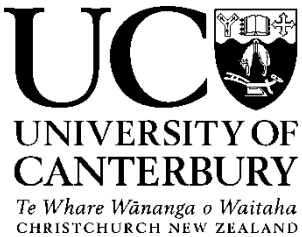
Efficient Algorithms for the Maximum Convex Sum Problem

A thesis
submitted in partial fulfilment
of the requirements for Degree
of
Doctor of Philosophy
in the
University of Canterbury
by
Mohammed Thaher

Supervision Committee

Prof. Tadao Takaoka, University of Canterbury	Supervisor
Prof. Jennifer Brown, University of Canterbury	Co-Supervisor



Department of Computer Science and Software Engineering

University of Canterbury

2014

This thesis is dedicated to
my parents, wife and daughters for their endless love and constant support.

Abstract

This research is designed to develop and investigate newly defined problems: the *Maximum Convex Sum* (*MCS*), and its generalisation, the *K-Maximum Convex Sum (K-MCS)*, in a two-dimensional (2D) array based on dynamic programming. The study centres on the concept of finding the most useful informative array portion as defined by different parameters involved in data, which is generically expressed in this thesis as the *Maximum Sum Problem* (*MSP*). This concept originates in the *Maximum Sub-Array* (*MSA*) problem, which relies on *rectangular regions* to find the informative array portion. From the above it follows that *MSA* and *MCS* belong to *MSP.* This research takes a new stand in using an alternative shape in the *MSP* context, which is the *convex shape*.

Since 1977, there has been substantial research in the development of the *Maximum Sub-Array* (*MSA*) problem to find informative sub-array portions, running in the best possible time complexity. Conventionally the research norm has been to use the *rectangular shape* in the *MSA* framework without any investigation into an alternative shape for the *MSP*. Theoretically there are shapes that can improve the *MSP* outcome and their utility in applications; research has rarely discussed this. To advocate the use of a different shape in the *MSP* context requires rigorous investigation and also the creation of a platform to launch a new exploratory research area. This can then be developed further by considering the implications and practicality of the new approach.

This thesis strives to open up a new research frontier based on using the *convex shape* in the *MSP* context. This research defines the new *MCS* problem in 2D*;* develops and evaluates algorithms that serve the *MCS* problem running in the best possible time complexity; incorporates techniques to advance the *MCS* algorithms; generalises the *MCS* problem to cover the *K-Disjoint Maximum Convex Sums (K-DMCS)* problem and the *K-Overlapping Maximum Convex Sums (K-OMCS)* problem; and eventually implements the *MCS* algorithmic framework using real data in an ecology application.

Thus, this thesis provides a theoretical and practical framework that scientifically contributes to addressing some of the research gaps in the *MSP* and the new research path: the *MCS* problem*.* The *MCS* and *K-MCS* algorithmic models depart from using the *rectangular shape as* in *MSA,* and retain a time complexity that is within the best known time complexities of the *MSA* algorithms. Future in-depth studies on the *Maximum Convex Sum (MCS)* problem can advance the algorithms developed in this thesis and their time complexity.

CONTENTS

List of Algorithms

List of Tables

List of Figures

Acknowledgments

The words “thank you” are not a strong enough statement of my gratitude. The work on this thesis has been an inspiring, often exciting, sometimes challenging, but always interesting experience. It has been made possible by many other people, who have supported me.

I wish to express sincere thanks to my Ph.D. supervisors. Thank you to Professor Tadao Takaoka for providing the inspiration for this research, and for his patience, guidance and support of my work. His passion for Algorithm Engineering and Formal Methods Research; dedication to ‘good science’; enthusiasm to challenge the advanced algorithms and an unrelenting attention to detail are an inspiration to me and to all of his students. Thank you also to my co-supervisor, Professor Jennifer Brown, for her guidance and support during this project.

Thank you to all of my family and friends for your unwavering support and encouragement. A special thanks to my parents, Shaban Atieh Thaher and Mudellah Abdel Fattah Thaher, for their emotional support and encouragement throughout this Ph.D. and my entire education. Thank you also to my brothers and my parents-in-law for their endless support and encouragement. Special thanks to my wife, Balsam Obaid, I wish to express my deepest gratitude for her constant support, understanding and love. Thank you to my lovely daughters (Deemah and Razan) for the cheerful moments during my study; their smiles kept me going forward.

To all of my colleagues, both staff and students, in the Department of Computer Science and Software Engineering (CSSE) at the University of Canterbury. You have made my time in the department that much more enjoyable. I feel honoured to have been a part of this department.

I would like to thank both Associate Professor Jon Harding and Danladi Umar from the *Freshwater Ecology Research Group (FERG)*, University of Canterbury, for supporting my research and for their contribution in the application section of my thesis. I would also like to thank all the people behind our collaborative work with biological scientists. Thanks go to Dr Hazel Chapman, Director of Nigerian Montane Forest Project Ngel-Nyaki and Professor M.J., Winterbourn for identifying the benthic invertebrate’s fauna in the Nigerian streams.

Thanks to Linda Morris, Pete McHugh, Katie McHugh, Frank Burdon, Milen Marinov (University of Canterbury) and Misa Zubairu (NMFP) who provided additional assistance at various stages during our collaborative work. Support for this collaboration came from the University of Canterbury, Gombe State University and the Education Trust Fund, Nigeria.

Thank you to the programme of *Optimization and its Applications in Learning and Industry* (*OptALI*), and the people who facilitated my research visit to the University of Göttingen in Germany (August 2012): Professor Anita Schöbel, Professor Jutta Geldermann, Dr. Marie Schmidt, Dr**.** Marc Goerigk, Ruth Hübner, Robert Schieweck, and Sybille Dühring. Thank you to Professor Horst W. Hamacheris, Professor Stefan Ruzika, Dr. Lara Turner and Bob Grün, of the Technical University of Kaiserslautern, Germany who also facilitated my research visit in September 2012. Thanks also to Professor Jesper Larsen from Technical University of Denmark for his supported of my visit in October 2012. Special thanks to Professor Matthias Ehrgott, Olga Perederieieva, Antony Phillips and Siamak Moradi from Auckland University for their support during the *OptALI* Summer School in Germany on Optimization with a focus on Robust Optimization.

Special thanks to Dr Sung Eun Bae for his support and valuable research. Many thanks to the following people: Professor Takeshi Fukuda, Professor Yasuhiko Morimoto, Professor Shinichi Morishita, Professor Takeshi Tokuyama, Dr Elena Moltchanova, Philippa Drayton, and ChiangHau Tay.

Last, but not least, thank you to the *Royal Society of New Zealand*, Canterbury branch (*RSNZ*), and *OptALI* for their financial contributions to attend and present at the conferences and workshops.

I couldn't have done this without all of you. Thank you!

Chapter One

[*Introduction*]

Introduction

The *Maximum Sub-Array* (*MSA*) problem is used to find the most useful informative array portion that associates with different parameters involved in data [1, 2]. *MSA* was initiated to resolve pattern recognition problems and used as a likelihood estimator (the most-likely area for certain property) to find the brightest portion in an image [3]. Since then, the topic has been expanding to solve other problems that stem from the original concept, such as those associated with finding the ranking of the maximum sums. Let us define the *Maximum Sum Problem* (*MSP*) as one that finds some portion of the given array that maximises the sum in it. In this sense *MSA* belongs to *MSP.*

MSA can be computed for one-dimensional (1D) and two-dimensional (2D) arrays that have positive and negative numbers. The 1D case can be called the *Maximum Sum Segment Problem* (*MSSP*) which can be solved in linear-time using Kadane's algorithm [1]. This involves a process of selecting a segment of consecutive array elements that have the largest possible sum compared to all other segments in presented data [1, 2]. In the two-dimensional case, the task is to find a 2D sub-array that has the maximum sum compared to all other segments in a matrix [4].

Finding an *MSA* solution with optimum time complexity is being investigated by researchers worldwide and has been a challenge over the past three decades [1-41]. Since 1977, various techniques have been developed to advance algorithms to find a solution with the optimum computation time. For example, finding *MSA* exhaustively took $O(n^6)$ time in an earlier work [3]. In 1998, the time complexity was reduced to subcubic time $O(n^3(\log \log n/\log n)^{1/2})$ by applying the *Distance Matrix Multiplication* (*DMM*) method [12], and its algorithm was then simplified in 2002 [14]. It was not until 2012 that an $O(n^3 \log \log n/\log^2 n)$ time was achieved [41].

In previous research focused on *MSA*, algorithms for searching *rectangular regions* have been primarily used for finding the maximum sub-arrays [1-41]. There has been almost no investigation of using alternative shapes in implementing the *MSP* framework. In this research, the *MSP* approach takes a new turn to explore and investigate such a component, using a shape that will potentially improve the maximum sum outcomes for the *MSP* problem. It uses the *convex shape*. This new approach provides a significant step towards discovering alternative robust methods to solving *MSP* problems. The newly defined problem in this research is called the *Maximum Convex*

Sum (*MCS*). Below is a simple example to explain the difference between the maximum sums obtained using *MSA* and *MCS* approaches, as shown in Figure 1.1.

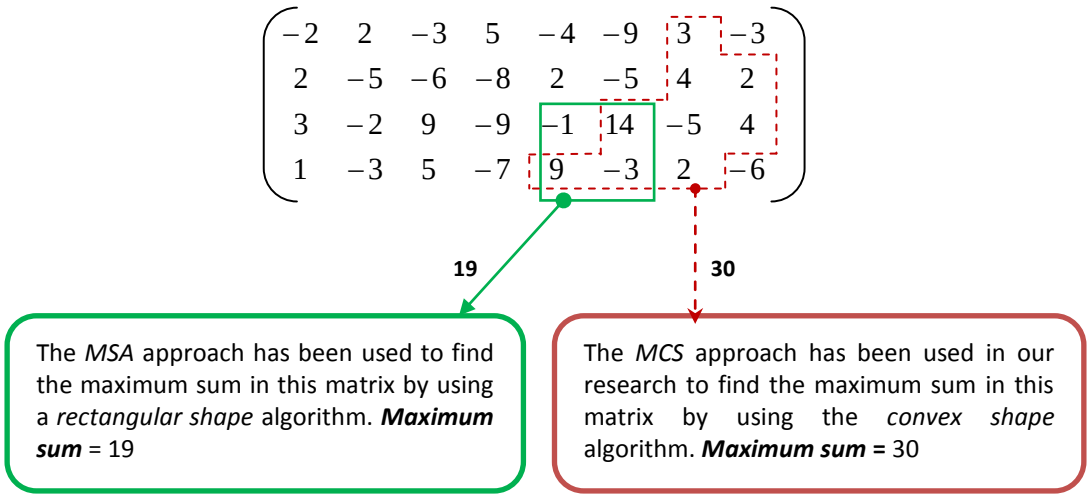


Figure (1.1): Maximum sums obtained from using two shapes in a matrix: *rectangular shape* and *convex shape*. The algorithms used to illustrate this example have been adapted from those that appear in the following chapters.

In addition to investigating a new shape in the *MSP* context, this research advances the *Maximum Convex Sum* problem to cover generalised cases. The research establishes the *K-Disjoint Maximum Convex Sum (K-DMCS)* and the *K-Overlapping Maximum Convex Sum (K-OMCS)* problems. The newly designed algorithms achieve $O(Kn^3)$ time. Furthermore, using real datasets from ecology, this research implements the newly developed *K-DMCS* method to finding *K*-maximum threshold values for investigating the relationships between different environmental elements. We collaborated with the *Freshwater Ecology Research Group (FERG)* at the University of Canterbury to investigate the effects of change in land use on benthic stream communities in the highland tropical streams of Nigeria. The use of an algorithmic approach in a real-life study contributes to both the study of ecology and the study of *MSP* in a practical context.

The organisation of this chapter is as follows: Section 1.1 deals with the thesis goals and motivation; Section 1.2 discusses the specific contributions of this study; Section 1.3 outlines the publications that have resulted from this research; Section 1.4 acknowledges the awards and funds received as a part of this project; Section 1.5 lists the internships and research visits; and Section 1.6 provides a very brief chapter outline for the thesis.

1.1 Thesis Goals and Motivation

The main goal of this thesis is to investigate, develop and evaluate a framework for solving the *MCS* problem. This framework can be broadly applicable and used to find the maximum sub-arrays in 2D as the most promising portions in datasets. The framework is based on the challenging dynamic programming approach. Previously, *MSA* frameworks have approached the problem of finding maximum sums using *rectangular regions*. This is explored and widely covered within many different research areas [1-41]. There is significant scope, however, for other shapes to be used in the *MSP* context; this has been rarely discussed in previous studies. The new research scope can potentially enhance outcomes for the *MSP*. The newly designed algorithms for the *MSP* using an alternative shape can take into account achieving time complexity that is in the current range of the *MSA* algorithms' time complexity using the *rectangular shape*. Thus introducing an alternative framework has been the primary motivation for our investigation of the *MCS* problem and more specifically our proposition of using the *convex shape* in the *MSP* framework. This opens up a new research area in terms of using alternative shapes in the *MSP* context.

The specific goals of this thesis are to:

- Develop an independent algorithmic framework to find regions that maximise the sum within the same time complexity of the current algorithms for *MSA*.
- Explore and integrate various techniques to speed up the computation time for the newly developed algorithms.
- Generalise the *MCS* problem to cover the *K-Disjoint Maximum Convex Sums (K-DMCS)* and the *K-Overlapping Maximum Convex Sums (K-OMCS)* problems in 2D, having the best possible time complexity.
- Evaluate the proposed *MCS* algorithms using experimental analysis.
- As a practical application, implement the theoretical algorithmic framework using real data in an ecology application.

1.2 Ph.D. Contributions

The main contributions of this Ph.D. thesis are:

- Investigating and utilising the *convex shape* in the *MSP* context and creating a new *MCS* framework. This research simplifies the algorithm of T. Fukuda *et al.* [42] to find the maximum convex sum using the *bidirectional* approach. The simplified algorithm, using the

convex shape to find the maximum sums, returns a sum at least as large as that returned by maximising over all *rectangular regions* (*Chapter 3*).

- Evaluating the simplified version of the *MCS* algorithm and selectively applying pre-processing and post-processing techniques to boost the applicability of the newly defined algorithms. The *prefix sum* method enhances the algorithms' performance by bypassing surplus *addition* operations. The *backtracking* technique, to determine the shape boundaries, permits the generalisation of the newly developed *MCS* algorithm to cover cases involving computing the *K* maximum sums (*Chapter 3*).

- Developing new algorithms using the *MCS* framework and generalising those for the *K-Maximum Convex Sum* (*K-MCS*), which finds the *K* maximum *convex regions* in an input two-dimensional array. This research develops two *K-MCS* algorithms to cover the *K-Disjoint Maximum Convex Sum* (*K-DMCS*) and the *K-Overlapping Maximum Convex Sum* (*K-OMCS*) concepts. The former requires the *convex regions* to be disjoint while the latter covers the case where the *convex regions* to overlap – both algorithms run in $O(Kn^3)$ time *(Chapters 3 and 4).*

- Analysing experimental implementation of the devised algorithms using simulated data. In this research experimental comparisons are made between algorithms developed as part of the newly defined *MCS* problem and their corresponding counterparts of the traditional *MSA* algorithms *(Chapters 3 and 4).*

- Applying and assessing the validity of the defined *MCS* problem in a real-life application. The *K-DMCS* algorithm is applied to an ecology setting to find the *K*-maximum regions that represent the impact of relationships in data. The research investigates the effects of change in land use on benthic stream communities in highland tropical streams of Nigeria. The results of this research are compared with those obtained using the traditional method (*rectangular shape* in the *K-DMSA)*. Using the *K-DMCS* algorithm, biologically significant relationships between the studied variables have been successfully identified, which essentially demonstrates the robustness of the tested approach (*Chapter 5*).

1.3 Ph.D. Publications

1.3.1 Journal article

- M. Thaher, T. Takaoka, (2013) Efficient algorithms for the Maximum Convex Sum problem. S*ubmitted* to *The Computer Journal*.

1.3.2 Conference contributions (peer-reviewed papers)

Materials from this thesis have been published in peer-reviewed articles in International Conferences, ranked 'A'[1]. The ranking is based on the *Excellence in Research for Australia 2010* (*ERA 2010*) evaluation conferences worldwide list*,* which can be found following this link: http://www.core.edu.au/. The publications are:

- M. Thaher, T. Takaoka, (2010) An efficient algorithm for the k maximum convex sums. In *Proceedings of the International Conference on Computational Science (ICCS)*, 1-2 June, 2010, Amsterdam, Netherlands, Procedia CS 1(1): 1474-1483 (2010), (*Elservier*). (*Chapter 3*)

- M. Thaher, T. Takaoka, (2011) An efficient algorithm for computing the K-Overlapping maximum convex. In *Proceedings of the International Conference on Computational Science (ICCS)*, 1-3 June, 2011, Singapore, Singapore, Procedia CS 1(1): 1288-1295 (2011), (*Elservier*). (*Chapter 4*)

- M. Thaher, T. Takaoka, (2013) Application of the maximum convex sum algorithm in determining environmental variables that affect Nigerian highland stream communities. In *Proceedings of the International Conference on Computational Science (ICCS)*, 5-7 June, 2013, Barcelona, Spain, Procedia CS 1(1): 909-918 (2013) (*Elservier*). (*Chapter 5*)

- M. Thaher, T. Takaoka, (2012) Improved algorithms for the K overlapping maximum convex sum problem. In *Proceedings of the International Conference on Computational Science (ICCS)*, 1-3 June, 2012, Omaha, USA, Procedia CS 1(1): 754-763 (2012) (*Elservier*). (*Chapter 6*)

[1] The Australian Research Council (ARC) regards 'A' conferences as equivalent to A* and A journals

1.3.3 National conferences (papers peer-reviewed by students)

- M. Thaher, T. Takaoka, (2008) Maximum Convex Sum Problem. In *the 6th New Zealand Computer Science Research Student Conference*, University of Canterbury, Christchurch, 14-16 April 2008 (*NZCSRSC* 2008).

- M. Thaher, T. Takaoka, (2010) A study of the Maximum Convex Sum Problem. In *the 8th New Zealand Computer Science Research Student Conference*, Victoria University, Wellington, 12-15 April 2010 (*NZCSRSC* 2010).

- M. Thaher, T. Takaoka, (2011) Efficient algorithm for the Maximum Convex Sum Problem. In *the 9th New Zealand Computer Science Research Student Conference*, Massey University, Palmerston North, 18-21 April 2011 (*NZCSRSC* 2011).

- M. Thaher, T. Takaoka, (2012) Improved algorithm for the *K-OMCSP*. In *the 10th New Zealand Computer Science Research Student Conference*, University of Otago, Dunedin, 11-13 April 2012 (*NZCSRSC* 2012).

- M. Thaher, T. Takaoka, (2013) Application for the *K-DMCSP*. In *the 11th New Zealand Computer Science Research Student Conference*, Waikato University, Hamilton, 15-19 April 2013 (*NZCSRSC* 2013).

1.3.4 Other participations

- S. Weddell, T. Read, M. Thaher, and T. Takaoka, (2013) Maximum subarray algorithms for use in astronomical imaging. *Journal of Electronic Imaging*, vol. 22, 2013.

- M. Thaher, T. Takaoka, (2013) Investigating Efficient Algorithms for Maximum Convex Sum Problem. Presented at a symposium of *Optimization and its Applications in Learning and Industry (OptAli)*, 27-31 August, 2012, Gottingen, Germany.

- M. Thaher, T. Takaoka, (2010) *MCSP show case*. *2010 Departmental Post-Grad conference*, University of Canterbury, Christchurch, 31 August-1 September 2011.

- M. Thaher, T. Takaoka, (2011) Investigating efficient algorithms for the maximum convex sum problem. 2011 *Departmental Post-Grad conference*, University of Canterbury, Christchurch, 1- 2 September 2011.

- University of Canterbury *Show Case* 2010.

- University of Canterbury *Show Case* 2013.

1.4 Awards and Funds

- Best Ph.D. presentation in COMS2013, ICCS 2013, Barcelona, Spain.
- The *Royal Society New Zealand* fund, Canterbury branch (RSNZ).
- The *Optimization and its Applications in Learning and Industry* (*OPTALI*) program fund.

1.5 Internships/Research visits

- **August, 2012: Research visit to the Institute for Numerical and Applied Mathematics - Research Group Optimization at Georg-August University Goettingen**, **Goettingen**, **Germany**. I attended the *OptALI* Summer School where I received training which helped to optimise my frame work and understand different methods.

- **September, 2012: Research visit to the Department of Mathematics. A research visit to Technical University of Kaiserslautern in Germany** - Optimization research group. I attended workshops as well as presented my research.

- **October, 2012: Research visit to the Department of Management Engineering. A research visit to Denmark Technical University** - Optimization research group. I received training on different methods in optimisation.

1.6 Outline of the thesis

Chapter 2 – *MSA* Background: covers a comprehensive review of the *MSA* theoretical and algorithmic frameworks.

Chapter 3 – The *K-DMCS* problem: describes the newly defined *MCS* problem and its generalisation to cover the disjoint case.

Chapter 4 – The *K-OMCS* problem: describes the generalisation of *MCS* to cover the overlapping case.

Chapter 5 – Application of the *MCS* (*K-DMCS* algorithm) to an ecology problem.

Chapter 6 – Conclusion and future work: provides a summary of the thesis contributions, limitations and future directions.

Chapter Two

[***Background information***]

This chapter provides a comprehensive review of the *MSA* theoretical and algorithmic frameworks. The chapter is organised as follows: Section 2.1 discusses the state-of-the-art of the *Maximum Sub-Array* (*MSA*) problem*;* Section 2.2 reviews the *MSA* problem in 1D and 2D; Section 2.3 provides a summary of the main points in the chapter.

2.1 History of the *Maximum Sub-Array* (*MSA*) problem

MSA is an approach that many researchers have investigated since 1977 and is an area of research with a plethora of publications [1-41]. Researchers worldwide have contributed to the advancement of this topic with research from IBM, AT & T Bell Labs, Kenstrel Institute, Cornell University, University of Minnesota, University of Alabama, National Taiwan University, University of Tokyo, University of Central Florida, University of Washington and University of Canterbury.

UIf Grenander at Brown University initially encountered problems in pattern recognition in 1977 [3] and these problems led to more investigation into *MSA* algorithms [2]. One of the challenges that Grenander faced was finding the maximum sum over all *rectangular regions* of a given $m \times n$ array of real numbers. The maximum likelihood estimator of a certain kind of pattern in digitised pictures is represented by the maximum sum or maximum sub-array [3]. Grenander initiated the computation of the maximum sum by implementing an algorithm of $O(n^6)$ time for an array size of $n \times n$ [3]. Following this algorithm, in an attempt to reduce the time factor and to gain more understanding of the structure, he simplified the problem to 1D [3]. The input is a one-dimensional array of n real numbers; the output is the maximum sum obtained in any consecutive portion (sub-array) of the input. Grenander managed to obtain $O(n^3)$ time using a one-dimensional array, but eventually terminated his research on the maximum sub-array to solve the problem of pattern matching, because of the high complexity of the attempted algorithms. The seed that he planted, however, has proven to be fruitful.

Following Grenander's attempts, Shamos and Bentley have improved the time complexity to $O(n^2)$ and later implemented an *O(n log n)* time algorithm. Kadane [4] and Gries [6] presented other linear time algorithm solutions to the time complexity problem. The two-dimensional version ($n \times n$ array) to solve the *MSA* problem has been achieved in $O(n^3)$ by extending Kadane's algorithm [2]. Smith presented an $O(n)$ time for the 1D *MSA* problem, and achieved an $O(n^3)$ time solution for the 2D problem using the *divide-and-conquer* technique [7].

The current state-of-the-art in terms of the optimal time for the 1D version is $O(n)$ [2]. The time complexity of $O(n^3)$ for 2D *MSA* has been the best achievable time until Tamaki and Tokuyama [12] devised an algorithm which achieves a subcubic time of $O(n^3(\log \log n/\log n)^{1/2})$. They achieved this time by adopting the *divide-and-conquer* technique, and applying the fastest known *Distance Matrix Multiplication* (*DMM)* algorithm [5]. Takaoka later simplified this algorithm and presented an even faster *DMM* algorithm [14]. More recently, Han and Takaoka achieved $O(n^3 \log \log n/\log^2 n)$ time in [41].

Bae studied the *MSA* problem using the *rectangular shape* in 2D [15-21]. He has presented methodologies and techniques to speed up computation time for the *K-Overlapping Maximum Sub-Array (K-OMSA)* and the *K-Disjoint Maximum Sub-Array* (*K-DMSA)* problems. Bae designed mesh algorithms for the 2D *MSA* and *K-OMSA* problems [15, 16]; established $O(K^2 + n \log K)$ time for the 1D *K-OMSA* through sampling before candidate generation [17]; applied the sampling for the 1D *K-OMSA* [18] to the 2D case; and designed a tournament for the next *K-DMSA* and extended this to the 2D *K-DMSA* [19].

