } L_1 \text{ and } y \text{ is in } L_2\}$.

Algorithm (3):

F_W (0, [s,t])=(0, - ∞…, -∞) for all s ≤ t // 0 followed by (*K*-1) -∞
F_W (k, [s,t])=(- ∞, ….., -∞) for all s > t and for all k
for k=1 to n do
  for all intervals of [s, t] in increasing
   order of t-s where s ≤ t do
    F_W (k, [s,t]) =
    max{ F_W (k-1, [s,t])+sum[k,s,t] (extended case 1)
        F_W (k, [s+1,t])+a[s, k], (extended case 2)
        F_W (k, [s, t-1])+a[t, k]} (extended case 3)
    F_N is similarly computed from right to left.
/** Finalization **/
  For k=1 to n do
    For s=1 to n do
      For t=s to n do
        F_{WN} (k,[s,t])= (F_W (k,[s,t]) +F_N (k,[s,t]) – sum[k,s,t])

We call the array portion from index s to t in the k^{th} column the anchor column of F_{WN} (k,[s,t]), denoted by (k, s, t).

The computation of F_W and F_N takes $O(Kn^3)$ time, since the max operation is placed inside a triply nested structure, and the max operation takes O(K) time.

In Algorithm 3, Finalization takes $O(Kn^3)$ time. This is because finding F_{WN} (k,[s,t])= (F_W (k,[s,t]) +F_N (k,[s,t]) – sum[k,s,t]) is placed in a triply nested structure and the addition operation, "+", of two *K*-tuples can be calculated in O(K) time [29].

6. Results and analysis

Previous research used concepts of the MSP to solve the K^{th} overlapping maximum sums, whilst using the rectangular region. The region of the rectangular shape has limitations when used in the MSP because it lacks the flexibility to cover various data distributions. This paper's research adds to the existing knowledge in finding an optimal solution for the *K* maximum sums by using the convex shape in the overlapping case. A new algorithm was developed in this paper to find the *K*-OMCS.

To compare results obtained from using the rectangular shape with that of the convex shape, whilst using the overlapping case in finding the *K*-OMSP, two algorithms were implemented and run. The first algorithm is the rectangular shape *K*-OMSP [8], and the second algorithm is Algorithm (3) from this paper. The time complexity for the both algorithms is $O(Kn^3)$. To compare the algorithms' outcomes for the maximum gain, the same input matrix was used. The matrix elements were generated via using a random number generator that provides positive, negative and 0 numbers. The matrix size is 300x300. Outcomes from the obtained output show that the new algorithm gives a better solution, when compared to the rectangular shape results. Additionally, it was found that a percentage in the range of 40-70% represents an increase in the maximum sum for the first and second and third. These results are depicted in Figure 7.

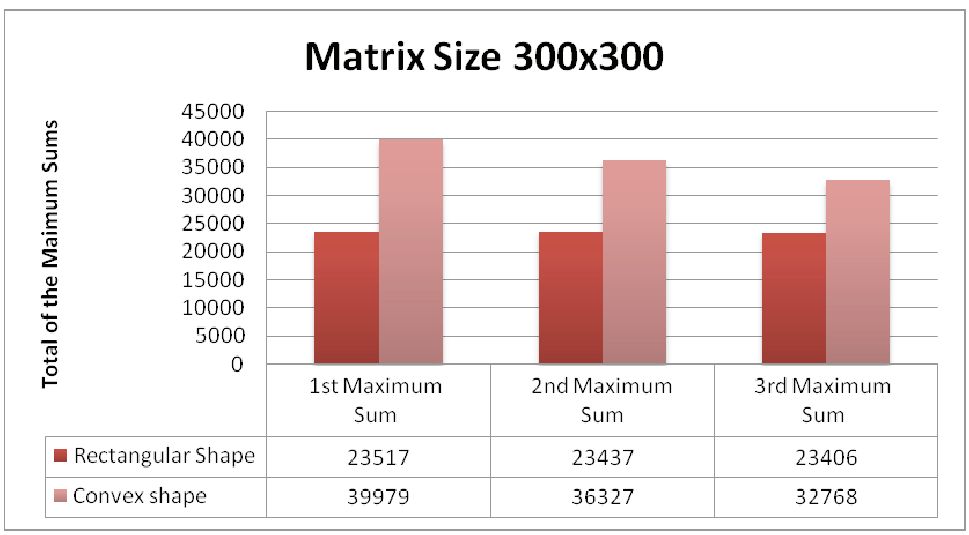

Fig. 7. comparison between first, second and third maximum sums in both rectangular and convex shape

7. Conclusions

Previous research used the rectangular shape to find the *K* overlapping sums. This paper presents a new algorithm that optimises results obtained from the rectangular *K* overlapping sums, which is using the convex shape in the overlapping case to find the ranking maximum sums (the first, second, third up to K^{th} maximum sum). The new algorithm uses the same time complexity, while returning an better solution; the current time complexity is $O(Kn^3)$ for both algorithms.

Our experiments show that the first, second and third maximum convex sum regions heavily overlap. If we impose the condition that subsequent sums cannot overlap with previous results more than 50%, we may have more practical algorithms for many applications.

