New Data Structure and Process Model for Automated Watershed Delineation

A Thesis

Submitted to the College of Graduate Studies and Research

In Partial Fulfillment for the Degree of

Master of Science

In the

Division of Environmental Engineering

University of Saskatchewan

Saskatoon, Saskatchewan, Canada

By

Naveen Mudgal

असतो मा सदगमय /
तमसो मा ज्योर्तिगमय //
मृत्यु मा अमृतंगमय /
ॐ शांति शांति शांति //

Transliteration:

Asatho Maa Sad Gamaya.
Thamaso Maa Jyothir Gamaya.
Mrithyur Maa Amritham Gamaya.
Om Shanti, Shanti, Shanti.

Translation:

From untruth lead us to Truth.
From darkness lead us to Light.
From death lead us to Immortality.
Om Peace, Peace, Peace.

PERMISSION TO USE

In presenting this thesis in partial fulfillment of the requirements for a postgraduate degree from the University of Saskatchewan, I agree that the Libraries of this University may make it freely available for inspection. I further agree that permission for copying of this thesis in any manner, in whole or in part, for scholarly purposes may be granted by the professor or professors who supervised my thesis work or, in their absence, by the Head of the Department or the Dean of the College in which my thesis work was done. It is understood that any copying or publication or use of this thesis or parts thereof for financial gain shall not be allowed without my written permission. It is also understood that due recognition shall be given to me and to the University of Saskatchewan in any scholarly use of the material from my thesis.

Requests for permission to copy or to make other uses of material in this thesis in whole or part should be addressed to:

Chair, Division of Environmental Engineering
College of Engineering, Room 2A20E
57 Campus Drive
University of Saskatchewan
Saskatoon, Saskatchewan, S7N 5A9
Canada

ABSTRACT

DEM analysis to delineate the stream network and its associated subwatersheds are the primary steps in the raster-based parameterization of watersheds. There are two widely used methods for delineating subwatersheds. One of these is the Upstream Catchment Area (UCA) method. The UCA method employs a user specified threshold value of upstream catchment area to delineate subwatersheds from the extracted network of streams. The other common technique is the nodal method. In this approach, subwatersheds are initiated at stream network nodes, where nodes occur at the upstream starting point of streams and at the point of intersection of streams in the network.

The UCA approach and the Nodal approach do not permit watershed initiation at points of specific interests. They also fail to explicitly recognize drainage features other than single channel reaches. That is, they exclude water bodies, wetlands, braided channels and other important hydrologic features.

TOPAZ (**TO**pographic **PA**rameteri**Z**ation) [Garbrecht and Martz, 1992], is a typical program for raster based, automated drainage analysis. It initiates subwatersheds at source points and at points of intersection of drainage channels. TOPAZ treats lakes as spurious depressions arising out of errors in DEM, and removes them. This research addresses one important limitation of the currently used modeling techniques and tools. It adds the capability to initiate watershed delineation at points of specific interest other than junction and source points in the delineated networks from the Digital Elevation Models (DEMs). The research project evaluates qualitative and quantitative aspects of a new Object Oriented data structure and process model for raster format data analysis in spatial hydrology. The concept of incorporating a user-specified analysis in extraction and parameterization of watersheds is based on the need to have a tool to allow for studies specific to certain points in the stream network including gauging stations. It is also based on the need for an improved delineation of hydrologic features (water bodies) in hydrologic modeling.

The research project developed an interface module for TOPAZ [Garbrecht and Martz, 1992] to offset the aforementioned disadvantages of the subwatershed delineation techniques. The research developed an internal hybrid, raster-based, Object Oriented data structure and processing model similar to that of vector data type. The new internal data structure permits further augmentation of the software tool. This internal data structure and algorithms provide an improved framework for discretization of the important hydrologic entities (water bodies) and the capability of extracting homogenous hydrological subwatersheds.

ACKNOWLEDGEMENTS

I would like to thank my supervisor, Dr. Lawrence Martz, for his unrelenting support, advice and patience. I would also take this opportunity to thank my committee members Dr. Alain Pietroniro, Dr. Dirk H. de Boer for their inputs into my project. I also thank Prof. J.A. Gillies (former Chair, Division of Environmental Engineering) for his support.

I would also like to acknowledge those who provided financial support during this project. I thank Dr. L.W. Martz for Research Assistantship, Division of Environmental Engineering and College of Graduate Studies and Research for Graduate Teaching Fellowships. I also thank Ms. Gina Kohen (Department of Computer Science) for teaching assistantships during the course of my M.Sc. program.

My special thanks to my parents, sisters Chitra and Chhaya, Brothers-in-Law Shiv Sharma and Sarvesh Vashishtha for believing in me and for their emotional support. I also thank my wife Anupama for her support during the last part of my thesis work.

I also take this opportunity to thank all friends and acquaintances who were always there to help me during the ups and downs.

TABLE OF CONTENTS

List of Tables

List of Figures

List of Abbreviations

ASCII	American Standard Code for Information Interchange. (p.7)
CASE	Computer Aided Software Engineering (p.32)
CSA	Critical Source Area (p.20)
CSS	Continuous Streamflow Simulation (p.11)
DEDNM	Digital Elevation Drainage Network Model – A TOPAZ module (p.20)
DEM	Digital Elevation Model (p.2)
EBSS	Event Based Streamflow Simulation (p.11)
ER	Entity Relationship (p.19)
GIS	Geographical Information System (p.1)
HEC	Hydrologic Engineering Centre : A Hydrologic Software (p.15)
HEC-HMS	Hydrologic Engineering Centre - Hydrologic Modeling System : A Hydrologic Software (p.15)
HILO	High and Low Point Detection- A FORTRAN Program (p.11)
IDEF	Integrated Definition (p.32)
INTER	Intermediate Output module for TOPAZ (p.38)
MSCL	Minimum Source Channel Length (p.20)
NSSTAT	Network and Subcatchment Statistics – A TOPAZ module (p.20)
PARAM	Parameterization Module- A TOPAZ module (p.20)
RASBIN	RASter to BINary- A module of TOPAZ (p.20)
RASFOR	RASter Formating- A module of TOPAZ (p.20)
RASPRO	RASter PROperties- A module of TOPAZ (p.20)
SCS	Soil Conservation Service (p.16)
TIN	Triangulated Irregular Network (p.2-3)
TOE.C	Terrain Object Extraction-in C language (p.12)
TOPAZ	TOpographic PArameteriZation -A raster-based software module(p.2)
TOPAZ-N	TOPAZ with Node Manipulation Capabilities (p.28)

UCA…………...	Upstream Catchment Area (p.1)
UML…………..	Universal Modeling Language (p.19)
USDA………....	United States Department of Agriculture (p.15)
USGS………….	United States Geological Survey (p.7)
UTM…………...	Universal Transverse Mercator (p.7)

CHAPTER 1

INTRODUCTION

1.1 Statement of Problem

The exponential growth of computer hardware and software has resulted in a rapid increase in computing power with diminishing costs. The decrease in the costs of computational power is not only in terms of the processing time (Mark 1983) but also monetary costs. The availability of inexpensive computational power has boosted the growth of digital-terrain-based analysis and modeling techniques in various engineering and science streams (Sivapalan and Kalma, 1995; Band and Moore, 1995; Wood et al., 1988).

Hydrology is a spatial science (Sivapalan and Kalma, 1995; Garbchet et al. 1996). The increase in computing power is being utilized in hydrology for fast and efficient analysis. There is an ever-increasing availability of spatial data in digital format (Sivapalan and Kalma, 1995). This has further promoted the process of computerization in hydrology (Djokic and Ye, 1999). Spatial hydrology is the combination of hydrology, Geographical Information Systems (GIS) and computer science. An important aspect of spatial hydrology is the raster-based digital terrain modeling of watersheds. In raster based digital–terrain-modeling, delineation and parameterization of the stream network and subwatersheds form the preliminary steps (Band, 1986; Jenson and Domingue, 1988; Garbrecht and Martz, 1992; Wolock and McCabe, 1995; Tarboton et al., 1991). Currently, there are two ways of arriving at the subwatershed extraction from the delineated stream network. In the first method, subwatersheds are initiated on the basis of a pre-defined, user-specified threshold value of upstream catchment area (UCA). In the second approach, which is node-based initiation, the subwatersheds are initiated at starting point, and points of intersection of channels in the delineated network. (Note- these are related as the UCA is used to initiate channels in the node-approach.)

Both the threshold UCA approach and the nodal approach suffer from various limitations. Both approaches exclude the possibility of watershed initiation at points of specific interests like gauging stations, which may or may not satisfy the conditions of subwatershed initiation of either of the approaches. They also eliminate the water bodies, an important hydrologic feature, from the analysis process.

TOPAZ (TOpographic PArameteriZation) (Garbrecht and Martz, 1992), a raster-based delineation and parameterization tool, initiates subwatersheds at source points and at points of intersection of drainage channels. TOPAZ, because of its typical raster techniques and robustness, is widely used by hydrologists. TOPAZ removes all the depressions by treating them as spurious and arising out of errors in the Digital Elevation Model (DEM). TOPAZ does not recognize lakes, but rather treats them as they are modeled in the DEM, i.e., like flat areas or depressions. TOPAZ treats these areas by adjusting the elevation values of the involved cells so as to maintain the flow continuity over these areas. TOPAZ employs the node-based approach in the delineation of subwatersheds, thus inheriting the limitations of this approach. This approach is very restrictive in analysis. It prevents inclusion of water bodies, an essential hydrologic feature, in the analysis process. It also precludes the delineation of subwatersheds around specific points of interest if it does not coincide with either of the subwatershed initiation points, (i.e., source and junction node points.)

The pure raster-data-model has a number of limitations. The raster data model lacks the ability to organize process data around feature entities. Instead, most data handling and storage is on a pixel or grid cell basis only. On the other hand, a vector data structure is purely feature based, thus making it less suitable for continuous data with spatial and temporal variability. Currently, for most hydrological modeling, raster data and processing techniques are used, with a few exceptions of Triangular Irregular Network

(TIN) based models (Wilson and Gallant, 2000; Wise, 2000; Moore et al., 1991). However, the availability of raster data sets makes it the first choice of the hydrologists.

The principal goal of this research is to add the capability of initiating watershed delineation at points of specific interest other than junction and source points in the delineated networks from the DEMs. The concept of incorporating a user-specified analysis framework in the extraction and parameterization of watersheds is based on the need to enhance and refine the point specific studies' capabilities along with an improved delineation for water bodies like lakes. For instance, hydrologists and engineers may be interested in carrying out studies for flood estimates or predictions around a gauge station which may not be at any naturally occurring junction or source. This research project would strive to initiate a new set of software tools for use in these studies. Also, the research project evaluates qualitative and quantitative aspects of a new data structure and process model for raster format data in spatial hydrology.

1.2 Research Objectives

The objectives of this research project are to develop an interface module for TOPAZ (Garbrecht and Martz, 1992) in order to incorporate the capability of initiating the subwatershed delineation at user-specified points. This will offset the aforementioned disadvantages of the two approaches of the subwatershed delineation techniques.

The research shall develop a hybrid, raster-based data structure and process model similar to the vector data model. This will involve development and evaluation of new algorithms using the object-oriented programming language Visual C++.

The project will develop a modeling tool with the capability of facilitating a comparative study of watersheds upstream of different points of interest. The study intends to fill the functional gaps between the general geography based spatial concepts and hydrology. It will identify the fundamentals of the spatial analysis in generic GIS which are incompatible with hydrology. It will also suggest remedial approaches to overcome these hurdles in spatial hydrology.

1.3 Scope of Research

The research project involves the implementation of a new module to interface with TOPAZ with all the desired characteristics and processing capabilities outlined in the objectives. The new interface will also focus on the generation of an open ended framework capable of serving as a base for any future development of decision support system in the field of raster-based-spatial-hydrology.

The Wolf Creek basin was chosen as the study site for the test runs of the software with the purpose of extracting Coal Lake, with the intent of demonstrating the application of addition and subsequent merger of subwatershed capabilities of the software.

The new data structure and algorithms will provide an improved framework for:

(1) Discretization of the watershed entities for efficient processing and information extraction.

(2) Easy storage and access of "Flow Direction" information.

(3) Rapid processing of changes within the hydrologic framework.

CHAPTER 2

LITERATURE REVIEW

2.1 General Background

The delineation and parameterization of landscape drainage features is an important area of study in geomorphology studies and hydrology (Band, 1986; Zevenbergen and Thorne, 1987; Chorowicz, Ichoku, Riazanoff, Kim and Cervelle, 1992). The delineation of channel networks involves their representation by a set of measurable physical properties. These properties are then used in various models to extract other characteristics, and also to ascertain the response of the drainage system to other variables.

The process of parameterization involves the demarcation of the drainage network on a base map. Once the drainage network is laid out on the map, parameters like channel lengths and angular measurements are measured (Garbrecht and Martz, 1992). The parameters being extracted depend on the model for which these parameters are determined and on the method of extraction. The process of acquiring these parameters can be either manual or by the use of Geographical Information System (GIS). These methods are very tedious, and require intensive labour (Band, 1986). Due to intensive human involvement and the tedious nature of the work, these methods are prone to errors. Moreover, the GIS-based techniques have limited hydrologic scope, as most of the GIS based techniques are developed for general purposes. The next few sections of this chapter will examine field based manual methods and automated methods of extracting these parameters.

2.2 Field Based Manual and Semi-Manual Methods

The blue line stream network that is seen on topographic maps is a “best-estimate” of the actual drainage network. The blue-line stream network only comprises permanent and major channels. The minor, seasonal channels are usually omitted from the blue line network (Coates, 1958; Morisawa, 1959; Coffman et al. 1972; Montgomery and Foufoula-Georgiou, 1993).

The field based manual method of extracting the physiographic parameters starts with the outline of the major channel sections of the drainage network. Traditionally, this outline was drawn by land surveying techniques, which are time consuming and labour intensive. Despite high accuracy of field measurements, this method is prone to errors and lack of precision due to unavoidable human and technical errors and precision limitations.

Land surveying techniques can be replaced or supplemented by photogrammetry. The aerial photographs can be used to establish the outline network. This, however, requires human intervention by experts, and is also dependent upon the limits of what can be seen on photos. Still, the land surveying techniques are needed, as aerial photographs only assist in the identification of major channels. Thus, with the combination of intensive landscape profile (close range contour map) and major drainage channels, the other seasonal and minor channel links are identified and marked. Once the required level of detail is obtained in the network, the next process involves measuring the basic parameters like link lengths, angular measurements etc. of the drainage networks and its watershed.

The process of parameter extraction can be done either manually or by computer with the use of a co-ordinate digitizer (Javaris, 1977). A co-ordinate digitizer can be employed to generate digital data to carry out the further processing, measurements, and calculations by computers. Co-ordinate digitizers are special machines which are employed to convert conventional paper-based data into digital data. Despite the availability of high precision co-ordinate digitizers, their use still introduces many errors in the data due to the involvement of intensive, tedious human labour in controlling these machines. However, with data converted into digital format the extraction process becomes faster and easier. Many parameters that are difficult to measure and calculate by pure manual methods can be determined with reasonable accuracy and precision after digitization.

2.3 Digital Elevation Model (DEM): An overview

Digital Elevation Models (DEMs) are a raster-geographic-digital-data-model developed by the US Geological Survey (USGS) (Kessler, 1992). It is an ordered array of the elevation values of an area sampled at regular horizontal intervals (Doyle, 1978).

According to the United States' Geological Survey (USGS):

> *"Digital elevation model (DEM) data consist of a sampled array of regularly spaced elevation values referenced horizontally either to a Universal Transverse Mercator (UTM) projection or to a geographic coordinate system. The grid cells are spaced at regular intervals along south to north profiles that are ordered from west to east."*
>
> (http://interactive2.er.usgs.gov/faq/get_answer.asp?id=367)

In its simplest form, a DEM can be in a two dimensional American Standard Code for Information Interchange (ASCII) or binary format. It is represented as a flat file with a two-dimensional computer array data structure holding elevation data. The one-dimensional ASCII or binary format file has a header, which usually specifies the lower left-hand corner, the cell size, the number of cells in both the X and Y directions and the orientation of the grid. It also lists the null values followed by the elevation values listed in either one single row or column.

DEM are divided into two basic types based on their procedure of production, namely, Level-1 and Level-2 DEM (Garbrecht and Starks; 1993). Level-1 DEM are those which are produced from manual profiling techniques using photogrammetric stereomodels or by Gestalt Photo Mapper-II techniques. Level-2 DEM are produced using modern techniques of Digital Line Graphs (DLG). The Level-2 DEM are more accurate than Level-1 DEM, and are also checked against the presence of systematic errors and hence tend to have fewer systematic errors.

2.3.1 DEM: Structure and Suitability and Availability

Structurally, DEM are divided into two broad categories based on the data structure employed to represent the elevation. These categories are:

1. Irregular Grid
2. Regular Grid Structure

In the regular grid structure, the grid comprises regular geometrical shapes to cover the area, whereas in irregular grid, delauney irregular triangles are used to create surface (Lee and Schacter, 1980) (Watson, 1981). Theses irregular triangle based grid are referred to as Triangulated Irregular Network (TIN). The triangles in a TIN format have a 'facet' created by the three different elevation values of its vertices (Wilson and Gallant, 2000). Contour based grid are another irregular grid structure format which is formed by the laying of flow lines over the contours (Moore et al., 1991 and Maunder, 1999).The irregular grid format is more suitable for complex and highly variable landscape (Wise, 2000). For terrain modeling purposes TIN is ideal as it generates a surface profile very close to the actual one (Moore et. al., 1991). TIN was used by Jones, Wright and Maidment (1990) in watershed delineation. The anisotropic nature of TIN structure makes it less suitable for use in hydrology. For simulation of flow of water on a terrain, a contour based format is more suitable (Moore and Grayson, 1991). The regular grid structures are described in subsequent subsections of this chapter.

2.3.2 DEM Accuracy: Sources and Types of Errors

In any DEM based analysis system, the DEM's horizontal resolution as well as its vertical resolution and accuracy is of utmost importance (Gyasi-Agyei, Willgoose and Troch, 1995). Contrary to general belief, the DEM's are not free from errors (Saunders, 1999). DEMs have a number of error sources depending upon their method of creation (Blackmore, 1984).

The vertical positional errors in DEM are classified as blunder, systematic and random errors. Blunder is a vertical error identified by its pronounced scale. Blunder errors are caused due to different surveying techniques of data collection (Bie and Beckett, 1973; Salome et al. 1982). The common causes are incorrect contour reading, numerical errors during transposing, and correlation errors.

Systematic errors are characterized by the presence of a fixed pattern or rule. These error patterns have consistent magnitude and/or repeat occurrences. The systematic errors are caused by bias in processing. Vertical elevation shift is an example of a systematic error in DEMS. The systematic errors often are results of incorrect interpretation of the elevation values due to building, tree shadows, or numerical errors in computer processing like rounding off (Gruenberger, 1984; Franklin 1984). Random errors are characterized by the lack of any prominent character. They are caused by unknown reasons often best described as 'accidental' causes. They are characterized by their random occurrences.

2.4 Hydrologic Modeling

A model is a simplification of a complex real-time process, and is an abstraction of processes and environments. The models serve to depict, to a maximum semblance, a concept or a theory. The models are used to predict, explain, and simulate a process/situation. A model is best described as a mathematical representation of a simplified form of a complex real-time process. Mathematical models have been defined as:

> *"….. a simplified representation of a complex system in which the behavior of the system is represented by a set of equations, perhaps together with logical statements, expressing relations between variables and parameters"*
>
> *(Clarke, 1973, p.1)*

or

> *"………… a numerical system inter-relating in a given time reference a sample of input, cause or stimulus of matter, energy or information and a sample of output, effect or response of information, energy or matter "*
>
> *(Flemings, 1975, p.18)*

Hydrologic modeling is an attempt to explain and simulate the responses to precipitation. These models not only predict the run-off response but also other effects. Models are classified into various categories based on their level of spatial variability and degree of precision. Models are classified as lumped models and distributed models based on their grouping, degree of spatial variability, and complexity. Distributed models tend to be spatially extensive with intensive data requirements. Lumped models, on the other hand, are less spatially variable and require less data. Models can also be classified based on the level of precision and accuracy of their results.

In reality, models do not completely fall into any one category. Also, within a category they may fall in between two different sub-types. For instance, a model that falls in between the lumped and distributed sub-types is called a 'semi-distributed' model. A 'semi-distributed' model requires less data than a distributed model and is more precise model than a lumped model. The 'type' of model is a function of the nature of investigation. Availability of data is another factor that has a bearing on the model type that can be employed for a particular study. Just like overlapping of lumped and distributed models give rise to 'semi-distributed' models, similarly overlapping of stochastic and deterministic models gives rise to an intermediate models referred to as 'parametric' model.

No one type of model can fit into the requirements of all studies. A good model is characterized by its level of universality. A good model will have a wide range of adaptability and applicability. In the words of Wood and O'Connell (1985, p.556):

> *"……there is no 'best' model for hydrological forecasting. Different types of models are required to fulfill different roles and objectives."*

Apart from space and randomness, time is another component that is crucial in some hydrologic studies. Incorporation of all these factors (spatial variability, randomness and temporal variability) in one single model is not possible, despite powerful computational capabilities. Based on the temporal variability there are two main types of model, namely the ‘Event Based Stream flow Simulation’ (EBSS) model and the “Continuous Stream flow Simulation’ (CSS) model (Singh, 1988, 1989, 1995).

2.4.1 DEM and Watershed Modeling

Availability of DEM data introduced a new era of automated techniques to extract hydrologic, geomorphic, and other terrain properties. The initial work in automated extraction of stream network was based on the topology alone. Puecker and Douglas (1975) were amongst the pioneers to use DEM to identify potential streams and ridges. In their approach for identifying streams and ridges, Puecker and Douglas employed the ‘upward concave’ and ‘convex’ area concept in relation to flow. Based on this algorithm, a FORTAN program called ‘HILO’ (High and Low Point Detection) was developed by Kikuchi, et al. (1982).

Hydrologically, the stream network is characterized by dominance of fluvial processes over slope processes indicating threshold runoff quantity, thereby identifying a stream network. Speight (1968) manually applied this hydrologic approach to extract drainage networks. Based on Speight’s approach, O’Callaghan and Mark (1984) developed an algorithm for automated extraction of drainage network. The program, based on their algorithm, was very time consuming due to several repetitive passes/evaluations of DEM cells. The program was found suitable only for small sized DEMs. Also, closed depressions were a major obstacle in extracting drainage networks.

Jenson (1984) developed an algorithm which employed a three by three (3x3) matrix with the cell being investigated at the centre of the matrix. The algorithm identified the local minimum by doing a pass of this 3x3 window for all DEM cells, which was much less time consuming than Mark and O’Callaghan’s (1984) as it just needed a single pass in the identification process. The same 3x3 matrix pass is then used on the ‘identified’ cells, beginning from the lowest elevation as a root. Its unmarked drainage neighbor is assigned the same identification number and the matrix then shifts to this cell and the process is repeated until no linked drainage cell is found. The entire process is done for every drainage cell with no identifying label assigned. In the final stage, all cells are given the same identification number as that of the drainage link into which they drain. Finally, based on incremental tolerance of elevation and distance between the roots, reassigning of the identification number is done to non-drainage cells to generate larger basins from these mini-basins. Depression filling algorithms were presented by Mark et al. (1984) and Jenson and Trautwein (1987) as a preprocessing for the DEM. TOE.C (Terrain Object Extraction), a C language program, was one of the first attempts at extracting features as objects from a DEM. Developed by Lammers and Band (1990), TOE.C is based on generating a database of spatial and functional relationships of basic geomorphic features identified as objects. It follows the rules and conventions of a formal geomorphic model (Band, 1989).