Bengtsson and Chen studied *K-OMSA* using the *rectangular shape* in sorted and non-sorted orders of the final *K maximum sums* and proposed an $O(n + k \log n)$ time algorithm for ranking the *K maximum sum* sub-sequences [22-24]. In a further study, Chen *et al.* [25, 26] achieved the equivalent complexity of $O(n + K \log \min (K, n))$ time. Bashar and Takaoka developed an algorithm to generalise the *MSA* by using average case analysis whilst utilising the *rectangular shape* [27]. In a different study, Ruzzo and Tompa [28] suggested a linear time algorithm that finds the *K-DMSA* in a one-dimensional array by locating all high scoring segments.

In addition to the aforementioned research, T. Fukuda and his colleagues from IBM discussed data-mining techniques based on the association rules of two numeric attributes and one Boolean value in an acceptable region [42]. They proposed an algorithm to compute regions that give optimal association rules for *support* and *confidence*. Their main aim was to generate two-dimensional association rules that represented the dependence on a pair of numeric attributes. For example, the relationship between the attributes will depend on the objective condition [42]. T. Fukuda's study was based on Agrawal and his colleagues' research to find all *confidence rules* in vast data [43]. Improved versions of Agrawal and his colleagues' algorithm have been reported [44, 45], in which the *rectilinear convex shape* was investigated for the first time. Furthermore, T. Fukuda *et al*. [46-53] considered three classes of geometric regions: *x-monotone regions, y-monotone regions*

and *rectilinear regions*. The key findings of this study, based on thorough investigations, indicate the limitation of using the *x-monotone* and *y-monotone regions.* These shapes are likely to over fit the training database dramatically, and therefore fail to give a good prediction on the unseen dataset [42, 46, 53, 54]. Sprague's research [29] attempted to investigate extracting optimal association rules over numerical attributes by using the *anchored convex shape*. His technique, however, involves a limited shape that lies at the edges of the matrix and does not take into account that the solution may lie in the inner portions of a 2D array. Researchers continue to use the *rectangular shape* in research and applications of the *MSA.*

K. Fukuda and Takaoka's research discussed applications for using 2D *MSA* algorithms [30, 31]. In their study, they used a *K-DMSA* algorithm in a health and environmental science setting. They investigated the associations between particulate air pollution and acute respiratory hospital admission counts in Christchurch, New Zealand [55-60]. Data gathering involved collecting daily measurements of particulate air pollution with a diameter less than 10 μm (PM_{10}) over a four year period (1998-2002). Additionally, counts of acute respiratory hospital admissions for all age groups from 0 to 98 years in Christchurch were collected. The morbidity rate was extracted from hospital records of residents within a 2 km radius of the air pollution monitoring site, which was located in a residential area. The *K-DMSA* detects associations among different (PM_{10}) levels and the variation according to different age groups, gender, and different seasons or weather changes. They stated in their study that the *K-DMSA* is potentially an encouraging methodology to investigate how various air pollution levels are related to health or climate, and this can contribute to policy making with regard to the problem of air pollution. This study shows the potential for a transition from traditional statistical approaches to finding rectangular portions which represent data associations.

In addition to the above study, *MSA* algorithms have been applied to other real-life applications [32-35]. One such example is the application of *K-DMSA* algorithm to investigate suicide rates in association with several factors [32]. This is used to identify a range of thresholds to explain the maximum associations of the suicide rates, age groups, and social factors (including bankruptcies, unemployment, divorce and orphan numbers). The study was conducted in collaboration with the Ministry of Health in New Zealand, and the data have been collected for over 20 years (1983-2003). The study detected the range of threshold criteria (maximum sums) to describe the maximum associations of the study variables. This included determining associations for specific ranges and factor levels by introducing a new approach: changing the weight parameter, *w*. The weight *w* is subtracted from the given array elements. This is normally the mean value, but for

fine tuning various *w* values are used in [31, 32]. The general trend is that association patterns obtained from suicide rates have been found to be similar across the studied social factors for females and males. The study reveals that despite the general trends, there are differences between the two genders. For example, the age group for the suicide range in females has a wider age range (15-64 years) compared to the males (15-59). In this study, using the *K-DMSA* has led to detection of detailed threshold associations between the suicide age groups and levels of the study factors [31, 32].

Another study that used *MSA* algorithms is an investigation of the spatial distribution patterns of hawthorn, grown over a 100 year period, on Porters Pass in the South Island of New Zealand [33]. Results show how the weed's distribution pattern has changed over the 40 year period from 1966 to 2008. *MSA* algorithms have further been used in other areas such as bioinformatics, genomic sequence analysis (GSA), computer vision (CV) and data mining (DM). Readers may refer to [4, 35, 36] for details regarding these algorithms.

More recently, *MSA* algorithms were applied to optical and radio telescope applications (e.g. wavefront detection and slope estimation, and efficient image processing) [34]. *MSA* algorithms were used to develop a centroid estimator for improving the computational efficiency to compensate for atmospheric turbulence in real-time. These algorithms improved performance in terms of the number of clock cycles compared to using the serial processing configuration. The input image size becomes limited, however, because to achieve the improved performance there is an increased cost of the *MSA* algorithm time complexity. The *MSA* algorithms were also applied to astronomical images, which were obtained for the Australian square kilometre array pathfinder (ASKAP) project. There was a need to find *MSA* portions of the radio telescope images to efficiently classify source objects. The use of *MSA* algorithms demonstrated operational gain, but this has a limitation in terms of the present loading method, and is still under investigation [34].

The preceding paragraphs outline the history and the state-of-the-art research and development of *MSA* problems and algorithms; the succeeding sections address the background and conceptual information of *MSA* algorithms.

2.2 Overview of the *MSA* problem in 1D and 2D

2.2.1 Finding *MSA* in 1D

In a given array $a[1..n]$ containing real numbers with a mixture of zero, positive and negative numbers, the maximum sub-array contains consecutive array elements having the greatest sum. Let $\mathcal{F} = max(S)$, where $max(S)$ is the operation that selects the maximum sum of contiguous elements in a list S, which is given by Equation (2.1),

$$S = \left\{ \sum_{x=i}^{j} a[x] \quad | \quad 1 \le i \le j \le n \right\} \quad \ldots\ldots\ldots\ldots\ldots\ldots \quad (2.1)$$

There are different algorithms that can compute the maximum sum in linear time $O(n)$ [4]. Kadane from Carnegie-Mellon University invented the first algorithm to solve the *MSA* problem in *O(n)* time [1, 2]. His well-known algorithm, Algorithm (1), uses the accumulation/reset method to find the largest maximum contiguous subsequence amongst all other segments in a 1D array of real numbers.

Algorithm (1): Kadane's algorithm $O(n)$ /* 1D *MSA* $a[k..l]$ of $a[1..n]$ */

1: Initialisation: $(k, l) \leftarrow (1, 0)$ (empty) ; $s \leftarrow 0$; $t \leftarrow 0$; $j \leftarrow 1$;

2: for $i \leftarrow 1$ to n do begin

4: $t \leftarrow t + a[i]$

5: if $t > s$ then begin $(k, l) \leftarrow (j, i)$; $s \leftarrow t$ end;

6: if $t \le 0$ then $t \leftarrow 0$; $j \leftarrow i + 1$ end /*reset the accumulation*/

7: end for

Algorithm (1) scans a given 1D array ($a[1..n]$), accumulating a tentative sum in t. In the case of t being greater than the current maximum s, s will be updated by t. If t becomes negative, it will be reset to zero. The variables k and l keep track of, respectively, the beginning and ending positions of the sub-array that has the sum s. The indices (k, l) and maximum sum s are initialised to zero. If all

of the elements of the array $a[1..n]$ are negative, the resulting s will be zero having an empty interval. An illustrated example is given in Figure 2.1.

Figure (2.1): An example showing Kadane's algorithm in 1D.

The computing time for Algorithm (1) is linear ($O(n)$). If we consider the time complexity as a function of the input size n, then the time complexity will be $O(f(n))$; that is proportional to $f(n)$.

2.2.2 Finding *MSA* in 2D

2.2.2.1 *Strip separation*

The *MSA* in a two-dimensional array can be computed for a given array of size $m \times n$ (we assume that $m \leq n$). The 2D input matrix has real numbers such as $a[1, \ldots, m, 1, \ldots, n]$. An approach to solve an *MSA* problem in 2D is to extend Algorithm (1), which is the 1D Kadane's algorithm [2]. The extended Kadane's algorithm used a *rectangular portion* to find the maximum sub-array in 2D, which is given by Algorithm (2) [4]. Example (2.1) depicts the extended Kadane's algorithm to find *MSA* for a two-dimensional array. Suppose we find a rectangular sub-array such that $a[k..i, l..j]$, where coordinates (k, l) and (i, j) are indices for tracing the portion area. In this example, we let index (1, 1) to be the top-left corner. The located rectangular region in this example is given by (3, 4) and (5, 6); and the sum is 18.

Example (2.1): Let a be given by

$$a = \begin{pmatrix} -2 & 2 & -3 & 5 & -4 & -9 & 3 & -3 \\ 2 & -5 & -6 & -8 & 2 & -5 & 4 & 2 \\ 3 & -2 & 9 & -9 & \begin{pmatrix} -1 & 13 \\ 8 & -2 \end{pmatrix} & -5 & 4 \\ 1 & -3 & 5 & -7 & & 2 & -6 \end{pmatrix}$$

The extended Kadanes's algorithm employs the *strip separation* approach to find a maximum sub-array in 2D [4]. This process involves horizontally separating the two-dimensional

array into every possible row *strip*, followed by applying the 1D Kadane's algorithm on each *strip*. The principal operation of the extended Kadane's algorithm is shown in Figure 2.2.

In this figure, the rectangular area that is defined by the bottom-right corner ($x1$, $y1$) and top-left corner ($x2$, $y2$) would have a continually updated maximum sum, S. As for the rectangular area defined in the *strip* from x to z, the tentative solution, t for this 1D area is from j to i. While iterating through the row bounded by x and z, we assign the most updated maximum sum in the current 1D *strip* to s. In other words, S keeps track of globally optimum, whereas s is for locally optimum. Employing the *strip separation* approach in 2D, *MSA* incorporates a triply nested structure as shown in Algorithm (2), which results in a total computational time of $O(m^2n)$. When $m = n$ the time complexity becomes a cubic time $O(n^3)$ [2].

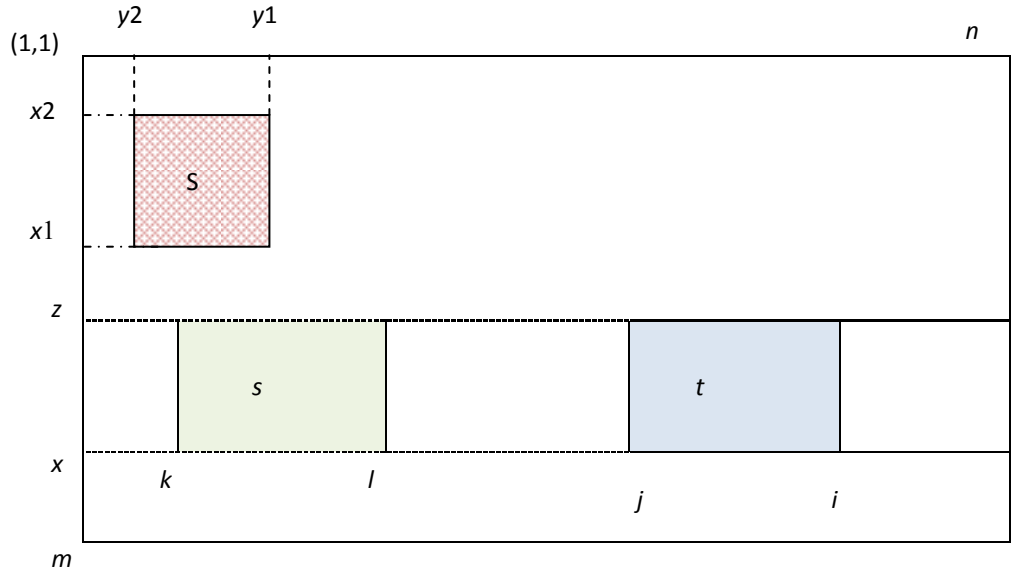


Figure (2.2): An example demonstrating Kadane's algorithm in 2D.

Algorithm (2): 2D version of Algorithm (1) (the extended Kadane's Algorithm); $O(m^2n)$

1. Initialisation (($x1$, $y1$),($x2$, $y2$)) ← ((1, 1),(0, 0)); S ← 0; /*initial candidate is empty */

2. for z ← 1 to m do begin

/** initialise *column*[][]**/

3. for i ← 1 to n do begin *column*[z-1][i] ← 0;

```
4.    for x ← z to m do begin
5.      t ← 0; s ← 0; (k, l) ← (1, 0);              /* initial candidate is empty */
6.      j ← 1
7.      for i ← 1 to n do begin
8.          column[x][i] ← column[x-1][i] + a[x][i];
9.          t ← t + column[x][i];
10.         if(t > s ){ s ← t ;(k, l) ← (i, j); }
11.         if(t < 0){ t ← 0; j ← i+1; }          /*reset the accumulation*/
12.     end for
13.   if (s > S) {S ← s; x1 ← x; y1 ← k; x2 ← z; y2← l;}
14.  end for
15.  end for
```

2.2.2.2 *Distance Matrix Multiplication*

In this section, we will briefly review the *Distance Matrix Multiplication (DMM)* framework to compute the *MSA* in 2D. Takaoka [14] simplified the algorithm of Tamaki and Tokuyama [12] and achieved a subcubic time. He used two concepts to improve the time complexity to subcubic for the 2D *MSA* problem: firstly, he translated *prefix sums* as distances for *DMM*; and secondly, he used the *divide-and-conquer* (splitting and merging) technique.

The *DMM* method was used in Takaoka's algorithm [14] as the key computation process to compute the distance product, which is outlined as follows. For two n x n matrices, $A = [a_{i,j}]$ and $B = [b_{i,j}]$, the product $C = AxB$ is defined by,

$$c_{i,j} = \min_{1 \le k \le n} \left\{ a_{i,k} + b_{k,j} \right\} \ (i,j = 1,..,n) \qquad (2.2)$$

In Equation (2.2), *A* and *B* are called distance matrices, and the right-hand side operation is called *DMM*. The *DMM* algorithm in [14] runs in $O(n^3(\log \log n / \log n)^{1/2})$ time, which is subcubic. The most recent improvement of this is given in [41], which has $O(n^3 \log \log n/\log^2 n)$ time. For the purpose of this study, we refer to [14] in this chapter to explain improving the time complexity for the newly defined problem of this study. Algorithm (3) outlines the 2D *MSA* algorithm using the *DMM* [14].

Algorithm (3):Takaoka's Algorithm for two-dimensional array; $O(n^3(\log \log n/ \log n)^{1/2})$ time

1: If the array becomes one element, return its value.

2: Otherwise, if $m > n$, rotate the array 90 degrees.

/* Now we assume $m \leq n$ */

3: Let A_{Left} be the solution for the left half.

4: Let A_{Right} be the solution for the right half.

5: Let A_{Centre_column} be the solution for the column-centred problem

/* The column-centred problem is to obtain a solution array that crosses over the vertical centre line */

6: Let the solution be the maximum of these three solutions (A_{Left}, A_{Right}, A_{Centre_column}) .

The time complexity is improved by transforming *prefix sums* to distances. The *prefix sum* of a one-dimensional array *a* at position *i*, is given by *s*[*i*], which is the sum of *a*[1], … , *a*[*i*]. The *prefix sum* array is computed in linear time *O*(*n*) in 1D, Algorithm (4).

Algorithm (4): 1D *prefix sum* algorithm; *O*(*n*) time

1.Initialisation $s[0] \leftarrow 0$;

2. for $i \leftarrow 1$ to n do begin

3. $s[i] \leftarrow s[i\text{-}1] + a[i]$;

4. end for

The *prefix sum* of a given two-dimensional array can be defined similarly as that of the 1D *prefix sum* algorithm, Algorithm (4). The *prefix sum* at position $s[i][j]$ in a two-dimensional $m \times n$ array (a) is the sum of the array portion $a[1, \ldots, i][1, \ldots, j]$ for all i and j with a boundary condition $s[i][0] = s[0][j] = 0$. This was computed in $O(mn)$ time [4].

Considering Equation (2.3), the sum of $a[k \ldots i][l \ldots j]$ is computed by the subtraction of the *prefix sums* as in Equation (2.4) and depicted in Figure 2.3. This equation is obtained from the *inclusion-exclusion* principle.

$$s[i][j] = \sum_{p=1,q=1}^{i,j} a[p][q] \quad \ldots\ldots (2.3)$$

$$\sum_{p=k,q=l}^{i,j} a[p][q] = s[i][j] - s[k][j] - s[i][l] + s[k][l] \quad \ldots\ldots (2.4)$$

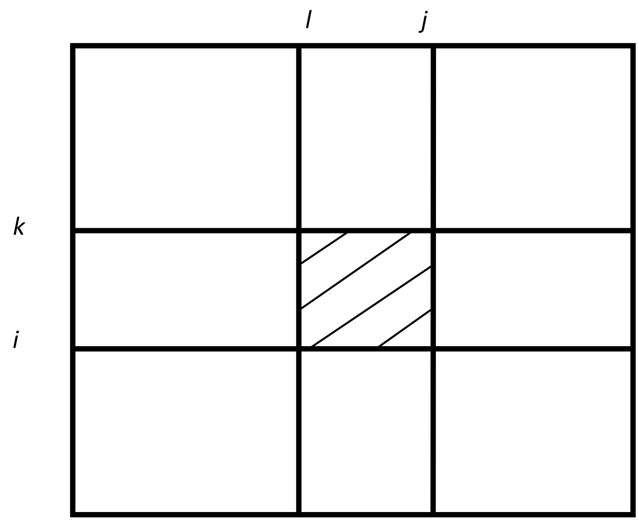


Figure (2.3): *MSA* of the area having indices (*k, l*), (*i, j*) using the p*refix sum* method in 2D

To maximise the sum from a two-dimensional array, we have to find indices (*k, l*) and (*i, j*) that maximise the $\sum_{p=k,q=l}^{i,j} a[p][q]$. Using *prefix sum* array *s*, the maximum sub-array is defined by Equations (2.5) and (2.6).

$$Maximum\ Sum = \max_{k=0,l=0,i=1,j=1}^{m-1,n-1,m,n} \left\{ s[i][j] - s[k][j] - s[i][l] + s[k][l] \right\} \quad \text{.....(2.5)}$$

$$= \max_{i=1,j=1,k=0}^{m,n,\ i-1} \left\{ s[i][j] - s[k][j] \right\} - \min_{i=1,l=0,k=0}^{m,j-1,\ i-1} \left\{ s[i][l] - s[k][l] \right\} \quad \text{.....(2.6)}$$

The outer framework of Takaoka's algorithm [14] was presented in Algorithm (3). The column-centred problem is to obtain an array portion that crosses over the central vertical line (A_{Centre_column}) with the maximum sum, as in Figure 2.4. The centre problem is as follows:

$$A_{Centre\_column} = \max_{\substack{0\le k\le i-1 \\ 0\le l\le n/2-1 \\ 1\le i\le m \\ n/2+1\le j\le n}} \left\{ s[i][j] - s[i][l] - s[k][j] + s[k][l] \right\} \quad \text{.....(2.7)}$$

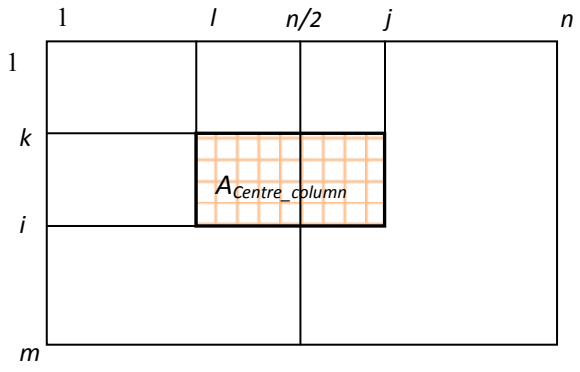


Figure (2.4): The column-centred problem (A_{Centre_column})

In Equation (2.7), the indices *k* and *i*, are fixed and the equations discussed above are maximised by changing *l* and *j*. Thus the above problem is equivalent to maximizing the following, Equation (2.8). For $i = 1..m$ and $k = 0..i-1$,

$$A_{Centre\_column}[i,k] = \max_{\substack{0\le l\le n/2-1 \\ n/2+1\le j\le n}} \left\{ -s[i][l] + s[k][l] + s[i][j] - s[k][j] \right\} \quad \text{.....(2.8)}$$

Let $s^*[i][j] = -s[j][i]$, then Equation (2.8) results in Equation (2.9) which is,

$$A_{Centre\_column}[i,k] = \max_{n/2+1\le j\le n} \left\{ s[i][j] + s^*[j][k] \right\} - \min_{0\le l\le n/2-1} \left\{ s[i][l] + s^*[l][k] \right\} \text{.... (2.9)}$$

In Equation (2.9), the first part is computed by the max-version of *DMM* and the second part is computed by the *DMM* as noted in Equation (2.2).

In Equation (2.10), the first and the second terms are the max-version and the min-version, respectively. Equation (2.10) is expressed in terms of matrices, S_1 and S_2 having elements at position (*i*, *j*). The *(i, j)* elements of S_1 and S_2 are given by s[*i*][*j* - 1] and s[*i*][*j* + n/2], respectively for $i = 1..m$; $j = 1..n/2$. Assume an arbitrary matrix *T* and its negated transpose *T**. As the range of the index *k* is [0 ..*m* - 1] which is same in both S_1^* and S_2^*, it is shifted to [1..*m*]. Thus, the above is computed as,

$$\underbrace{S_2S_2^*}_{max} - \underbrace{S_1S_1^*}_{min} \text{..... (2.10)}$$

max -version of the *DMM* min-version of *DMM*

In Equation (2.10), the multiplication of matrices, S_1 and S_1^* is performed by the min-version of *DMM*. Similarly, the multiplication of matrices, S_2 and S_2^* is performed by the max-version of the *DMM*. This is followed by the subtraction of the distance products, obtained component-wise. Finally A_{Centre_column} is computed by extracting the maximum from the lower triangle of the resulting matrix [14].

Algorithm (3) is applied on a square array of size $n \times n$, where n is assumed to have a power of 2 for simplicity. It was observed by Takaoka [14] that the algorithm divided the array vertically and then horizontally. The procedure of computing the A_{Centre_column} (line 7, Algorithm (3)) through this recursion of depth 2 is defined at level 0. The algorithm then splits the array horizontally and then vertically through the next recursion of depth 2, which is called level 1. This process of splitting continues to a higher level following the same pattern.

Takaoka analysed the time of the process at level 0 [14]. The time is measured by the number of comparisons. $M(n)$ is assumed to be the time for multiplying two $(n/2, n/2)$ matrices. The multiplication of $(n \times n/2)$ and $(n/2 \times n)$ matrices is substituted by four multiplications of size $(n/2 \times n/2)$, which takes $4M(n)$ time. In Equation (2.10), A_{Centre_column} considers each of the min- and max-version of multiplications. Thus, A_{Centre_column} involving $(n \times n/2)$ and $(n/2 \times n)$ matrices requires $8M(n)$ time, and computing two smaller A_{Centre_column} solutions involve $(n/2 \times n/2)$ matrices with $4M(n)$ time. For each level, three solutions are calculated, one A_{Centre_column} and two smaller A_{Centre_column}, which accounts for $12M(n)$ time. The following recurrence is computed for the total time $T(n)$ as

$$T(1) = 0$$

$$T(n) = 4T(n/2) + 12M(n) \quad \ldots\ldots\ldots\ldots\ldots \quad (2.11)$$

Takaoka presented a lemma for this process [14]. The β is assumed as an arbitrary constant such that $\beta > 0$, $M(n)$ is supposed to satisfy the condition $M(n) \geq (4 + \beta)M(n/2)$. Thus, the above $T(n)$ satisfies $T(n) \leq 12(1 + 4/\beta)M(n)$.

It is obvious that the complexity of $O(n^3(\log \log n/\log n)^{1/2})$ for $M(n)$ satisfies the condition of the lemma with some constant $\beta > 0$. Thus, the *MSA* is achieved in $O(n^3(\log \log n/ \log n)^{1/2})$ time, which is subcubic. An extra term of $O(n^2)$ in the recursive process to count the number of operations is also required because the maximum sum of several matrices component-wise in the algorithm is computed during the process [14], which is absorbed in the main complexity. Studies are still emerging in this area to build on the preceding research and to advance the *MSA* algorithms. In the following chapter, we will introduce a more flexible shape to maximise the sum within the cubic time complexity.

2.3 Chapter summary

This chapter provided an introduction and background to the *MSA* problem. Previous research addressing the *MSP* and the use of the *rectangular shape* algorithms in the *MSA* framework and applications were discussed. We proposed that outcomes for the *MSP* in terms of maximum sum could be improved by using an alternative shape to the traditional *rectangular shape*. The following chapters present a new approach that addresses using an alternative shape in the *MSP*.