Acknowledgements

The work on this paper has been an inspiring, often exciting, sometimes challenging, but always interesting experience. It has been made possible by many other people, who have supported us. Finally, thanks to Dr. Sung Bae and his valuable research.

References

1. J.Bentley, Programming pearls: algorithm design techniques. *Commun. ACM* 27 (9) (1984) 865.
2. J. Bentley, Programming pearls: perspective on performance. *Commun. ACM 27 (11)* (1984) 1087.
3. J. Bentley, Programming pearls: solutions for September's problem, *Common. ACM* 27 (11) (1984) 1090.
4. D.R. Smith, The design of divide-and-conquer algorithms, *Sci. Comput. Programming* **5** (1985) 37.
5. D. Gries, A note on a standard strategy for developing loop invariants and loops, *Sci. Comput. Programming* **2** (1982) 207.
6. U. Grenander, Pattern Analysis, Springer, New York (1978).
7. R. Agrawal, T. Imielinski, A. Swami, Mining association rules between sets of items in large databases, in: Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data, (1993) 207.
8. S. E. Bae, T. Takaoka, Algorithms for the problem of k maximum sums and a VLSI algorithm for the k maximum subarrays problem, in: Proceedings of the 7th International Symposium on Parallel Architectures, Algorithms and Networks, (2004) 247.
9. F. Bengtsson, J. Chen, Efficient algorithms for k maximum sums, Algorithmica 46 (1) (2006) 27.
10. C. Cheng, H. Liu and K. Chao, Optimal algorithms for the average-constrained maximum-sum segment problem, Information Processing Letters, 109 (2009) 171.
11. M. Thaher and T. Takaoka, An efficient algorithm for the k maximum convex sums, Proceedings of the 2010 Elservier ICCS 2010., Amsterdam, (2010) 1469.
12. S. E. Bae, Sequential and Parallel Algorithms for the Generalized Maximum Subarray Problem. PhD Thesis, University of Canterbury, (2007).
13. K. Perumalla and N. Deo, Parallel algorithms for maximum subsequence and maximum subarray, *Parallel Process. Lett. 5* (1995) 367.
14. T. Fukuda, Y. Morimoto, S. Morishita and T. Tokuyama, Data Mining with optimized two-dimensional association rules**,** *ACM Transactions on Database Systems* (TODS)**,** (2001) 179.
15. L. Allison, Longest biased interval and longest non-negative sum interval, *Bioinformatics* 19, (2003) 1294.
16. K.-Y. Chen and K.-M. Chao, Optimal algorithms for locating the longest and shortest segments satisfying a sum or an average constraint,, *Inform.Process. Lett.* 96, (2005) 197.
17. T.-H. Fan, S. Lee, H.-I. Lu, T.-S. Tsou, T.-C. Wang and A. Yao, An optimal algorithm for maximum-sum segment and its application in bioinformatics, in: Proc. Eighth Internat. Conf. on Implementation and Application of Automata, Lecture Notes in Computer Science, Vol. 2759, (2003) 251.
18. X. Huang, An algorithm for identifying regions of a DNA sequence that satisfy a content requirement, Comput. Appl. Biosci. 10, (1994) 219.
19. Y.-L. Lin, X. Huang, T. Jiang and K.-M. Chao, MAVG: locating non-overlapping maximum average segments in a given sequence, *Bioinformatics 19*, (2003) 151.
20. Y.-L. Lin, T. Jiang and K.-M. Chao, Efficient algorithms for locating the length-constrained heaviest segments with applications to biomolecular sequence analysis, J. Comput. System Sci. 65, (2002) 570.
21. L. Wang and Y. Xu, SEGID: identifying interesting segments in (multiple) sequence alignments, *Bioinformatics 19*, (2003) 297.
22. K. Fukuda and T. Takaoka, Analysis of air pollution (PM10) and respiratory morbidity rate using K-maximum sub-array (2-D) algorithm**,** Proceedings of the 2007 ACM symposium on Applied computing**,** Korea, (2007).
23. S. E. Bae and T. Takaoka, Improved algorithms for the k-maximum subarray problem, *Computer Journal 49*, (2006) 358.
24. F. Bengtsson and J. Chen, Efficient algorithms for the k maximum sums, ISAAC 2004, LNCS, *Springer 3341* (2004) 137.
25. S. E. Bae and T. Takaoka, Improved algorithm for the k- maximum subarray problem for small k, In Proc. of International Computing and Combinatorics Conference (COCOON 2005) , LNCS 3595, (2005) 621.
26. M. Bashar and T. Takaoka, Average Case Analsis of Algorithms for the Maximum Subarray Problem. Masters Thesis, University of Canterbury, 2007.
27. G. Cooper and R. Hausman, R. *The Cell: A Molecular Approach,* SA, US, 4 (2006) 1-200.
28. A. Alberts, A. Johnson, J. Lewis, M. Raff, K. Roberts and P. Walter. *Molecular Biology of the Cell*, GS, US, 4 (2002) 1-1463.
29. G. Frederickson, and D. Johnson, The complexity of selection and ranking in x+y and matrices with sorted rows and columns. *Journal of Computer and System Sciences 24* (1982) 197.