In 1990 and 1991 work on a new way to extract drainage network was published by Tribe. Tribe introduced the ‘threshold’ slope value to ascertain the steepness of valley sides. Her algorithm also included cells other than the adjacent eight neighbors in evaluating the profile of a cell. The treatment of flat surface was different than non-flat surface for which she used the Jenson and Domingue (1988) method. Tribe’s method treated closed depressions as ‘spurious’ and filled them to generate flat surface and then assigned them flow directions. For flat areas, the flow directions are assigned based on an interpolated straight line, thereby implementing a flow pattern to drain into the interpolated channel link. Tribe’s method introduces new problems and increases the problem of channel straightening and causes a straight cut flow path across high elevations (Martz and Garbrecht, 1995). Tribe’s method produces unrealistic, straight cut stream channel sections across high elevation areas (Martz and Garbrecht, 1994). Furthermore Tribe’s approach, instead of solving the problem of unrealistic straightening of channel links, increases it. Tribe’s method also suffers from spatial variability in landscape characteristic, as pointed out by Martz and Garbrecht (1994).

Profile Scan, based on the principle of differential geometry (Papo and Gelbman, 1984) and the mathematical model for hydrologic flow developed by Riazanoff et al. (1990), were employed by Chorowicz et al. (1992) in their algorithm for automated drainage network. The algorithm was focused solely on linear features.

2.4.2 Lumped Models

Lumped models do not take into consideration spatial variability. Lumped Models, sometimes also referred to as the ‘Black Box’ models, are described by a chosen set of adjustable (fitted) parameters. It is the most simplified watershed model based on spatial averaging (grouping) (Kite, 1994, p.191). Lumped modeling techniques group together various individual process threads into one. Lumped models assume that the watershed is one homogenous unit subject to uniform processes (Linsley et al. 1986, p.340). The parameters are averaged, ignoring their spatial variability over the entire watershed unit (Flemming, 1979, p. 8-9). The lumped models are based on large generalization between input and output relationships which are arrived at from a representative range of basins. These models are ‘macroscopic’ in nature, as they relate various parameters to the watershed as one single unit. In a lumped watershed model the mathematical relationships only comprise of temporal factors and depth (Blackie and Eeles, 1985, p.313).

Lumped models are best suited for small watersheds which have little spatial variability and can be considered ‘homogeneous’ for the purpose of modeling (Blackie and Eeles, 1985, p. 313). For instance, in a small watershed, the variability of parameters like soil type, soil moisture, vegetation type, and precipitation is very low. Due to this low variability, these can safely be averaged over the entire watershed without adversely affecting the outcome. This does not hold for larger basins which have inherent variability in all the parameters. However, there have been successful cases of lumped model application to large heterogeneous watersheds (Blackie and Eeles, 1985:313).

2.4.3 Semi-Distributed to Distributed Models

Distributed models sub-divide the drainage basin into elementary units. The physically based distributed model is one such model. It is based on the physics governing the hydrological process for each elementary unit to generate channel flow in a watershed (Beven, 1985, p. 405). The fully distributed model was arrived at by improvement over the semi-distributed model. Nash (1957) implemented a series of reservoirs in conjunction with the lumped model. In this model, a series of 'lumped' watersheds were linked to arrive at a bigger watershed. Dooge (1959) improved it with flow routing giving rise to a 'cascading linear reservoirs' type of model. This was followed by a 'multiple linear elements' model (Eagleson) and then 'multiple non linear storages'. In a distributed model, the watershed is sub-divided into a finite difference grid mesh or a finite element mesh. The finite element mesh could be a regular element mesh or it can be an irregular element mesh. Each element in this model is simulated separately and then the results for each element are compounded to arrive at the result for the entire watershed response (Fleming, 1979). Due to spatial intensity of this model, the data requirements for it are very large and the lack of data availability does not result in much precision and accuracy and is close to lumped model results (Linsley et al., 1986). Apart from requiring time-consuming computational processes, the distributed models also require spatially distributed parameters and variables. The distributed models suffer from the lack of availability of precise data sets to simulate watershed response; this is partly because such intensive and extensive data collection has high costs associated with them.

2.4.4 Deterministic Models

A deterministic model is characterized by its replicable nature. Under certain environmental conditions, a deterministic model will produce the same output for a given input (Kirkby et al., 1993, p. 3). These models are closely linked to distributed models. Deterministic models are complex and precise compared to stochastic models, and are based on complete knowledge of the involved variables (Kirkby et al., 1993:14). In real world situations it is impossible to achieve a completely deterministic model because of the complexity created by the spatial variability of the properties and the processes. Furthermore, it is not possible to map out all the relationships between the variables in a system. According to Becker and Serban (1990), hydrological models comprise of both deterministic and stochastic elements. Deterministic models are physically based models governed by fundamental, scientific laws. Deterministic models are also known as 'White Box' models.

2.4.5 Parametric Models

Parametric models are the models which lay midway between the stochastic models and the deterministic models. As per Kirkby et al., (1993, p. 14):

> *"…..it must be stressed that this classification is somewhat arbitrary: all models are really a shade of 'grey' since even the most realistic models involve some simplification of the real world."*

Parametric models are characterized by the inclusion of temporal components in addition to spatial components (Fleming, 1979). The parameters chosen in a parametric model are based on the fundamentals of the water movement, but their values are derived by methods which are either statistical or empirical in nature instead of direct deterministic measurements.

2.4.6 Stream Flow Simulation Models

Stream Flow Simulation Models are classified into two distinct categories, namely:

1. Event based streamflow simulation models (EBSS Models).
2. Continuous streamflow simulation models (CSS Models).

EBSS models are simple as compared to the Continuous flow models because the latter represents more physical processes. Both models suffer from the lack of mass closure for the complete hydrologic cycle. HEC (1990, 1995) Software (Hydrologic Engineering Center), HEC-HMS (1995, 2001) (HEC-Hydrologic Modeling System) and USDA Soil

Conservation Service's (SCS) TR-20 are widely used computer modeling programs that employ EBSS modeling. USGS PRMS/Stanford, NWS/Sacramento, HEC-Continuous Flow Simulation are a few of the computer modeling programs that are based on the Continuous Flow Simulation model. Whereas EBSS models are suitable for flood control, real time hydrologic problems, continuous flow simulation is better suited for planning of the water resources and to study the effects of climate change on the flow of streams.

2.5 Spatial Hydrology

Spatial hydrology is the union of GIS and hydrology. GIS describes the geographical and spatial variations of a hydrologic system. Surface hydrology, on the other hand, deals with the effect of the variations of the surface profile on the flow of water over the terrain. A hydrologist is interested in modeling the overland flow of precipitation to develop effective flood control systems, water supply systems, and water resources management systems. Spatial hydrology is the branch that develops and employs the GIS based tools for the purpose of working out these hydrologic systems.

Terrain characteristics are spatially intensive and so is their effect on the overland flow of water. This spatially intensive characteristic facilitates the use and development of GIS based automated techniques for surface water modeling. Spatial hydrology serves as a link between the spatially varying terrain and the surface water flow. It has two functions:

1) To generate / develop spatial data sets.
2) Support hydrologic modeling.

2.6 Primary Spatial Data Models and Structures

An ideal, GIS based hydrology is characterized by its ability to take into account the locational factor in its manipulation and processing of data to extract information and conduct information search. The efficacies of these processes are based on three factors:

1) Data Models,
2) Data Structure and Algorithms and,
3) Database management techniques.

Geographically referenced data is spatially referenced. In the most basic format, spatially referenced data can be modeled by two different spatial addressing techniques. In the first technique, the graphical entities which represent the geographic features form the fundamental building blocks. These entity objects are spatially indexed. Each entity has ‘location’ as one of its attributes. In the second technique, it is the location which is the fundamental building block of the model. In this technique, every location has a set of attributes associated with it. The first method resulted in development of a ‘Vector’ data model whereas the second one in the development of ‘Raster’ data model.

2.7. Vector data model

The fundamental units in vector data models are the graphical geographic entities. These graphical entities are defined by a pair of co-ordinate values. A line is defined by a join between two set of co-ordinate values, one representing the beginning of the line and the other representing the end of the line. Three or more closed lines form an area or a polygon. Within this model there are variations. The models are further classified as unlinked models and topological models.

2.7.1 Unlinked Vector data model

Unlinked vector models are the simplest form of vector data model in which a map is abstracted by a set of separately encoded entities. These entities are not referenced with their neighbors. There is no concept of neighbors, hence there does not exist any spatial relationships in this model. This model lacks continuity of structure, and hence is very expensive for spatial analysis. It does have an advantage of data compression due to chain length encoding technique (Freeman, 1974).

2.7.2 Topological Vector data model

In the Topological vector model, entities are referenced with their neighboring entities. This model has structural continuity, permitting better analysis, like optimization of networks, finding the shortest path and flow path analysis. This model allows for the definition of complex spatial relationships.

2.8 Raster data model

The Raster data model is a tessellation of a plane. In this model, the entire space is partitioned into small cells. Every cell representing a unique space has a set of properties associated with it. Raster data models have implicit spatial relationships. The continuity is also an implicit property of this data model. In such models, there are no separate graphical/ geographical entities. Raster can be based on different geometric shapes for tessellation. The best tessellation shapes should have the basic quality of producing an infinite, repetitive pattern which can completely cover the plane being tessellated. They should also have the property of infinite recursive decomposition into similar smaller units. Further rater data models can be of regular or semi-regular tessellation types. Squares, triangles, and hexagons are the three regular tessellation logical units.

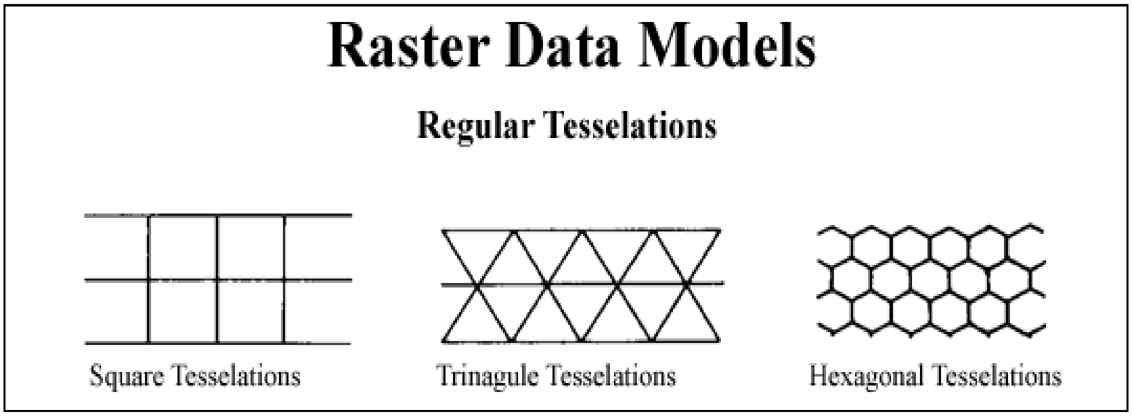


Figure 2.1: Raster Data Models: Regular Tessellations-Square, Triangle, Honeycombed

In a regular tessellation, the same number of sides and the same number of cells meet at any vertex. Each geometrical figure has its inherent draw backs. Hexagons cannot be further decomposed while offering uniformity in adjacency and orientation. Only squares and triangles are capable of infinite decomposition into similar types. Squares are best suited for hierarchical models. Hierarchical models can have various data structure configurations. Quad tree and pyramid are the two principal data structures used in hierarchical models.

2.9 Spatial Data and Structure Modeling

Spatial data is different from most of the other types of data such as business or medical data. Spatial data is characterized by the presence of a unique ‘location’ attribute. Not only just the location variability, but within the same domain of location there is an element of temporal variability for a given dataset. These unique properties add two more dimensions to the data. Unlike most data which are flat in structure, the spatial data is three-dimensional. It has a temporal dimension, a location dimension and then the attributes.

To model a data structure for spatial dataset, two distinctly different approaches are used. The models used are:

1) ER Model (Entity Relationship Model).
2) UML (Universal Modeling Language).

The ER model is more suitable to conventional data. However, it has been successfully employed in the vector based spatial data domain. The onset of the Object-Oriented and Object-Based approach to spatial data modeling has also lead to using the UML. It is important to mention here that both the ER and UML approach are employed for the vector data model.

2.10 TOPAZ: An overview of the framework

TOPAZ is a raster based spatial hydrology tool. TOPAZ processes Digital Elevation Models (DEM) and delineates drainage network, extracts watersheds, subdivides watersheds into subwatersheds, and extracts their hydrologically and topologically important parameters. TOPAZ is a suite of six software programs. The constituent software modules of TOPAZ are:

1. DEDNM Module (**D**igital **E**levation **D**rainage **N**etwork **M**odel Module)
2. RASPRO Module (**RAS**ter **PRO**perties Module)
3. RASBIN Module (**RAS**ter to **BI**nary **N**etwork Module)
4. NSSTAT Module (**N**etwork and **S**ubcatchment **STAT**istics Module)
5. PARAM Module (**PARAM**eterization Module)
6. RASFOR Module (**RAS**ter **FOR**matting Module)

These six modules are classified into three categories, based on their inter-dependency and execution sequence. These categories are Primary, Secondary, and Tertiary Module. Primary modules run independent of any other module of the TOPAZ family whereas secondary modules can run using some output from the primary modules. Tertiary modules require inputs from primary and secondary modules' outputs. The DEDNM module is independent of the rest of the other five modules of the TOPAZ family. The DEDNM module is the very first module to be executed. It requires two files to commence processing and subsequent analysis. One file is the DEM data file and the other file is the one containing processing options selected by the user. DEDNM generates 'input' data files for other modules. DEDNM is thus classified in the Primary Level of the TOPAZ Suite. DEDNM delineates the preliminary channel network and subwatersheds based on two user-provided network parameters: the critical source area (CSA) and the minimum source channel length (MSCL). The CSA parameter defines the drainage density. An increase in CSA value results in the decrease of the drainage density in generated network. On the other hand MSCL the parameter defines the minimum permissible channel length. Thus an increase in MSCL value results in removal of short source channels (1st order channels). The CSA and MSCl values which are provided by the user can be estimate from maps or field surveys. Their values are selected in accordance with the scale and resolution of the particular application under consideration.

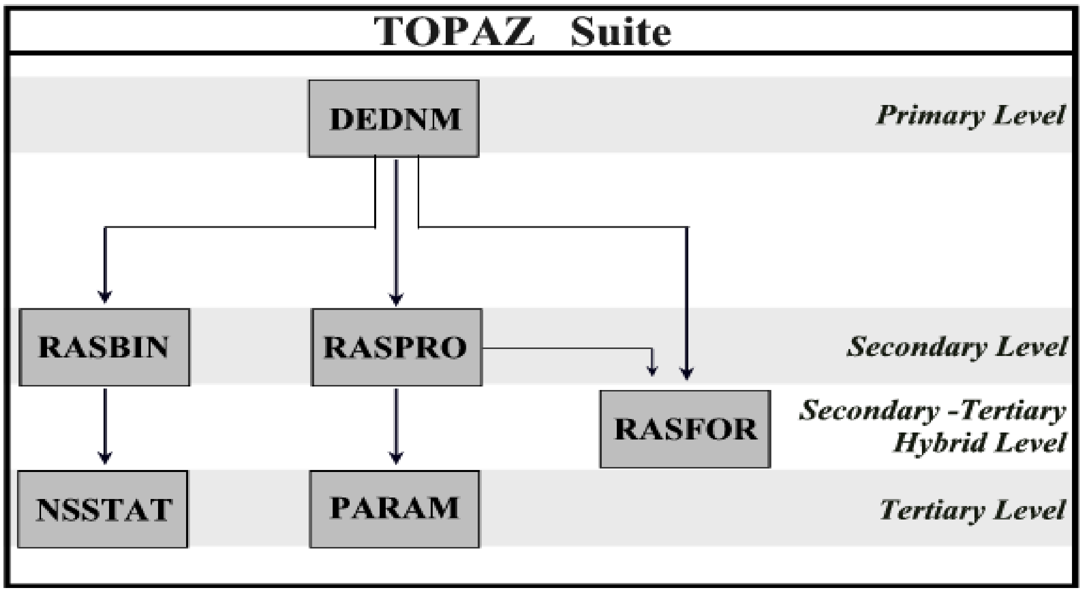
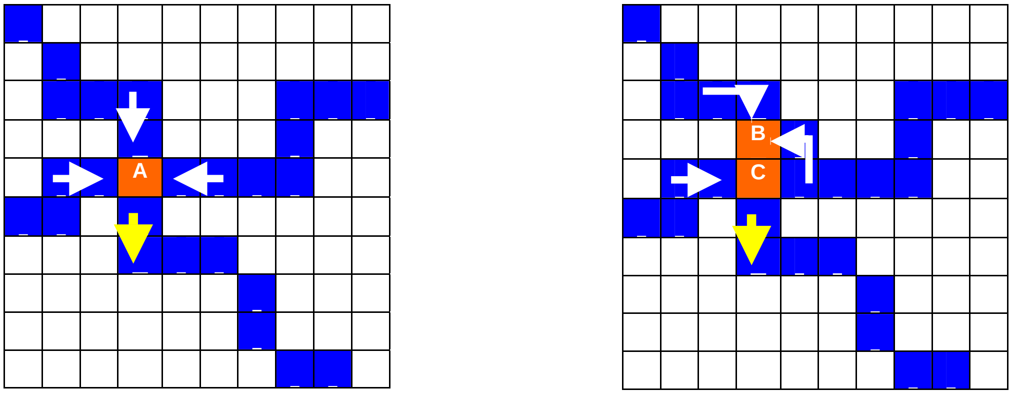


Figure 2.2: TOPAZ Suite. TOPAZ has six modules which are differentiated into three hierarchical categories based on the inter dependency. Hence the execution sequence.

(After: Martz and Garbrecht, 1998)

The RASPRO and the RASBIN constitute the Secondary Level of TOPAZ. Both these modules can only be run if the DEDNM module has already been executed. These modules are dependent on DEDNM for their input data. RASPRO generates a set of spatially intensive physical characteristics of the terrain which are of hydrological importance e.g. slope, aspect etc.

RASBIN, on the other hand, enlists the channel and subcatchment properties for the binary network. RASBIN converts the raster stream network into a binary network. Raster networks can have more than two inflows at any junction, but TOPAZ only allows two inflows per node. A node with more than two inflows is defined as a ‘complex’ node. RASBIN resolves the complex node to arrive at a binary network. Complex nodes arise due to the coarse horizontal resolution of the DEM. RASBIN decomposes the complex junction nodes into a set of simple junction nodes with only two inflows per junction node.



(a) (b)

Figure 2.3 (a) Depiction of a Complex node named ‘A’ in the network with three inflows, (b) RASBIN decomposes the Complex node ‘A’ into two simple nodes- ‘B’ and ‘C’ with only two inflows each.

NSSTAT and PARAM form the tertiary level of TOPAZ. These modules can be executed once the modules in the Primary and Secondary levels of TOPAZ have been run. NSSTAT produces statistical information for each of the channel segments and subcatchment, for both the raster and the binary channel network.

The PARAM (PARAMeterization) module generates the properties for the subwatersheds which are created by the primary level modules, namely DEDNM and RASPRO. PARAM calculates mean flow path length, the total drainage area and various slope alternatives. RASFOR is a secondary-tertiary hybrid level. It can be run immediately at the secondary level, immediately after the DEDNM execution. It can also be run at the tertiary level after the execution of RASPRO. It is so because RASFOR can use ‘output’ data of either DEDNM or RASPRO as its ‘input’ data. As the name suggests, RASFOR is a raster formatting module. It can be used to generate outputs in formats which are compatible with Geographical Information Systems (GIS) and other software programs.

2.11. DEDNM-An Analytical Overview

2.11.1 DEDNM-Breaching of Closed Depressions

DEDNM pre-processes the DEM to rectify any value or set of values in the DEM which causes formation of closed depressions. Closed depressions prevent the continuity of flow which is required in channel delineation. These closed areas serve as a sinks. Conventionally, these sinks are treated as spurious due to errors in elevation values in the DEM. It is assumed that these are caused by the error/underestimation of the elevation value for a grid cell or sets of grid cells. Based on this approach, these depressions are corrected by a simulated filling with a local outlet. Closed depressions, however, are caused not only by the underestimation of the elevation values, but also by overestimation.

However, it is not possible to identify the cause of closed depressions in a DEM. DEDNM's approach to filling these spurious sinks is more realistic. Apart from considering underestimation as a cause of depression, the DEDNM also accommodates the overestimation factor against the conventional method which completely ignores overestimation factor. In DEDNM the process of breaching is restricted to approximately two grid cells. The whole premise is that lowering the elevation value for any of the peripheral local outlets for the depression could shrink the depression. DEDNM's breaching process is a three-step procedure. The very first step is the determination of the size of the depression. This is followed by locating and breaching the lowest outflow from the depression. Finally, the remaining grid cells of the depression are filled to the elevation level of the outflow.