Chapter Three

[*K-Disjoint Maximum Convex Sum* problem]

This chapter presents an approach that has not been previously investigated to generalise the *MSP* problem in 2D using the *convex shape*. This study defines a new problem: the *K-Disjoint Maximum Convex Sum (K-DMCS*). A new approach to solve this problem uses the *WN convex shape* as a core component for implementing algorithms to find the disjoint maximum sums. The following sections present a detailed overview of the theoretical and algorithmic frameworks for the new approach. Research on the new approach from this study has been published in peer-reviewed papers [38, 61-64].

The chapter is organised as follows: Section 3.1 discusses the new approach in the *MSP* context, the *WN* C*onvex shape*; Section 3.2 presents the newly developed algorithm for the *K-Disjoint Maximum Convex Sum (K-DMCS)* problem; Section 3.3 illustrates the experimental analysis; and Section 3.4 provides a summary of the key points discussed in the chapter.

3.1 The c*onvex shape*: a new approach in the *MSP* context

Traditionally, maximum sub-arrays in a matrix are computed using *rectangular regions* that include elements which return the largest possible sums in the *MSA* context. The *convex shape* is proposed by this research as a new method to find the regions that return the maximum sums. The advantage of using a *convex shape* is due to its flexibility in covering diverse data distribution regions. The *rectangular shape* has less flexibility than the *convex shape*, which can result in underestimating the maximum sums of the data within the enclosed region, thereby compromising important data portions that could otherwise maximise the sums. The T. Fukuda *et al.* [42] investigation of the *rectilinear convex shapes* has motivated the use of a more flexible shape, compared to the conventionally used *rectangular shape*, to find the maximum sub-array among a multitude of sub-arrays. The T. Fukuda *et al.* investigations were limited to computing a solo region of interest, but without the capability of further extension to find *K* regions in two cases: the disjoint and overlapping cases. Our research incorporates techniques to simplify the T. Fukuda *et al.* algorithm [42] for finding the maximum sums, and this is followed by the generalisation of the simplified algorithm to iterate up to *K*th maximum sum. The algorithms use the *WN convex shape* to find the maximum sums. Although there exist other types of *rectilinear convex shapes* these are outside the scope of this research. Instead the focus is on the main objective of this study, which is to investigate outcomes of the initiated *MCS* and *K-MCS* and to depart from using the *rectangular shape* in *MSP context*. For simplicity and convenience, in this thesis, the "*WN convex shape"* and *"convex shape"* terms are interchangeable.

The start of this chapter provides general definitions and concepts. The *WN convex shape* is comprised of the *W* and *N* shapes, which are illustrated in Figures 3.1 (a) and (b). The *WN convex shape* algorithm is based on the dynamic programming method. In this method, complex problems are broken down into sub-problems, and their collective solutions are combined to yield an overall solution.

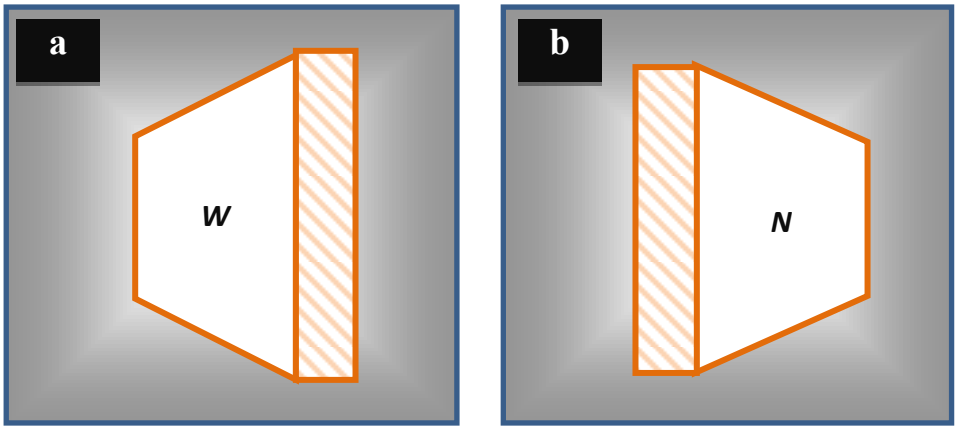


Figure (3.1): (a) *W shape*; (b) *N shape.*

Definition 3.1.1 The *WN convex shape* is defined, in this research, as a shape that has a centre (*anchor*) column linked to *W* and *N* shapes (Figure 3.2). A *W shape* (Figure 3.1 (a)) can be described as a region with a top contour inclining or remaining horizontal and a bottom contour declining or remaining flat from left to right, whereas the *N Shape* (Figure 3.1 (b)) is a mirror image of the *W shape*. In this study, the definition does not follow the geometrical *convex shape* definition [61]. In Figure 3.2, at *anchor column k*, calculating the *W shape* is computed from *s* to *t* to find a maximum sum for each *s* and *t*.

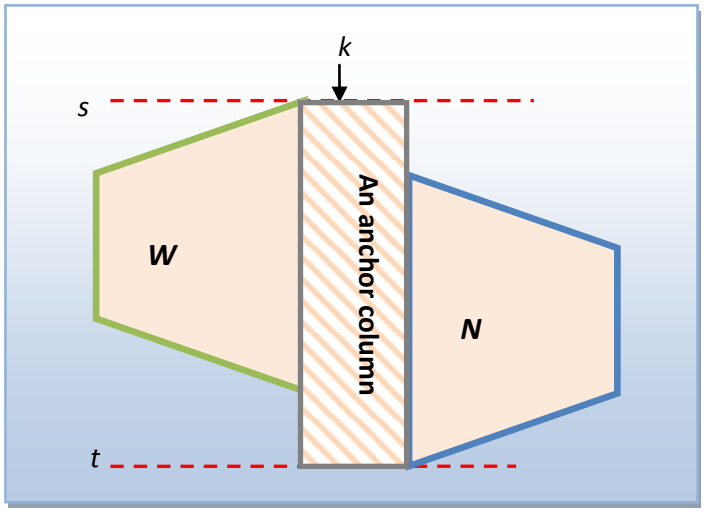


Figure (3.2): The *WN convex shape*

The maximum sum of the *W shape* can be calculated based on three cases: in the first case solution of the *W shape* is computed by the addition of the current *anchor column k* from rows *s* to *t*, denoted by (*k*, *s*, *t*), to the best solution that is given in the previous column *k*-1 (the light red shaded area) of the same interval. Likewise the *anchor column k*-1 is computed recursively from the summations of the preceding solutions obtained based on the three cases discussed herewith. This is shown in Figure 3.3.

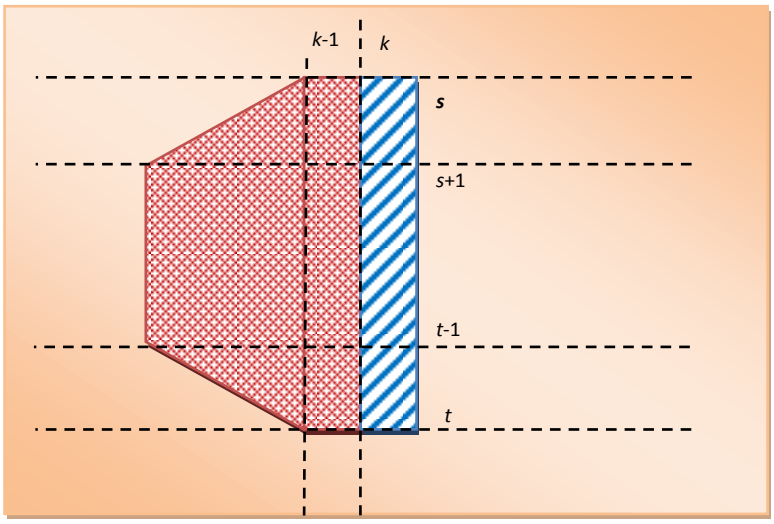


Figure (3.3): The first case of the *W shape* solution

The second case is given by the addition of the best solution obtained from the interval of rows *s*+1 to *t* at column *k* (the light red shaded area) and the element in row *s* at column *k*. This is shown in Figure 3.4.

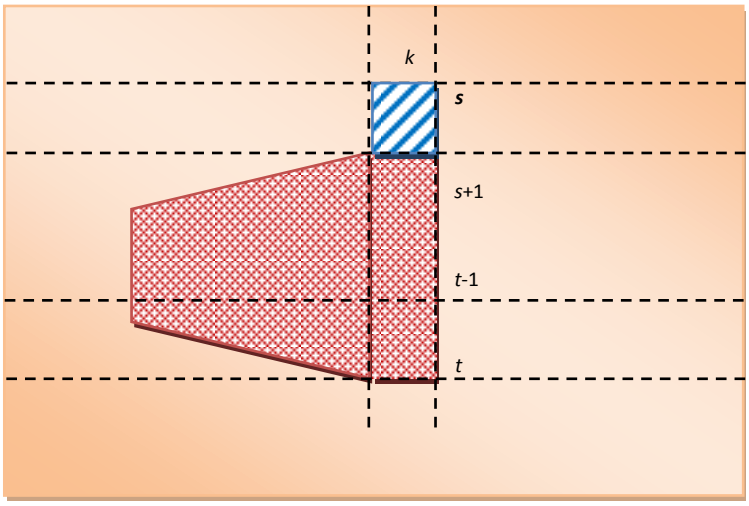


Figure (3.4): The second case of the *W shape* solution

The third case is given by the addition of the best solution obtained from the interval of rows *s* to *t*-1 at column *k* (the light red shaded area) and the element in row *t* at column *k*. This is shown in Figure 3.5.

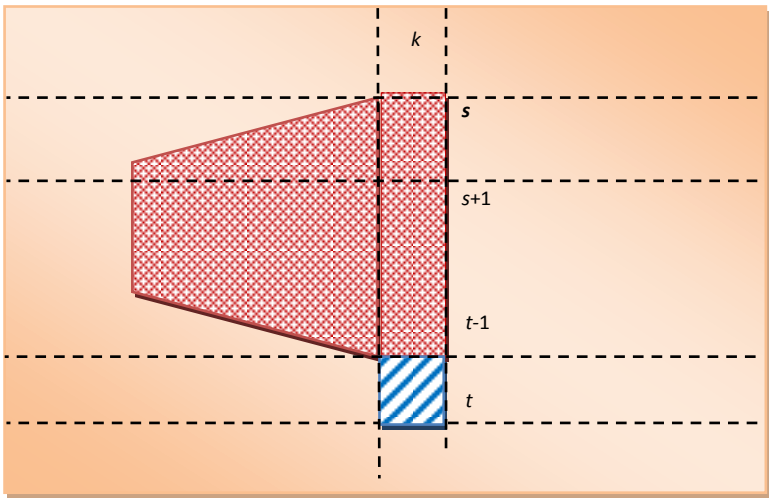


Figure (3.5): The third case of *W shape* solution

The computation process for each of the three cases, using the *W shape*, yields solutions in the form of three maximum sums for a given input of a 2D array. The *W shape*'s final solution for the *anchor column* (*k*, *s*, *t*) returns the greatest value of these three maximum sums. Algorithm (5) outlines this process, and incorporates a pre-processing procedure using the *prefix sum* method of Algorithm (4). This procedure is important to avoid repeating the addition process by utilising existing results from the previous (column-wise) summation of elements.

In Algorithm (5), a column from position *s* to position *t* in the *k*th column is written as (*k*, *s*, *t*), which is an *anchor column* when it is located on the rightmost of a *W shape*. The maximum sum with *anchor column* (*k*, *s*, *t*) is denoted by $f_W(k, [s, t])$. The f_W is computed as follows:

Algorithm (5): *W shape* Algorithm

1: *prefix_sum*[0, *k*] ← 0 for all *k* /* compute *prefix sum* column-wise */

2: for *k* ← 1 to *n* do

3: for *s* = 1 to *m* do *prefix_sum*[*s*, *k*] ← sum[*s*-1, *k*] + *a*[*s*, *k*]

/* compute f_W value */

4: $f_W(0, [s, t])$ ← 0 for all $s \leq t$

5: $f_W(k, [s+1, s]) \leftarrow -\infty$, for all k and s

6: for $k \leftarrow 1$ to n do

7: for all intervals of $[s, t]$ in increasing order of $t-s$ where $s \leq t$ do

8: $\text{sum}[k, s, t] \leftarrow prefix\_sum[t, k] - prefix\_sum[s-1, k]$

9:
$$f_W(k, [s, t]) \leftarrow \max \begin{cases} f_W(k-1, [s, t]) + \text{sum}[k, s, t] & \text{(case 1)} \\ f_W(k, [s+1, t]) + a[s, k] & \text{(case 2)} \\ f_W(k, [s, t-1]) + a[t, k] & \text{(case 3)} \end{cases}$$

where,

$f_W(k,[s, t])$ is the *anchor column* which has the maximum value out of the three cases;

$\text{sum}[k, s, t]$ is the sum of the kth column from position s to position t used in the first case;

$a[s, k]$ is the element in row s at column k;

and $a[t, k]$ is the element in row t at column k.

Theorem (3.1.1) *$f_W(k, [s, t])$ is the maximum value of the sum of a W shape with an anchor column (k, s, t).*

Proof. The proof is based on a double induction on k and $t-s$ from smaller to larger *anchor column*s. Let $k = 1$ for the basis. It is easy to verify $f_W(1, [s, s]) = \text{sum}[1, s, s]$, noting $f_W(1, [s+1, s]) = 0$. Now for general s and t, case (1) is $\text{sum}[1, s, t]$, since $f_W(0, [s, t]) = 0$. Suppose the Theorem is true for $t-s = c-1$ for some c, where c is an arbitrary value. We prove $f_W(1, [s, t]) = \text{sum}[1, s, t]$. From induction we have $f_W(1, [s+1, t]) = \text{sum}[1, s+1, t]$. Thus, case (2) is $f_W(k, [s+1, t]) + a[s, k] = \text{sum}[k, s, t]$. Case (3) is similar. From the program (algorithm (5), line 9), $f_W(1, [s, t]) = \max\{\text{sum}[1, s, t], \text{sum}[1, s, t], \text{sum}[1, s, t]\} = \text{sum}[1, s, t]$. Similarly we can prove $f_W(k, [s, s])$ is correctly computed.

Now suppose Theorem (3.1.1) is true for $k-1 > 0$ and $t-s-1 > 0$. The maximum value of the *W shape* with the *anchor column* (k, s, t) is obtained from three smaller *anchor column*s of *W shapes:* the *W shape* with the *anchor column* $(k-1, s, t)$, the *W shape* with the *anchor column* $(k, s+1, t)$, and the *W shape* with the *anchor column* $(k, s, t-1)$. From the induction hypothesis, the f_W value of these *W shapes* is maximum with respect to the three parameters. The three cases in the program (line 9,

Algorithm (5)) correspond to those three *W shapes* and the maximum f_W (*k*, [*s*, *t*]) is successfully computed by Algorithm (5).

3.1.1 *Mono-directional convex shape* algorithm: The algorithm of T. Fukuda *et al.*

Using a *mono-directional* approach, the algorithm of T. Fukuda *et al.* (T. Fukuda*'s* algorithm) uses seven cases to compute a *WN shape.* Three of these cases are covered as part of Algorithm (5); the remaining four cases are depicted in Figure 3.6 and computed in Algorithm (6). This algorithm processes the array in one direction from left to right; hence, the name *mono-directional*. The fourth case is based on the f_W in line 6 of Algorithm (6) and Figure 3.6(a), which was earlier explained in the *W* shape definition. To compute the solution for this case, we firstly need to find the three cases from Algorithm (5) (line 9), as in Figures 3.3, 3.4 and 3.5. The fifth case is based on the addition of the solution of f_{WN} from position *s* to position *t* at column *k*-1 to the sum from position *s* to position *t* at column *k* (Figure 3.6(b)). The sixth case (Figure 3.6(c)) is based on the solution of f_{WN} for the interval from position *s*-1 to *t*; the value *a*[*s*-1, *k*] is subtracted from the interval at the *k*th column. The seventh case (Figure 3.6(d)) is based on the solution of f_{WN} for the interval from position *s* to *t*+1 at the *k*th column. The value *a*[*t*+1, *k*] is subtracted from the interval at the *k*th column. A *max* operation was used to select the largest value of the four sums.

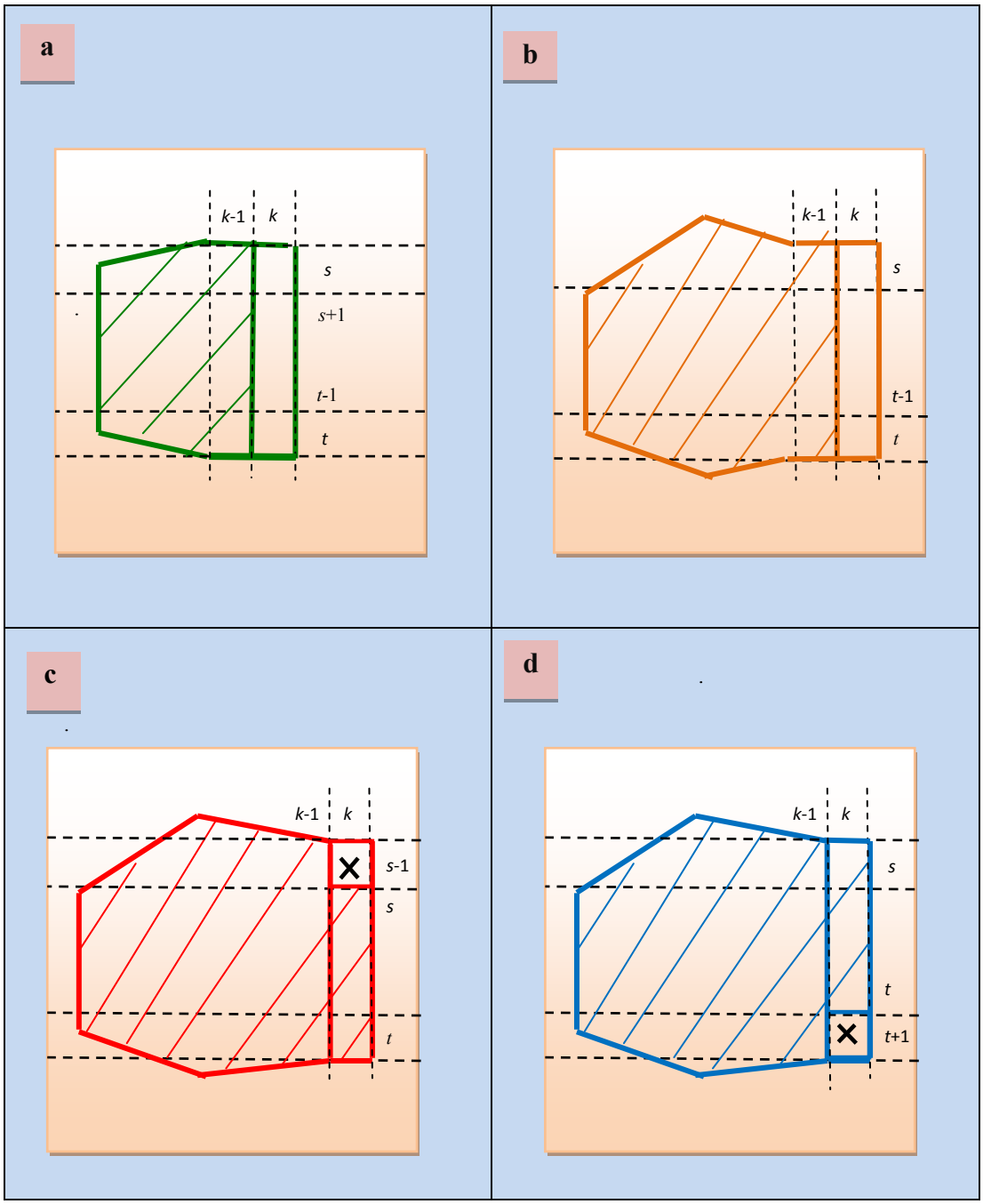


Figure (3.6): (a) to (d) figures depict the four remaining cases of the *WN shape*, out of seven (Figures 3.3, 3.4, 3.5, 3.6), required to compute f_{WN} using the *mono-directional WN convex shape* algorithm (T. Fukuda's *WN shape* algorithm).

In algorithmic terms, f_{WN} in Algorithm (6) is computed as follows: Let *a* be an array of size *m* x *n* containing real numbers. Due to the triply nested structure in Algorithm (6), when $m = n$, the time complexity is $O(n^3)$:

Algorithm (6): T. Fukuda's *WN shape* algorithm $O(n^3)$.

1: Initially, assume all $f_W(k, [s, t])$ are computed from Algorithm (5)

2: $f_{WN}(0, [s, t]) \leftarrow 0$

3: For all $s < t$

4: For $k \leftarrow 1$ to n do

5: For all intervals of $[s, t]$ where, $s < t$ do

6:
$$f_{WN}(k, [s, t]) \leftarrow \max \begin{cases} f_W(k\text{-}1, [s, t]) + \text{sum}[k, s, t] & \text{(case 4)} \\ f_{WN}(k\text{-}1, [s, t]) + \text{sum}[k, s, t] & \text{(case 5)} \\ f_{WN}(k, [s\text{-}1, t]) - a[s\text{-}1][k] & \text{(case 6)} \\ f_{WN}(k, [s, t\text{+}1]) - a[t\text{+}1][k] & \text{(case 7)} \end{cases}$$

where,

f_{WN} is a function to find the *WN* shape such that $f_{WN}(k,[s, t])$ is the max value of the four cases from position s to position t in column k at the rightmost of the shape*;*

sum$[k, s, t]$ is the sum of column k from position s to position t*;*

$a[s\text{-}1, k]$ and $a[t\text{+}1, k]$ are the values subtracted in the third and fourth cases, respectively.

In Algorithm (6), the *anchor column* is placed at the rightmost column of each *WN shape*. As discussed, this algorithm deals with seven cases; three for the *W shape* and four for the *WN shape*. In the next section we address how we can deal with essentially three cases to replace the seven cases, considering the fact that the *W* and *N* shapes are symmetrical.

3.1.2 The simplified c*onvex shape* algorithm: *bidirectional* approach

The *WN convex shape* is an amalgamation of the *W shape* and its mirrored image, the *N shape*. The *W shape* in the left part of the *WN convex shape* is considered as the primary shape in the creation of the *N shape* due to the property of its widening from left to right. This was outlined in Algorithm (5). The *bidirectional* technique is used to co-join the *W shape* with the *N shape* at the common *anchor column* (k, s, t) to find the solution of the *WN convex shape*. The algorithm used to compute the *W shape* is now used in its modified form to compute the *N shape*, which is widening

from right to left. The *anchor column* (*k, s, t*) of the *W shape* is located on its rightmost position, whereas for the *N shape* it is located on its leftmost position. As discussed above, when the two shapes are co-joined at the common *anchor column* (*k*, *s*, *t*), it results in a duplication from the overlapping at the column. Thus one of the sums at the common *anchor column* (*k*, *s*, *t*) is subtracted from the *WN convex shape* to compute the final solution. Our approach simplifies T. Fukuda's algorithm by discarding four of the seven cases yet computing the final solution using three cases. Mathematical proof of this is given in Section 3.1.2.1. The simplification and mathematical proof of this study have been presented in [61].

The f_{WN} can be derived using our version of the simplified algorithm, Algorithm (7), to compute the *WN convex shape*. This algorithm has a triply nested structure. When $m = n$, the time complexity is $O(n^3)$.

Algorithm (7): The simplified *WN convex shape* algorithm $O(n^3)$

1: Let sum(*k*, [*s*, *t*]) be the column sum from *s* to *t* of the *k*th column.
2: Compute *W shape* from left to right for each *k*, *s* and *t* in f_W.
3: Compute *N shape* from right to left for each *k*, *s* and *t*, resulting in f_N.
4: For $k \leftarrow 1$ to *n* do
5: For $s \leftarrow 1$ to *n* do
6: For $t \leftarrow$ s to *n* do
$f_{WN}(k,[s, t]) \leftarrow (f_W(k,[s, t]) + f_N(k,[s, t] - \text{sum}(k,[s, t])$ /** the subtraction removes the duplication **/
7: Take the maximum of $f_{WN}(k,[s, t])$ for all *k*, *s*, *t*.

The following section outlines some of the challenges faced in reducing the time complexity to subcubic for the *WN convex shape*. The time complexity to find the maximum *WN* convex sum is $O(n^3)$. We attempted to improve this time complexity using the concept of the *Distance Matrix Multiplication* (*DMM*) approach, used for reducing the time complexity of the *rectangular shapes* as discussed earlier in Chapter 2. Here, we demonstrate the ways in which this approach cannot readily be applied to the *WN convex shape*.

In the *MSA* approach, Takaoka [14] reduced the time complexity from $O(n^3)$ to $O(n^3(\log \log n/\log n)^{1/2})$ using Algorithm (3). This was achieved by finding the maximum value of the three

solutions, A_{Right}, A_{Left} and A_{Centre_column} shown in Figure 3.7. Using Algorithm (3), the A_{Right} and A_{Left} can be found recursively on the right and left sides of the input array. The crucial part in Takaoka's solution is based on solving the A_{Centre_column} problem. This problem is defined as finding the solutions that cross over the central vertical line (i.e. in the middle of the input array) discussed in Chapter 2. The A_{Centre_column} region is divided into two sub-arrays, namely $A_{Centre\_column\_Left}$ and $A_{Centre\_column\_Right}$. These are homogenous *rectangular portions* and can be easily connected after the *DMM* is applied. Thus, the solution becomes readily available because mathematically the two sub-matrices can be added in $O(n^2)$ time to obtain the A_{Centre_column} solution, as in Equation (2.10).

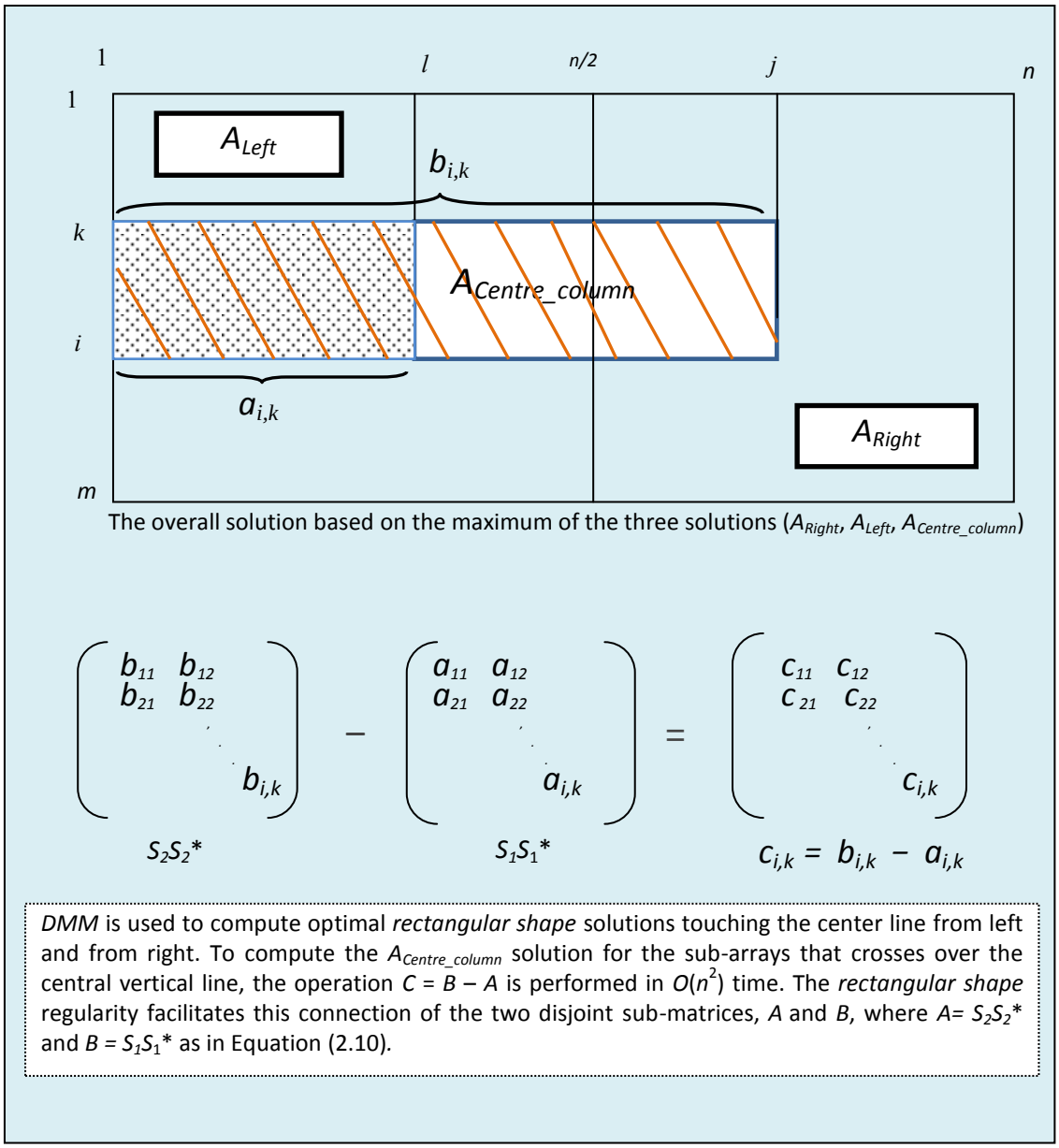


Figure (3.7): Process for solving the column-centred problem by using *DMM*.

The primary reason for the difficulty of using *DMM* in achieving subcubic time for the *MCS* problem is that the *anchor column* of the maximum *WN shape* may not necessarily fall on the central vertical line (see Figure 3.7). Thus the approach of computing the best *W shape* touching the centre line from the left, and the best *N shape* touching from the right cannot be undertaken, to find the solution for the centre problem. This is illustrated in Figure 3.8, which highlights the co-joining of the *W shape* and *N shape* to form the *WN convex shape* at the *anchor column (k, s, t)*, at any position within the input matrix.

The *convex shape* algorithm connects the two shapes (*W shape* and *N shape*) at an *anchor column* (*k*, *s*, *t*) at any position within the input matrix. The anchor column coordinates of the *convex shape* (the boundaries of the shape) keep changing in an irregular manner based on the solution, and the anchor column may not necessarily fall on the central vertical line. This makes the column-centred problem difficult to define in the *MCS* context.

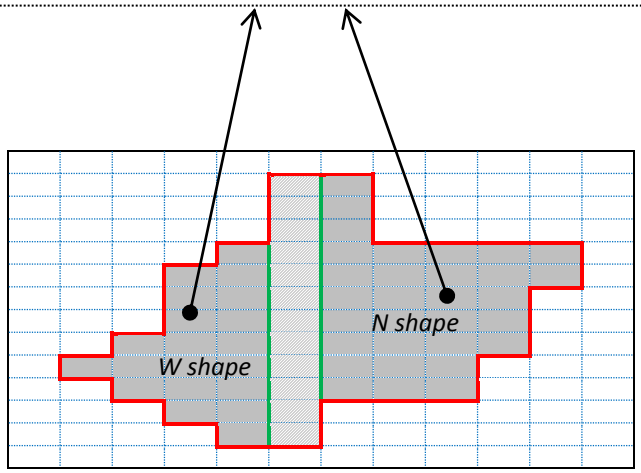


Figure (3.8): An illustration of the *convex shape* region

3.1.2.1 Mathematical proof of the simplified algorithm using *bidirectional* computation

The following is a proof of the correctness of the *bidirectional* approach to compute the *WN convex shape* at an *anchor column* (*k, s, t*): $f_W = a + c$, $f_N = b + c$ and $f_{WN} = a + b + c$ (Figure 3.9(a)) [61]. Suppose f_{WN} is not maximum for the *anchor column* (*k*, *s*, *t*), then there must be another shape (Figure 3.9(b)) at the same *anchor column* with the sum $x + y + c > a + b + c$.

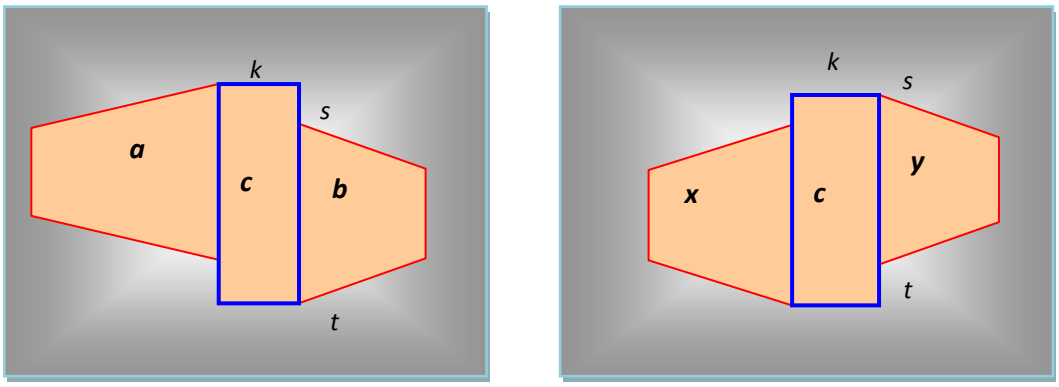


Figure (3.9): (a) the *WN convex shape* $a + b + c$; (b) the *WN convex shape* $x + y + c$

This implies, if $x + y > a + b$, then $x > a$ or $y > b$, otherwise $x \leq a$ and $y \leq b$, which would be a contradiction. If $x > a$, then $x + c > a + c$, contradicts with maximum W. If $y > b$, this contradicts with maximum N. Thus, $f_{WN}(k,[s, t])$ is the maximum *WN* at *anchor column* (k, s, t).

3.1.2.2 The *backtracking* technique to determine the *WN convex shape* boundaries

To generalise *MCS* to *K-MCS*, which is to compute K maximum convex sums, two cases can be defined namely, disjoint and overlapping. The *backtracking* technique can be applied to identify the boundaries of the *WN convex shape* regions for the above two cases; this chapter focuses on the disjoint case. Locating the boundaries involves *backtracking* the recorded solutions from the three cases for f_W and f_N. Whilst in the process of finding the maximum sums (a process outlined in Algorithm (7)), the *anchor column* coordinates and the solutions that contribute to the results of f_W and f_N are recorded as preparation for the *backtracking* process. This is presented in Algorithm (8) for f_W. Hence, the *backtracking* approach locates the actual *WN convex shape* reversely, column by column, until converging where the solution originated.

The *backtracking* technique allows the *MCS* problem to be extended to cover the disjoint case that branches out from it. Determining the shape boundaries is important to generalise the problem to cover the disjoint case. Such a generalisation is essential not only to locate the first selection of segments of consecutive array elements that have the largest possible sum, but also to find the second, third and up to the Kth region. Computing these regions makes it feasible to apply the generalised *MCS* algorithms to real-life datasets. These generalised problems are discussed in Chapters 3 and 4 of this thesis; here the primary focus is the *backtracking* technique.

Algorithm (8) outlines the *backtracking* process of the *W shape* solutions. The *backtracking* of the *N shape* solutions are processed likewise in the opposite direction to the *W shape*. In Algorithm (8), let the *anchor column* indices (*k, s*, *t*) of the maximum sums be k_m, s_m, and t_m, respectively. These indices are used to *backtrack* the *W convex shapes.* Let array *bw* store the values of the three cases (1, 2, or 3) in Algorithm (5) to keep a record for the *backtracking* directions. In case 1, the current solution at column *k* is *backtracked* one step back to its previous solution at *k*-1 having the same interval (*s, t*). The previous solution at column *k*-1 is located on the left-side of the current solution at column *k* in the *W shape* (*backtracked* case 1 of Algorithm (5)). In case 2, the solution at column *k* is *backtracked* to its previous solution at *s*+1 (*backtracked* case 2 of Algorithm (5)). In case 3, the current solution at column *k* is *backtracked* to its previous solution at *t*-1 (*backtracked* case 3 of Algorithm (5)). An array *bn* is used to store the values of the three cases (1, 2, or 3) to *backtrack* the *N shapes*. Let *region[]* (in line 1 of Algorithm (8)) be an *m* x *n* array to identify the boundaries of a shape. The array elements are all initialised to zeros (0s). In the *backtracking* process, for each column *k*, the upper boundary and the lower boundary of the shape are set to 1. The output of Algorithm (8) returns the solution boundaries (f_W) assigned as 1s and the rest of the array remains set to 0s.

Algorithm (8): *Backtracking* Algorithm

1: initialise elements in region as zeros
2: set *k* as k_m /* Where, (k_m, s_m, t_m) is the *anchor column* of the maximum sum*/
3: set *s* as s_m
3: set *t* as t_m
4: while $k > 0$ and $f_W(k, [s, t]) > 0$
5: *region*[s][k] ← 1
6: *region*[t][k] ← 1
7: decrement *k* if *bw* at (*k, s, t*) is case 1
8: increment *s* if *bw* at (*k, s, t*) is case 2
9: decrement *t* if *bw* at (*k, s, t*) is case 3
10: end loop

3.2 The newly developed algorithm for the *K-Disjoint Maximum Convex Sums (K-DMCS)* problem

The problem of *MCS* has been generalised to cover the disjoint case. The newly defined *K-Disjoint Maximum Convex Sum* (*K-DMCS*) problem is to find the *K* maximum sums that are *disjoint* by using the *WN convex shape*. The *K-Disjoint Maximum Sub-Array* (*K-DMSA*) problem using the *rectangular shape* was previously introduced by Bae and Takoaka [4]. In the present study, our newly developed algorithm uses the *convex shape* to find the first maximum convex sum, second maximum convex sum, and up to the *K* maximum convex sum from the remaining portions. The *K-DMCS* algorithm computes this process in a time complexity $O(Kn^3)$, which is same as that of the disjoint case of the *K-MSA* algorithm [4]. The developed *K-DMCS* algorithm is published as part of [61]. The new algorithm for the *K-DMCS* problem returns higher maximum sums than the *K-DMSA* which is later verified in Section 3.3.

After reviewing the background to the *K-DMSA* problem and corresponding algorithms, the new *K-DMCS* algorithm is presented later in this section. The *K-DMSA* problem was initially studied by Ruzzo and Tampa in 1D [28]. They designed a linear time algorithm that found all disjoint maximum sub-arrays for a one-dimensional array [28]. Later studies investigated the problem in 2D [4, 17, 18]. Bae and Takaoka [4] extended the 2D Kadane's Algorithm (Algorithm (2)) to address the disjoint case, which is classified as a greedy approach by Anzai *et al.* [39]. The *K-DMSA* algorithm was implemented in this research and is given in Algorithm (9). The process of finding the first *rectangular shape* was explained in Chapter 2 (Section 2.2.2.1). Algorithm (9) extended the problem to find the *K*th maximum sub-array. In this algorithm, the next maximum sub-array was found, whilst avoiding any duplication by assigning negative infinity to the first located *rectangular shape*. This procedure was repeated for the second, third, and up to the *K*th maximum sub-arrays. The time complexity of Algorithm (9) was $O(Kn^3)$ having a three *for-loop* (nested) structure [4]. This algorithm was adopted in K. Fukuda's data mining studies to find peaks of data using real-life applications [30, 31]. The algorithm captured regions that potentially represented the data [30, 31].

Algorithm (9): Bae and Takoaka's algorithm for the disjoint case using the *rectangular shape*; $O(Kn^3)$, when $m = n$

1: Initialisation $((x1, y1),(x2, y2)) \leftarrow ((1,1), (0, 0))$; $S \leftarrow 0$; $iter \leftarrow 0$ /* the initial candidate is empty */

2: while ($iter < K$){ /* K is the limit of the disjoint *rectangular shapes* */

3: for $z \leftarrow 1$ to $z \leq m$ do{

4: for $i \leftarrow 1$ to $i \leq n$ do $column[z\text{-}1][i] \leftarrow 0$;

5: for $x \leftarrow z$ to $x \leq m$ do {

6: $t \leftarrow 0$; $s \leftarrow -\infty$; $k \leftarrow 0$; $l \leftarrow 0$; $j \leftarrow 1$;

7: for $i \leftarrow 1$ to $i \leq n$ do {

8: $column[x][i] \leftarrow column[x\text{-}1][i]+a[x][i]$;

9: $t \leftarrow t + column[x][i]$;

10: if($t > s$){$s \leftarrow t$; $k \leftarrow$ i; $l \leftarrow j$;}

11: if($t < 0$){$t \leftarrow 0$; $j \leftarrow i+1$;} /* reset the accumulation*/

12: end for /* i */

13: if($s >$ S){$S \leftarrow s$; $x1 \leftarrow x$; $y1 \leftarrow k$; $x2 \leftarrow z$; $y2 \leftarrow l$;}

14: end for /* x */

15: end for /* z */

/ * S is the *Maximum Sub-Array* using the *rectangular shape;* bottom-right corner is ($x1$, $y1$); top-left corner is ($x2$, $y2$) */

16: Output ($x2$, $y2$),($x1$, $y1$), S

17: for ($i \leftarrow x2$; $i \leq x1$; i++)

18: for ($j \leftarrow y2$; j$\leq y1$; j++) $a[i][j] \leftarrow -\infty$;

19: $iter$++; /* iterate to find the next *rectangular shape*/

20: end while

The new Algorithm (10) incorporates the aforementioned approaches to obtain the maximum sums by using the *WN convex shape* to compute the *K-DMCS*. The process includes finding the *W shape*, *N shape* and the combined *WN shape*, and integrates the *prefix sum* and *backtracking* techniques. Algorithm (10) is implemented to find the K disjoint shapes having the maximum sums of the first (K = 1), second (K = 2), third (K = 3) and up to the Kth regions. Once the first maximum sum is found, the algorithm excludes the K = 1 region in the search for the second maximum sum (K

= 2) as illustrated in Figure 3.10. The array is then processed to find the third maximum sum (K = 3), discarding the regions at both K = 1 and K = 2. The algorithm consecutively processes the remaining array portions following the same procedure, while discarding the accumulated regions that return the maximum sums. The algorithm identifies these discarded regions via the assigned negative infinity. This procedure continues until it reaches the Kth maximum convex sum. The *WN convex shapes* that were assigned negative infinity in every iteration are placed in triply nested loop structure giving a time complexity of $O(Kn^3)$.

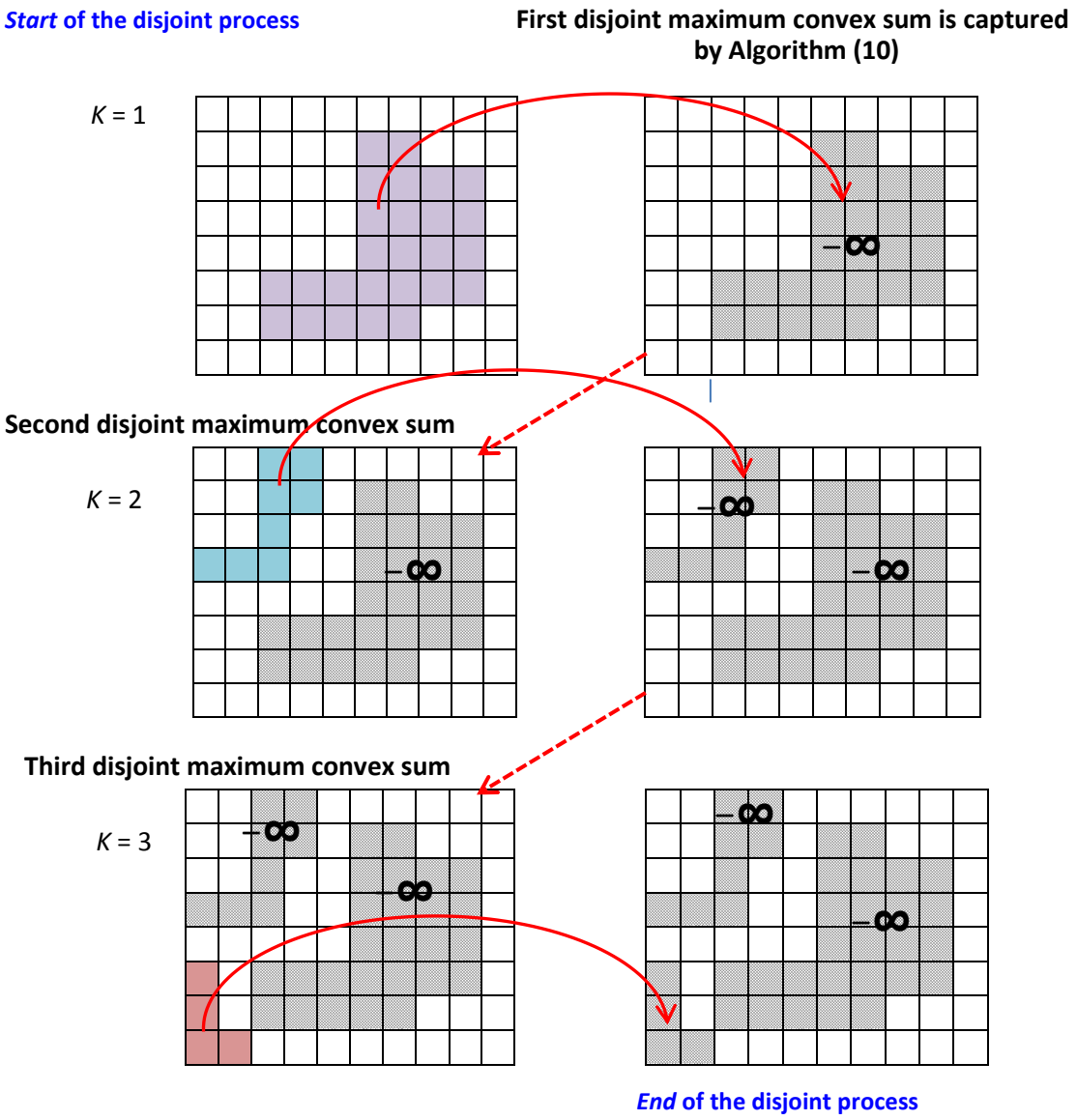


Figure (3.10): An illustrative example of the process of the original *K-DMCS* algorithm. This shows the first, second and third regions of the located disjoint maximum sums.

The following pseudo code outlines the *K-DMCS* algorithm, with an output of the located disjoint *convex shapes*, and the maximum convex sums of the regions:

Algorithm (10): *K-Disjoint Maximum Convex Sums* algorithm; ($O(Kn^3)$), when $m = n$

1: read input matrix, *a*
2: While not all *K* solutions are found
3: perform *WN shape* calculation
4: for each column *k*, $1 \le k \le n$
5: for each top index s, $1 \le s \le m$
6: for each bottom index t, $s \le t \le m$
7: set sum of overlap *anchor column* as $ac \leftarrow prefix[t][k] - prefix[s][k]$
8: *WN shape* $\leftarrow$ *W shape* solution at (k,s,t) + *N shape* solution at $(k,s,t) - ac$
10: end top index *s*
11: end bottom index *t*
12: end column *k*
13: replace *WN solution* with negative infinity by *backtracking W* and *N* shapes
14:end while

3.3 Experimental analysis

The previous sections outlined the theoretical framework and implementation of *MSA* and *MCS* algorithms in the *MSP* context. This section[2] discusses two experimental phases, where we made experimental comparisons to assess the differences in outcomes obtained using the *MSA* and *MCS* approaches. The first phase focused on an analysis of experiments designed to assess the differences in outcomes obtained using the *rectangular shape* algorithm (Algorithm (2)) and the *convex shape* algorithm (Algorithm (7)). The second phase focused on comparing outcomes of the two algorithms that covered the disjoint case: the *rectangular shape K-Disjoint Maximum Sub-Array* algorithm (Algorithm (9)) and the *K-Disjoint Maximum Convex Sums algorithm* (Algorithm (10)). The comparisons were based on outcomes from finding a selection of segments of consecutive array elements that have the largest possible sum compared with all other segments in the presented data. Large matrices of different sizes were used in the experiments to find the maximum sum.

[2] We consulted the University of Canterbury statistician, research consultancy services, to ensure that this section's statistical procedures adequately fulfilled the study aims.

The underlying procedures to compute the maximum sums and running times for the two algorithms differed from each other. It was expected that using the *convex shape* algorithms would yield maximum sums that are no smaller than those of the *rectangular shape* algorithms. The experiments' running times of each algorithm were expected to be constant across all experiments for a given matrix size; but even so, due to processor precision this would vary by an order of milliseconds for large matrix sizes. The *convex shape* algorithms included more operations to find the maximum sum and required more running time compared to the *rectangular shape* algorithms. During the measurement of the running time in the experiments, we considered the length of time that was taken to locate the exact shapes because in addition to returning the sum, we returned the exact shape boundaries. In the *convex shape* algorithms, the *backtracking* process to find the shape coordinates (explained in Algorithm (8)) was applied after computing the sums. The *convex shape* algorithms required post-processing of a manifestation of three-dimensional (3D) coordinates represented by the three indices: *s*, *t*, and *k,* where *s* and *t* determine the interval, and *k* determines the current column. For the *rectangular shape* algorithms, the region location was found by determining the two indices *x* and *y* in 2D (such that, (*x2*, *y2*) and (*x1*, *y1*) in Algorithm (2)). The running time to find the actual *convex shape* and its boundaries is more than that for the *rectangular shape* because of the complexity in processing the *convex shape* using three coordinates compared to using two coordinates for the *rectangular shape*. Although the *convex shape* algorithms were more complicated and involved more operations compared to the *rectangular shape* algorithms, the time complexities were asymptotically the same for the corresponding algorithms.