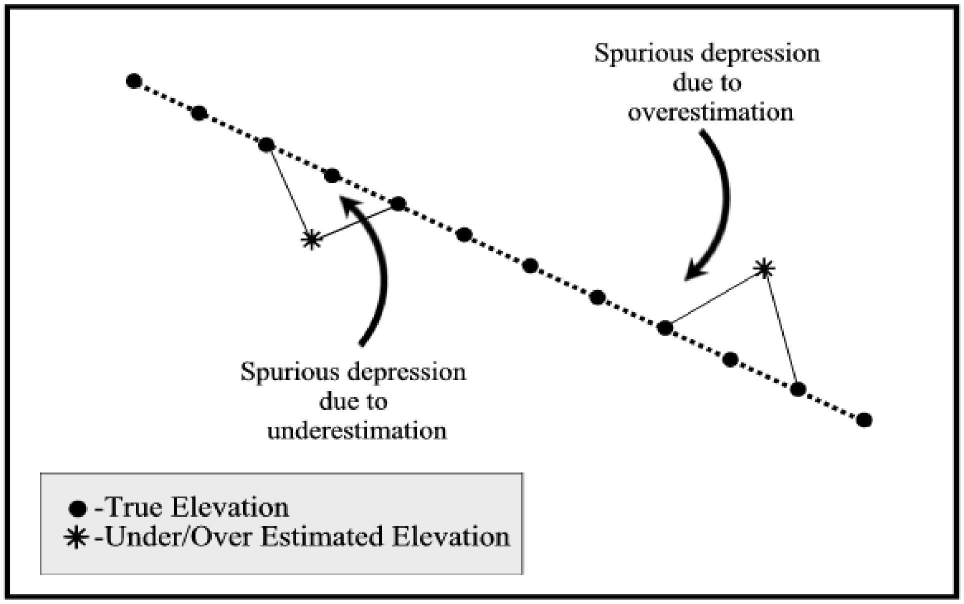
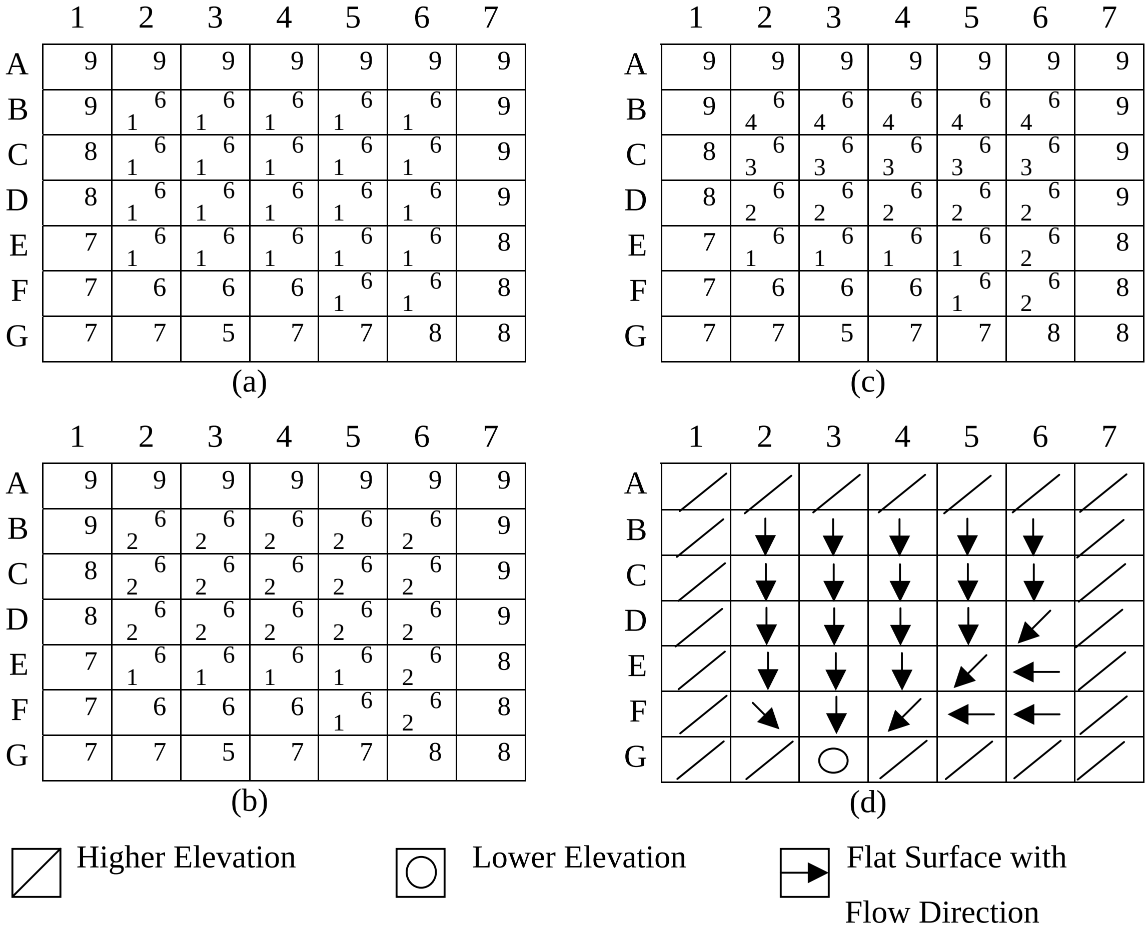

Spurious depression due to overestimation
Spurious depression due to underestimation
● -True Elevation
✱ -Under/Over Estimated Elevation

Figure 2.4: Two dimensional illustration of spurious depressions along valley bottom (After: Martz and Garbrecht, 1998)

2.11.2 DEDNM-Treatment of Flat Areas to obtain Flow Vectors

Flat surfaces pose a problem in drainage network extraction due to the inability to determine the flow direction across a flat area (Speight, 1974; Tribe 1992). To overcome this problem of flow routing on flat surfaces in the DEM, the DEDNM employs an approach based on the law of physics that the movement of water is away from higher areas and is directed towards lower elevations. This law is always true for any landscape in nature. DEDNM achieves flow path over a flat surface in broad steps. In the first step the elevation is increased for all the cells which are adjacent to cells already not having a down-slope gradient. This process is repeated recursively until an overall gradient in the direction of the low lying area is achieved.



	1	2	3	4	5	6	7
A	9	9	9	9	9	9	9
B	9	6 1	6 1	6 1	6 1	6 1	9
C	8	6 1	6 1	6 1	6 1	6 1	9
D	8	6 1	6 1	6 1	6 1	6 1	9
E	7	6 1	6 1	6 1	6 1	6 1	8
F	7	6	6	6	6 1	6 1	8
G	7	7	5	7	7	8	8

(a)

	1	2	3	4	5	6	7
A	9	9	9	9	9	9	9
B	9	6 4	6 4	6 4	6 4	6 4	9
C	8	6 3	6 3	6 3	6 3	6 3	9
D	8	6 2	6 2	6 2	6 2	6 2	9
E	7	6 1	6 1	6 1	6 1	6 2	8
F	7	6	6	6	6 1	6 2	8
G	7	7	5	7	7	8	8

(c)

	1	2	3	4	5	6	7
A	9	9	9	9	9	9	9
B	9	6 2	6 2	6 2	6 2	6 2	9
C	8	6 2	6 2	6 2	6 2	6 2	9
D	8	6 2	6 2	6 2	6 2	6 2	9
E	7	6 1	6 1	6 1	6 1	6 2	8
F	7	6	6	6	6 1	6 2	8
G	7	7	5	7	7	8	8

(b)

	1	2	3	4	5	6	7
A	╱	╱	╱	╱	╱	╱	╱
B	╱	↓	↓	↓	↓	↓	╱
C	╱	↓	↓	↓	↓	↓	╱
D	╱	↓	↓	↓	↓	↙	╱
E	╱	↓	↓	↓	↙	←	╱
F	╱	↘	↓	↙	←	←	╱
G	╱	╱	○	╱	╱	╱	╱

(d)

╱ Higher Elevation

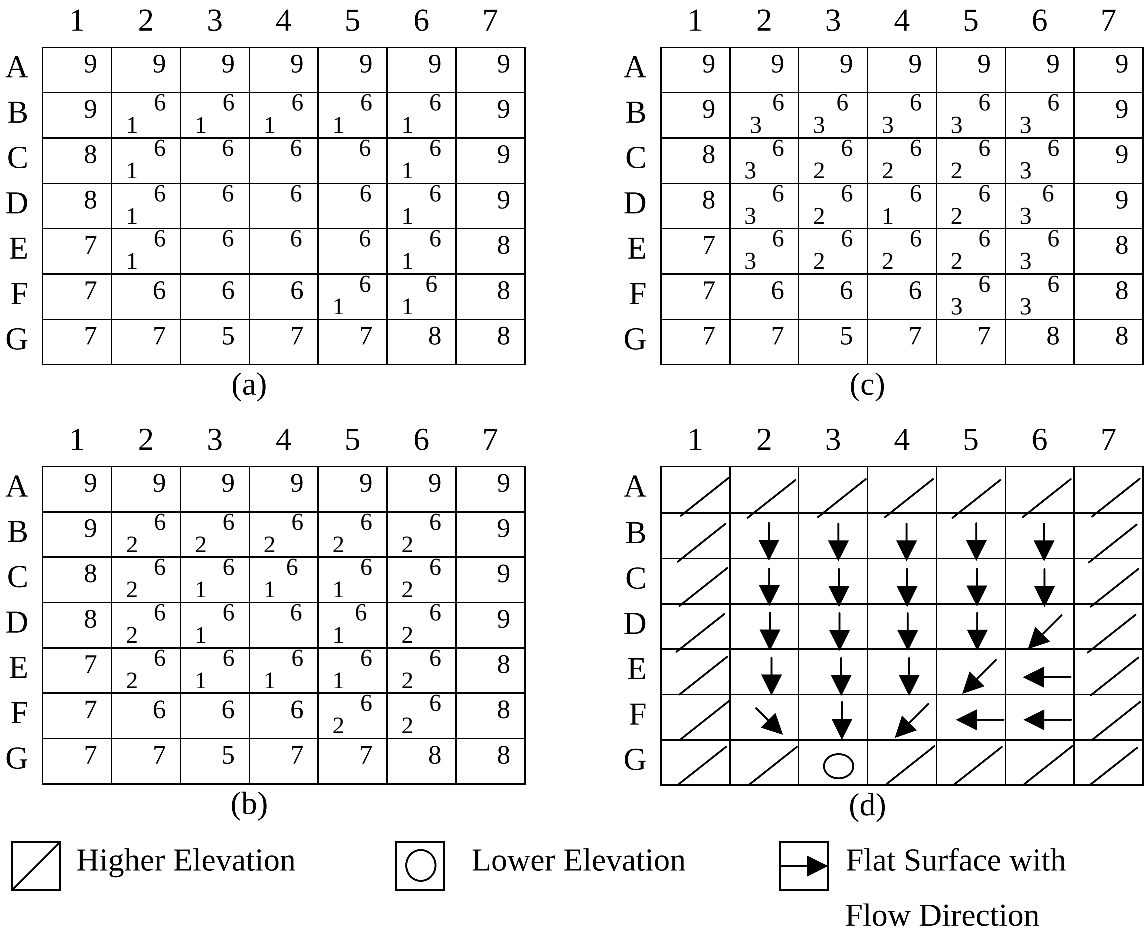
○ Lower Elevation

→ Flat Surface with Flow Direction

Figure 2.5: (a) Lower left hand number is the first pass of the depression incrementing elevation of cells neighbor to one not having any down slope (b) and (c) depict successive passes there by achieving the resultant flow directions as shown in (d).
(Source: Garbrecht and Martz, 1997)

The amount of increments is 2/100 000 of the vertical resolution of the DEM in-order to prevent ridge formation at the boundaries. This process of the backward growth of the gradient from potential outlets for the flat area generates parallel flow vector directions ultimately leading to the outlets Fig 2.5 (d).

In the second step, the goal is to achieve a gradient in a direction away from higher elevation areas. Thus, the elevation increment is applied to the cells which are adjacent to cells with higher elevation values and no neighboring cell at a lower elevation. This generates a downward slope away from the higher areas.



	1	2	3	4	5	6	7
A	9	9	9	9	9	9	9
B	9	6 1	6 1	6 1	6 1	6 1	9
C	8	6 1	6	6	6	6 1	9
D	8	6 1	6	6	6	6 1	9
E	7	6 1	6	6	6	6 1	8
F	7	6	6	6	6 1	6 1	8
G	7	7	5	7	7	8	8

(a)

	1	2	3	4	5	6	7
A	9	9	9	9	9	9	9
B	9	6 3	6 3	6 3	6 3	6 3	9
C	8	6 3	6 2	6 2	6 2	6 3	9
D	8	6 3	6 2	6 1	6 2	6 3	9
E	7	6 3	6 2	6 2	6 2	6 3	8
F	7	6	6	6	6 3	6 3	8
G	7	7	5	7	7	8	8

(c)

	1	2	3	4	5	6	7
A	9	9	9	9	9	9	9
B	9	6 2	6 2	6 2	6 2	6 2	9
C	8	6 2	6 1	6 1	6 1	6 2	9
D	8	6 2	6 1	6	6 1	6 2	9
E	7	6 2	6 1	6 1	6 1	6 2	8
F	7	6	6	6	6 2	6 2	8
G	7	7	5	7	7	8	8

(b)

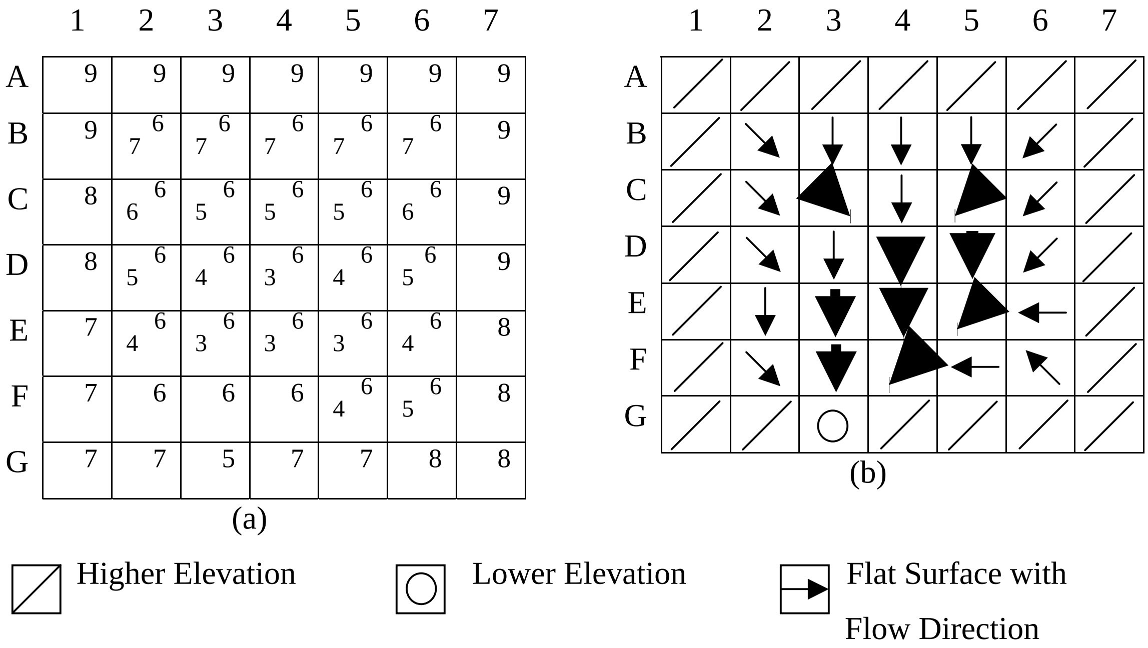
	1	2	3	4	5	6	7
A	╱	╱	╱	╱	╱	╱	╱
B	╱	↓	↓	↓	↓	↓	╱
C	╱	↓	↓	↓	↓	↓	╱
D	╱	↓	↓	↓	↓	↙	╱
E	╱	↓	↓	↓	↙	←	╱
F	╱	↘	↓	↙	←	←	╱
G	╱	╱	○	╱	╱	╱	╱

(d)

╱ Higher Elevation ○ Lower Elevation → Flat Surface with Flow Direction

Figure 2.6: (a) Lower left hand number is the first pass of the depression incrementing elevation of cells generating slope away from high areas (b) and (c) depict successive passes there by achieving the resultant flow directions as shown in (d).
(Source: Garbrecht and Martz, 1997)

In the third and final step, the elevation increments of the previous two procedures are added linearly and applied to the cells. Since both the increments are inline with the law of physics (i.e. flow is always away from higher areas and towards low areas), the flat area is replaced by a downward slope towards an outlet, removing indeterminacy of flow paths.



	1	2	3	4	5	6	7
A	9	9	9	9	9	9	9
B	9	6 7	6 7	6 7	6 7	6 7	9
C	8	6 6	6 5	6 5	6 5	6 6	9
D	8	6 5	6 4	6 3	6 4	6 5	9
E	7	6 4	6 3	6 3	6 3	6 4	8
F	7	6	6	6	6 4	6 5	8
G	7	7	5	7	7	8	8

Figure 2.7: (a) Final incremental elevations achieved by linear addition of the increments achieved in the first two steps. (b) Final drainage pattern with the size of arrows indicative of the size of respective upstream drainage area.
(Source: Garbrecht and Martz, 1997)

2.11.3 DEDNM-Flow Path Assignment using D8 method.

Once the DEM is free from sinks and flat areas, the flow vectors are determined using the D8 method. The D8 method (Douglas, 1986; Fairchild and Leymarie, 1991) defines the landscape properties for each individual raster cell by the evaluation of itself and eight immediately adjacent cells.

1	2	3
4	X	5
6	7	8

Figure 2.8: D8 method matrix, Flow from ‘X’ can flow in any of the eight adjacent cells depending on the steepest slope direction.

In this approach, the down-slope flow angle for each cell is determined. This angle is measured counter-clockwise from the steepest downward slope. The flow direction is the resultant of the two successive steepest directions which are not diagonals. TOPAZ however, adopts a much simpler approach wherein the steepest slope is assigned as the flow path direction and incases of a tie the direction is chosen at random out of the two or more equally sloping directions.

2.12 Watershed Segmentation - Hydrologic Response Approach

In hydrological modeling, the segmentation of the land surface into spatial units based on their homogeneity with regard to their hydrological response is an emerging research area. In recent years there have been efforts to segment watershed based on hydrologic response unit (HRU) approach. This approach is effective in processing large watersheds. In HRU approach the areas with same responses are lumped together irrespective of their location. (Kouwen et al., 1993). There have been evaluative studies on segmentation of watersheds based on hydrologic response units based on derived drainage direction (Shaw, 2002). Derived drainage direction is defined as the drainage direction in line with the hydrography of the area. This derived drainage direction is limited to the four fundamental directions. WATPAZ Software (Shaw, 2002) has this cardinality limitation.

CHAPTER-3

METHODOLOGY

3.1 Framework

The need for node manipulation capabilities for point specific studies, to delineate subwatersheds at points of specific interest rather than junction nodes and the need for the inclusion of water bodies in the extraction process lead to the development of the concept of a new software module in the TOPAZ module called TOPAZ-N (TOPAZ with node manipulation capabilities). A select set of hydrological data is required by the various TOPAZ-N functions to achieve the desired analysis. The source of this data for TOPAZ-N is not only external but also the intermediate data obtained from the internal processing. It was thus important to identify the data requirements for each step of the TOPAZ-N execution process. The Information Engineering (IE) (Martin and Finkelstein, 1981) based structured analysis approach was adopted to ascertain the data requirements for TOPAZ-N. The analysis was subdivided in following two logical categories:

1) Functional Requirements.
2) Technical Requirements.

3.2 Functional Requirements

The functional requirements for TOPAZ-N involved the identification of the scope and hence the framework of the new module/model. While doing so, the open-end structural requirement for possible future additions was kept in mind. This being the preliminary stage, a broad system domain was defined. The system’s domain was characterized as its ability to carry out automated extraction and parameterization of watersheds along with water bodies in a more efficient and flexible way. This domain was further decomposed to a user-subsystem level. The subsystem was defined as the users' ability to initiate subwatershed delineation at user specified points. In other words, it meant the ability to analyze and manipulate:

1) Subwatersheds
2) Nodes
3) Channel Links

Spatial hydrology identifies only the channel links and the subwatersheds as hydrological features, where nodes are treated only as a tool for achieving certain objectives. In designing TOPAZ-N, the need to account for the hydrological importance of the nodes was felt. Nodes can be identified with gauge stations, and thus the nodes can become a hydrologic entity. TOPAZ-N involved dealing with three distinct, yet hydrologically linked entities: channel-links, subwatersheds, and nodes. The subsystem was further decomposed into functions. The major functions identified were the functions to add nodes in the channel network, functions to divide channel segments and thus, subwatersheds draining into those segments, and functions to remove nodes, thus merging channel segments and the corresponding watersheds. Figure 3.1 depicts the major object interactions for the system. TOPAZ-N Central Control is the controlling object.

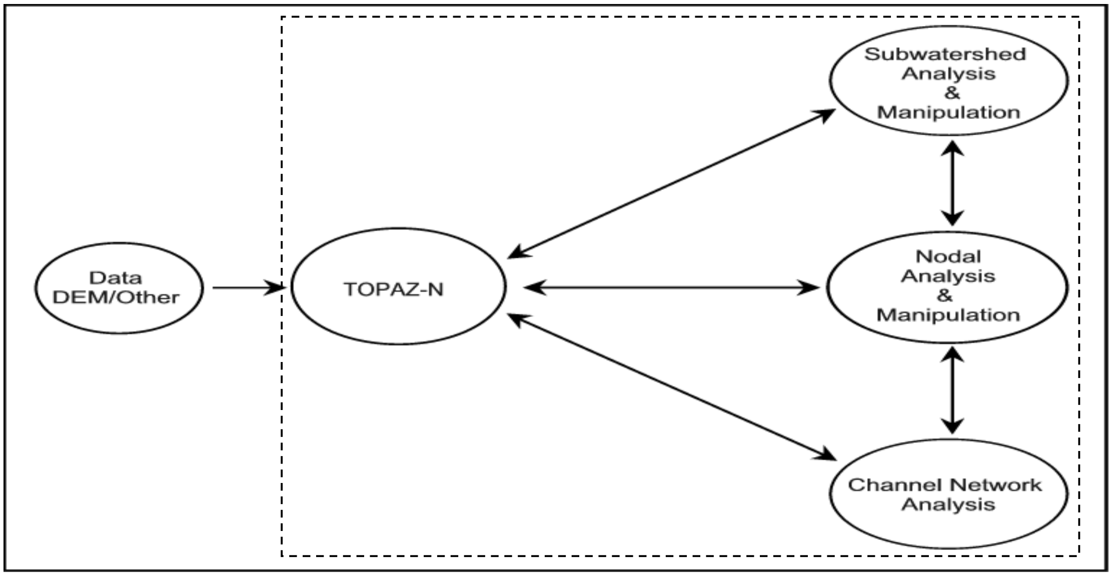


Figure 3.1: Functional Requirement for the TOPAZ-N. The 'driver' module received data from DEM/DEDNM and interacts with three basic internal functional modules.

Since all other objects are hydrologically linked they are also linked, in the implementation. It is important to note that in raster processing there is no such relationship. Thus, the objects were to be designed for raster data, yet they should have the vectorial relationship or multiple attribute associativity. Figure 3.1 depicts the Subwatershed Analysis and Manipulation function which interacts with the TOPAZ-N Central Control function to receive subwatershed merger or division related commands.

Subwatershed Analysis and Manipulation function then interacts with the Nodal Analysis and Manipulation function directly. Subwatershed Analysis and Manipulation interacts with Channel Network Analysis function through the Nodal Analysis and Manipulation Function or through the TOPAZ-N Central Control function to maintain feature, data and parameter information integrity across the whole system in carrying out subwatershed manipulation and analysis. Also, since node-related manipulation has affects the subwatershed configuration as well as the channel network, the node analysis and manipulation function interacts directly with the other two modules on receiving any node manipulation or analysis instructions from the central control function. As the arrow indicates the interaction is two-way. The information is exchanged between the layer objects. The system, however, does not permit any direct manipulation of the channel network. The channel network is only manipulated through manipulation of subwatershed and node manipulation. However, independent interaction between the central control function and the channel network layer object is permitted for analysis. The same degree of analytical interaction is also permitted between the other two functionalities and the central control function. The central control function for TOPAZ-N also serves as a data input or data filtering and channeling function for the system. The central control does intermediate processing of internal data as well as of the external data on requests by client functions.

3.3 Technical Requirements

The assessment was complex and was therefore broken down into a number of simple assessments. At the first level, a very coarse data flow within the system was charted out. In the functional requirements’ assessment, the major object interactions were identified. The data flow chart laid down the groundwork for the modeling of the data and the data structure. The raster data was classified into three major layers: Channel Network Layer, Subcatchment Layer, and Node Layer (Figure 3.2). These layers were named after their constituent objects. Object interactions were used as a blueprint to chart out the layer interactions. The classification was based on major features as done in vector-based spatial hydrology. These features are interlinked. Any node change has a corresponding change in channel layer and subcatchment layer and vice-a-versa.