The experiments were conducted using a 2.66 GHz Intel® Core™2 Quad Processor Q8400 cache size: 4MB 1333MHZ 2 DUO Processors, each processor had CPU of 2.66GHz with 8GB 667MHz DDR2 RAM. The two algorithm implementations were written in C programming language. The system was installed with Fedora 18 Linux Operating System equipped with kernel version 3.11.10-100. The matrices were randomly generated using a built-in function in C programming language: rand(). This function returns a random number in the range of 0 to a very large number (2^{31}); we used modular arithmetic to wrap around the random number (*r*) to range between 0 and 198. We then subtracted *r* from 99, which gives a random number in the range between -99 to 99. The GNU compiler gcc (Version 4.1.3) was used in the random number generation process.

We randomly generated *N* matrices ranging in size with *n* x *n* dimensions. Determining an adequate sample size *N* to detect any significant differences, if any, could have been based on knowledge and information obtained from previous research, but because studies are lacking in the

project's field we were unable to use this option. Alternative methods were available to estimate *N* using power analysis [65]. We used *a priori* power analysis using the G*Power 3.1.9 statistical package to determine a suitable sample size *N* [65]. This analysis required specifying three factors *alpha*, *beta* and *effect size* (*Es*) [65-67]. The *alpha* is the probability of a *type I error* (i.e. probability of rejecting a correct null hypothesis), *beta* is the probability of a *type II error* (i.e. probability of accepting a false null hypothesis), and *effect size* is the standardised magnitude of the difference between the two groups [65, 67]. Because the magnitude of the difference between the groups is not known, a conservative approach was adopted in determining parameter values for the analysis [65]. We let *alpha* = .05; power (1 - *beta*) = .95; and *effect size* = 0.30. Based on these parameters, the analysis recommended a sample size of 100 and therefore 100 random matrices were generated in each matrix size to compare the differences in outcomes obtained using *the convex shape* and *rectangular shape* algorithms.

We introduced the following notation for convenience: S_{conv} and S_{rect} denote the maximum sum outcome obtained using the *convex* and *rectangular shapes* algorithms, respectively; μ_{conv} and μ_{rect} denotes the mean maximum sum outcome obtained using the *convex* and *rectangular shapes* algorithms, respectively. Size *n*, (for *n* = 50, 100, …, 600) denotes the matrix size, from 50 x 50 to 600 x 600 (e.g. Size 100 = 100 x 100).

The *convex shape* and *rectangular shape* experimentation groups were related because the quantitative measurements of the sums (S_{conv} and S_{rect}) were obtained from the same matrix. A dependent-sample statistical test was used as part of the inferential investigation. SPSS (version 20) was the software package used for the statistical analysis [68, 69]. The paired-samples t-test in SPSS was used to evaluate the statistical significance of the difference between the two related groups' means and its standardised magnitude [70]. Throughout the analysis, we used a significance level (*p*-value) of .05 (Confidence Interval (CI) was set to 95%).

Prior to using the paired-samples t-test, we checked that the assumptions were not violated [70]. This was necessary to ensure the validity of the test results [66, 71, 72]. The assumptions were: the dependent variable is continuous, the related groups were in two categorical levels (i.e. two shapes*: rectangular* and *convex*), influential outliers were detected and adequately dealt with, and the distribution of the paired differences (in the dependent variable) was approximately normally distributed [70, 73, 74]. The paired t-test is not very sensitive to deviations from normality for sample sizes $N \geq 30$ because of the Central Limit Theorem (CLT) [70, 73, 75].

The first two assumptions were directly checked from the observations. The latter two assumptions were assessed using SPSS functions, including numerical (Shapiro-Wilk test) and graphical tests (boxplot, histogram, and qq-plots) [70, 74]. Extreme outliers were removed if they obscured the normality distribution of the dependent variable, to prevent threats to the validity of the statistical test. The normality distribution of the paired differences was checked using the Shapiro-Wilk test and qq-plots [70, 74]. The former is a robust numerical test to check the normality distribution of a variable, whereas the qq-plot involves decisions based on visual inspections [74].

3.3.1 Results and interpretations

3.3.1.1 The *convex shape* and the *rectangular shape* algorithms: Algorithms (7) and (2)

The descriptive results from running the two algorithms were in accordance with the theoretical prospects. Figure 3.11 and Table 3.1 demonstrate the differences between the two algorithm outcomes and their directions. The observed differences in Figure 3.11 and Table 3.1 comply with expectations, which had been based on the outlined underlying processes and their mechanisms. The paired-samples t-test was used to assess the statistical significance of the differences.

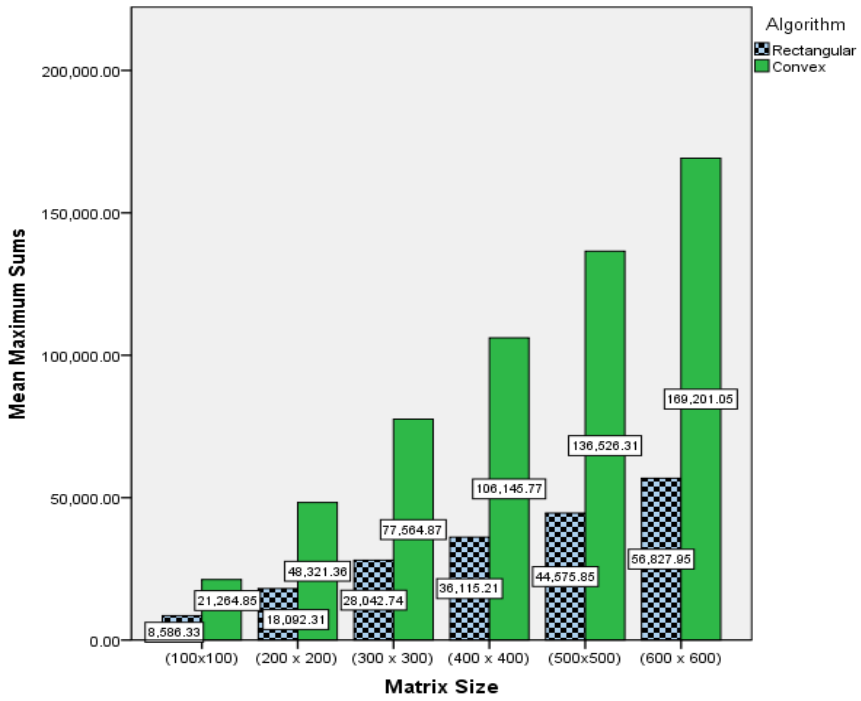


Figure (3.11): Descriptive comparisons of the mean maximum sums obtained using the *convex shape* (Algorithm (7)) and the *rectangular shape* (Algorithm (2)) algorithms. Sample size = 100.

Table 3.1: The mean running time results for finding the maximum sums and the shape boundaries obtained using the *convex shape* Algorithm (7) and the *rectangular shape* Algorithm (2). Sample size N = 100. The running time is measured in seconds.

Matrix Size	Shapes	Min	Max	Mean	Standard Deviation
(600 x 600)	*Convex**	49.45	50.91	49.74	0.21
	Rectangular†	1.32	1.44	1.36	0.02
(500 x 500)	*Convex*	28.5	29.5	28.72	0.17
	Rectangular	0.78	0.84	0.8	0.01
(400 x 400)	*Convex*	14.29	14.73	14.41	0.07
	Rectangular	0.4	0.44	0.42	0.01
(300 x 300)	*Convex*	5.89	6.58	5.97	0.07
	Rectangular	0.18	0.19	0.19	0.00
(200 x 200)	*Convex*	1.75	1.81	1.77	0.01
	Rectangular	0.06	0.06	0.06	0.00
(100 x 100)	*Convex*	0.11	0.16	0.13	0.01
	Rectangular	0.01	0.01	0.01	0.00

**Convex*: Algorithm (7) running time to obtain S_{conv} and to *backtrack* the shape boundaries using the three indices: s, t, and k.
†*Rectangular*: Algorithm (2) running time to obtain S_{rect} and to locate the region using the two coordinates x and y.

The paired t-test assumes that the differences between S_{conv} and S_{rect} are approximately normally distributed. To ensure the normality assumption has been met, we interpret the box plot and histogram for the differences between S_{conv} and S_{rect} for all six matrix sizes. These showed the sample data was approximately normal for matrix sizes 200, 300, 400, 500, and 600. Removing an outlier in matrix size 100 resolved a slight deviation from normality. The qq-plots for each matrix size also suggest no significant deviations from normality. This was numerically confirmed using the Shapiro-Wilk test. This was statistically non-significant (p > .05) for all matrix sizes (Table 3.2), which gave no evidence against the hypothesis that the underlying population differences are normally distributed. Hence, there is no evidence against the assumption that the sample data are normally distributed and that there is no threat to the statistical analysis.

Table 3.2: Results of the Shapiro-Wilk test for testing the normal distribution of the differences between S_{conv} and S_{rect} for all six matrix sizes.

Matrix Size	Shapiro-Wilk *p*-value
600	0.916 > 0.05
500	0.410 > 0.05
400	0.771 > 0.05
300	0.552 > 0.05
200	0.156 > 0.05
100	0.239 > 0.05

Findings from the paired-samples t-test (Table 3.3) showed strong evidence of the difference between the maximum sums obtained from the *convex shape* and the *rectangular shape* algorithms for each matrix size. The results show that there was a statistically significant increase ($p < .01$) in the maximum sums outcome of the *convex shape* compared to the *rectangular shape* algorithm's maximum sums outcome. The standardised effect sizes indicate a large magnitude of difference between the two algorithms' outcomes, ranging from *Es* = 10 to *Es* = 14 across the various matrix sizes. These findings are important in confirming positive implications of using the *convex shape* to find the maximum sums in comparison to using the *rectangular shape* in the *MSP* framework. The new approach increased the likelihood of scoring improved results, since maximising over all of the *convex regions* return maximum sums that are no smaller than those maximised over the *rectangular regions.* In other words, in the case of not obtaining equivalent results it is inevitable that there will be larger overlapping maximum convex sums. It is essential to use methods that permit making the most of presented data and prevent missing potentially important regions.

Table 3.3: The dependent-samples t-test results (N = 100*)* of the paired differences (μ_{conv} - μ_{rect}) of the various matrix sizes using the *convex shape* Algorithm (7) and the *rectangular shape* Algorithm (2).

Matrix Size	Paired Differences (μ_{conv} - μ_{rect}) 95% Confidence Interval [Lower, upper]	Sig. (*p*-value)
600x600	112373 [110818, 113928]	<.01
500x500	91950 [90594, 93307]	<.01
400x400	70031 [68913, 71148]	<.01
300x300	49522 [48760, 50284]	<.01

200x200	30229 [29723, 30735]	<.01
100x100	12742 [12491, 12992]	<.01

3.3.1.2 The *K-DMSA* and *K-DMCS* algorithms: Algorithms (9) and (10)

In the previous section evidence was presented for the analytical differences in the maximum sum outcomes using Algorithms (2) and (7). This section outlines experiments to assess differences of results for the *K-DMSA* and *K-DMCS* algorithms: Algorithms (9) and (10), respectively. The expectation for this section is that the pattern of results observed above will be similar. Using these findings as guidelines and adapting the power analysis calculation, a sample size $N = 30$ was selected. This sample size was chosen as being sufficient to detect differences and dismiss potential threats to the validity of the statistical test. Table 3.4 describes the outcomes obtained using the *K-DMCS (*Algorithm (10)) and *K-DMSA* (Algorithm (9)), where K=1, 2, and 3 are, in order, the first, second, and third located regions. Differences and their directions between the two algorithms' outcomes were clear from the observed categorical figures of the corresponding K digits. The differences, as noted previously, are because of the distinctions between the algorithmic framework, their core shapes, and the ways in which the two algorithms function.

Table 3.4: Descriptive results for the maximum sums and running times (in seconds) obtained using the *K-DMSA* (Algorithm (9)) and *K-DMCS* (Algorithm (10)); N = 30. K = 1, 2, and 3 are, in order, the first, second, and third located regions.

		Disjoint maximum sub-arrays (***rectangular shape***)†			Disjoint maximum convex sums (***convex* shape**)*		
Matrix Size	Mean Outcome	K=1	K=2	K=3	K=1	K=2	K=3
300x300	Sum	27483	11665	8240	76994	31675	21779
	Running Time	0.54			7.42		
250x250	Sum	21247	9936	7477	60914	26433	18496
	Running Time	0.31			4.45		
200x200	Sum	17775	7338	5501	48093	19867	14103
	Running Time	0.15			2.04		
150x150	Sum	13090	6257	3961	33940	14365	9179
	Running Time	0.06			1.04		
100x100	Sum	8792	3843	2690	21061	8088	5861
	Running Time	0.03			0.24		
50x50	Sum	3923	1820	1235	9160	3314	2269
	Running Time	0.002			0.02		

†*Rectangular shape*: Algorithm (9) K-disjoint maximum sub-arrays and the experiments running time. The time complexity is $O(Kn^3)$.
**Convex shape*: Algorithm (10) K-disjoint maximum convex sums and the experiments running time. The time complexity is $O(Kn^3)$.

The normality assumption of the differences between S_{conv} and S_{rect} for all six matrix sizes was checked. The boxplot and histogram of the differences both showed that the sample data was approximately normal for the disjoint outcomes of the matrix sizes 300, 250, 200 (K = 1 and K = 3), 150, 100, and 50. Removing two influential outliers in matrix size 200 in the second disjoint region (K = 2) outcome resolved a deviation from normality. The patterns in the qq-plots for each matrix size seem to be consistent with the assumption of normality. The Shapiro-Wilk test gave no evidence (p > 0.05) against the hypothesis that the underlying population differences are normally distributed (Table 3.5), for all matrix sizes. Thus, there appears to be no evidence against the assumption that the sample data were approximately normally distributed.

Table 3.5: Results of the Shapiro-Wilk test for testing normal distribution of the differences between S_{conv} and S_{rect} for all six matrix sizes on K = 1, 2, and 3.

	Shapiro-Wilk p-value (p > .05)		
Matrix Size	K=1	K=2	K=3
300x300	.096	.461	.253
250x250	.666	.179	.924
200x200	.982	.449	.759
150x150	.796	.832	.183
100x100	.566	.778	.740
50x50	.958	.064	.303

Findings from the paired-samples t-test (Table 3.6) show evidence of the difference between the maximum sums outcome obtained from the *K-DMCS* algorithm and those from the *rectangular shape* algorithm, for each matrix size. There were statistically significant increases (p < .01) in the K disjoint maximum sums outcomes of the *convex shape* algorithm compared to those of the *rectangular shape* algorithm. The standardised magnitudes of differences were relatively large (1.5 < *Es* < 13) for the given matrix sizes, which suggests a potential advantage of using the *convex shape* to obtain the K disjoint maximum sums compared to using the *rectangular shape* in *the MSP* algorithms. The interpretation of this can be attributed to the flexibility of the *convex shape* when seeking maximum convex regions, which otherwise could not be covered using the rectangular segments. Thus, from both the experiment findings and the underpinning theoretical knowledge, the *convex shape* provides a 'win-win' situation because the *convex regions* maximum sums will be at least equivalent, if not exceed those of the *rectangular regions* maximum sums.

Table 3.6: The dependent-samples t-test results of the paired differences (μ_{conv} - μ_{rect}) of the various matrix sizes using *K-DMSA* and *K-DMCS* algorithms; N = 30. K = 1, 2, and 3 are, in order, the first, second, and third located regions.

Matrix Size	K	Paired Differences ($\mu_{conv} - \mu_{rect}$) 95% CI [Lower, upper]	Sig. (p-value)
300x300	1	49512 [48014,51010]	< 0.01
	2	20010 [17827,22194]	< 0.01
	3	13539 [11991,15088]	< 0.01
250x250	1	39667 [38403,40931]	< 0.01
	2	16497 [14796,18198]	< 0.01
	3	11019 [9311,12727]	< 0.01

200x200	1	30318 [29295,31342]	< 0.01
	2	11446 [10189,12703]	< 0.01
	3	8602 [7514,9690]	< 0.01
150x150	1	20849 [20031,21668]	< 0.01
	2	8108 [6795,9421]	< 0.01
	3	5219 [4337,6100]	< 0.01
100x100	1	12269 [11786,12752]	< 0.01
	2	4245 [3594,4896]	< 0.01
	3	3171 [2751,3591]	< 0.01
50x50	1	5236 [5014,5459]	< 0.01
	2	1494 [1192,1797]	< 0.01
	3	1033 [808,1259]	< 0.01

3.4 Chapter summary

All previous *MSA* studies have focused upon using the *rectangular shape* to obtain the maximum sums. In our research, a different approach to this focus is presented. This approach is manifested by using the *convex shape* in the *MSP* context, and defining a new problem, the *MCS*. The rationale behind using the *convex shape* is its flexibility to cover at least the same regions as those processed by the *rectangular shape* algorithms, but with the added advantage of the possibility of detecting potentially overlooked maximum sum regions.

This study's contributions to the field of *MSP* are described here. The new *MCS* problem is defined and a simplified preliminary *MCS* algorithm is implemented. A *bidirectional* approach is used to implement a more efficient algorithm which has the same time complexity compared to the base algorithm developed by T. Fukuda *et al*. The new version of the *MCS* algorithm has been evaluated and integrated techniques to advance its operational utility. The *prefix sum* has been used as a pre-processing technique to overcome the costly repetition of unnecessary operations. *Backtracking* is used as a post-processing technique to determine the maximum convex sum region boundaries, which thereby allows for further development of the algorithm. Most significant is the development of an original algorithm that generalises the *MCS* problem to cover the *K* disjoint maximum convex sums, running in $O(Kn^3)$ time complexity.

Chapter Four

[*K-Overlapping Maximum Convex Sum* problem]

A new problem of the *K-Overlapping Maximum Convex Sum* (*K-OMCS*) using the *WN convex shape* is presented in this chapter, and is also published in [62], a paper that is an outcome of this research. Prior to this, Bengtsson and Chen studied the overlapping *MSA* problem in 1D [22-24]. Their algorithm produced a list of *K* maximum sub-arrays in sorted order in a time complexity of $O(n+K\log n)$ [22-24]. Bae also investigated the *K-OMSA* problem in 1D; no particular order was assumed in his work [16]. Takaoka [14] further advanced Bae's algorithmic framework in 1D based on the *DMM*. He developed an algorithm to compute a very small *K*: $K \leq O((\alpha \log n/\log\log n)^{1/2})$, where $0 \leq \alpha \leq 1$. The problem was thereby extended to 2D with a time complexity $O(Kn^3)$ [16]. Bae and Takaoka [19] developed an algorithm to solve the problem in subcubic time with a very small *K*: $K \leq O((\log n/\log\log n)^{1/2})$. The field of *MSA* research to find the maximum sums in the best time complexity has evolved and there is now a robust body of scholarly works and publications addressing the problem using the *rectangular shape* [17-21]. None of the existing *MSA* studies have attempted to investigate this problem by using a shape other than a *rectangular* one. In this research, however, we have deviated from using the traditional *rectangular shape* for the *MSA* problem. Rather we use the *convex shape*, and thus initiate a new research area in the *MCS* field. This new approach improves the maximum sums outcomes within the time complexity of the existing algorithms [38, 61, 63, 64]. We have continued our research beyond the newly formulated problems (*MCS* and *K-DMCS*) to address the *K-OMCS* problem [62]. In this Chapter, we formalise the definition of the new overlapping problem and present the *K-OMCS* algorithm in $O(Kn^3)$ time.

This chapter is organised as follows: Section 4.1 defines the problem; Section 4.2 presents the newly developed algorithm for the *K-Overlapping Maximum Convex Sum* (*K-OMCS*) problem; Section 4.3 discusses the experimental analysis, and Section 4.4 provides a chapter summary.

4.1 Problem definition

We start by defining the *K*-overlapping maximum sub-array problem using a basic case of a one-dimensional array which provides a preliminary insight into the problem. The problem is then elaborated to the two-dimensional array, also using the *rectangular shape*, before discussing our new approach, using the *convex shape* (Algorithm (11)).

For a given array a[1..*n*] containing positive and negative numbers and 0, the maximum sub-array is the consecutive array elements that add up to the greatest sum. Let MAX(*K*, *L*) be the operation that selects the *K* largest elements in a list *L*. The definition of the *K* overlapping maximum sub-arrays is given in Equation (4.1) below:

$$R = MAX\,(K, L), \text{ where } L = \left\{ \sum_{x=i}^{j} a[x] \quad | \quad 1 \le i \le j \le n \right\} \ldots\ldots\ldots\ldots\ldots \quad (4.1)$$

Here, the *K* maximum sums are stored in *R*[1..*K*]. Note that the solution set, *R,* is in sorted order.

Example (4.1):

Let array *a* = {3, 51, -41, -57, 52, 59, -11, 93, -55, -71, 21, 21}, where the first element is indexed as 1 and the last element is indexed as 12 i.e. a[1] = 3 and a[12] = 21. The size of array *a* is 12 and a total of 78(= 12(12+1)/2) sub-arrays exist. Amongst this set, the first maximum sub-array is 193. This results from *a*[5]+*a*[6]+*a*[7]+*a*[8]. We denote the first maximum sub-array by 193(5, 8). When overlapping is allowed, the second and third maximum sub-arrays are 149(1, 8) and 146(2, 8). The 78th overlapping maximum sub-array is −126 (9, 10). Using this example, we can elaborate the problem in 2D.

Bae and Takaoka [16] utilised the *rectangular shape* to find the *K*-overlapping maximum sub-arrays in 2D. They devised an algorithm based on *prefix sum* for 1D where *sum*[*i*]=*a*[1]+..+*a*[*i*], which is pre-computed as *sum*[*i*-1]+*a*[*i*] in *O*(*n*) time as highlighted in Algorithm (4). Their 1D algorithm maintains a list of the preceding *K minimum prefix sums* amongst *sum*[0],..,*sum*[*i*-1] for every position *i* (i.e. min_i=min{*sum*[0]..*sum*[*i*-1]}), where *sum*[0]=0. This results in *K candidates* for every position *i* by computing *sum*[*i*]-*x*, where *x* is the *K minimum prefix sums* ($cand_i$ = *sum*[*i*] - min{*sum*[0]…*sum*[i-1]). Finally the maximum of these candidates is selected i.e. *max_sum* = *max* {$cand_1$,$cand_2$, … $cand_{n-1}$,$cand_n$}. The same steps can further be extended to be applied in 2D. This algorithm has *O*(*Kn*) time for 1D and $O(Kn^3)$ time for 2D. The readers may refer to [16] for the detailed procedure. An example is given in Figure 4.1 to outline the *K-OMSA* solution in 2D. This is to find all possible *K* overlapping sub-arrays such that the overlapped rectangular segments are ranked from highest to lowest based on their sums in a matrix *a*. The first maximum sum in Figure 4.1 is 114 and the next overlapping maximum sum is 111. This process continues until all of the *K* overlapping maximum sub-arrays are found.

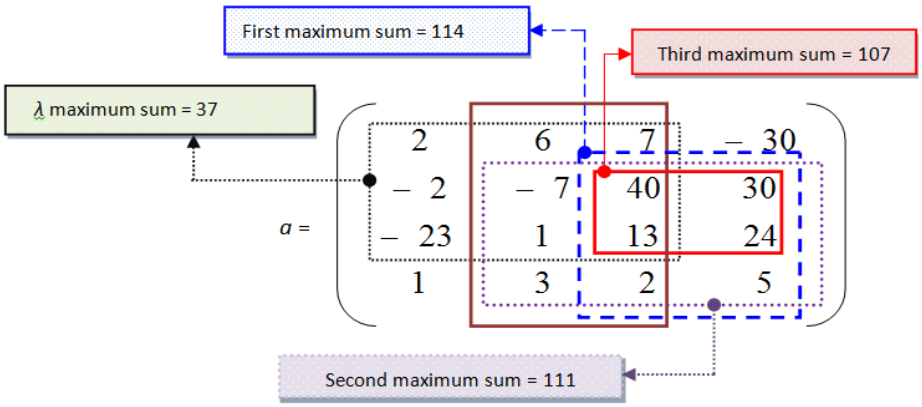


Figure (4.1): Finding the *K-OMSA* by using the *rectangular shape*, where *λ,* the maximum sum represents one of the maxima before finding the *Kth* overlapping maximum sums.

4.2 The newly developed algorithm for the *K-Overlapping Maximum Convex Sum* (*K-OMCS*) problem

This section presents the new approach of finding the *K* overlapping maximum sums using the *WN convex shape*. The implementation of the *convex shape* algorithm, which finds the first maximum convex sum, was discussed in Chapter 3. Finding the first maximum sum can be advanced to finding the *K* overlapping maximum convex sums, as shown in Figure 4.2. The newly developed algorithm of the *K-OMCS* problem is outlined in Algorithm (11).