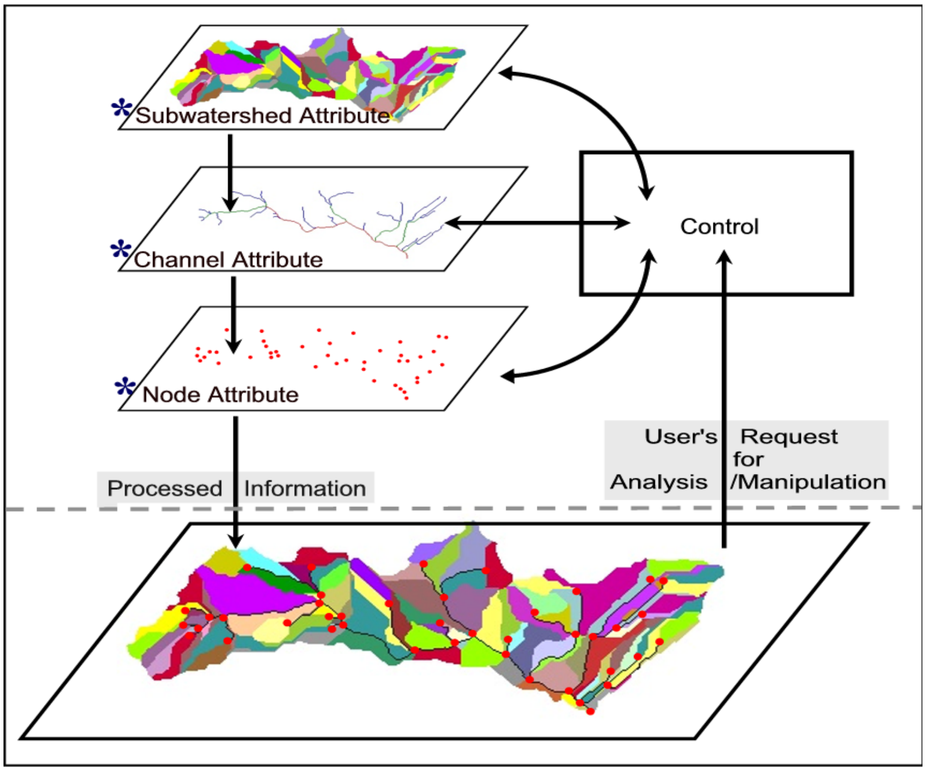


Figure 3.2: Architectural Level Data Flow in the TOPAZ-N, The new data structure views and processes entities and attributes as interlinked layers. Links between the layers are based on their hydrologic relationship

To achieve the same hydrological dependency between the data structure, and to maintain data and information integrity, it was imperative to link these layers based on their hydrological dependencies. Thus, a control mechanism was put into place to process the user’s requests in accordance with the hydrological constraints. Also, each layer itself was identified as a distinct object for implementation. Figure 3.2 depicts the macro or the architectural layout and data/information flow for the TOPAZ-N. Each layer is an object comprising of a collection of a particular raster unit.

3.4 Data Modeling

In order to develop TOPAZ-N to be reliable, adaptable and open to future growth it was essential to base it on a data model with all these desired characteristics identified in the earlier analysis. The data model forms the foundation of an information system or decision support system based analytical tool. To develop a sound data model for TOPAZ-N, the following goals were set forth to be achieved in the data modeling:

1) Due to spatial variability of the processed data it is essential that the data model establishes a clear picture of the data structure that would evolve.
2) The data model should accurately and completely adhere to the problem and solution space.
3) The data model should be transparent and should reveal, in entirety, the architecture of the framework/solution.

To achieve these goals CASE (Computer Aided Software Engineering), Chen's ER (Entity Relationship) (Chen, 1977, 1983, 2002) and IDEF (Integrated DEFinition) (Mayer, Richard J., et al.; Wisnosky, Dennis E., Allen W. Batteau, 1990) modeling methodologies were studied to generate an amalgamated approach suitable to the spatial domain.

CASE is a collective term used for a collection of software tools available to produce diagrams and models. CASE tools analyze component relationships and generate the analysis and design in narrative form. ER diagrams are a significant and important part of any CASE tool. ER diagrams provide a conceptual visual of a data structure being modeled. There are three basic elements in ER models:

1. Entities are the "things" about which we seek information.
2. Attributes are the data we collect about the entities.
3. Relationships provide the structure needed to draw information from multiple entities.

These however are not enough to completely model the spatially variable hydrologic model entities and their dynamic relationships. IDEF is the modeling technique used to model functional processes. IDEF comprises of the Data Model, which is a model of the entities and their relationships involved in the design. Data Model also involves 'information' model, which is a set of activities (functions) acted upon the entities and by the entities based on their attributes, roles and relationships. In IDEF a combination of data and information model is used to develop the data structure and the process model. IDEF has few limitations in modeling dynamic spatially variable hydrologic relationships between the hydrologic model entities. A complimentary use of CASE tools with ER diagrams and IDEF modeling methodologies helped in offsetting their individual inherent limitations. This helped in achieving a near perfect abstraction with easy analytical ability along with formality and executability of the data model. Since the model being developed is a hybrid raster model, it was not possible to completely adhere to the Calkin's (1996) modified ER symbology for spatial dataset which is more suitable for vector data only.

Figure 3.3 represents the complex relationships as exist between the different entities of the TOPAZ-N. Unlike traditional relational structures, entities in TOPAZ-N are inter-dependent for their definition. For instance 'channel' is defined by 'starting node' and 'ending node', a 'subcatchment' is identified by the channel reach it drains into, thus a change in a node location affects all the three entities i.e. the node itself, the channel and the subcatchment. In traditional relational structures entities remain constant in their definition. For example in case of a doctor-patient-ward entity relationship, a patient is not dependent on doctor for its definition and vice-a-versa; same is true for patient-ward and doctor-ward entity relationships.

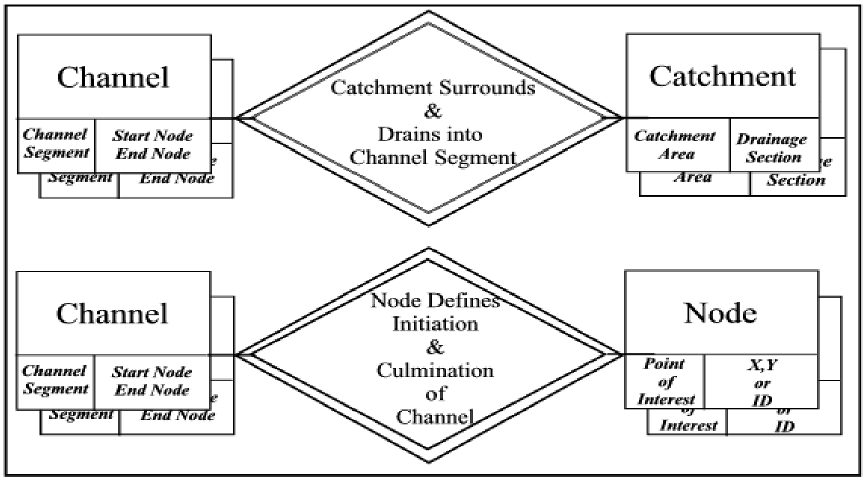


Figure 3.3: Entity Relationship Diagrams for TOPAZ-N

3.5 Hydrological issues in Data Structure & Process Modeling

TOPAZ-N, being developed as a raster-based tool, would be processing and analyzing spatially extensive data. Thus, it appears to fit into the generic geometric based spatial analysis. This, however, is not true, as TOPAZ-N does not fit into the typical geometric based approach. There were many hydrologic issues that are completely different from vector data model based geometric manipulation. The merging of subwatersheds in hydrology cannot be designed as a simple geometric assimilation based on geometric adjacency definition. The node classification issue, and thus the channel order issue, was also identified.

3.5.1 Subwatershed Aggregation and Adjacency Factor

While adopting the vector-based approach of processing for raster data, since in pure raster there are no separate entities, it was realized that the vector-based approach for processing of raster data cannot be implemented in its entirety.

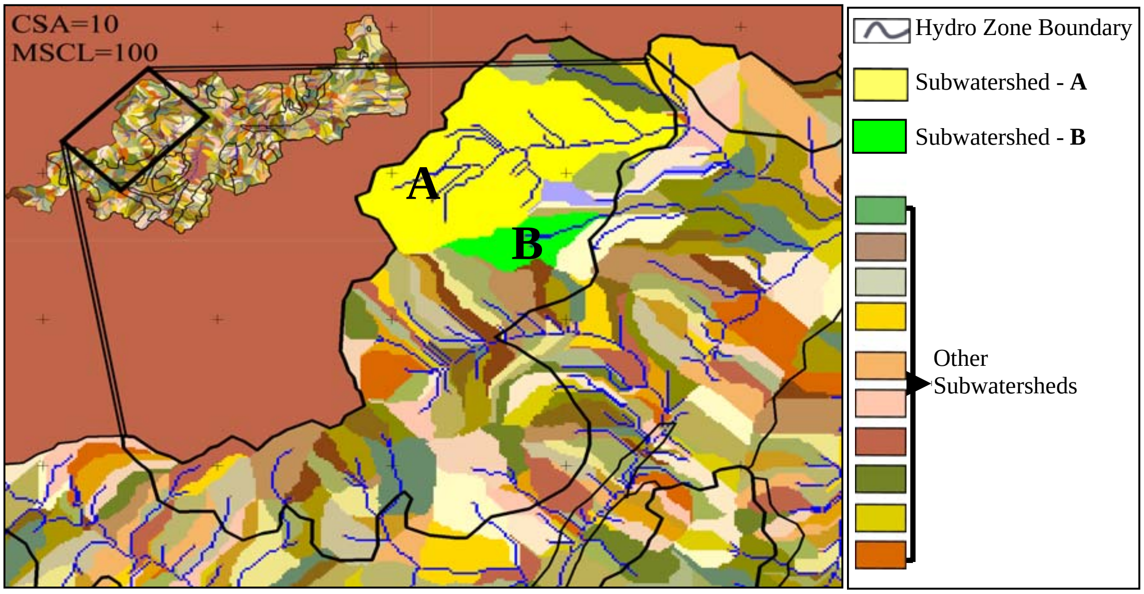


Figure 3.4: Adjacency Issues in TOPAZ-N and Hydrology in general: Neither do the subwatershed ‘A’ & ‘B’ flow into one another nor do they have a common outlet subwatershed, hence cannot be characterized as hydrologic neighbor and thus cannot further be merged.

It was felt that the issue of adjacency for this raster-vector hybrid approach needs to be defined from scratch and needs to be tailor made based on the hydrologic principles. For instance, as in Figure 3.4, subwatershed (A) and subwatershed (B) are sharing a common boundary; hence, as per geometric principals, they can be classified as adjacent or neighbors. Thus, the two watersheds can be merged together. But hydrologically, the two watersheds are far from being neighbors as they do not flow into each other nor do they have a common outlet. Merging subwatershed (A) and subwatershed (B) would lead to ambiguity in flow direction. Both subwatersheds would contribute to the channel network at two different points. This would make the flow routing and parameterization vague. Thus, subwatershed (A) and subwatershed (B) cannot be merged.

Hydrologically speaking, two or more subwatersheds can be classified as adjacent or neighbors, thus facilitating their merger if and only if one of the following criteria is satisfied:

1) They all contribute to a common node.
2) One flows into another.

Mere sharing of geometric boundaries, however, is not sufficient for hydrological adjacency. In implementation, the adjacency issue was worked out by establishing relationships by way of spatial operations. These operations involved all three major layer objects, namely the Subwatershed Layer Object, the Channel Network Layer Object, and the Node Layer Object. Any change in the configuration of any of these objects/layers was based on the central spatial operation threads. This was essential to maintain the data integrity in hydrological terms.

3.5.2 Subwatershed Aggregation and Bank Side Factor

The hydrological analysis also identified that in certain cases, the merger of two or more subwatersheds leads to peculiar situations, in that it becomes impossible to adjudge the left and right sides of the new watershed. This is encountered when two watersheds on the top end of a typical ‘Y’ formation are merged. First, to get past many of such merger related issues, it was decided to make merger a node independent function. In other words, merger would not necessarily mean the omission of any node. The node would still be retained in other channel related parameterization processes. Also, merger would strip the watersheds of their bank identity. In other words, if the user chooses to merge, the resulting watershed would comprise of subwatershed with no distinction as to their left or right banks.

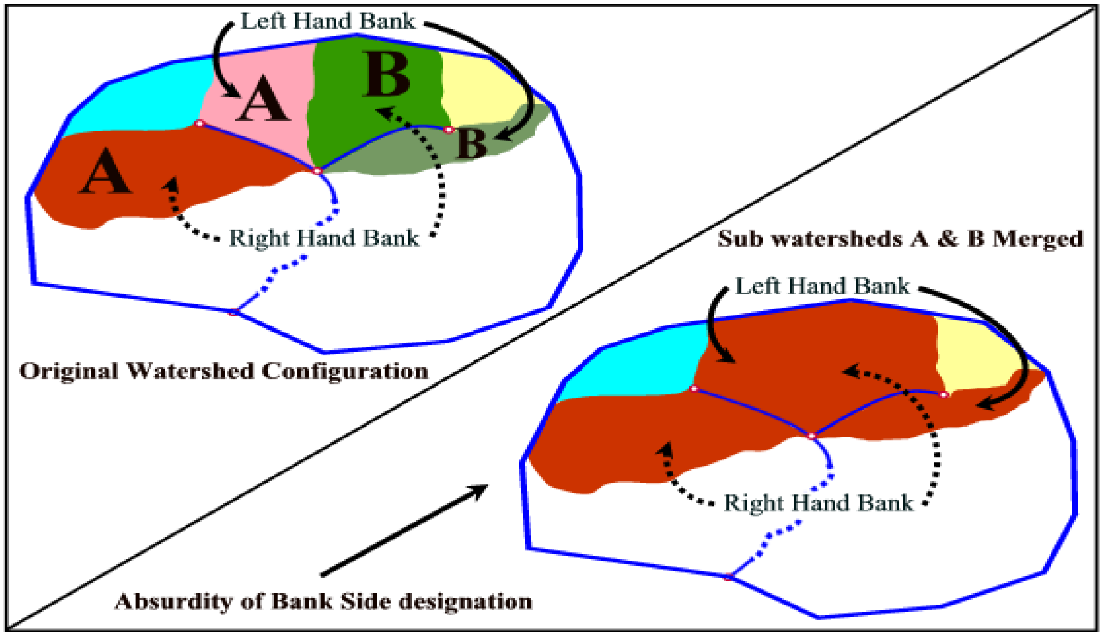

Left Hand Bank
A
B
B
A
Right Hand Bank
Original Watershed Configuration
Sub watersheds A & B Merged
Left Hand Bank
Right Hand Bank
Absurdity of Bank Side designation

Figure 3.5: Adjacency Issues in TOPAZ-N and Hydrology in general. Retaining the left-right bank identity after the merger of ‘A’ and ‘B’ subwatersheds creates ambiguity as there are two channel segments in one subwatershed.

3.5.3 Subwatershed Aggregation And Channel Network Issue

Merger of subwatersheds also has an impact on the channel network. There were two options available to carry out the subwatershed merger. One involved altering the node system of one of the merging subwatersheds, which in turn affects the channel network. Another approach was to aggregate the subwatersheds and restrict this manipulation only to the subwatershed layer object and thus retain the node system to preserve the channel network. If, on the other hand, the aggregation involved simultaneous changes in the node system, and hence changes in the channel network, it would lead to absurd channel definitions, i.e., a channel with two sources but one end. In some complex aggregation cases, as in a typical ‘Y’ formation, there would be problems with channel network parameterization, for example, designating the stem part of channel section in a 'Y' configuration. Figure 3.6 illustrates that, in some cases, aggregation leads to a channel segment in the new subwatershed that has two initiatory points but only one end point. Although the situation is not absurd in reality but it does create absurdity in the current parameterization methodology.

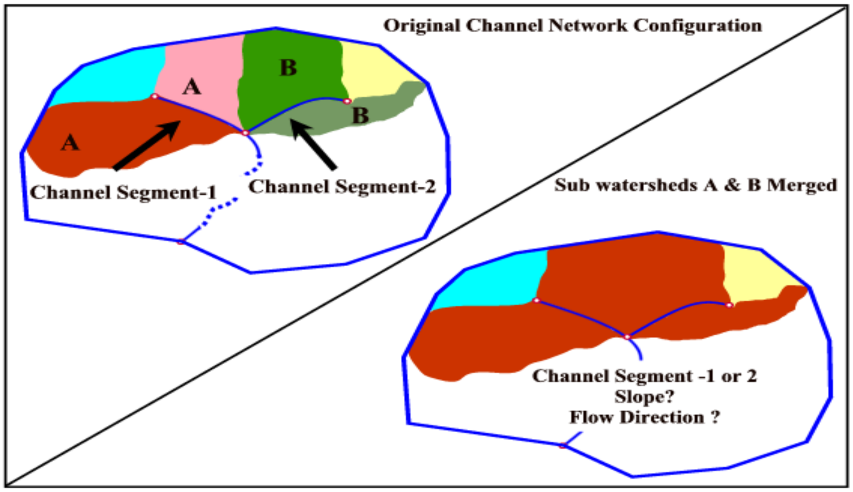


Figure 3.6: Adjacency Issues Merger of subbasins may cause 'Y' shaped channel segments in a subwatershed, distinct identity for each component of the segment requires retaining node as pseudo-node.

Thus, by retaining the junction node in the merger process, TOPAZ-N maintained the sanctity of the channel network along with channel order allocation issues.

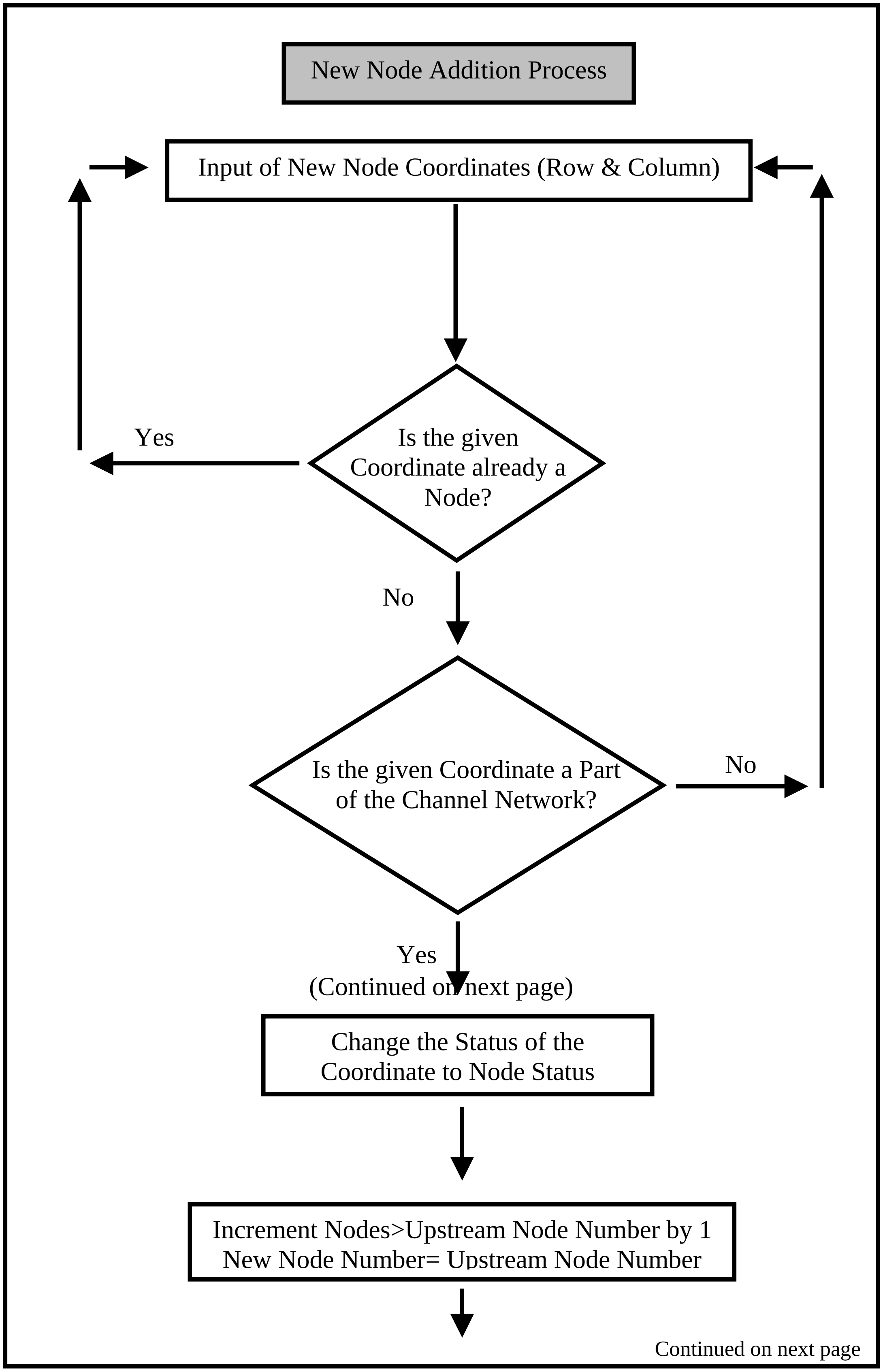
3.6 Process Modeling

Raster data has a flat tabular structure. In a raster there are grid cells with values. Each set of grid cells (a layer) corresponds to some attribute value associated with that location. Thus, raster structure lacks any real-life abstraction. In TOPAZ-N, a new, raster-based data structure was developed exclusively tailored for the spatial hydrology. It was felt that adhering to commonly available data and process models for spatial analysis does not suit spatial hydrology. The most pronounced was the adjacency issue. In any topological application, sharing of a boundary qualifies features to be classified as adjacent or neighbors. This, however, is not true in hydrology as topologically adjacent might not be hydrologically adjacent. In hydrology, adjacency is based on the final drainage point and is linked to the channel section in to which the watersheds finally drain.

3.7 Algorithm for Information Extraction and Utilization

TOPAZ–N receives input from the output of the DEDNM preprocessing module. TOPAZ-N receives the ‘flow direction grid’ called the “Flovec.dat”. Flovec.dat is a flat file containing the flow vector directions for each of the grid cells, which are listed as numbers 1 through 9 as per D8 methodology convention. Also received is the data regarding subwatershed identities of the non-channel cells from the “Subwta.dat” file. The “Netw.dat” file provides the skeletal stream network. TOPAZ-N also requires a file called “Channel.dat” an output from an addendum TOPAZ program called INTER which lists node numbers and their coordinates as extracted by the preliminary run of TOPAZ. The data from all these sources is organized in the new data structure modeled with the grid cell as the basic unit, node status as a class of this basic unit, ‘Channel’ and ‘Subcatchment’ forming the next level of hierarchy.

Each grid cell is assigned an additional attribute of ‘final flow’ point/direction. This is the point at which a particular cell drains into the stream. Jenson and Domingue (1988) do make a reference to this property but never has it been utilized in any process modeling or analysis. This property is retrieved by starting with the top left hand corner of the grid. A trace is performed so as to ascertain where the cell is draining, based on its flow direction value. On reaching its drain point, a check is done if the drain cell is a part of a channel or not. If not then the process is continued until a channel cell is reached. When the trace leads to a stream cell, then the address of this stream cell is assigned to all the non-stream cells visited in the trace process. This process is continued until all of the un-flagged cells are assigned ‘final flow’ point/direction. At the same time, the traversal of the channel network is done and information of upstream and downstream node is collected and stored. The addition of node is accomplished by asking for user input of the row and column for the new node to be inserted. First, a check is done if the given cell is a part of the channel network or not. If not, the user is asked to provide another set of coordinates. If the user-supplied coordinates (Row and Column) are a part of the channel network they satisfy one of the criteria for assigning the new node status.


New Node Addition Process
Input of New Node Coordinates (Row & Column)
Is the given Coordinate already a Node?
Yes
No
Is the given Coordinate a Part of the Channel Network?
No
Yes
(Continued on next page)
Change the Status of the Coordinate to Node Status
Increment Nodes>Upstream Node Number by 1
New Node Number= Upstream Node Number

Continued on next page

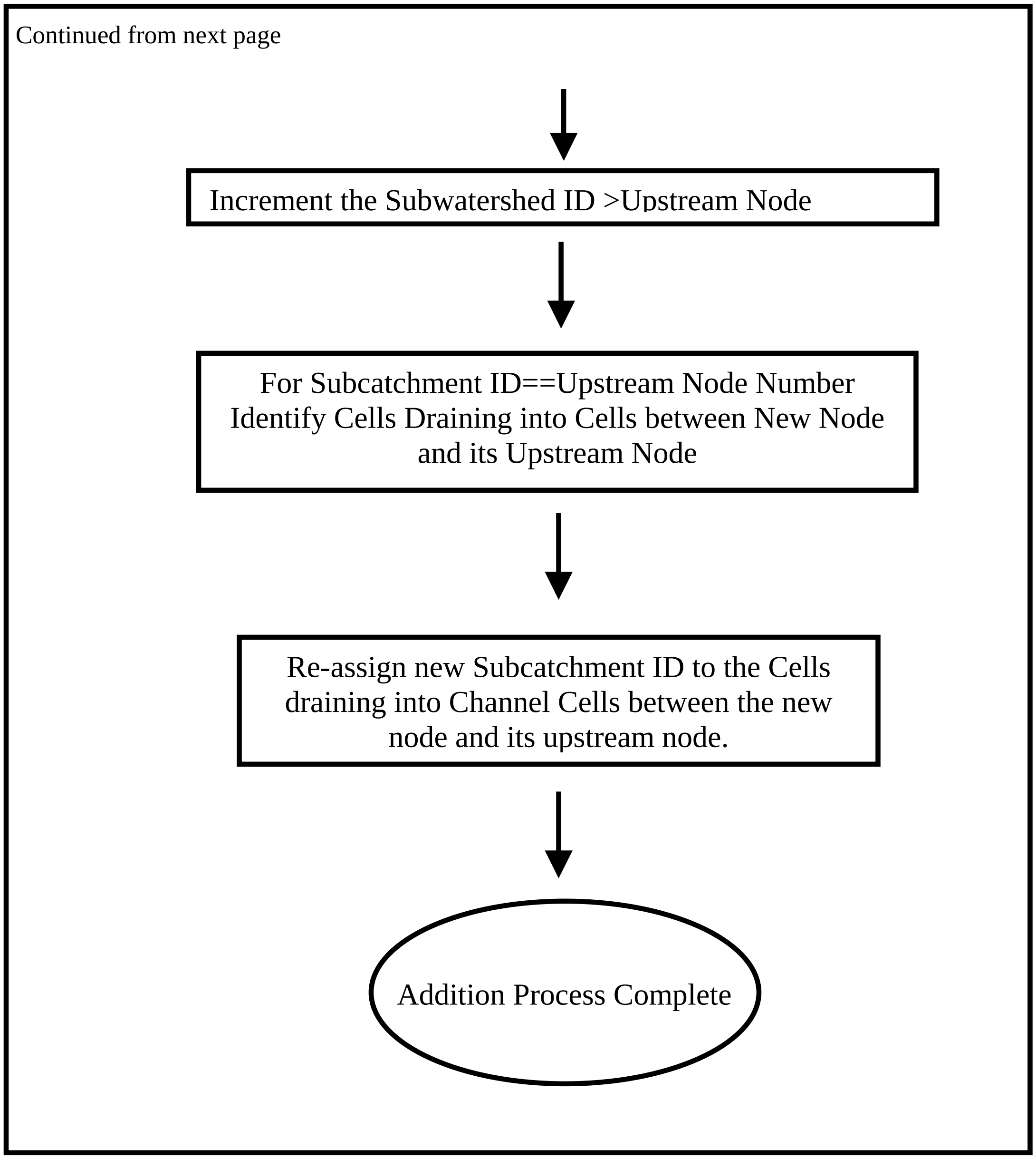

Continued from next page
Increment the Subwatershed ID >Upstream Node
For Subcatchment ID==Upstream Node Number Identify Cells Draining into Cells between New Node and its Upstream Node
Re-assign new Subcatchment ID to the Cells draining into Channel Cells between the new node and its upstream node.
Addition Process Complete

Figure3.7: Flow Chart of the New Node Addition Process in the TOPAZ-N Module.

Then their current node status is verified. The given coordinate has to be a channel cell but with no node status. Once the non-node status is confirmed, the process for the addition of a new node at that cell is initiated. This is worth mentioning although exterior to the program the process appears to be a node "addition" or "insertion" process, but internally it is designed as a "re-assigning" process.

Once the user-supplied coordinates are verified, their node status is immediately changed, and the upstream and downstream nodes of this new node are determined. The new node is assigned a node ‘I.D.’ one value higher than its downstream value. Then all the nodes with an ‘I.D.’ greater than the ‘I.D.’ of the new node get a one unit-increment in their ‘I.D.’. This also changes the subcatchment ‘I.D.s’ for all the non-channel cells with ‘I.D.’ higher than two unit values of the new node ‘I.D.’.

Next a trace is started from the u/s (upstream) node of the new node flagging all the channel cells until the new node is reached. Then a pass is made on the cells with subcatchment ‘I.D.’ of the new node ‘I.D.’. All the cells which drain into any of the flagged cells between the u/s node and the new node are then given a single unit increment in their subcatchment ‘I.D.’, thereby partitioning the old subcatchment in two.

Merger of two subwatersheds is designed to merge only two hydrologically adjacent subwatersheds, and not topographically adjacent subwatersheds. The user is asked to provide the ‘I.D.s’ of the subwatersheds that need to be merged. Upon receiving the two ids for example ‘A’ and ‘B’ a check is performed to verify whether A is d/s of B or B is d/s of A. Once this is confirmed then the two subwatersheds are merged, retaining the outermost subcatchment ‘I.D.’. No other subwatershed gets any change in its ‘I.D.’, nor is the node number of any other node changed. Upon merger, the ‘removed’ node is ‘de-recognized’ as an active node and is assigned a pseudo-node status. This serves only one purpose, i.e., if the merged watersheds are the two arms of a ‘Y’ junction, then the pseudo-node maintains a distinction between two different channel segments to avoid ambiguity in the channel network. The merger of subbasins is not purely a process of node removal. Also, during the merger, the process rids the subbasins of their left and right bank associativity. This is again meant to avoid ambiguity in case of ‘Y’ junction mergers as described in section 3.5.3.

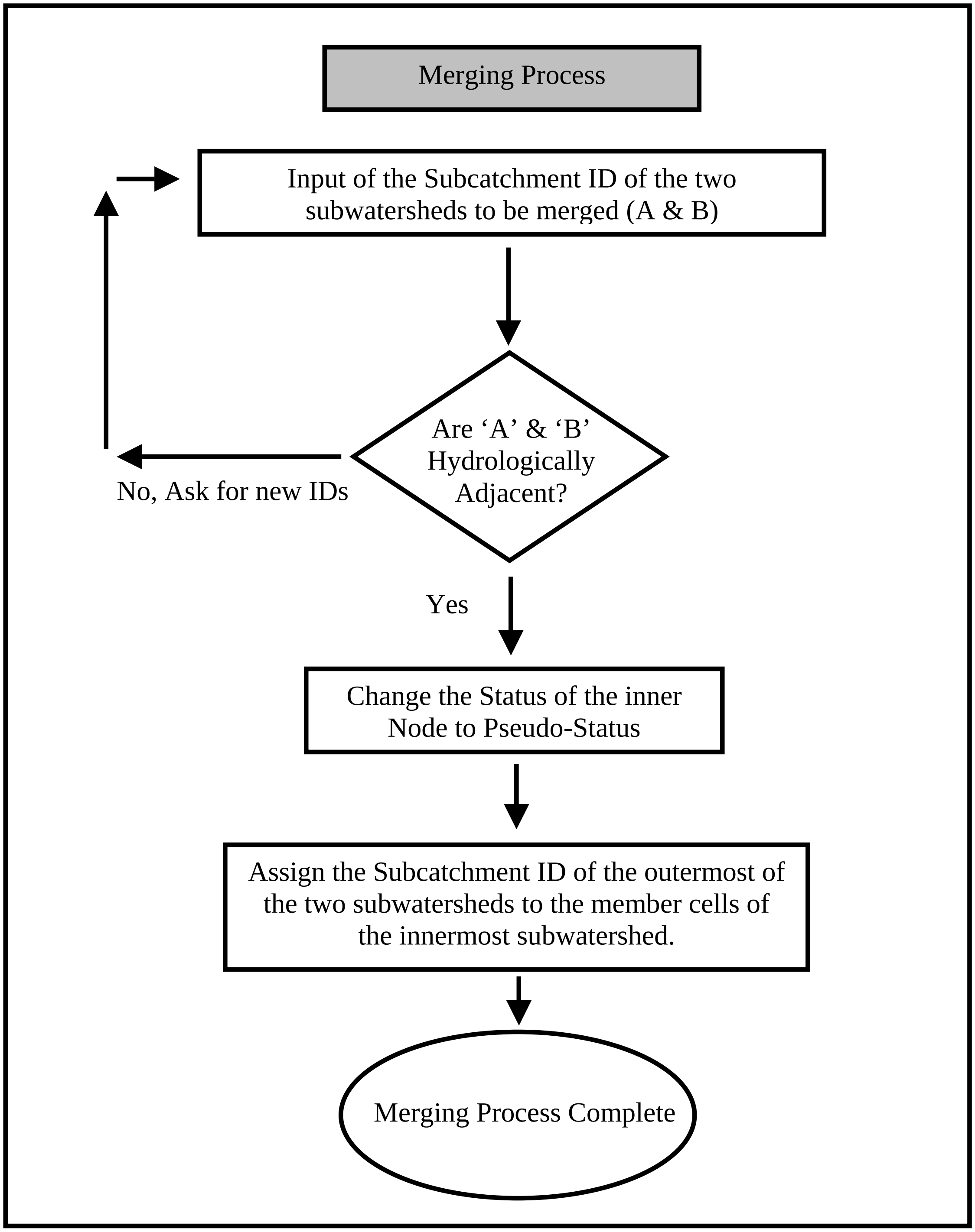

Merging Process
Input of the Subcatchment ID of the two subwatersheds to be merged (A & B)
Are ‘A’ & ‘B’ Hydrologically Adjacent?
No, Ask for new IDs
Yes
Change the Status of the inner Node to Pseudo-Status
Assign the Subcatchment ID of the outermost of the two subwatersheds to the member cells of the innermost subwatershed.
Merging Process Complete

Figure3.8: Flow Chart of Merging Process in the TOPAZ-N Module.

Using these two capabilities, another process is designed to delineate water bodies. It is to be noted that the accuracy of the extraction or the homogeneity of the extracted feature is a factor of right combination of MSCL and CSA values which is evident from test runs. For extraction, the nodes are inserted at the intersection of streams with the lake boundary. The identification of coordinates of new nodes to be inserted is done externally, by performing an overlay analysis on an intermediate output of TOPAZ-N and the lake boundary dataset. The identified new node insertion coordinates are fed into the TOPAZ-N by a simple text file, and these new nodes get inserted. Next, a new 'intermediate' output from TOPAZ-N is taken out and all the subwatersheds lying within the lake boundary are identified by overlay analysis. The 'I.D.s' of these subwatersheds are fed into TOPAZ-N through a file. These subwatersheds are checked and merged to define a new 'feature'. The user is then asked to supply an 'I.D.' for this lake. However the cells which are channel cells and are inside the lake boundary retain their status, as it is necessary to maintain flow continuity over the lake

CHAPTER-4

MODEL DEVELOPMENT

4.1 Introduction

The software implementation of TOPAZ-N is done using Visual C++. There is a abundance of programming languages in the information technology market today. FORTRAN, which is still to some extent, a programming language of preference for the development of scientific and engineering applications is being replaced by more popular object oriented programming languages. Why Visual C++ over FORTRAN? C++ and Fortran 90 both are object-oriented languages used for scientific computing. C++ is a fully developed and full-featured, object-oriented language that provides support for inheritance and polymorphism. Inheritance allows code reusability feature and polymorphism is the ability of a method to behave differently depending upon the object it is acting on. Fortran 90 provides some object-oriented features through combinations of its TYPE and MODULE syntax elements, but it lacks inheritance and thus does not permit code reuse to the same extent as C++.

FORTRAN has only standard data types like characters, integers, real etc. It does not permit user defined data types. Visual C++ on the other hand allows creation of user defined abstract data types also called ADT. This data abstraction capability enables the development of data type closely resembling the real world entity being mapped into the software. This close resemblance is in terms of characteristics and functionality. These complex data types enable the application developer to simplify the over all application model.

The other most important feature missing in FORTRAN 90 is the template. Template functionality in C++ allows a developer to build a portable and reusable code and to improve the efficiency of evaluation of complex expressions using user defined data types. C++ also offers advantage of a low-level language as well as a high level language due to its dual nature. It can interact directly with the system hardware almost without any limitation like a low-level language, at the same time it can be used like any high level language with supporting specific libraries. C++ codes are much concise as compared to the same code written in any other language. In summary C++ allows to take advantage of modern software design principles. It allows developers to write well-structured code that can be easily modified as needed for changes in scientific research.

4.2 Conceptual Design

Projects in software domain, just like any other domain, require development of a conceptual design. The conceptual design describes the system in terms of major design elements and relationship among them. It involves determining user requirements and functional requirements of software. This requires gathering of information, based on which, the application responses are derived as per user requirements and actions.

A Conceptual design maps the activities of the process being modeled and the tasks that a user might need to perform to address a certain issue. The information which is collected in the conceptual design process is transformed into the logical design and finally into the physical design.

4.2.1 General & User Inputs, User Requirements and Application Responses.

The information collected for the conceptual design of TOPAZ-N consisted of general inputs, user inputs, user requirements and the corresponding application responses. The first step was to identify the user requirements, which were:

1. The ability to partition a subwatershed, and
2. The ability to merge subwatersheds.

The next step in this process was to ascertain the inputs that the application would need from the user to accomplish the aforementioned requirements. These inputs were identified as:

1. User specification of the co-ordinates where a new node needs to be inserted.
2. User input of the sub catchment identification numbers to merge the subwatersheds.

Based on these two information pieces, next was to determine the other non-user inputs for TOPAZ-N. DEDNM (a module of TOPAZ) was identified as the source for these non-user inputs. This was because the DEDNM pre-processes and extracts the preliminary subwatersheds. This would also ward off any redundancy in the functional design of TOPAZ-N. The DEDNM outputs which would enable and equip TOPAZ-N to satisfy user requirements were identified as:

1. File NETW.OUT - It contains the information about the definition of the drainage network. This file comprises sequential numerical values, where '0' value corresponds to non-channel cells and any other positive numerical value correspond to the 'Strahler' order of the channel to which the representative cell belongs.

2. File SUBWTA.OUT – It contains the unique subcatchment indices of each of the delineated subcatchment. The encoding of subcatchment follows the following rule to include the left and right hand designations for the banks:

   Source Node Subcatchments = (NODN * 10) + 1

   Right Bank Subcatchments = (NODN * 10) + 2

   Left Bank Subcatchments = (NODN * 10) + 3

   Channel Network Cells = (NODN * 10) + 4

   where NODN is the index assigned to the nodes or channel links of the generated drainage network.

3. File FLOVEC.OUT – This file contains the drainage direction at each raster cell (local flow vectors) in one of eight principal directions of the raster grid (D-8 method). Values of flow vectors are 0 through 4, and 6 through 9. The value of 0 defines cells with indeterminate flow direction.

4. File BOUND.OUT File - BOUND.OUT contains the watershed definition above the user specified watershed outlet point. Cells with a value of 0 are outside the watershed boundaries, and cells with a value of 1 define the area within the watershed boundaries.

5. Another TOPAZ-DEDNM addendum whose out put was recognized to be needed by TOPAZ-N was INTER. INTER is an intermediate module which gives node information. Channel.int, an output of Inter, lists nodes, their coordinates and their numbers for the initial DEDNM delineated channel network.

4.2.2 Data and Process Model Design

Data Structure is a logical way of organizing data. Data in a data structure can be of fundamental data type or it can be a user defined data type. An important step in the development of a software program and paradigm is the design of model. A software model comprises the activities, methods, and practices necessary to develop the system (Humphrey, 1989). Just like any other model, it is an abstraction of the methodology. These models are employed as a mechanism to reduce the complexity of a real world entity by removing irrelevant details.

Data structure modeling or design is the formal layout of the application data in the computer memory. TOPAZ-N deals with four distinct and yet co-related entities namely:

1. Cell: It is the basic raster data entity. Each cell in a raster corresponds to a particular geographic location.
2. Channel: Channel is a hydrologic entity, which is a set of linear sequence of cells satisfying the threshold criteria for their upstream contributing areas.
3. Subcatchment: It is a collection of cells which have a common drainage outlet.
4. Node: Node is a cell that lies at the junction of two or more channel segments.

Thus, TOPAZ-N requires these aforementioned sets of data abstractions. In addition for smooth data flow, file handling and container structures are required. TOPAZ-N thus was designed with the following base classes:

1. Cell
2. Node
3. Subcatchment
4. Channel
5. File Manager
6. TOPAZ_N

The inheritance structure for the classes which form the basic entities of the model is as follows:

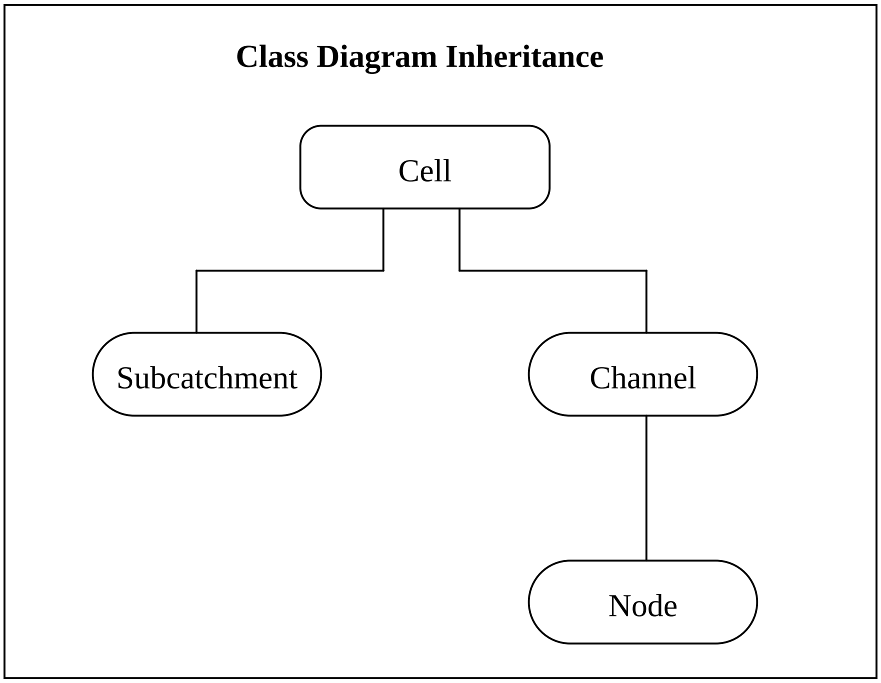


Fig 4.1- Inheritance hierarchy of the classes comprising TOPAZ-N

Inheritance allows a class to use properties and methods of another class while adding its own functionality. The base class is known as the parent class and the inheriting class is known as the child class. In case of TOPAZ-N cell is the parent class while Subcatchment and Channel are the child classes inheriting from the Cell class. Channel class is the base class for the Node. Node class in this case inherits all the properties and functionality of the immediate parent class Channel and also from the Cell class from the hierarchy. Inheritance hierarchy allows reusability of code and makes software design simpler and cleaner. The abstraction of each of the entity comprising TOPAZ-N required identification of the attributes and the behaviors for that entity. Classes are the result of the abstraction of these hydrologic entities into their programming language incarnates the abstract data types.

The Cell Class was abstraction of ‘cell’ of a raster. A cell has a unique geographical location. Also cells have a number of attributes depending upon the study. These attributes could be soil type, vegetation type but in case of TOPAZ-N the basic attribute that was encoded into cell class was its flow vector direction, thus cell class in addition to its cell ID has the row and column co-ordinates of the cell into which it flows.

Cell	
Attributes-	***Behaviors-***
-cellid; *-flowrow;* *-flowcol;*	*+setflowDir (long unsigned int, long unsigned int, long unsigned int): void* *+colfunct (long unsigned int, long unsigned int): int* *+rowfunct (long unsigned int, long unsigned int): int* *+cellid (long unsigned int, long unsigned int, int, int): long unsigned int* *+setFinalFlowDirExtraction (long unsigned int, long unsigned int): void*

Fig 4.2: Class abstraction of raster entity cell

The Channel class was characterized by its unique ID and the behavior to identify the channel status. The other attributes like the Strahler order were excluded from the encoding as they were not critical to characterize any channel segment. Similarly, the Subcatchment class is abstracted primarily on the basis of unique ID. The behaviors of Subcatchment class were in accordance with the need to set, alter and seek the attributes.

Channel	
Attributes-	***Behaviors-***
+ int: ChannelID *-int: tempValue( )*	*+DefineChannelStatus (long unsigned int, long unsigned int): void*

Fig 4.3: Class abstraction of hydrologic entity channel.

Subcatchment	
Attributes-	***Behaviors-***
+ int: SubcatchID	*+getSubID ( ): int* *+setID (int): void* *+Mergesheds (int, int): void*

Fig 4.4: Class abstraction of hydrologic entity subcatchment.

Node, in TOPAZ-N, is defined as the junction of two or more channel segments, point of initiation or culmination of channel segments. Nodes numbers serve as the identification for the subcatchments. Node class is an important class as plays a central role in the functionality of TOPAZ-N. Major behaviors for the Node class comprise of seeking, altering/assigning the node status or node number for a particular valid cell.

Node
Attributes- *Type=initialValue*
Behaviors-
+setNodeNumber (int): void *+setNodeX (int): void* *+setNodeY (int): void* *+setNodeID (unsigned long int): void* *.....Continued on next page*

.....Continued from previous page

+nsidfill (vector<int long unsigned> &, vector<int>, vector<int>, int long unsigned const, int long unsigned const): void

+printNodeInfo (): void
+setNodeStatus (): void
+validate_Add (): void

+getdsnode (int, vector<int long unsigned, vector<int>, vector<bool>, vector<bool>, vector<int long unsigned>): int

+getusnode (int, vector<int long unsigned, vector<int>, vector<bool>, vector<int long unsigned>, vector<int>, int long unsigned, int long unsigned): int

+load_dim (int long unsigned &, int long unsigned &, vector<int> &, vector<int> &, vector<int> &, int long unsigned &, int long unsigned &, int &): void

+Pre_load_dim (int long unsigned &, int long unsigned &, vector<int> &, vector<int> &, vector<int> &, int long unsigned &, int long unsigned &, int &): void

Fig 4.5: Class abstraction of Node.

The two other major functionality internal to TOPAZ-N were data handling i.e. input from files and subsequent pre-processing to populate the data structure. File Manager class serves that purpose in the system. It creates input and out put stream objects for the entire TOPAZ-N. This class is more of a system requirement and has no hydrologic significance.