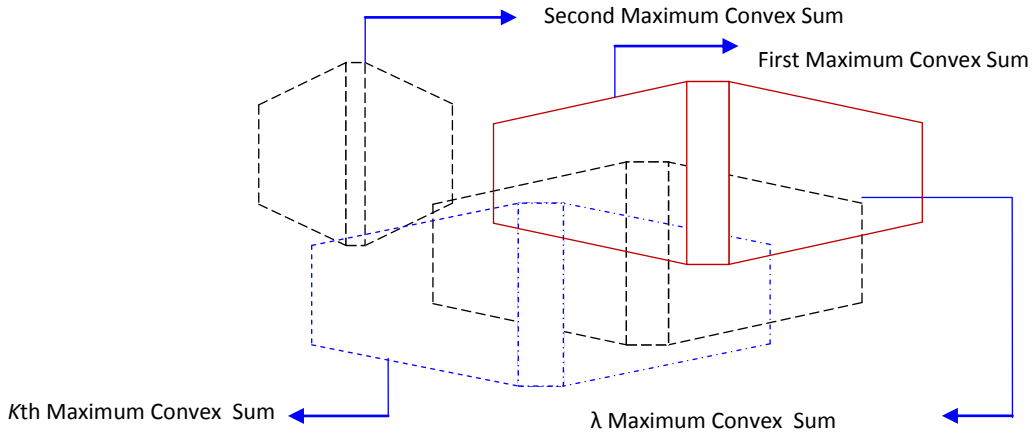


Figure (4.2): The *K* overlapping maximum convex sums in a two-dimensional array where *λ* represents one of the maxima before finding the *Kth* overlapping maximum sums.

In Algorithm (11), we extend the single values of f_W and f_N of Algorithm (7) to *K*-tuples expressed by F_w *and* F_N. The tentative maximum sum and the currently accumulated sum are also

extended to *K*-tuples in Bae and Takaoka's study, which used the *rectangular shape* [16]. Using the *convex shape* approach, Algorithm (11) employs the *K*-tuples which are as follows: Suppose *L* is a *K*-tuple (a_1, a_2, …, 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, …, L_m,, max{L_1, L_2, …, L_m} are the largest *K* numbers in the merged list of L_1, L_2 , …, L_m. For two *K*-tuples L_1 and L_2, $L_1 + L_2$ are the largest *K* numbers from the set and the Cartesian sums are used as {$x + y$ | *x* is in L_1 and *y* is in L_2} [76]. The time complexity to find the *K-OMCS* solution can be computed in $O(Kn^3)$. The proof of the time complexity is explained in the proof of Theorem 4.1. In the following, capital *F* is used to signify *K*-tuples.

Algorithm (11): *K-Overlapping Maximum Convex Sums* algorithm using the *K*-tuples $O(Kn^3)$

/* compute *prefix_sum* */

1: *prefix_sum*[0, *k*] = 0 for all *k*

2: for k=1 to *n* do

3: for s=1 to *m* do *prefix_sum*[*s*, *k*]=sum[*s*-1, *k*] + a[*s*, *k*]

/* compute F_W */

4: F_W (0, [*s*, *t*])←(0, - ∞…, -∞) for all $s \leq t$ // 0 followed by (*K*-1) -∞

5: F_W (*k*, [*s*+1, *s*])←(- ∞,- ∞, ….., -∞) for all *k* and *s*

6: for *k*←1 to *n* do

7: for all intervals of [*s*, *t*] in increasing order of *t*-*s* where $s \leq t$ do

8: sum[*k*, *s*, *t*] = *prefix_sum*[*t*, *k*] – *prefix_sum*[*s*-1, *k*]

9:
$$F_W(k,[s,t]) \leftarrow \max \begin{cases} F_W(k\text{-}1,[s,t])+\text{sum}[k,s,t] & \text{(extended case 1)} \\ F_W(k,[s\text{+}1,t])+a[s,k], & \text{(extended case 2)} \\ F_W(k,[s,t\text{-}1])+a[t,k] & \text{(extended case 3)} \end{cases}$$

10: end for *s, t*

11: end for *k*

12: F_N is similarly computed from right to left.

/** *Finalisation* **/

13: For *k*←1 to *n* do

14: For *s*←1 to *n* do

15: For $t \leftarrow s$ to n do

16: $F_{WN}(k,[s, t]) \leftarrow (F_W(k,[s, t]) + F_N(k,[s, t]) - \text{sum}[k, s, t])$

17: end for t

18: end for s

19: end for k

20: $F = \max\{F_{WN}(k,[s, t])\}$ for all k, s, t

In Algorithm (11), *Finalisation* takes $O(Kn^3)$ time (lines 13-19). This is because to compute $F_{WN}(k, [s, t]) = (F_W(k, [s, t]) + F_N(k,[s, t]) - \text{sum}[s, t, k])$, the result is placed in a triply nested structure. The addition operation '+' of the two K-tuples ($F_W(k, [s, t])$ and $F_N(k, [s, t])$) can be calculated in $O(K)$ time [76]. If sorted order is required for the K solutions, they can be sorted at the end taking $O(K\log K)$ time.

Theorem 4.1 *$F_W(k, [s, t])$ is the K maximum values of the sum of the W shape having the anchor column (k, s, t).*

Proof. The proof is based on double induction on k and t-s from smaller to larger *anchor column*s. This proof is similar to that of Theorem 3.1.1 in Chapter 3. However, Theorem 3.1.1 is related to single values, whereas, Theorem 4.1 deals with K-tuples.

Let the basis be $k = 1$. It is easy to verify $F_W(1, [s, s]) = (\text{sum}[1, s, s], -\infty, \ldots, -\infty)$, noting $F_W(1, [s+1, s]) = (-\infty, -\infty, \ldots, -\infty)$. Now for general s and t, case 1 is $(\text{sum}[1, s, s], -\infty, \ldots, -\infty)$, since $F_W(0, [s, t]) = (0, -\infty, \ldots, -\infty)$. Suppose Theorem 4.1 is true for t-$s = c$-1 for some c, where c is an arbitrary value. We prove $F_W(1, [s, t]) = (\text{sum}[1, s, t], -\infty\ldots, -\infty)$. From induction we have $F_W(1, [s+1, t]) = (\text{sum}[1, s+1, t], -\infty\ldots, -\infty)$. Thus, case 2 is $F_W(1, [s+1, t]) + a[s, 1] = (\text{sum}[1, s, t], -\infty, \ldots, -\infty)$. Case 3 is similar. Based on the program (algorithm (11), line 9).Thus

$F_W(1, [s, t]) = \max\{(\text{sum}[1, s, t], -\infty, \ldots, -\infty), (\text{sum}[1, s, t], -\infty, \ldots, -\infty), (\text{sum}[1, s, t], -\infty, \ldots, -\infty)\}$.

Similarly we can prove $F_W(k, [s, s]) = (f_W[k, s, s], -\infty, \ldots, -\infty)$.

Now suppose Theorem 4.1 is true for k -1 > 0 and t-s-1 > 0. The K maximum values of the *W shape* with the anchor, *column*(k, s, t), are obtained from three smaller *W shapes:* the *W shape* with *column*(k-1, s, t), the *W shape* with *column*(k, s+1, t), and the *W shape* with *column*(k, s, t-1). From the induction hypothesis, the F_W values of those *W shapes* are maximum with those three *anchor*

*column*s. The three cases in the program correspond to those three *W shapes*. The maximum values for F_W (k, [s, t]) are correctly computed by Algorithm (11), because the algorithm chooses maximum K values out of the three K values obtained from all possibilities of the three cases.

Let F_{WN} (k, [s,t]) be the K maximum values of the *WN convex shapes* with the *anchor column* (k, s, t).

Finalisation (Algorithm (11))

For k = 1 to n

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]

F=max{ F_{WN} (k,[s, t])} for all k, s, t

To prove the correctness of *Finalisation*, we need the following lemma by Frederickson [76].

Lemma 4.1 Let X={x_1, ..., x_n} and Y={y_1, ..., y_n} be sets of n numbers each. Let X + Y, be called the Cartesian sum, and defined by X + Y = {$x_i + y_j$ | $x_i \in X$, $y_j \in Y$}. The largest n numbers from X + Y can be computed in $O(n)$ time. Let us denote the set of n largest numbers by X + Y for simplicity.

Theorem 4.2 *Finalisation computes F_{WN} (k, [s,t]) correctly in $O(Kn^3)$ time. Then K maximum WN sums are computed.*

Proof. The proof in Section 3.1.2.1 is generalised to K-tuples. Proof is by contradiction. In the following f_W and f_W' are the f_W values of the two *W shapes* with the same *anchor column* (k, s, t). f_N and f_N' are similar. Suppose there is an *anchor column* (k, s, t) such that

$$f_W(k, [s,t]) + f_N(k, [s,t]) - \text{sum}[k, s, t] < f_W'(k, [s,t]) + f_N'(k, [s,t]) - \text{sum}[k, s, t] \quad (*)$$

where $f_W(k, [s,t]) + f_N(k, [s,t]) - \text{sum}[k, s, t]$ is in the K-tuple and $f_W'(k, [s,t]) + f_N'(k, [s,t]) - \text{sum}[k, s, t]$ is not in the K-tuple. The inequality (*) is characterised in the same way as $a+b+c < x+y+c$ in Section 3.1.2.1, which leads to a contradiction. The last part of Algorithm (11), line 20, can be computed by repeating the framework of $F \leftarrow \max\{F_{WN}(k,[s, t])\}$ for all k, s, t, taking $O(Kn^3)$ time.

4.3 The *K-OMSA* and *K-OMCS* algorithms experimental comparisons

The previous section outlined the theoretical framework and implementation of the *K-OMCS* problem, Algorithm (11). In this section, the experimental comparisons to assess the differences are discussed for outcomes obtained using the *K-OMCS* algorithm and the *K-OMSA* algorithm [16]. The procedures are the same as those which were outlined in Chapter 3 for running the experiments and conducting the statistical analysis. Comparisons of the differences between results of the *K-OMCS* and *K-OMSA* algorithms using inferential statistics are presented. Based on the preceding analytical findings and the theory supporting it, there was an expectation to see patterns that are generally similar to those outlined in the experimentation section of Chapter 3. A sample size of 30 was used based on current expectations, previous findings and the underpinning theoretical knowledge. Table 4.1 describes the outcomes obtained using the *K-OMCS* and *K-OMSA*, where *K* = 1, 2, and 3 are, in order, the first, second, and third located regions.

Table 4.1: Descriptive results of the maximum sums and running times (in seconds) obtained using the *K-OMSA* and *K-OMCS* algorithms; *N* = 30. *K*=1, 2, and 3 are, in order, the first, second, and third located regions.

		Overlapping maximum sub-arrays **(*rectangular shape*)**†			Overlapping maximum convex sums **(*convex* shape)***		
Matrix Size	Mean Outcome	*K*=1	*K*=2	*K*=3	*K*=1	*K*=2	*K*=3
300x300	Sum	28719	28604	28512	77304	77303	77303
	Running Time		1.23			8.96	
250x250	Sum	21412	21279	21206	60658	60657	60656
	Running Time		0.72			5.56	
200x200	Sum	17381	17255	17187	47470	47469	47469
	Running Time		0.37			2.82	
150x150	Sum	13473	13365	13307	34238	34237	34236
	Running Time		0.16			1.17	
100x100	Sum	8887	8830	8774	21225	21224	21223
	Running Time		0.05			0.35	
50x50	Sum	4619	4581	4538	9987	9986	9984
	Running Time		0.01			0.04	

†*Rectangular shape*: the *K* overlapping maximum sub-arrays algorithm using the *rectangular shape* runs in $O(Kn^3)$ time complexity.
**Convex shape*: the *K* overlapping maximum convex sums algorithm runs in $O(Kn^3)$ time complexity.

The normality assumption of the differences between S_{conv} and S_{rect} for all six matrix sizes was checked. The boxplot and histogram were checked for differences; both showed the sample data were approximately normal for the overlapping maximum sums of each of the matrix sizes. The qq-plots for each matrix size were consistent with the assumption of normality. The Shapiro-Wilk test gave no evidence against the hypothesis that the underlying population differences are normally distributed (Table 4.2), for all matrix sizes (Shapiro-Wilk $p > 0.05$). Thus, there is no evidence against the assumption that the sample data was approximately normally distributed.

Table 4.2: Results of the Shapiro-Wilk test for testing normal distribution of the differences between S_{conv} and S_{rect} for all six matrix sizes on K = 1, 2, and 3.

	Shapiro-Wilk p-value ($p > .05$)		
Matrix Size	K=1	K=2	K=3
300x300	.906	.936	.915
250x250	.296	.267	.247
200x200	.648	.657	.710
150x150	.108	.120	.122
100x100	.386	.438	.485
50x50	.306	.315	.287

Findings from the paired-sampled t-test (Table 4.3) for each matrix size showed strong evidence of the differences between the K overlapping maximum sums obtained using the *K-OMCS* and *K-OMSA*. There were statistically significant increases ($p < .01$) in the K overlapping maximum sums outcomes obtained using the *K-OMCS algorithm* compared to using *K-OMSA* algorithm. The effect sizes showed that there were large differences between the two algorithms' K overlapping maximum sums ($8 < ES < 13$). Findings from the experiments signify the notion of using the *convex shape* in the *MSP* context, which would improve maximising over the 'promising' regions compared to using the *rectangular shape* in the same context.

In the main, these findings endorse the new approach of this study as it provides potential benefits relating to the sensitivity of finding maximum sum regions. As noted above, maximising over all of the *convex regions* will return maximum sums that are no smaller than those maximised over all the *rectangular regions.* It is evident that results from both algorithms will be different, in the form of increased maximum convex sums, unless equivalent to those of the *rectangular shape* maximum sums. The likelihood of optimising the overall findings will increase, when using methods that will reach portions that were previously overlooked.

Table 4.3: The dependent-samples t-test results of the paired differences (μ_{conv} - μ_{rect}) of the various matrix sizes using *K-OMSA* and *K-OMCS* algorithms; *N* = 30.

Matrix Size	*K*	Paired Differences (µconv - µrect) (95% CI [Lower, upper])	Sig. (*p*-value)
300x300	1	48586 [46944,50227]	< 0.01
	2	48699 [47070,50329]	< 0.01
	3	48791 [47162,50420]	< 0.01
250x250	1	39246 [37927,40564]	< 0.01
	2	39378 [38058,40697]	< 0.01
	3	40139 [38312,41966]	< 0.01
200x200	1	30089 [29059,31119]	< 0.01
	2	30214 [29181,31248]	< 0.01
	3	30282 [29251,31313]	< 0.01
150x150	1	20765 [20017,21513]	< 0.01
	2	20872 [20121,21622]	< 0.01
	3	20929 [20184,21675]	< 0.01
100x100	1	12338 [11764,12912]	< 0.01
	2	12394 [11818,12970]	< 0.01
	3	12449 [11879,13019]	< 0.01
50x50	1	5368 [5206,5529]	< 0.01
	2	5405 [5247,5563]	< 0.01
	3	5446 [5293,5599]	< 0.01

4.4. Chapter summary

This chapter presented the original algorithm to find the *K* overlapping maximum convex sums: *K-OMCS* algorithm, with $O(Kn^3)$ time. This is the same time complexity as that of the *rectangular shape* algorithm [16]. The *convex shape* to find the *K* overlapping maximum convex sums was verified to return improved maximum sums compared with those of the *rectangular shape*. The work presented in this chapter has been published in a peer-reviewed paper [62]. The experiments showed that the first, second and third maximum convex sum regions overlap heavily. In future work, we could impose the condition that subsequent sums cannot overlap by more than 50%. This gives scope for further improvements to advance the usability of the algorithm for real-life applications.

Chapter Five

[*Application of the MCS (K-DMCS algorithm) to an ecology problem*]

In this chapter, we applied the newly developed *K-Disjoint Maximum Convex Sum* (*K-DMCS*) algorithm to a real-life application. The study presented in this chapter has been published in a peer-reviewed paper [63]. We investigated the effects of land use changes on benthic stream communities in highland tropical streams of Nigeria. A collaboration between computer scientists and freshwater biologists was established to implement and examine the robustness of this approach compared to a traditional method that used a *rectangular shape* in the *K-Disjoint Maximum Sub-Array* (*K-DMSA)* algorithm. This new approach*,* which uses a *convex shape* as compared to a *rectangular shape,* is expected to maximise the sums and to locate efficiently the maximum sum regions. This new approach was applied to data[3] (Appendix A) that were collated from 55 tropical highland streams on the Mambilla Plateau, Nigeria to investigate the interactions between substrate index, dissolved oxygen percentage (DO%) and temperature with number of macro-invertebrates (taxa). The *K-DMCS* algorithm located the *K*-maximum threshold values. This was achieved by implementing the algorithm to maximise the sum of elements of a selected portion of a 2D array on two sets of data and their interactions in relation to taxa: substrate index at a range of temperatures, and dissolved oxygen percentage at a range of temperatures. The *K-DMCS* algorithm successfully detected the various temperatures between the different categories of the substrate index and dissolved oxygen percentage, and the way in which these variables affect the numbers of macro-invertebrates. Furthermore, applying the algorithm revealed that the number of macro-invertebrates differs according to land use (e.g. forestry and agriculture). The new method used in this research is encouraging in its capability to find the relationship among various environmental parameters and macro-invertebrate distribution and diversity. This method can potentially be applied to other real-life applications that require finding associations between different parameters.

5.1 Background

It is essential for real-life studies to use analysis techniques and approaches that yield accurate outcomes, such as those when investigating relationships between different elements in ecological applications. This section reviews some of the key studies that are relevant to the ecological problem presented in this chapter, and the application of algorithms to real-life problems.

[3] The data was obtained from the *Freshwater Ecology Research Group* (*FERG*) in 2013, with permission from Professor Jon Harding (*FERG*, University of Canterbury). This data is not available for other studies, without the express permission of Professor Harding.

Habitat preference is widely viewed as a major process influencing the distribution and density of macro-invertebrates, but identifying and quantifying associated factors is not an easy task. Many hidden factors can be involved, and potential causal factors may be indefinable, such as physico-chemical parameters and land use. Studies of invertebrate interaction with habitat and environmental factors have focused on traditional ecological approaches. As an example, a study in New Zealand investigated the magnitude of effects of substrate particle size, and the influence of physical factors on benthic macro-invertebrates in 88 New Zealand Rivers [77]. In a second study, substrate size preferences of macro-invertebrates were more clearly demonstrated by the comparison of different substrates (e.g. boulders, cobbles) [78]. Although changes in light intensity were found to have a significant influence on the periodicity of drifting organisms [78], there is little information about other factors that may also affect the drift fauna of slow-flowing waters [79]. The importance of current velocity and substrate type in determining the benthic distribution of invertebrates is also well known [80-83].

The use of algorithmic methods has much to offer the biological sciences. Algorithms can contribute to knowledge discovery by extracting new aspects of results and permit new insights, in addition to those derived from general statistical methods. This real-life study provides new insights into the collaborative research, which is detailed as follows. Despite the presence world wide of a large number of studies on macro-invertebrate communities that relate substrate and other physical parameters with macro-invertebrates, apart from Olomukoro and colleagues' study [84], research on macro-invertebrate colonisation of the natural substrate in Nigerian streams is lacking. Reports informed by analysis that is based on appropriately chosen approaches and techniques help decision makers in policy development and for management processes in environmental studies. These approaches need to have attributes such as quality of performance, computation time, flexibility, and data applicability. A specific example is presented in this chapter. Researchers in benthic ecology and other environmental gradients traditionally apply statistical analysis to understand outcomes from their research data. An alternative approach for analysing data in ecology is to use advanced computer algorithms. Although the use of algorithmic computations is still in its infancy [31], there are emerging and promising approaches using algorithms to analyse datasets. The rationale behind using the *K-DMCS* approach for this particular study is two-fold. Firstly, the flexibility of this algorithm in adapting to the nature of the data in order to detect the maximum events of interest, and secondly, two-dimensional observations can take many multiple dimensions. In this research, we study the effects of two relationships: the correlation between substrate index and temperature and their influence on macro-invertebrates' habitat preference in streams; and the correlation

between dissolved oxygen percentage and temperature and their influence on the macro-invertebrates' habitat preference in streams.

In this study we applied the *K-Disjoint Maximum Convex Sum* algorithm (*K-DMCS*), which has been developed as part of this research, using the *convex shape*. Algorithms using the *rectangular shape*, the *K-MSA* algorithms, have been previously used in applications, as was outlined in Chapter 2. Studies using the *K-DMSA* included that of the K. Fukuda and Takaoka's study to investigate air pollution and health effects [30]. They used the *K-DMSA* algorithm to identify the association between PM_{10} and admission counts patients over a wide range of age groups (in 5 yearly intervals from 0 to 98 years old). They investigated this by detecting *rectangular regions* (*K*-maximum sub-arrays) with maximum hospital admission counts, rather than determining a regression coefficient relating the variables [30]. Use of the *K-DMSA* method was also explored elsewhere [4, 31-34]. In a given 2D array, the *K-DMSA* algorithm finds disjoint sub-arrays that maximize the sum. These disjoint sub-arrays are *rectangular regions* covering up to the *Kth* maximum sub-array, where *K* can be specified by the user or the application. These regions have the most promising portions in the dataset, which provide knowledge to represent large data inputs. Although using the *rectangular shape* could lead to the promising regions, it fails to provide a flexible enough shape that potentially could cover various data distributions. This motivated us to propose a new method using the *convex shape* and to apply it to situations such as the one described above. The *convex shape* is a relatively complex shape compared to the *rectangular shape* [61]. It requires three dimensions, resulting in cubic computing time, to determine the actual shape in a given matrix. Using the *convex shape* approach accurately determines the peaks of data by optimising the sums. This study advances our work reported in [61-63] but with the addition of a potential application for the *K-DMCS* algorithm.

5.2 Data collection and processing

The *convex shape* algorithm and its extended version (*K-DMCS*) were implemented in Chapter 3 using simulated data. We also implemented the *K-DMCS* algorithm using real data obtained from a study that was conducted in the tropical Montane mountain region of the Mambilla Plateau, in the south-eastern corner of Taraba state, Nigeria (11^{o}- 6^{o}E and 6^{o}- 7^{o}N). Streams sampled were near Yelwa village in the headwaters of the Donga River. The Mambilla Plateau experiences two main seasons: rainy and dry. The rainy season lasts on average 250 days, from late March until the end of October, and the dry season runs from November until mid-March. The mean annual rainfall exceeds 1780 mm, peaking in June and July, but due to its elevation, at nearly 1500m above

sea-level, the daily mean temperature rarely exceeds 30^{o}C. During the sampling period (October 2009 to March 2010), the mean air temperature was 24^{o}C. The vegetation varied markedly across the 55 streams. In the tropical Montane rainforest, the riparian vegetation was dominated by *Deinbollia pinnata, Santiria trimera, Rafania, Croton macrostachyus, Anthonotha noldeae,* and *Ficus 803*. On the intensive grazing and pasture land, however, trees such as *Acasia senegalensis, Polyscius fulva, Beeilschmedia, Pouteria altissima,* and *Bridelia speciosa* dominated. In the Montane rainforest streams, substrates were mainly larger substrate sizes particularly boulders, cobbles, and sand, while in the intensively grazed pastures and cropping streams, substrates were dominated by sand or mud.

A total of 55 headwater streams (first and second order) were sampled (Appendix A). Streams were selected to provide at least five replicates in nine different land uses. These land uses were: continuous tropical montane rainforest, forest fragment, intensive grazing, open pasture cabbage and maize fields, tea and banana plantations, and eucalyptus forests. Each stream was sampled for both physical and chemical parameters: fine particulate organic matter (FPOM), coarse particulate organic matter (CPOM), algae, and benthic invertebrates. All sites were sampled on a single occasion during the tropical dry season between October 2009 and March 2010.

To compare the diversity, density, and composition of benthic invertebrate communities among the different land uses both qualitative and quantitative samples were collected. At each site, five Surber samples (30 cm x 30 cm, 500 µm mesh size) were collected from riffles. In order to get a more extensive species list, a single composite kick-net sample (mesh 500 µm) was also collected from a range of different micro-habitats in each stream. Additionally, leaf packs, wood jams, and moss were included in this kick-net sample. Macro-invertebrates collected include mayfly (Leptophlebiidae), stonefly (Perlidae) and caddisflies (Hydropsychidae) (Figure 5.1). These species are pollution sensitive and are therefore useful as bio-indicators of environmental degradation.

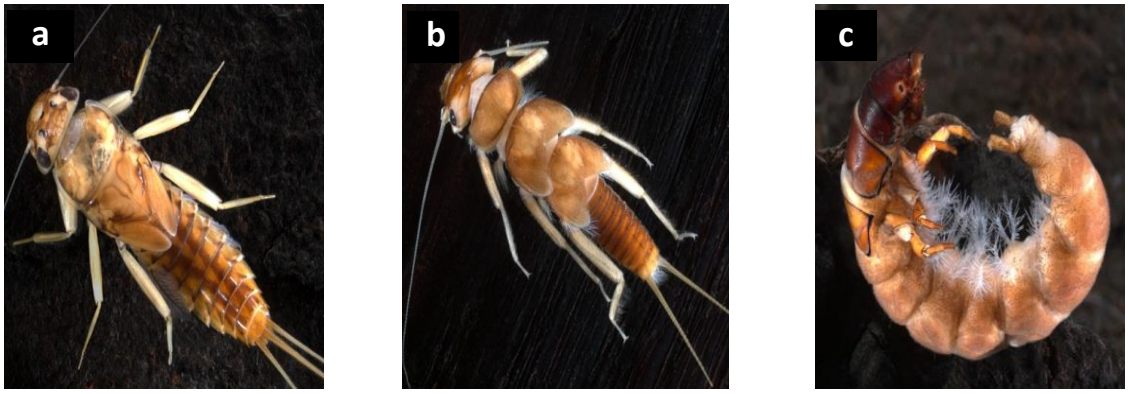


Figure (5.1): Macro-invertebrates collected in streams on Mambilla Plateau between October 2009 and March 2010. (a) Mayfly (Leptophlebiidae);(b) Stonefly (Perlidae); (c) Caddisflies (Hydropsychidae).

Ash-free dry mass (AFDM) of algae and moss were obtained by scraping stones with a wire brush and washing the slurry into clean bottles. These samples were later filtered using a syringe and filter paper and subsequently weighed in the laboratory. A composite FPOM sample was obtained by placing a 60 µm net in the stream for 20 minutes. The FPOM was transferred into a bottle in the field, and filtered; oven dried, and weighed in the laboratory. CPOM was obtained from the Surber samples. The CPOM was oven dried and ashed at 450^{o}C for three hours to calculate AFDM.

All collected benthic invertebrates were preserved in 70% ethanol in the field and returned to the laboratory for analysis. In the laboratory, samples were filtered using a 500µm sieve, sorted, and identified to the lowest taxonomic level possible. The taxonomy of Nigerian aquatic invertebrates is poorly known therefore we used a combination of keys by [85, 86].

At each site, representative reaches at least 10 m long containing a riffle/run/pool complex were selected. Furthermore, physico-chemical factors were measured in situ including temperature, pH, conductivity (using a 4-star Orion pH/conductivity portable meter), and dissolved oxygen percentage (using WTW oxygen meter). Filtered water samples were also collected. These were analysed in the laboratory for nitrate and phosphate content using the Molybdenum blue calorimetric method (*R.J. Hill laboratories Ltd*, *Hamilton New Zealand*).

In the field, parameters such as substrate size and composition, wetted width, depth, and current velocity were measured. Channel stability was also evaluated at each site using the method of Pfankuch [87]. The *K-DMCS* algorithm we had recently developed was then applied to the data to examine the impacts of the two relationships of particular interest, namely: the relationship between substrate index and temperature and the influence on the habitat preference of macro-

invertebrates in streams, and the relationship between dissolved oxygen percentage and temperature and the influence on the habitat preference of macro-invertebrates in streams.

5.3 Results and discussion

The *K-DMCS* algorithm, Algorithm (10) was implemented to find optimal association results between the research parameters using the *convex shape*. The results were compared with the results from Algorithm (9), which used the *rectangular shape*, to find the maximum sub-arrays. In Figures 5.2, 5.3, and 5.4, we set $K = 3$ and the three disjoint maximum sums are expressed as S_1, S_2 and S_3. A weight factor, w is introduced to show various degrees of maximum sub-array in different regions of the matrix as formulated in Equation (5.1). This weight factor in most cases is the mean value of all elements in matrix M, or can be determined by the user. This process is used to normalise the matrix so it has positive and negative values. In Equation (5.1), where M' is the matrix after normalisation, M is the original matrix and w is an anchor value.

$$M' = M - w \quad \text{..... (5.1)}$$

The number of macro-invertebrates were analysed based on the two environmental parameters under investigation: substrate index and dissolved oxygen percentage with temperature. First we investigated the relationship between substrate index and temperature and its influence on the macro-invertebrate habitat preference in streams. We started our procedure by normalising the original matrix (M) through subtracting the anchor value ($w = 28$) using Equation (5.1); the matrix elements are normalised numbers of macro-invertebrates. The normalised data are stored in Matrix (1) in Figure 5.2. Algorithms (9) and (10) were run on Matrix (1). The first three peaks from the data showed the same regions, suggesting that maximising over all *convex regions* returns a value that is at least as large as that returned by maximising over all *rectangular regions*.

The *rectangular shape* Algorithm (9) and the *convex shape* Algorithm (10) return the same result for $K = 1$, $S_1 = 117$.

Algorithm (9) and Algorithm (10) return the same results for $K = 2$, $S_2 = 54$.

Algorithm (9) and the Algorithm (10) return the same result for $K = 3$, $S_3 = 37$.

Temperature °C

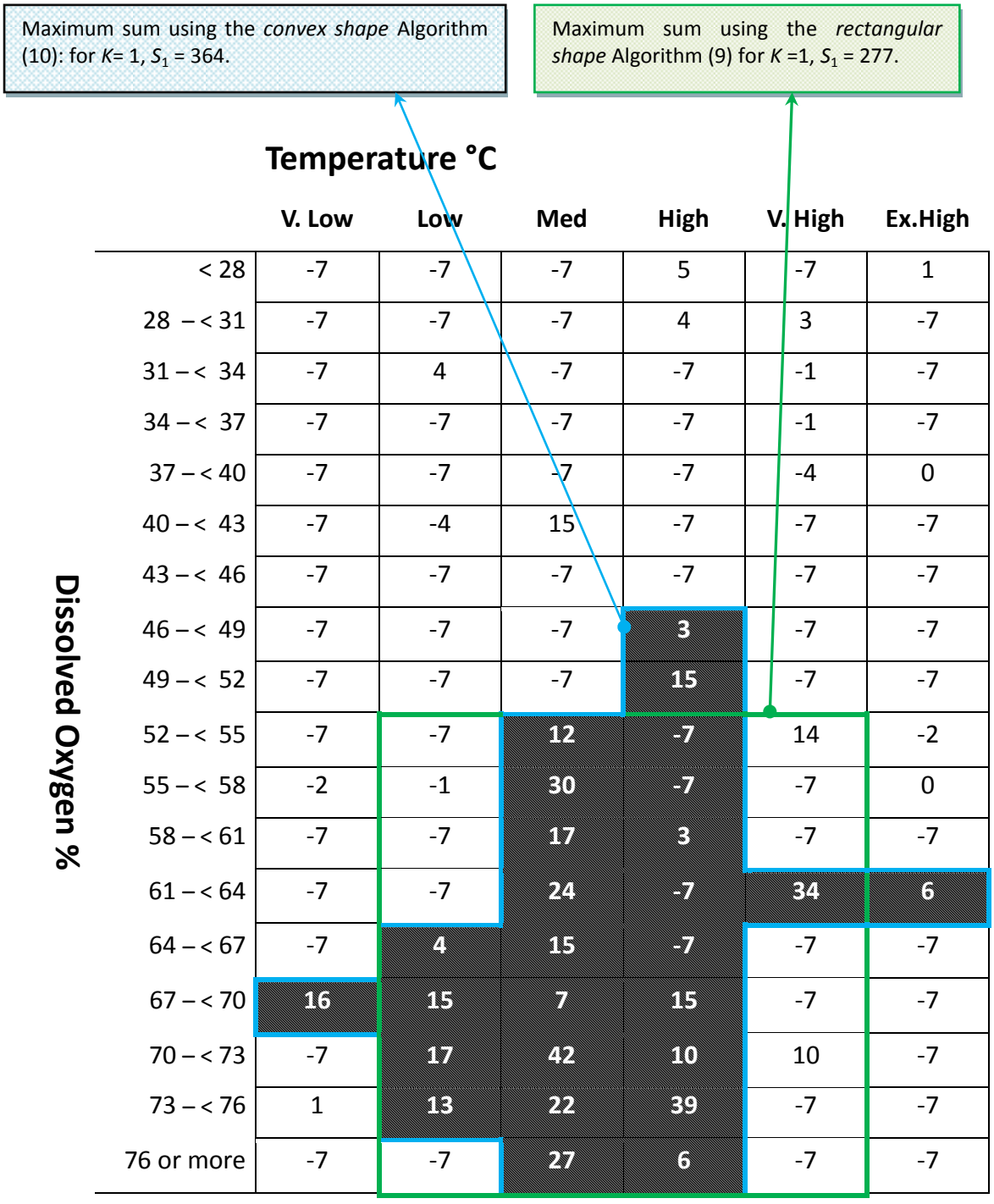
Substrate Index	V. Low	Low	Med	High	V. High	Ex.High
3 -- < 3.5	-28	**3**	**10**	**-11**	**35**	-7
3.5 -- < 4	-23	-28	-19	-17	-28	-21
4 -- < 4.5	-25	-28	-7	-28	-25	-21
4.5 -- < 5	-28	-19	-11	-28	-28	-28
5 -- < 5.5	0	-4	-13	-28	-28	-28
5.5 -- < 6	-28	-28	**54**	-28	-22	-28
6 -- < 6.5	-28	-15	-16	-20	-28	-28
6.5 -- < 7	-28	-28	-15	-11	-28	-28
7 -- < 7.5	-28	-28	-28	-16	-22	-28
7.5 -- < 8	-5	-28	**39**	**78**	-5	-28

Figure (5.2): Matrix (1) includes the normalised data collated from the Mambilla Plateau (Nigeria). In the matrix are three variables: substrate index, temperature, and numbers of macro-invertebrates.

The findings of this study suggest that macro-invertebrates have a preference for large substrate size. Three major peak areas were captured by the two Algorithms (9) and (10). These peaks were based on the interactions between temperature, substrate index and macro-invertebrates (Figure 5.2). The first peak was the medium to high temperature range in habitats with large substrate size (e.g. boulders) which supported the highest density of macro-invertebrates; this region had the first maximum convex sum ($S_1 = 117$). The second peak was the medium temperature in habitats with moderate substrate type (e.g. cobbles) which was the next favourable; this region had the second maximum convex sum ($S_2 = 54$). The third peak was at a very high temperature and low substrate index (e.g. mud and sand) where the lowest macro-invertebrate density was recorded; this region had the third maximum convex sum ($S_3 = 37$).

Results from this research also showed the interaction of macro-invertebrates with temperature and dissolved oxygen percentage analysed using the *K-DMCS* algorithm – Matrix (2) in Figure 5.3. Matrix (2) represents the interactions between temperature, dissolved oxygen percentage and macro-invertebrates. The anchor value (*w = 7*) is shown in Figures 5.3 and 5.4. In these figures, the first peak region (S_1 = 364) is captured by Algorithm (10). We identified three important informative areas: (a) at medium and high temperature ranges habitats having DO% between 46 – <49 to 76 or more recorded the highest macro-invertebrate richness; (b) high temperature to extremely high temperature ranges in habitats with DO% 61 – <64 is next in macro-invertebrate richness; and (c) at very low temperatures in habitats having DO% 67 – < 70 the lowest macro-invertebrate richness was recorded. The *convex shape* successfully identifies these areas.

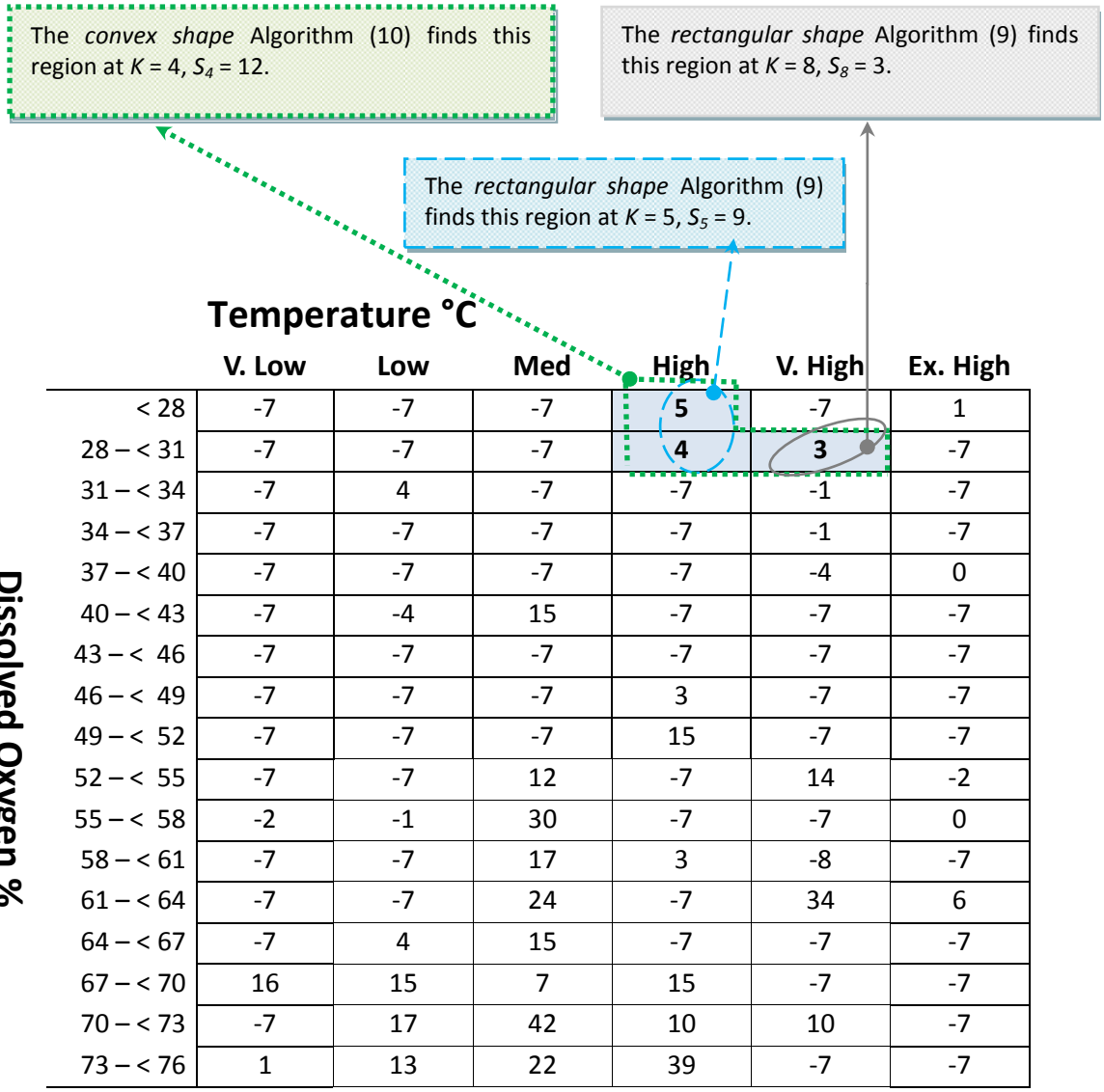
Others have had similar results in different geographic contexts. For example, Quinn *et al.* [77], who studied the magnitude of effects of substrate particle size on benthic invertebrates in 88 New Zealand Rivers, found that the distribution of common macro-invertebrates in the Mohaka River indicated significant substrate preference, where taxonomic richness and total densities of macro-invertebrates were least in patches of sand and small gravel. Jowett and Richardson [88] investigated microhabitat preferences of benthic invertebrates in Waingawa River, New Zealand. Their study reported that mayflies, stoneflies, cased caddisflies, and Diptera show a preference for a substrate index of more than 6 (boulder/cobbles) and beetles' preference for a substrate index of about 5.6 (gravel/cobble) habitat, while the operational taxonomic unites (OTUs) spread to occupy the full range of particulate substrate. The pollution sensitive macro-invertebrates (Mayflies larvae, stoneflies larvae and caddisflies larvae) were mostly associated with habitats having big substrates.



	V. Low	Low	Med	High	V. High	Ex.High
< 28	-7	-7	-7	5	-7	1
28 – < 31	-7	-7	-7	4	3	-7
31 – < 34	-7	4	-7	-7	-1	-7
34 – < 37	-7	-7	-7	-7	-1	-7
37 – < 40	-7	-7	-7	-7	-4	0
40 – < 43	-7	-4	15	-7	-7	-7
43 – < 46	-7	-7	-7	-7	-7	-7
46 – < 49	-7	-7	-7	**3**	-7	-7
49 – < 52	-7	-7	-7	**15**	-7	-7
52 – < 55	-7	-7	**12**	**-7**	14	-2
55 – < 58	-2	-1	**30**	**-7**	-7	0
58 – < 61	-7	-7	**17**	**3**	-7	-7
61 – < 64	-7	-7	**24**	**-7**	**34**	**6**
64 – < 67	-7	**4**	**15**	**-7**	-7	-7
67 – < 70	**16**	**15**	**7**	**15**	-7	-7
70 – < 73	-7	**17**	**42**	**10**	10	-7
73 – < 76	1	**13**	**22**	**39**	-7	-7
76 or more	-7	-7	**27**	**6**	-7	-7

Figure (5.3): Matrix (2) includes the normalised data collated from the Mambilla Plateau (Nigeria) taking into account three variables: DO%, temperature, and number of macro-invertebrates. It also shows the first maximum sum by running the Algorithms (9) and (10). The *convex shape* is more flexible in covering various data distributions and manifests a greater maximum sum in comparison to the *rectangular shape*, which not only returns a smaller maximum sum, but also contains less useful data (noise).

The choice of the *convex shape* supported a more efficient way of capturing all datasets at once; this is outlined in Matrix (2) of Figure 5.4.



Temperature °C

Dissolved Oxygen %	V. Low	Low	Med	High	V. High	Ex. High
< 28	-7	-7	-7	**5**	-7	1
28 – < 31	-7	-7	-7	**4**	**3**	-7
31 – < 34	-7	4	-7	-7	-1	-7
34 – < 37	-7	-7	-7	-7	-1	-7
37 – < 40	-7	-7	-7	-7	-4	0
40 – < 43	-7	-4	15	-7	-7	-7
43 – < 46	-7	-7	-7	-7	-7	-7
46 – < 49	-7	-7	-7	3	-7	-7
49 – < 52	-7	-7	-7	15	-7	-7
52 – < 55	-7	-7	12	-7	14	-2
55 – < 58	-2	-1	30	-7	-7	0
58 – < 61	-7	-7	17	3	-8	-7
61 – < 64	-7	-7	24	-7	34	6
64 – < 67	-7	4	15	-7	-7	-7
67 – < 70	16	15	7	15	-7	-7
70 – < 73	-7	17	42	10	10	-7
73 – < 76	1	13	22	39	-7	-7

Figure (5.4): A comparison between the maximum sum outcomes of the *rectangular shape* and *convex shape* algorithms. Algorithm (10), the *convex shape*, achieves the maximum sum $S_4 = 12$ at $K = 4$. Algorithm (9), the *rectangular shape,* achieves a maximum sum $S_5 = 9$ at $K = 5$, and a maximum sum $S_8 = 3$ at $K = 8$. The *convex shape* is more flexible than the *rectangular shape* in covering data distribution.

5.4 Chapter summary

Scientific investigations in this chapter were achieved by establishing collaboration between the *Algorithm Engineering and Formal Methods Research Group* (*AEFMRG*) and the *Freshwater Ecology Research Group* (*FERG)*, University of Canterbury (New Zealand). The collated data in this study were gathered by one of the originators of this research, who is a member of *FERG*, University of Canterbury. The collaboration investigated the effects of land use changes on benthic stream communities in highland tropical streams on Mambilla Plateau in North-East Nigeria using the newly developed *K-DMCS* algorithm. In previous research the *K-DMSA* algorithm was used in several applications. It has been natural for these studies to focus on using the *rectangular shape* in the *MSA* frameworks, because of the lack of studies exploring alternative shapes for *MSP*. The *MSP* has taken a new turn by using the concept of maximum convex sums. The *convex shape* has several useful attributes that increase the utility of *MSP* in real-life applications. In this chapter, we demonstrated that although there is the same time complexity for both algorithms the maximum sums are improved when using the *convex shape* in preference to the *rectangular shape* in the *MSP* context. This research applied the *K-DMCS* algorithm to an ecology setting, where it demonstrated improved results compared to outcomes obtained from applying the *rectangular shape*.

The *K-DMCS* algorithm successfully detected various temperatures between different categories of the substrate index or dissolved oxygen percentage, and the ways in which these variables affected the number of macro-invertebrates showed a preference for a large substrate size. The pollution sensitive macro-invertebrates (mayflies larvae, stoneflies larvae and caddisflies larvae) were most closely associated with habitats having a large substrate index. An experimental investigation using the *K-DMCS* algorithm by changing the weight parameter successfully returns specific and detailed results by maximising the sum to find the peaks of the data. Overall, this study provides a new approach that potentially can be applied to other *MSP* applications.

Chapter Six

[*Concluding remarks and future work*]

6.1 Concluding remarks and future work

Since 1977 *MSA* research has focused on using the *rectangular shape* for computing the maximum sub-arrays [1-41]. This approach, whilst useful in some contexts can be further enhanced for *MSP* by using a different shape. Using the *convex shape* in the *MSP* context offers the flexibility to locate regions that have the maximum sums. There is a lack of studies in exploring and applying a new shape in the *MSP* context; studies have rarely explored this frontier for examining a new shape to find the maximum sums with best possible time complexity. This research creates a different perspective for *MSP* and introduces a new shape for it. We have deviated from what has been the previous norm for exploring and investigating new methods to improve *MSP* outcomes and its efficacy for potential applications. Our approach to *MCS* provides a firm foundation and a new research path for developing further versatile algorithms for problems associated with *MSP*.

This chapter highlights the main contributions of this thesis to the *MSP*:

- Investigation and application of a *convex shape* in the *MSP* context and creation of a new *MCS* framework. We developed an independent *MCS* algorithmic framework to find regions that maximise the sum, running within the state-of-the-art time complexity of *MSA* algorithms. At the start, to explore the *convex shape* in the *MSP* context, T. Fukuda*'s* algorithm was simplified to find the maximum convex sums. This simplification created an algorithm that uses three cases compared to the seven required for T. Fukuda*'s* algorithm, and both yielded the same results. The *bidirectional* approach was applied to achieve this goal. The simplified algorithm, using the *convex shape*, returned a sum at least as large as that returned by the *rectangular shape* algorithm. This algorithm was then evaluated to create the groundwork for developing advanced *MCS* algorithms (*Chapter 3*).

- Integration of pre-processing and post-processing techniques to increase the effectiveness of the newly developed *MCS* algorithms and their utility in real-life applications. We used the *prefix sum* method to enhance the algorithms' performance by eliminating costly repetitions. We also devised the algorithms to use the *backtracking* technique to back trace and precisely determine the shape boundaries. This creative method provided adequate grounds for the generalisation of the newly developed *MCS* algorithms to cover cases involving computation of the *K* maximum sums (*Chapter 3*).

- Development of new algorithms using the *MCS* framework and the generalisation of those for the *K-Maximum Convex Sum* (*K-MCS*), which finds the *K* maximum convex regions in an input two-dimensional array. We developed two *K-MCS* algorithms to cover the *K-Disjoint Maximum Convex Sum* (*K-DMCS*) and the *K-Overlapping Maximum Convex Sum* (*K-OMCS*) – both algorithms running in $O(Kn^3)$ time (*Chapters 3 and 4*).

- Conducted scientifically designed experiments using simulated data to evaluate outcomes of the newly devised *MCS* algorithms in comparison with their *MSA* counterpart. We implemented the algorithms on the simulated data to make the comparisons and carry out statistical analysis. These revealed the relevance and vitality of exploring the diversion from using the *rectangular shape* in the *MSP* approach and the introduction of the *convex shape* in that context, because the *convex shape* algorithm returns significantly larger maximum sums (*Chapters 3 and 4*).

- Application of the newly defined *MCS* problem to a real-life setting. We established collaboration with the ecologists from the University of Canterbury. We implemented the theoretical framework and algorithms using data extracted from an ecological experiment. We used the *K-DMCS* algorithm to find the *K*-maximum regions that represent relationships and interactions in the data. We investigated the effects of changes in land use on benthic stream communities in highland tropical streams of Nigeria. We compared the results to those obtained by using the traditional method (*rectangular shape* in the *K-DMSA)*. Using the *K-DMCS* algorithm we successfully detected biologically significant relationships between the study variables, as well as demonstrating the applicability of our approach to real-life contexts (*Chapter 5*).

To conclude, this research provides the theoretical and practical framework that makes a significant scientific contribution to addressing some of the research gaps in *MSP* problem and the new research path, the *MCS* problem. The *MCS* algorithmic model departs from using the *rectangular shape* as in *MSP,* and retains a time complexity that is within the best known time complexities of the *MSA* algorithms. The study provides a strong foundation for further investigations in this new area of research. Future in-depth studies on the *Maximum Convex Sum (MCS)* problem can advance the algorithms developed in this thesis and their time complexity.