FileManager
Parameters -
- bool initializer_a *- ifstream InputCell1* *- ifstream InputCell2* *- ifstream InputCell3*

.....Continued on next page

....Continued from previous page (***FileManager)***
- ifstream InputCell4 *+ string ChannelDefFile* *+ string FlowPathFile* *+ string NodeFilePath*
Behaviors –
+openDataFiles ( ): void *+getChannelFile ( ): string* *+getFlowPathFile ( ): string* *+getNodeFilePath ( ): string* *+setChannelFile (string): void* *+setFlowPathFile (string): void* *+setNodeFilePath (string): void* *+output_11D (vector<int>, int long unsigned &, int long unsigned &, char, long unsigned int, long unsigned int, int): void* *+output_1 (vector<int>, int long unsigned &, int long unsigned &, vector<bool>,vector<bool>,char, long unsigned int, long unsigned int, int): void* *+output_2 (vector<bool>, int long unsigned &, int long unsigned &, vector<bool>,vector<bool>ns, char, long unsigned int, long unsigned int, int): void*

Fig 4.6: Class abstraction of FileManager.

TOPAZ_N is the container class for all these constituent classes of the TOPAZ-N program. TOPAZ_N encompasses all these classes to carryout intra class interactions. All the classes in TOPAZ-N are interdependent. Any change in anyone of the class element has effects on the instances of other classes. Thus any change in anyone of the class must be reflected accordingly in other dependent instances. TOPAZ_N class performs this container class functionality. TOPAZ_N, though a purely system requirement, is based on the hydrologic process and emulates the hydrologic model while performing internal task. The behaviors of TOPAZ_N class are designed to adhere to the hydrologic principals in their operations. For instance, it is encoded in the TOPAZ_N behavior to verify the hydrologic adjacency of subwatersheds.

TOPAZ_N
Parameters -
- char choice *- char ver* *+ int outrow* *+ int outcol* *+ int value* *+ int outcellid*
Behaviors -
+printMsg ( ): void *+readInputInfo ( ): void* *+mainmenu ( ): void* *+getNode ( ): Node* *+setNode (Node): void* *+getcell ( ): cell* *+setcell (cell): void* *+setFlagstaff (bool): void* *+setChar (char): void* *+printNodeInfo ( ): void* *+addNode ( ): void* *+mergeWatersheds (int, int): void*

Fig 4.7: Class abstraction of TOPAZ_N.

4.3 Class interactions

The smooth functioning of software requires clear definition of class roles and class interactions. There are certain class interactions which are implicit in nature and are an outcome of inheritance of the different classes involved. The other class interactions are much more explicit in their nature. Class interactions involve information exchange and subsequent alteration of data values amongst the different instances of the involved classes. In TOPAZ-N there are two tier interactions between the classes. The fundamental classes (Cell, Channel, Node, and Subcatchment) interact with each other to exchange information for the purpose of validation of values like confirming channel status, node status etc. These classes also interact with the TOPAZ_N and through TOPAZ_N with FileManager class for overall execution of functions and altering of data values. The interactions between various classes are shown below:

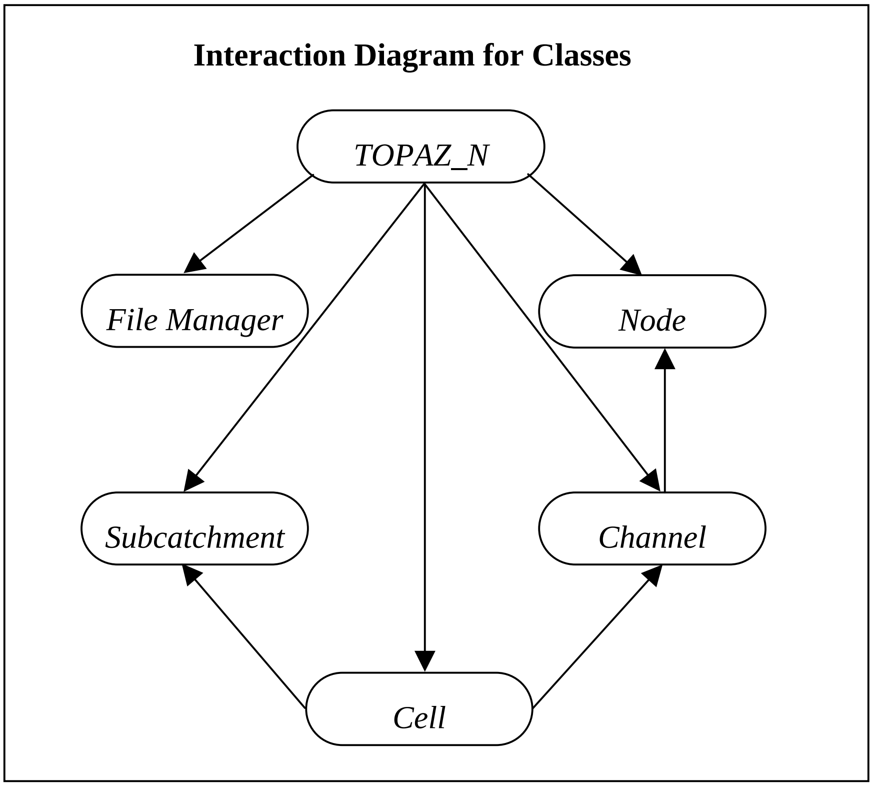


Fig 4.8- Interaction between different classes of TOPAZ-N

4.4 Process Model

TOPAZ-N extracts the final flow destination for each individual cell of the watershed. The Final flow destination for any given cell is the first channel cell into which it finally drains. DEDNM gives the out put of flow vector directions for each cell. Each cell is assigned a numeral value 1 through 9 excluding 5. These values are assigned based on flow direction derived from the D8 method. For instance a value of '9' for any given cell means it drains or flows into the lower right hand corner neighbor cell. As figure 4.9 shows the immediate flow destination (any of the 8 adjacent cells) and the final flow destination (the first channel cell into which any cell finally drains).

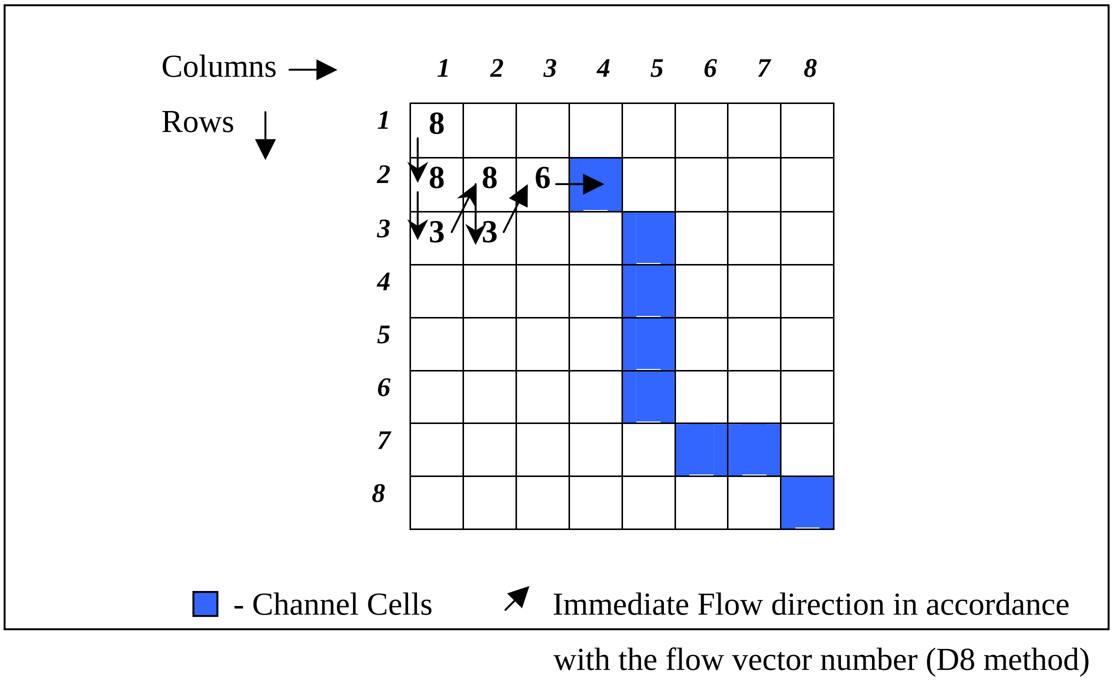

Columns
Rows
1 2 3 4 5 6 7 8
1 2 3 4 5 6 7 8
8
8
8
6
3
3
- Channel Cells
Immediate Flow direction in accordance with the flow vector number (D8 method)

Fig 4.9 – Final Flow direction for cell (1, 1) is the channel cell (4, 2). cell (1, 2), cell (1, 3), cell (2, 2), cell (2, 3) and cell (3, 2) all have the same final flow destination.

To arrive at the final flow destination (FFD) value, TOPAZ-N pre-processes the flow vector values as obtained from DEDNM out put. It extracts the FFD for each cell and then populates the data structure with these values. In this pre-processing, scanning begins with the first cell and control then moves to its immediate flow destination, a check is made if this destination is a channel cell or not. If it is not a channel cell then control moves to the immediate flow destination of this cell again the check is made, this process is continued till a channel cell destination is arrived. At that point all the previously passed cells are assigned the ID of this channel cell and are flagged. Then next cell is taken for scanning. This time a check is made to verify if this cell has been assigned FFD previously, if the cell is flagged from prior scanning it is skipped and control moves to the next cell performing the same check. This process is continued till all the cells are assigned FFD and are flagged off.

Addition of node by TOPAZ-N involved division of a subwatershed. It makes use of the final flow destination values of cells for this purpose. In the very first step after the user specifies the coordinates of the new node location, the control verifies if the given coordinate is that of a cell belonging to channel network. Once satisfied, a scan upstream of that location to the next node is done along the channel. All the channel cells between the new node location and its upstream node are identified and flagged. Then all the cells for that subwatershed (which is being divided) which have any of these flagged channel cells as their final flow destination are identified. A new subcatchment is created with ID same as the node number of the new node. The identified cells are assigned to that newly created subcatchment ID. Thus the old subcatchment gets divided.

In the watershed aggregation the control checks for the adjacency of the two subwatersheds being merged. It does so by validating the premise that in order to be hydrologic neighbors, one of the subwatersheds should flow into another or they both should have a common outlet. The control goes to that node location and makes a downstream scan along the channel to identify the downstream nodes for both the subcatchment, if one of them has the other as the downstream subcatchment or if they have the same outlet then they are considered neighbors and the merger is performed by re-assigning of indices and IDs.

CHAPTER-5

RESULTS AND DISCUSSION

5.1 Runs and Results

The test runs on TOPAZ-N were aimed at following two objectives:

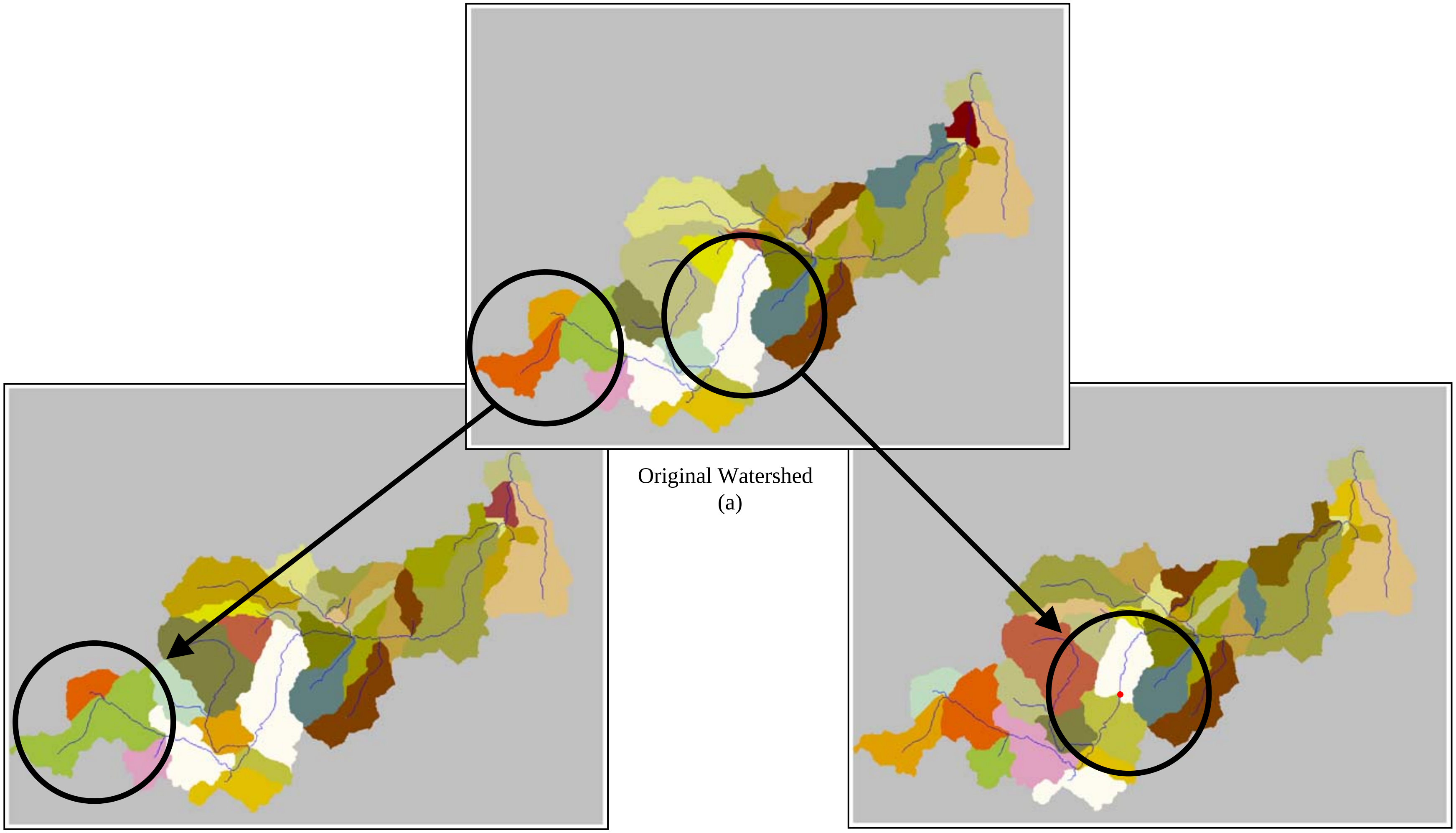
1) To evaluate the functional capabilities and data structure of the software.
2) To evaluate the hydrologic applications of these capabilities of the software.

Evaluation of functional capabilities and data structure meant to access if the basic functions for which the data structure and the software were designed are performed effectively. The second objective was to evaluate the hydrologic applications of these capabilities.

The first evaluation further looks at two different aspects of the data structure and functionality of the software, these were:

a) Add a node in the channel system and there by divide a subwatershed.
b) Merge two neighboring subwatersheds.

TOPAZ-N successfully inserts new node and merges of two subwatersheds as shown in figure 5.1.The second objective of the evaluation focuses on exhibiting and measuring up the hydrologic applications of TOPAZ-N. In hydrologic modeling it is desired to segment the watershed into units that are reasonably homogeneous with regard to hydrological response. With the functionality of node addition and merger of subwatersheds TOPAZ-N can initiate subwatersheds at points of specific interests and merge neighboring subwatersheds having common hydrologic responses. Another interest was to evaluate TOPAZ-N to arrive at a watershed segmentation based on the derived drainage direction.



Original Watershed
(a)

(b) Merger of two subwatersheds

(c) Insertion of new node & partitioning of a subwatershed

Figure 5.1- Simple Application of TOPAZ-N functionalities-(a) Original watershed delineated at CSA=200 and MSCL=400. (b) Watershed after merger of two subwatersheds. (c) Watershed after insertion of a new node and hence partitioning of subwatershed into two distinct subwatersheds.

In order to evaluate the capability of the software module to extract the water bodies in the parameterization process, we did runs on a set of five CSA values. The initial run was done at CSA of 50 which is comparable to the 30 m grid size of the parent DEM. It was then decided to do runs with 1, 2, 10 and 200 CSA values. The jump from 50 to 200 CSA value was done after evaluating the TOPAZ output for intermediate CSA values between 50 and 200, which revealed insignificant changes in the delineated network and subwatersheds from 50 to 200 CSA values. The number of nodes inserted at the intersection of streams and lake boundary almost remained constant over this range of CSA values. Within each CSA value, the MSCL values were varied from 100 to 400 in steps of 100 m. MSCL values below 100 did not show a significant effect on intersecting streams and lake boundary, hence 100 was adopted as the base value. The interval of 100 for MSCL values was chosen to get a fairly wide range.

As the CSA and MSCL values decrease the delineated network becomes increasingly dense. The denser the channel network becomes the more the number of intersection of channels with Lake Boundary. Thus denser networks give better extraction results. In general there is an overall trend of linear decrease in the number of nodes with increasing MSCL values for any given CSA value.

Table 5.1: The variation in the percentage of 'actual' lake in the delineated lake with MSCL and CSA value variation.

MSCL	**100**	**200**	**300**	**400**
CSA	**%**	**%**	**%**	**%**
1	94.68	82.28	75.91	63.42
2	67.68	66.25	58.39	52.66
10	30.53	30.53	24.16	20.64
50	12.25	10.43	10.43	10.43
200	9.221	9.221	9.221	9.221

In the table (Table 5.1) above, the column under each MSCL value category lists the percentage of the extracted lake as the 'actual' lake. The 'actual' lake refers to the lake area from the original (Shawn Francis's) digital data set. Table 5.1 was arrived at by an overlay analysis of the original data with the delineated lake DEM in Arc View. The data from the analysis was exported to an Excel file to calculate the percentage calculations vis-à-vis the actual lake area.

The percentage is representative of homogeneity of the extracted feature. A 100 percent indicates that all the area of the extracted feature is water; less than 100 percent reflects an inclusion of some non-water areas into the extracted feature.

Over all, the highest percent of homogeneity achieved was 94.68 percent at the CSA value of 1, and MSCL value of 100. For a given MSCL value, an increase in CSA value causes a sharp decrease in the homogeneity of the extracted feature. On the other hand, the variation in homogeneity for a given CSA value is not that sharp with increase in MSCL value.

The highest homogeneity of 94.68 percent means that 5.32 percent is the area which is not lake but directly drains into the lake, hence gets included into the extracted lake. This inclusion of area close to the shoreline of the lake is in line with the fact that Coal Lake resides in a Steep Sloped valley.

For a given CSA value, the percentage of 'actual lake' in the 'extracted lake' decreases, indicating inclusion of more of the surrounding area into the 'extracted lake'. The same is true when MSCL is kept constant and CSA values are changed (Figure 5.2).

Figure 5.3 to 5.7 show the delineation of Coal Lake at different CSA and MSCL values. It is observed that at low CSA value of 1 and MSCL value of 100 the delineated lake includes area which can be referred to as the riparian areas. This however cannot be said for higher CSA and MSCL values because the delineated lake includes vast areas beyond riparian zones. Thus TOPAZ-N would be a suitable tool to delineate water bodies with their riparian zones which drain directly into the lake without forming any significant channel links, but only at low CSA and MSCL values.

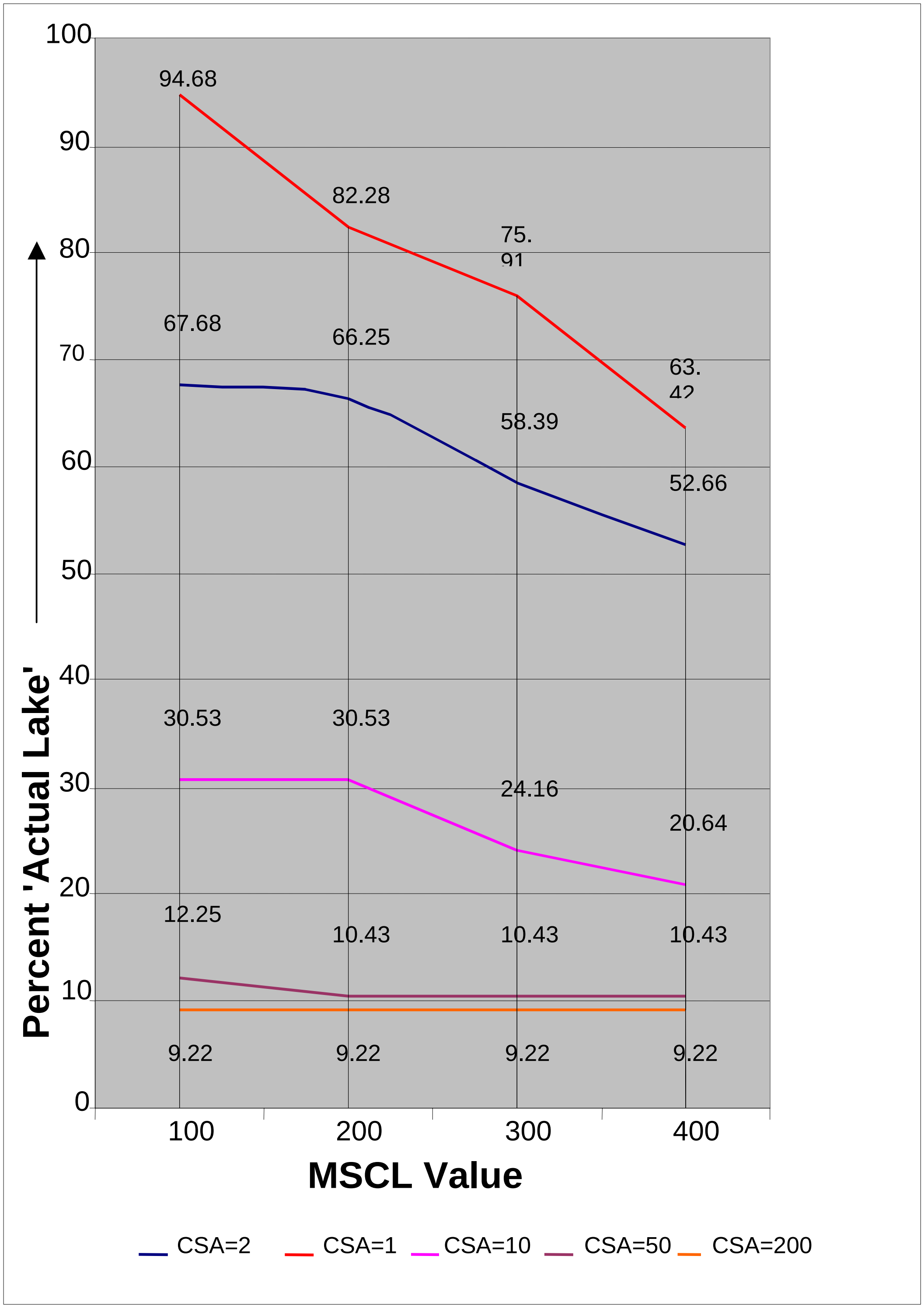


Figure 5.2: The graph showing the percent of homogeneity /percent of actual lake in the extracted feature as a function of MSCL value for different CSA values.