Future studies will include research to improve the time complexity of the simplified *convex shape* algorithm with the aim of achieving subcubic time. Our research has demonstrated computing the first maximum convex sum running *in* $O(n^3)$ time. Further research can investigate improving the time complexity of our original *K-OMCS* algorithm, from $O(Kn^3)$ to a subcubic time by reducing the time complexity for the heavily overlapped regions. Such a future study could impose the condition that the subsequent maximum sums within the matrix do not overlap by more than 50%. This may result in more efficient algorithms for real-life applications. We have already commenced preliminary investigations into this with the goal of improving the computational time of the algorithms developed in this thesis [38]. The possibility of improving the time complexity for the *K-OMCS* from $O(Kn^3)$ to $O(n^3+Kn^2)$ is demonstrated in this exploratory paper [38]. This algorithm is based on the conjecture that identical shapes are reported at most n times. That is, if there are n identical shapes, the duplication check process requires additional computation time. Improving the time complexity needs rigorous further investigations, specifically into the process of checking the identical shapes.

References

[1] J. Bentley, "Programming pearls: algorithm design techniques," vol. 27, ed: ACM, 1984, pp. 865-873.

[2] J. Bentley, "Programming pearls: perspective on performance," vol. 27, ed: ACM, 1984, pp. 1087-1092.

[3] U. Grenander, *Pattern analysis* vol. 2. New York: Springer-Verlag, 1978.

[4] S. E. Bae, "Sequential and parallel algorithms for the generalized maximum subarray problem," Ph.D.Thesis, University of Canterbury, New Zealand, 2007.

[5] T. Takaoka, "A new upper bound on the complexity of the all pairs shortest path problem," *Information Processing Letters,* vol. 43, pp. 195-199, 1992.

[6] D. Gries, "A note on a standard strategy for developing loop invariants and loops," *Science of Computer Programming,* vol. 2, pp. 207-214, 1982.

[7] D. R. Smith, "Applications of a strategy for designing divide-and-conquer algorithms," *Science of Computer Programming,* vol. 8, pp. 213-229, 1987.

[8] Z. Wen, "Fast parallel algorithms for the maximum sum problem," *Parallel Computing,* vol. 21, pp. 461-466, 1995.

[9] K. Perumalla and N. Deo, "Parallel algorithms for maximum subsequence and maximum subarray," *Parallel Processing Letters,* vol. 5, pp. 367–373, 1995.

[10] K. Qiu and S. Akl, "Parallel maximum sum algorithms on interconnection networks," Queen's University, Department of Computer and Information Science, Tech. Rep., 1999.

[11] K. Qiu and S. G. Akl, "Finding the maximum subsequence sum on interconnection networks," *Int. J. Parallel Emerg. Distrib. Syst.,* vol. 22, pp. 371-385, 2007.

[12] H. Tamaki and T. Tokuyama, "Algorithms for the maximum subarray problem based on matrix multiplication," in *Proceedings of the 9th Annual ACM-SIAM Symposium on Discrete Algorithms*, San Francisco, California, USA, 1998, pp. 446-452.

[13] T.-C. Lin and D. T. Lee, "Randomized algorithm for the sum selection problem," *Theoretical Computer Science,* vol. 377, pp. 151-156, 2007.

[14] T. Takaoka, "Efficient algorithms for the maximum subarray problem by distance matrix multiplication," *Electronic Notes in Theoretical Computer Science,* vol. 61, pp. 191-200, 2002.

[15] S. E. Bae and T. Takaoka, "Parallel approaches to the maximum subarray problem," in *Proceedings of the 7th Japan-Korea Workshop on Algorithms and Computation*, 2003, pp. 94-104.

[16] S. E. Bae and 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, pp. 247–253.

[17] S. E. Bae and T. Takaoka, "Improved algorithms for the K-maximum subarray problem for small K," in *Proceedings of the 11th Annual International Conference on Computing and Combinatorics*, 2005, pp. 621-631.

[18] S. E. Bae and T. Takaoka, "Algorithm for K disjoint maximum subarrays," in *Computational Science – ICCS 2006*. vol. 3991, V. Alexandrov, G. Albada, P. A. Sloot, and J. Dongarra, Eds.: Springer Berlin Heidelberg, 2006, pp. 595-602.

[19] S. E. Bae and T. Takaoka, "Improved algorithms for the K-maximum subarray problem," *Computer Journal,* vol. 49, pp. 358-374, 2006.

[20] S. E. Bae and T. Takaoka, "Ranking Cartesian Sums and K-maximum subarrays," Department of Computer Science and Software Engineering, University of Canterbury, Tech. Rep. TR-COSC 03/08, 2006.

[21] S. E. Bae and T. Takaoka, "Algorithms for K-disjoint maximum subarrays," *International Journal of Foundations of Computer Science,* vol. 18, pp. 319-339, 2007.

[22] F. Bengtsson and J. Chen, "Efficient Algorithms for k Maximum Sums," in *Algorithms and Computation*. vol. 3341, R. Fleischer and G. Trippen, Eds.: Springer Berlin Heidelberg, 2005, pp. 137-148.

[23] F. Bengtsson and J. Chen, "A note on ranking k maximum sums," Lulea University of Technology, Tech. Rep. 2005:08, 2005.

[24] F. Bengtsson and J. Chen, "Ranking k maximum sums," *Theoretical Computer Science,* vol. 377, pp. 229-237, 2007.

[25] C.-H. Cheng, K.-Y. Chen, W.-C. Tien, and K.-M. Chao, "Improved algorithms for the k maximum-sums problems," in *Algorithms and Computation*. vol. 3827, X. Deng and D.-Z. Du, Eds.: Springer Berlin Heidelberg, 2005, pp. 799-808.

[26] C.-H. Cheng, K.-Y. Chen, W.-C. Tien, and K.-M. Chao, "Improved algorithms for the k maximum-sums problems," *Theoretical Computer Science,* vol. 362, pp. 162-170, 2006.

[27] M. Bashar, "Average case analysis of algorithms for the maximum subarray problem," M.Sc. Thesis, University of Canterbury, New Zealand, 2007.

[28] W. L. Ruzzo and M. Tompa, "A Linear time algorithm for finding all maximal scoring subsequences," in *Proceedings of the 7th International Conference on Intelligent Systems for Molecular Biology*, 1999, pp. 234-241.

[29] A. P. Sprague, "Extracting optimal association rules over numeric attributes," in *Proceedings of the 36th annual Southeast regional conference*, 1998, pp. 59-63.

[30] K. Fukuda and T. Takaoka, "Analysis of air pollution (PM_{10}) and respiratory morbidity rate using *K*-maximum sub-array (2-D) algorithm," in *Proceedings of the 2007 ACM symposium on Applied computing*, Seoul, Korea, 2007, pp. 153-157.

[31] K. Fukuda, "Computer-enhanced knowledge discovery in environmental science," Ph.D. Thesis, University of Canterbury, New Zealand, 2008.

[32] K. Fukuda and T. Takaoka, "Investigation of the maximum association for suicide rate and social factors using computer algorithm," in *International Congress on Modelling and Simulation (MODSIM07)*, 2007, pp. 1381-1387.

[33] K. Fukuda, J. Brown, P. Williams, and J. Kean., "The K-maximum subarray algorithm as an alternative clustering analysis for the spatial weed aggregation pattern," presented at the 59th Annual Conference of the New Zealand Statistics Association, Hamilton, New Zealand, 2008.

[34] S. J. Weddell, T. Takaoka, T. Read, and R. Candy, "Maximum subarray algorithms for use in optical and radio astronomy," in *Proceedings of Optical Engineering & Applications: Image Reconstruction from Incomplete Data VII*, San Diego, CA, USA, 2012, pp. 8500-24.

[35] T. Takaoka, N. K. L. Pope, and K. E. Voges, "Algorithms for Data Mining," in *Business Applications and Computational Intelligence*: IGI Global, 2006, pp. 291-315.

[36] S. Weddell and B. Langford., "Hardware implementation of the maximum subarray algorithm for centroid estimation," in *Proceedings of the 21st Image and Vision Computing Conference New Zealand (IVCNZ 2006)*, 2006, pp. 511–515.

[37] S. M. Lee, "Sub-cubic time algorithm for the k-disjoint maximum subarray problem," M.Sc. Thesis, University of Canterbury, New Zealand, 2011.

[38] M. Thaher and T. Takaoka, "Improved algorithms for the K overlapping maximum convex sum problem," *Procedia Computer Science,* vol. 9, pp. 754-763, 2012.

[39] S. Anzai, J. Chun, R. Kasai, M. Korman, and T. Tokuyama, "Effect of corner information in simultaneous placement of K rectangles and tableaux," in *Proceedings of the 16th Annual International Conference on Computing And Combinatorics*, Nha Trang, Vietnam, 2010, pp. 235-243.

[40] S. E. Bae and T. Takaoka, "Improved algorithms for the K-maximum subarray problem for small K," in *Computing and Combinatorics*. vol. 3595, L. Wang, Ed.: Springer Berlin Heidelberg, 2005, pp. 621-631.

[41] Y. Han and T. Takaoka, "An $O(n^3 \log \log n / \log^2 n)$ time algorithm for all pairs shortest paths," in *Algorithm Theory – SWAT 2012*. vol. 7357, F. Fomin and P. Kaski, Eds.: Springer Berlin Heidelberg, 2012, pp. 131-141.

[42] T. Fukuda, Y. Morimoto, S. Morishita, and T. Tokuyama, "Data Mining with optimized two-dimensional association rules," *ACM Trans. Database Syst.,* vol. 26, pp. 179-213, 2001.

[43] R. Agrawal, T. Imielinski, and A. Swami, "Mining association rules between sets of items in large databases," *SIGMOD Rec.,* vol. 22, pp. 207-216, 1993.

[44] R. Agrawal and R. Srikant, "Fast algorithms for mining association rules in large databases," in *Proceedings of the 20th International Conference on Very Large Data Bases*, 1994, pp. 487-499.

[45] J. S. Park, M.-S. Chen, and P. S. Yu, "An effective hash-based algorithm for mining association rules," in *Proceedings of the 1995 ACM SIGMOD international conference on Management of data*, San Jose, California, USA, 1995, pp. 175-186.

[46] T. Fukuda, Y. Morimoto, S. Morishita, and T. Tokuyama, "Data mining using two-dimensional optimized association rules: scheme, algorithms, and visualization," in *Proceedings of the 1996 ACM SIGMOD International Conference on Management of Data*, Montreal, Quebec, Canada, 1996, pp. 13-23.

[47] T. Fukuda, Y. Morimoto, S. Morishita, and T. Tokuyama, "Constructing efficient decision trees by using optimized numeric association rules," in *Proceedings of the 22th International Conference on Very Large Data Bases*, 1996, pp. 146-155.

[48] T. Fukuda, Y. Morimoto, S. Morishita, and T. Tokuyama., "Data mining using two-dimensional optimized association rules Scheme and algorithms," Tech. Rep., IBM Tokyo Research Laboratory, 1996.

[49] T. Fukuda, Y. Morimoto, S. Morishita, and T. Tokuyama, "Mining optimized association rules for numeric attributes," presented at the Proceedings of the fifteenth ACM SIGACT-SIGMOD-SIGART symposium on Principles of database systems, Montreal, Quebec, Canada, 1996.

[50] T. Fukuda, Y. Morimoto, S. Morishita, and T. Tokuyama, "SONAR: system for optimized numeric association rules," *ACM SIGMOD Record,* vol. 25, p. 553, 1996.

[51] Y. Morimoto, T. Fukuda, S. Morishita, and T. Tokuyama, "Implementation and evaluation of decision trees with range and region splitting," *Constraints,* vol. 2, pp. 401-427, 1997.

[52] Y. Morimoto, H. Ishii, and S. Morishita, "Efficient construction of regression trees with range and region splitting," in *Proceedings of the 23rd International Conference on Very Large Data Bases*, 1997, pp. 166-175.

[53] K. Yoda, T. Fukuda, Y. Morimoto, S. Morishita, and T. Tokuyama, "Computing optimized rectilinear regions for association rules," in *Proceedings of the 3rd International Conference on Knowledge Discovery and Data Mining*, 1997, pp. 96–103.

[54] C. Apte, S. Hong, S. Prasad, and B. Rosen, "Ramp: Rules abstraction for modelling and prediction," IBM Research Division, T. J. Watson Research Center, Yorktown Heights, NY, Tech. Rep. RC-20271, 1996.

[55] B. Forsberg, N. Stjernberg, and S. Wall, "People can detect poor air quality well below guideline concentrations: a prevalence study of annoyance reactions and air pollution from traffic," *Occup Environ Med,* vol. 54, pp. 44-8, 1997.

[56] G. Touloumi, K. Katsouyanni, D. Zmirou, J. Schwartz, C. Spix, A. P. de Leon*, et al.*, "Short-term effects of ambient oxidant exposure on mortality: a combined analysis within the APHEA project. Air Pollution and Health: a European Approach," *Am J Epidemiol,* vol. 146, pp. 177-85, 1997.

[57] H. R. Anderson, S. A. Bremner, R. W. Atkinson, R. M. Harrison, and S. Walters, "Particulate matter and daily mortality and hospital admissions in the west midlands conurbation of the United Kingdom: associations with fine and coarse particles, black smoke and sulphate," *Occup Environ Med,* vol. 58, pp. 504-10, Aug 2001.

[58] R. T. Burnett, M. Smith-Doiron, D. Stieb, M. E. Raizenne, J. R. Brook, R. E. Dales*, et al.*, "Association between ozone and hospitalization for acute respiratory diseases in children less than 2 years of age," *Am J Epidemiol,* vol. 153, pp. 444-52, 2001.

[59] F. Dominici, "Invited commentary: air pollution and health-what can we learn from a hierarchical approach?," *Am J Epidemiol,* vol. 155, pp. 11-5; discussion 16, 2002.

[60] J. Q. Koenig, *Health effects of ambient air pollution: how safe is the air we breathe?* Boston: Kluwer Academic, 2000.

[61] M. Thaher and T. Takaoka, "An efficient algorithm for the maximum convex sums," *Procedia Computer Science,* vol. 1, pp. 1475-1483, 2010.

[62] M. Thaher and T. Takaoka, "An efficient algorithm for computing the K-overlapping maximum convex sum problem," *Procedia Computer Science,* vol. 4, pp. 1288-1295, 2011.

[63] M. Thaher, D. Umar, T. Takaoka, and J. Harding, "Application of the maximum convex sum algorithm in determining environmental variables that affect Nigerian highland stream benthic communities," *Procedia Computer Science,* vol. 18, pp. 909-918, 2013.

[64] S. J. Weddell, T. Read, M. Thaher, and T. Takaoka, "Maximum subarray algorithms for use in astronomical imaging," *Journal of Electronic Imaging,* vol. 22, 2013.

[65] F. Faul, E. Erdfelder, A.-G. Lang, and A. Buchner, "GPower 3: a flexible statistical power analysis program for the social, behavioral, and biomedical sciences," *Behavior research methods,* vol. 39, pp. 175-191, 2007.

[66] B. H. Cohen, J. Welkowitz, and R. B. Lea, *Introductory statistics for the behavioral sciences*. Chichester: John Wiley & Sons, 2011.

[67] G. Cumming, *Understanding The New Statistics: Effect Sizes, Confidence Intervals, and Meta-analysis*. Hoboken: Taylor & Francis, 2011.

[68] E. Huizingh, *Applied statistics with SPSS*. Thousand Oaks, Calif: SAGE, 2007.

[69] E. T. Berkman and S. P. Reise, *A conceptual guide to statistics using SPSS*. Thousand Oaks: Sage Publications, Inc, 2012.

[70] K. Black, J. Asafu-Adjaye, N. Khan, N. Perera, P. Edwards, and M. Harris, *Australasian business statistics*. Milton, Qld: John Wiley & Sons Australia, 2007.

[71] S. Nolan and T. Heinzen, *Statistics for the Behavioral Sciences*: Worth Publishers, 2011.

[72] F. E. Grubbs, "Procedures for Detecting Outlying Observations in Samples," *Technometrics,* vol. 11, pp. 1-21, 1969.

[73] J. Schwab. (2006). *Paired-Samples T-Test of Population Mean Differences* [Online]. Available: http://www.utexas.edu/courses/schwab/sw388r6_fall_2006/SolvingProblems/Homework%20Problems%20-%20Paired%20Samples%20T-Test.ppt [Accessed 1 April 2014]

[74] N. Razali and Y. Wah, "Power comparisons of Shapiro-Wilk, Kolmogorov-Smirnov, Lilliefors and Anderson-Darling tests," *Journal of Statistical Modeling and Analytics,* vol. 2, 2011.

[75] Boston University School of Public Health. (2014). *The Role of Probability: Sampling Distributions* [Online]. Available: http://sphweb.bumc.bu.edu/otlt/MPH-Modules/BS/BS704_Probability/BS704_Probability11.html [Accessed 1 April 2014]

[76] G. N. Frederickson and D. B. Johnson, "The complexity of selection and ranking in X + Y and matrices with sorted columns," *Journal of Computer and System Sciences,* vol. 24, pp. 197-208, 1982.

[77] J. M. Quinn and C. W. Hickey, "Magnitude of effects of substrate particle-size, recent flooding, and catchment development on benthic invertebrates in 88 New-Zealand rivers," *New Zealand Journal Of Marine And Freshwater Research,* vol. 24, pp. 411-427, 1990.

[78] T. Waters, "Invertebrate drift-ecology and significant to stream fishes," in *Symposium on Salmon and trout in streams, H. R. MacMillan lectures in fisheries, Vancouver. Edited by T.G. Northcote.*, University of British, Columbia, Vancouver, 1969, pp. 121-134.

[79] L. D. Corkum, "A comparative study of behaviour relating to differential drift of two species of mayflies," Ph.D. Thesis, University of Toronto, Canada, 1976.

[80] K. W. Cummins, "Factors Limiting the Microdistribution of Larvae of the Caddisflies Pycnopsyche lepida (Hagen) and Pycnopsyche guttifer (Walker) in a Michigan Stream (Trichoptera: Limnephilidae)," *Ecological Monographs,* vol. 34, pp. 271-295, 1964.

[81] K. W. Cummins and G. H. Lauff, "The influence of substrate particle size on the microdistribution of stream macrobenthos," *Hydrobiologia,* vol. 34, pp. 145-181, 1969.

[82] H. B. N. Hynes, *The ecology of running waters*. Liverpool: Liverpool University Press, 1970.

[83] W. Minshall, " Aquatic insect-sbstratum relationships.," in *The ecology of aquatic insects*, V. H. Resh and D. M. Rosenberg, Eds., New York, Praeger, 1984, pp. 358-400.

[84] J. O. Olomukoro and U. Okologume, "Macroinvertebrate colonisation of artificial substrates in a Nigerian river I: gravel and leaf litter," *African Journal of Aquatic Science,* vol. 33, pp. 143-143, 2008.

[85] M. J. Winterbourn, K. L. D. Gregson, S. New Zealand Limnological, and Z. Entomological Society of New, *Guide to the aquatic insects of New Zealand* vol. 9. Auckland, NZ: Entomological Society of New Zealand, 1989.

[86] T. J. Blakely, *An illustrated guide to the freshwater macroinvertebrates of Singapore*: University of Canterbury - School of Biological Sciences, 2010.

[87] J. Pfankuch, *Stream reach inventory and channel stability evaluation: a watershed management procedure*: U.S. Dept. of Agriculture, Forest Service, Northern Region, 1975.

[88] I. G. Jowett and J. Richardson, "Microhabitat preferences of benthic invertebrates in a New Zealand river and the development of in stream flow habitat models for Deleatidium spp," *New Zealand Journal of Marine and Freshwater Research,* vol. 24, pp. 19-30, 1990.

Appendix

Appendix (A): Real-life data collated by the *Freshwater Ecology Research Group* (*FERG*) and processed by the *Algorithm Engineering and Formal Methods Research Group* (*AEFMRG*) to study the effects of land use changes on benthic stream communities in highland tropical streams on Mambilla Plateau north-east Nigeria using the newly developed *K-DMCS* algorithm.

#	Site code	Site name	Land-use	Subtract index	temp.° C	%DO	% forest	% pasture	Taxa
1.	AG1	Canselor 2	Eucal	5.8	18.1	61.8	10	90	24
2.	AG10	Bapa Zubairu 1	Tea	7.9	20.1	61.8	5	95	8
3.	AG11	Mohamadu 1	Tea	3	23.5	63.3	2	98	13
4.	AG12	Godfree 1	Tea	3	21.2	70.6	20	80	17
5.	AG13	Mal. Ali	Banana	7.3	21.7	31.7	25	75	6
6.	AG14	Head Master	Banana	3.4	21.6	29.5	15	85	10
7.	AG16	Alh. Yusuf 1	Maiz	4	17.6	71	10	90	21
8.	AG17	Alh. Lawan	Grazing	3	15.8	64.9	10	90	11
9.	AG2	Canselor 3	Grass	5.6	18.8	54.4	10	90	19
10.	AG21	Alh. Yusuf 3	Grass	4.5	17.1	41.1	5	95	17
11.	AG22	Alh. Yusuf 2	Eucal	3	23.9	26.1	5	95	8
12.	AG23	Alh. Bayero 1	Grazing	5	14.2	74.5	3	97	8
13.	AG24	Alh. Bayero 2	Grazing	3	18.3	66.4	3	97	12
14.	AG25	Misa 1	Maiz	3.2	20	47.8	2	98	10
15.	AG26	Misa 2	Maiz	3.3	16.1	69.9	2	98	9
16.	AG27	Misa 3	Maiz	3	21.2	54.6	0	100	21
17.	AG28	Jacob 1	Cabage	3	20.6	50	0	100	7
18.	AG29	Jacob 2	Cabage	3	23	54.4	0	100	5
19.	AG3	Canselor 4	Grass	6	16.9	68.7	20	80	13
20.	AG30	Ruqaya 1	Cabage	3	17.8	68.2	0	100	14
21.	AG31	Ruqaya 2	Cabage	3	22.4	61.3	0	100	10
22.	AG32	Habiba K	Cabage	3.8	23.6	39.3	0	100	7
23.	AG33	Yelwa	Banana	4.3	21.1	38.8	0	100	3
24.	AG39	Alh. Kato 2	Grass	3.8	14.4	57.8	0	100	5
25.	AG4	Alh. Dambi 1	Grass	3	15.5	31.4	20	80	11
26.	AG5	Alh. Dambi 2	Eucal	4	15.9	40.7	5	95	3
27.	AG6	Alh. Isa 1	Banana	4.5	15.2	55.7	5	95	6
28.	AG7	Fragment. C	Grass	6.2	18.5	58.1	10	90	12
29.	AG8	Bobbo 1	Tea	3.8	17.3	74.2	10	90	9
30.	AG9	Bapa Saedu 1	Tea	3.8	20.2	30.9	20	80	11
31.	MG34	Mustapha 1	Mining	5	15.5	70.1	0	100	24
32.	MG35	Counselor 1	Mining	5	14.7	73.4	0	100	20
33.	MG36	Mustapha 2	Mining	5	17.3	66.3	0	100	10
34.	MG37	Alh. Umaru	Mining	5.6	17.1	75.6	0	100	20
35.	MG38	Jauro Saedu	Mining	5.8	18.5	57.6	0	100	19
36.	NF1	Fragment A	Maiz	7	18.5	77.1	100	0	13
37.	NF10	J.S.H 4	C/forest	3.2	17.2	58.7	100	0	12
38.	NF15	Luga Ginnaji 2	M/forest	6.4	20.8	69.2	100	0	8
39.	NF16	Luga ginnaji 3	M/forest	7.8	20.3	73.7	100	0	27
40.	NF17	Luga Ginnaji 1	M/forest	7.8	20.9	79	100	0	13
41.	NF18	Luga Ginnaji 8	M/forest	7.7	21.7	61.7	98	2	17
42.	NF19	Luga Ginnaji 9	M/forest	7.8	20.5	60.4	98	2	10
43.	NF2	Fragment B1	Grazing	5	18.78	40.5	100	0	5
44.	NF20	Luga Ginnaji 10	M/forest	7.6	22	61.7	95	5	6
45.	NF21	Luga Ginnaji 6	M/forest	6.7	20	70.1	95	5	17
46.	NF22	Luga Ginnaji 7	M/forest	7.9	20.2	73.5	95	5	19
47.	NF23	Luga Ginnaji 5	M/forest	7.5	20.4	50.6	95	5	15
48.	NF24	Luga Ginnaji 4	M/forest	7.7	19.4	67.6	96	4	14
49.	NF3	Fragment B2	Banana	5.5	22.7	35.4	100	0	6
50.	NF4	Fragment D	Eucal	7.3	19.1	15.1	100	0	12
51.	NF5	Alh. Isa 2	Eucal	4.4	24.3	55.5	100	0	7
52.	NF6	Augustine trans	C/forest	7.9	17.3	56	100	0	18
53.	NF7	J.S.H 1	C/forest	7.6	17.3	76.6	100	0	21
54.	NF8	J.S.H 2	C/forest	7.6	17.8	70.9	100	0	28
55.	NF9	J.S.H 3	C/forest	7.7	13	67.9	100	0	23