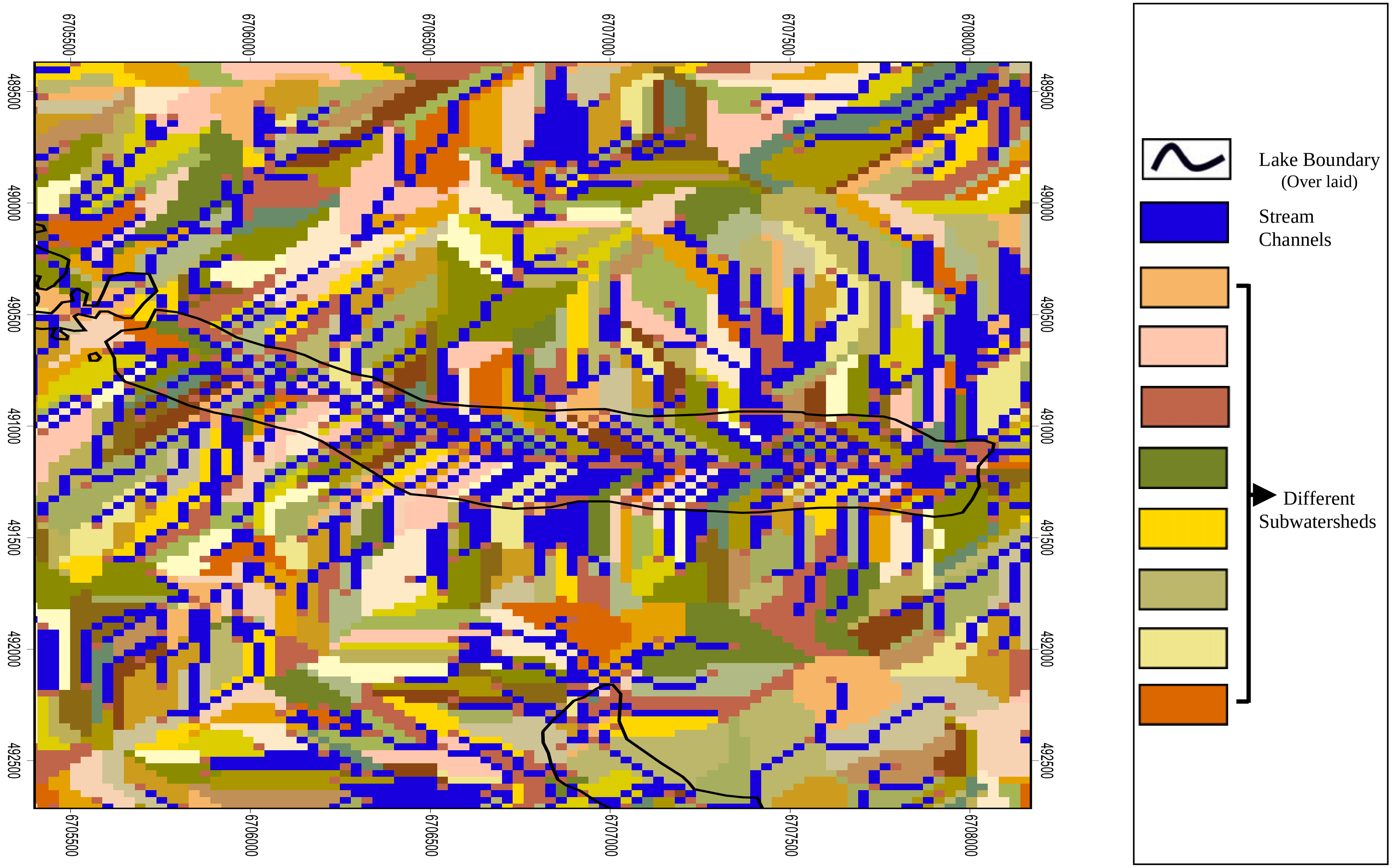


Figure 5.3 (a): The Overlay of lake boundary with the initial watershed prior to lake extraction at CSA=1, MSCL=100

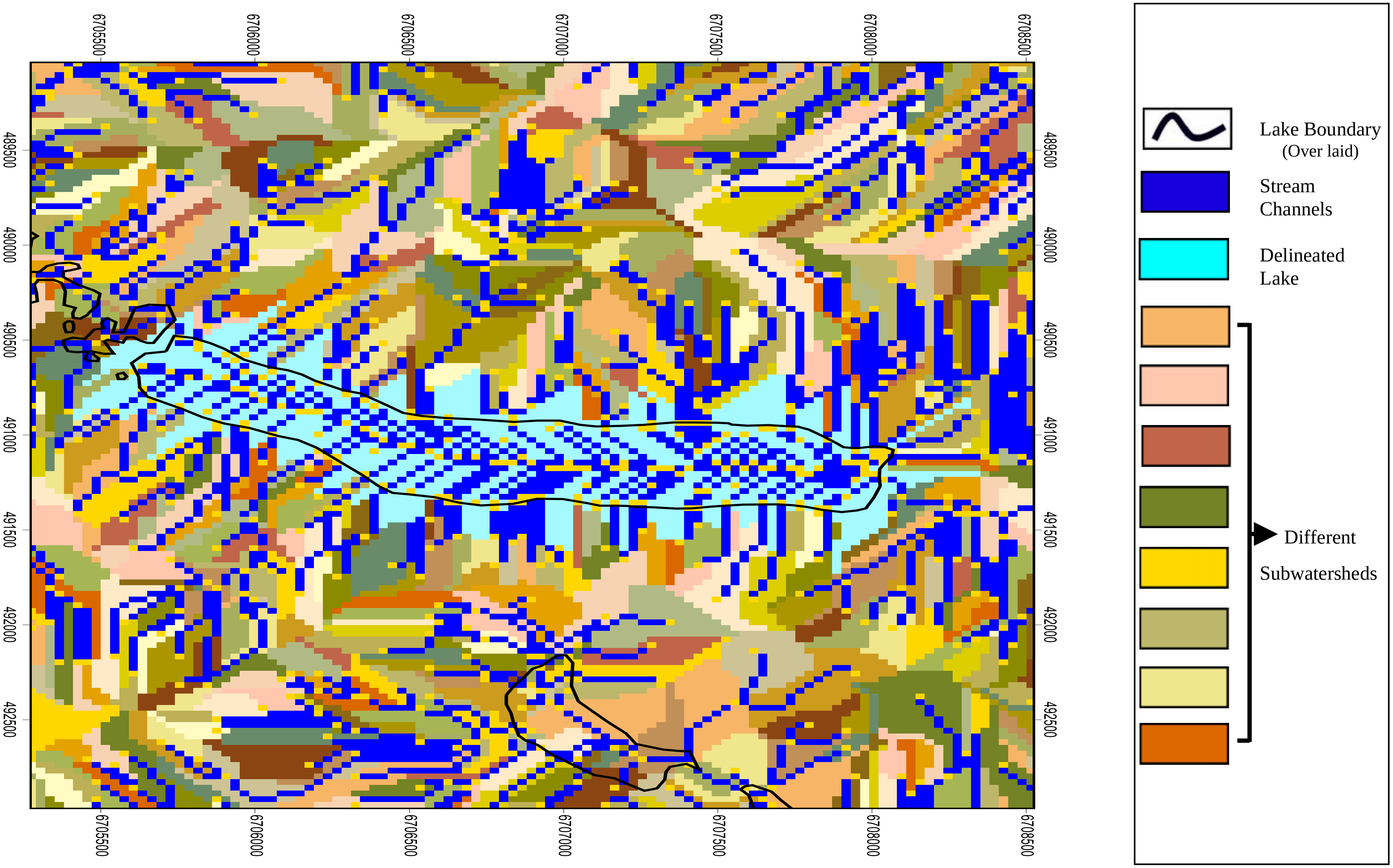


Figure 5.3 (b): The Overlay of lake boundary with the watershed after the lake extraction-CSA=1, MSCL=100-cyan-blue area is the extracted lake.

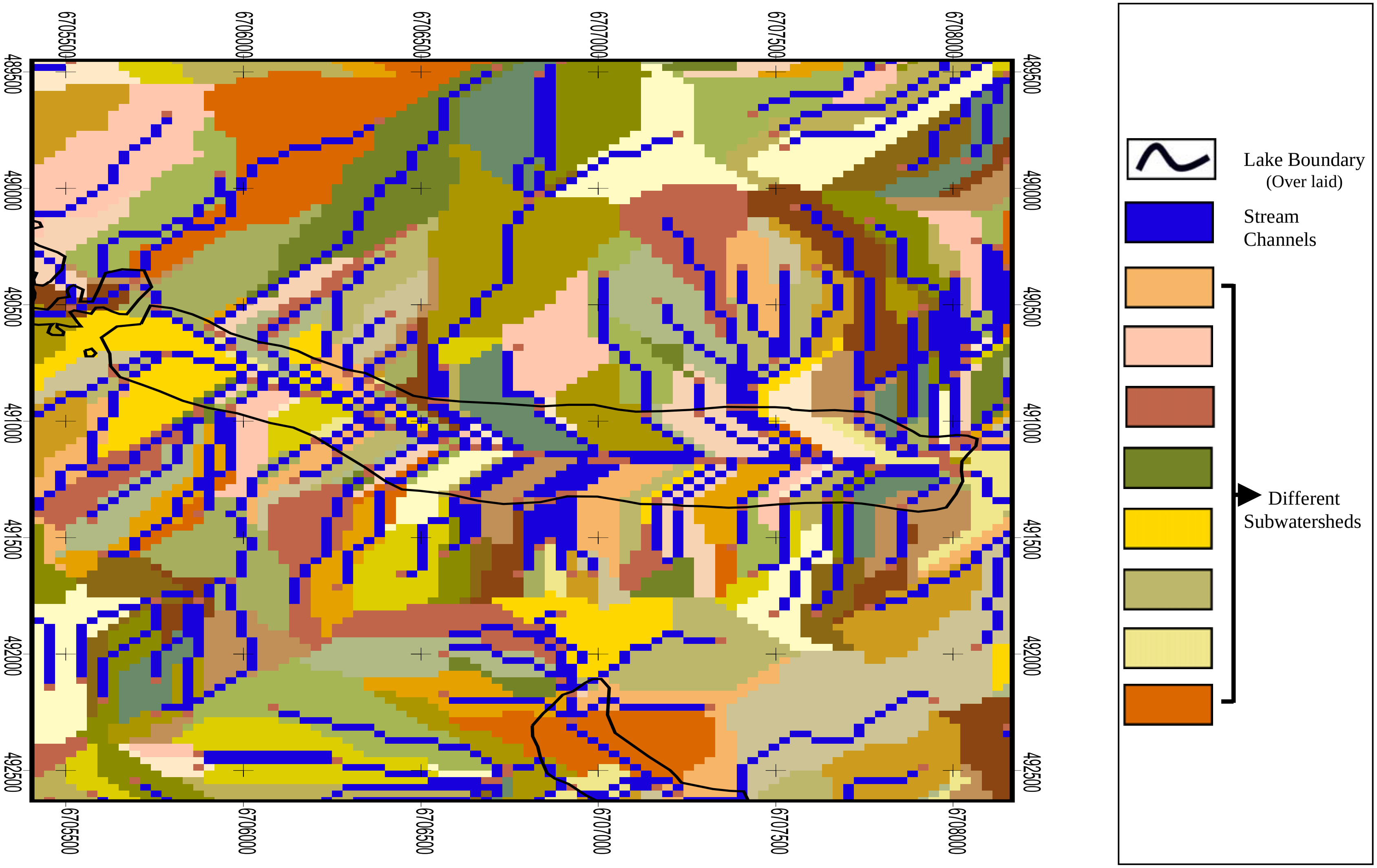


Figure 5.4 (a): The Overlay of lake boundary with the initial watershed prior to lake extraction-CSA=2, MSCL=100

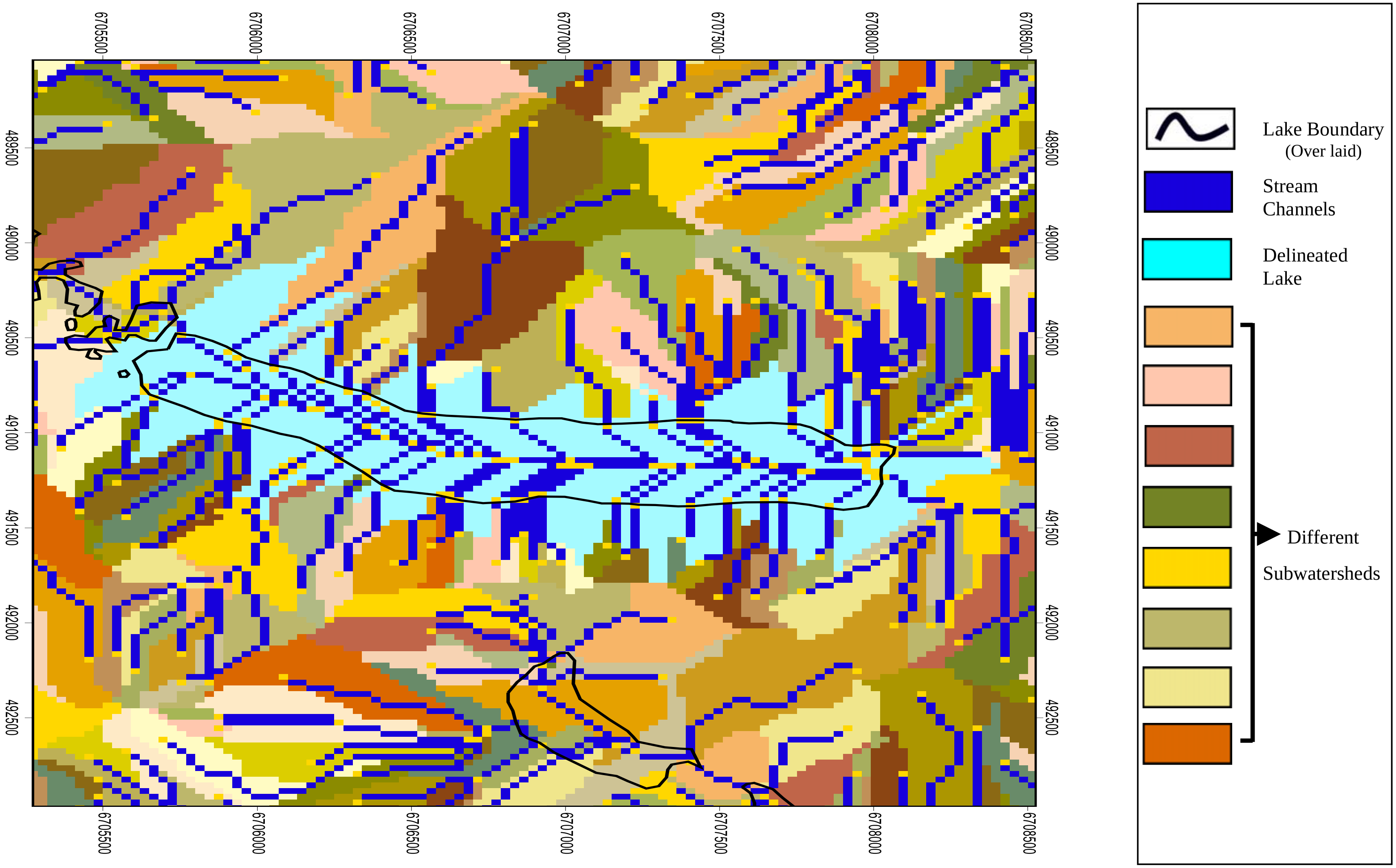


Figure 5.4 (b): The Overlay of lake boundary with the watershed after the lake extraction-CSA=2, MSCL=100-cyan-blue area is the extracted lake.

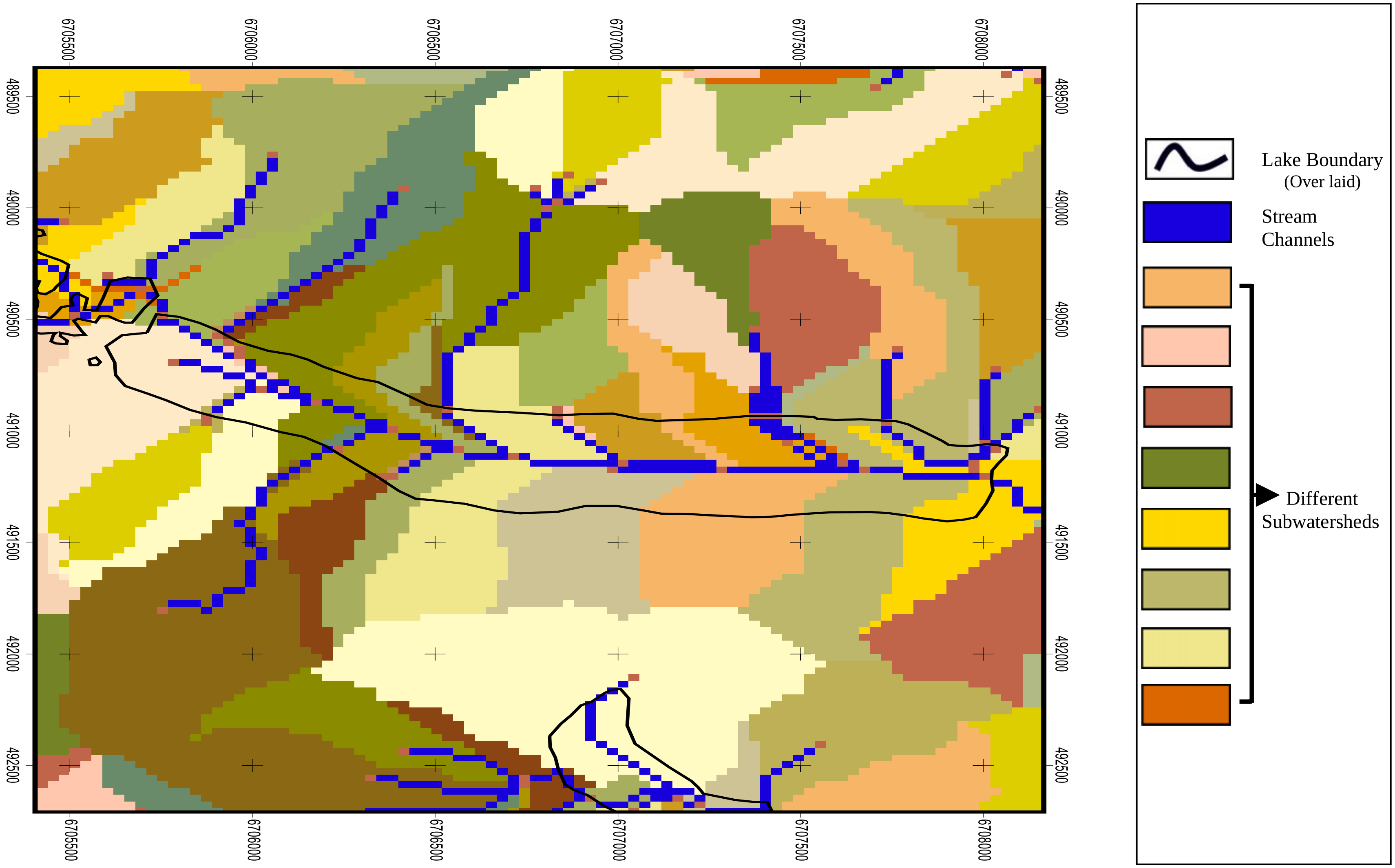


Figure 5.5 (a): The Overlay of lake boundary with the initial watershed prior to lake extraction-CSA=10, MSCL=100

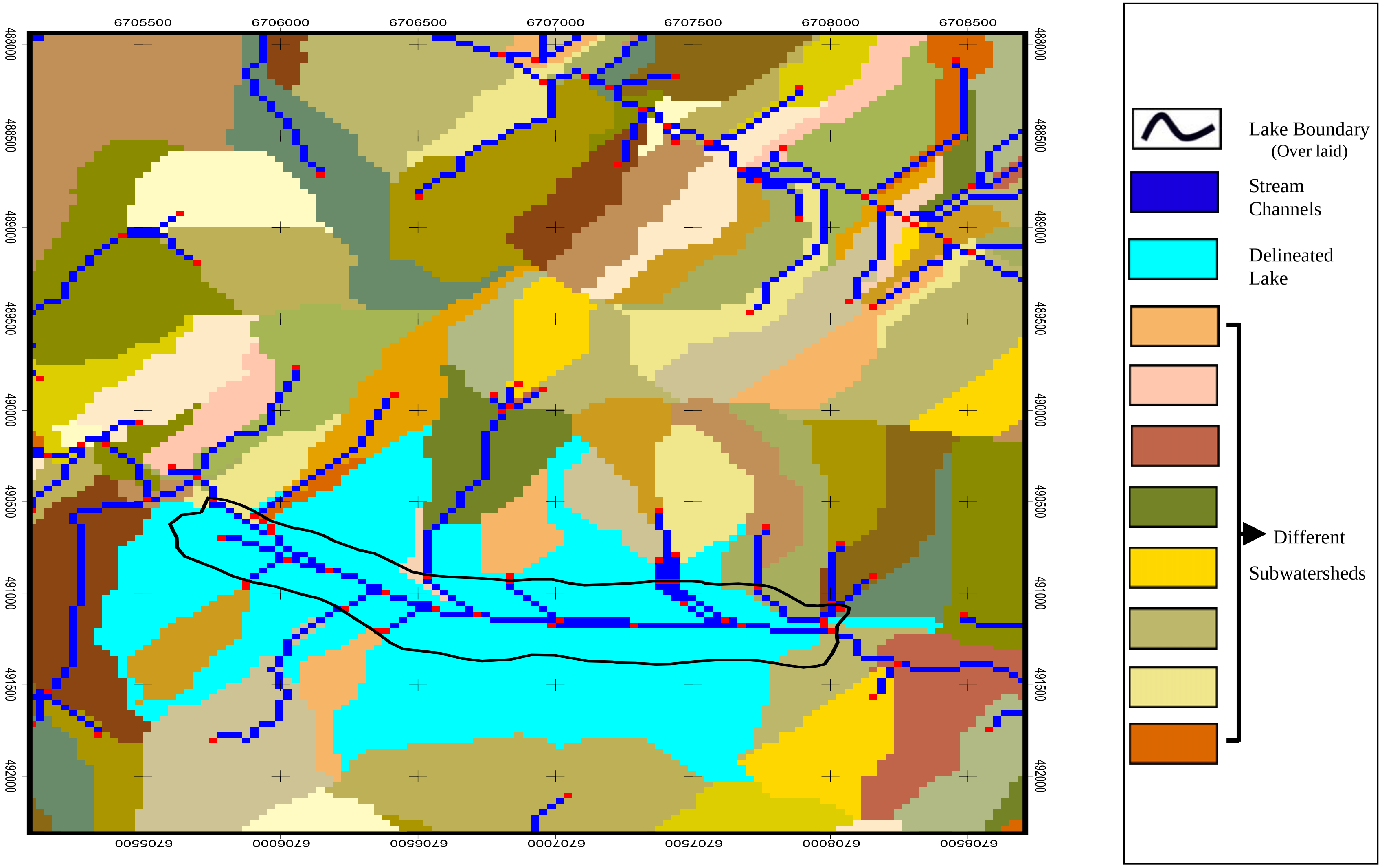


Figure 5.5 (b): The Overlay of lake boundary with the watershed after the lake extraction-CSA=10, MSCL=100-cyan-blue area is the extracted lake.

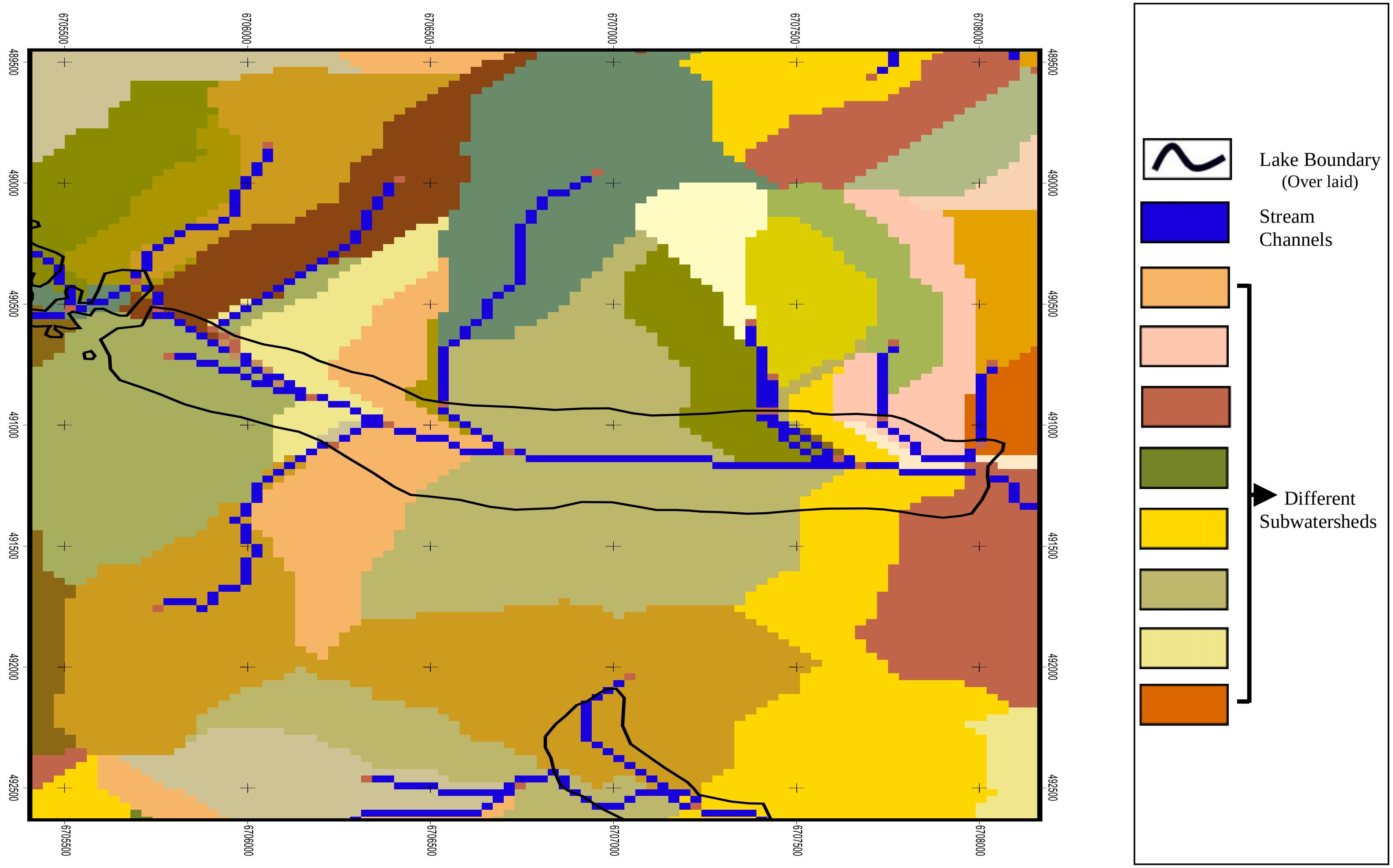


Figure 5.6 (a): The Overlay of lake boundary with the initial watershed prior to lake extraction-CSA=50, MSCL=100

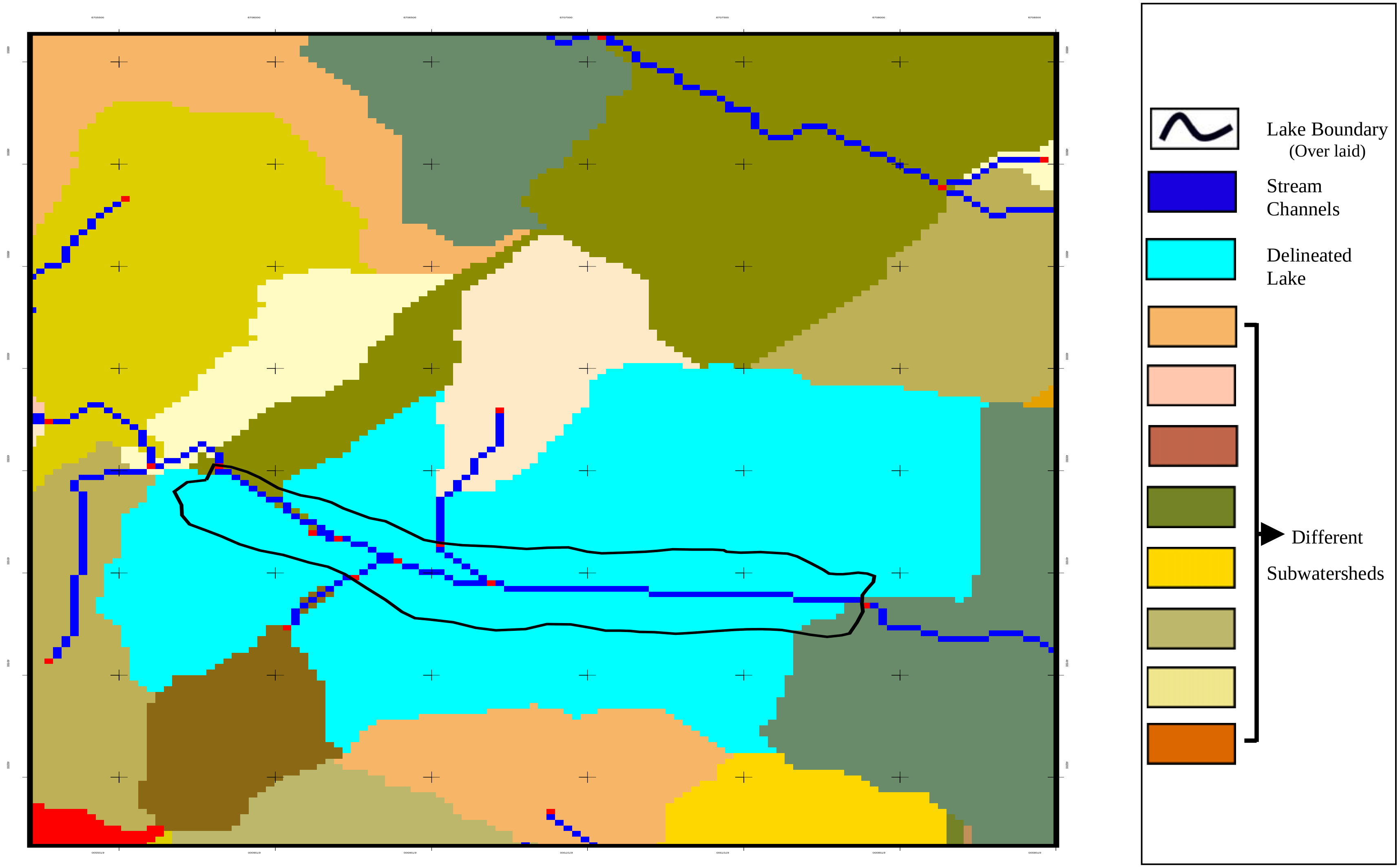


Figure 5.6 (b): The Overlay of lake boundary with the watershed after the lake extraction-CSA=50, MSCL=100-cyan-blue area is the extracted lake.

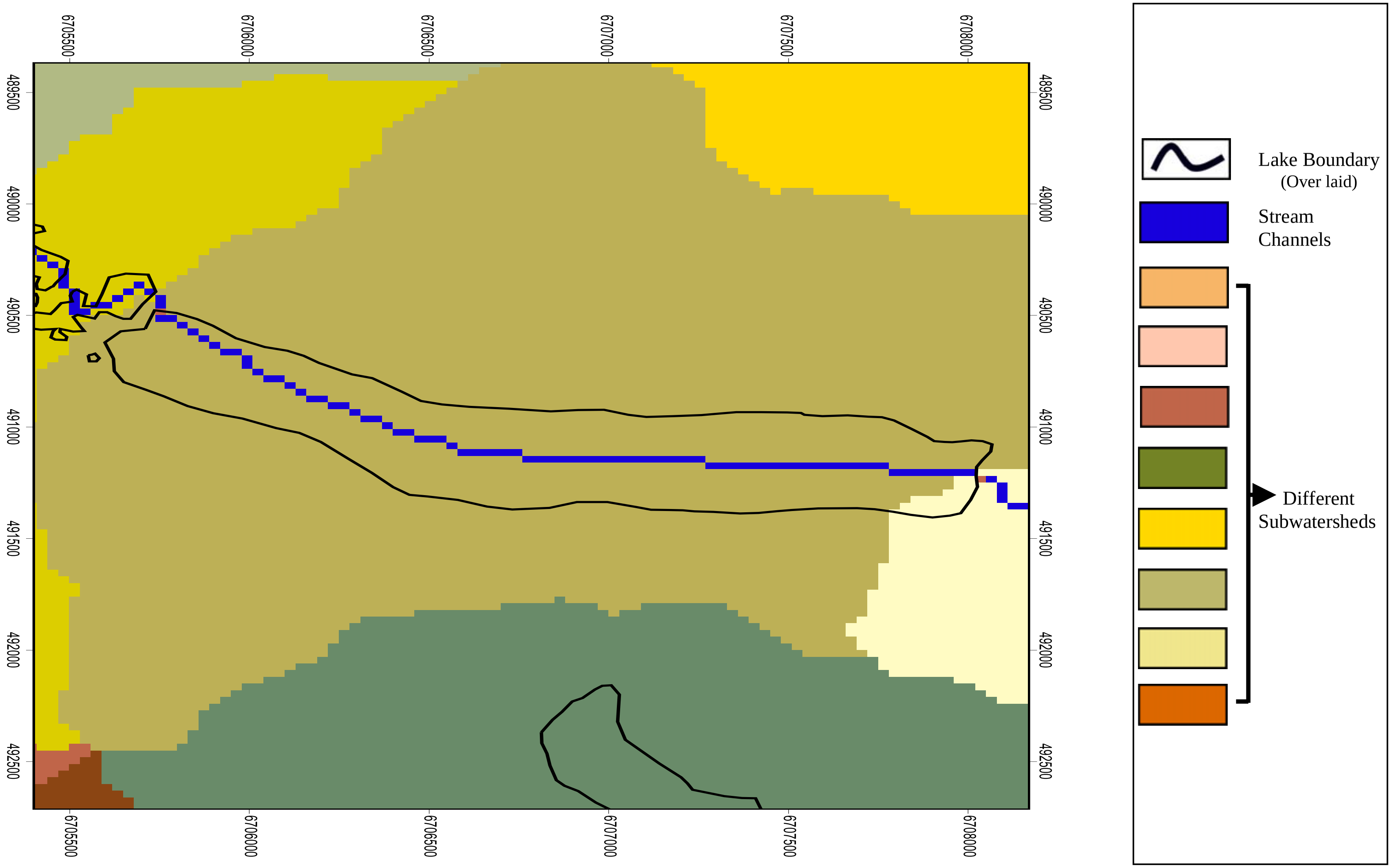


Figure 5.7 (a): The Overlay of lake boundary with the initial watershed prior to lake extraction-CSA=200, MSCL=100

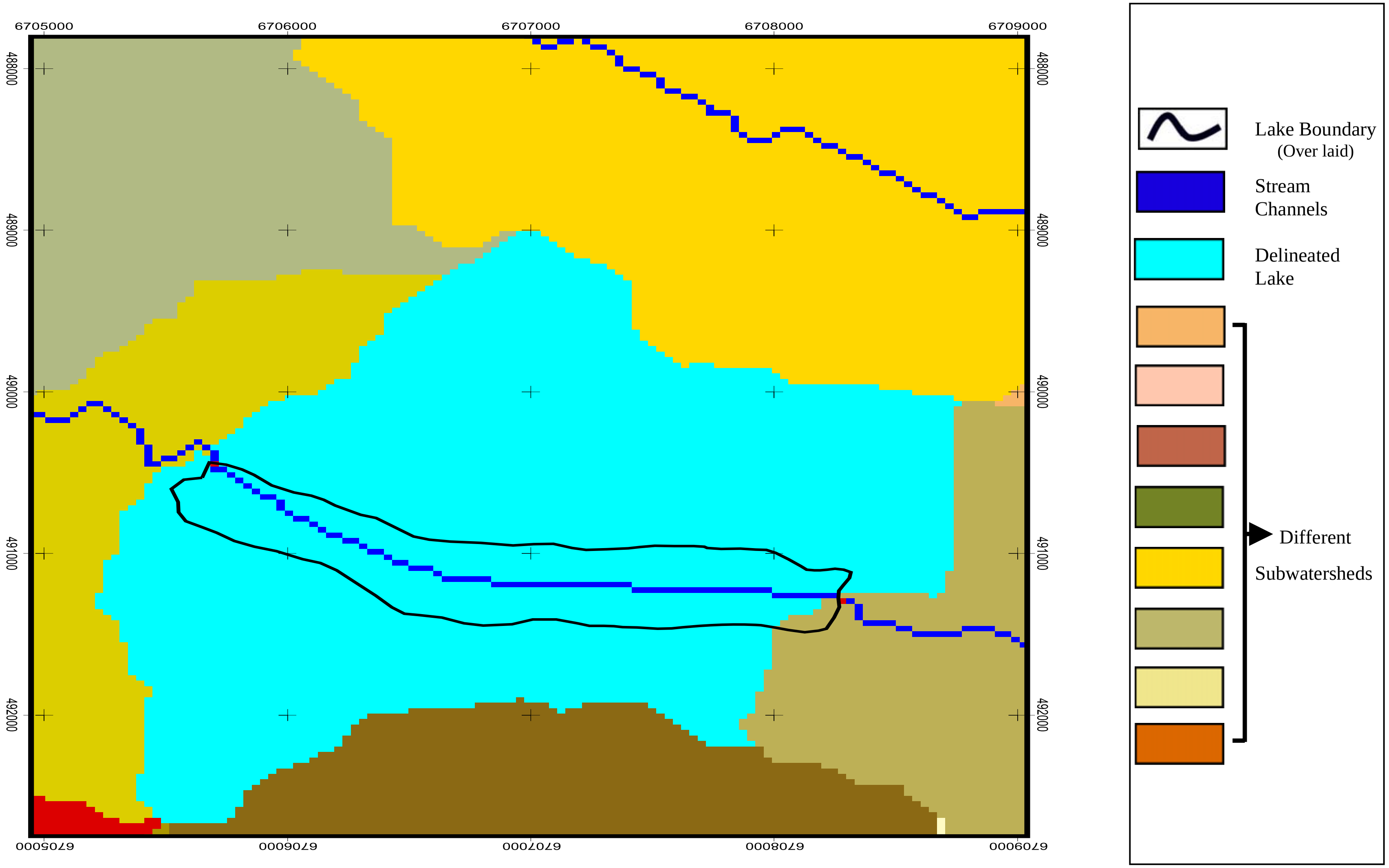
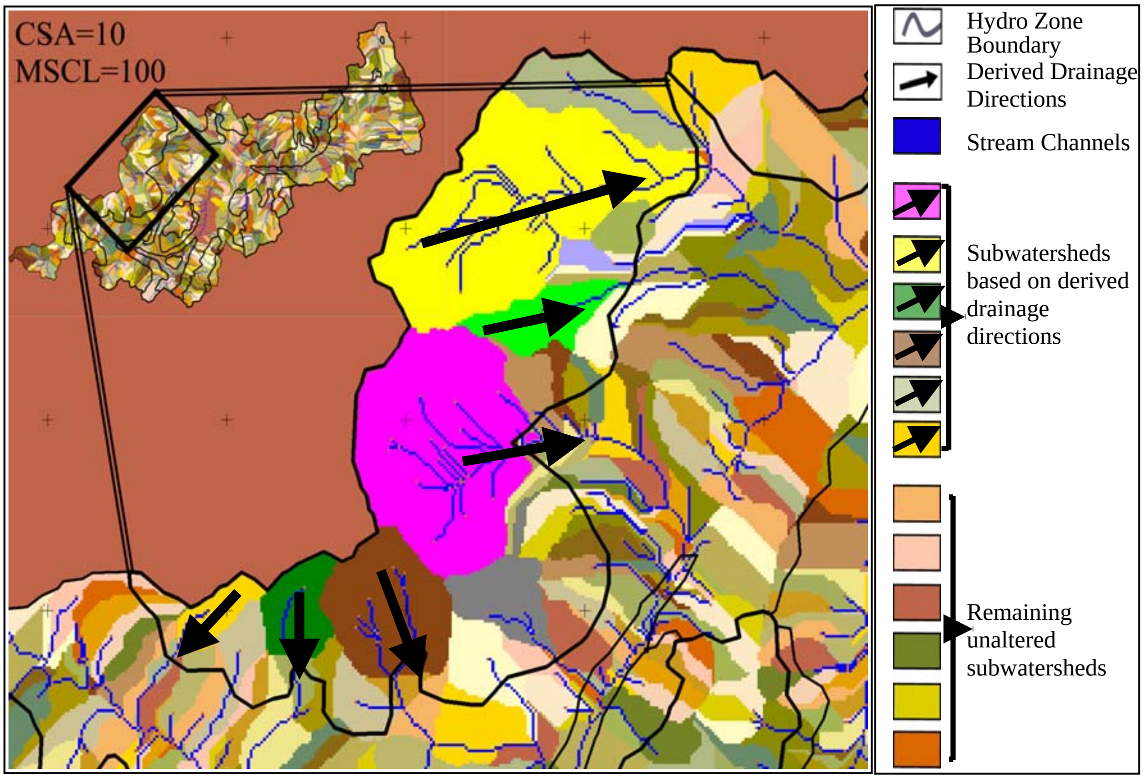


Figure 5.7 (b): The Overlay of lake boundary with the watershed after the lake extraction-CSA=200, MSCL=100-cyan-blue area is the extracted lake.

5.2 Application Runs

There have been attempts and comparative studies to arrive at watershed segmentation based on the derived drainage directions (Shaw, 2002). Derived drainage direction is defined as the one which emulates the hydrography. The derived drainage direction is the drainage direction of the final outlet for that subwatershed. The results demonstrate that TOPA-N can be effectively be used to generate watershed segmentation based on derived drainage direction as shown in figure 5.8.

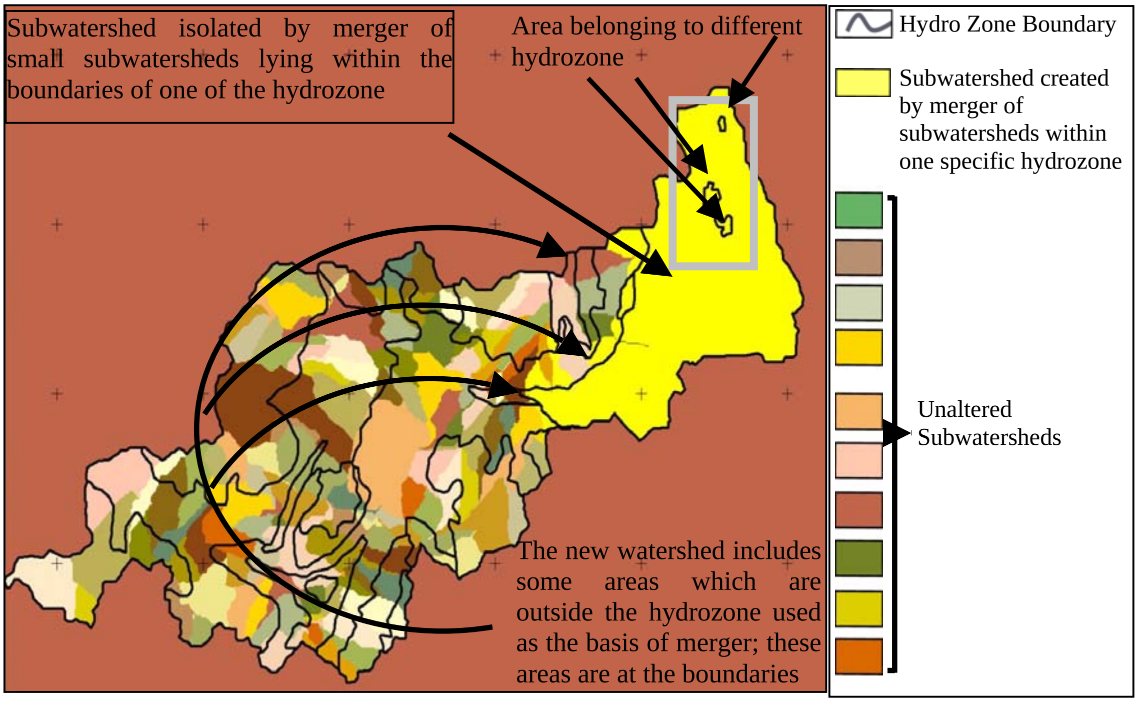


CSA=10
MSCL=100
Hydro Zone Boundary
Derived Drainage Directions
Stream Channels
Subwatersheds based on derived drainage directions
Remaining unaltered subwatersheds

Figure 5.8 – Segmentation of a section of watershed based on the

In another set of runs, an attempt was made to use the model to delineate subwatersheds based on some specific parameters. The specific parameter initially chosen was Ecozones. Thus, the runs were aimed at deriving subwatersheds that were fairly homogenous in terms of ecozones' characteristics. The first few runs revealed no pattern and the level of homogeneity was fairly low. This was due to the extremely complex pattern of the ecozones' mosaic which was used as an overlay. Thus, no further runs were done on this data set.

Next the ‘hydro-zones’ data was used to arrive at subwatersheds which were fairly homogenous in terms of the hydro-zones. Hydro zone is the classification of the watershed based on the hydrologic property. These hydrologic properties could be soil type, conductivity and porosity. Instead of hydro zones, simple vegetation based classification of the watershed could be another base for segmentation of watersheds. Again the results for hydro zone were the same as the ecozones’ runs. Both attempts, however, did isolate a few select subwatersheds with very high (almost 90-95%) accuracy with CSA of 1 and MSCL 100. However, this was not true for basin-wide application. For instance, in the Figure 5.9, one sees the outcome of the analysis that was performed on a DEM with CSA value of 10 and MSCL value of 100. A node was inserted at the intersection of the hydrozone boundary and the channel. Then all the subwatersheds from that node onwards, until the last node, which was in that hydrozone, were merged. The new watershed derived, however, has some areas outside the hydrozone. These ‘extra’ areas are downstream of the inserted node but do not completely fall in one hydrozone. It is not possible to initiate watersheds so that they can ‘mimic’ hydrozone boundaries. This is because the flow path may or may not conform to the hydrozone boundaries.


Subwatershed isolated by merger of small subwatersheds lying within the boundaries of one of the hydrozone

Area belonging to different hydrozone

The new watershed includes some areas which are outside the hydrozone used as the basis of merger; these areas are at the boundaries

Hydro Zone Boundary
Subwatershed created by merger of subwatersheds within one specific hydrozone
Unaltered Subwatersheds

Figure 5.9: Extraction of one subwatershed based on hydrozone overlay.

This is in agreement with lake extraction run in which the lake is treated as an isolated subwatershed and hence we get the same results as in ecozones and hydro-zones for isolated subwatersheds; i.e. 90-95 % homogeneity at low CSA values. The reason for a lack of any trend when the model was used for basin-wide application was due to that fact that the overlaying characteristics' layer has a mosaic which is different from that of the flow pattern mosaic. Any attempt to mold the watersheds based on overlay mosaic results in the conflicting flow patterns in the watershed.

The two zones in the inset (Figure 5.9) however do not belong to the Hydrozone under which they were aggregated. This is because the size of these zones is far too small as compared to the delineated subwatersheds in the underlying raster layer. The smallest feature to be extracted or delineated must be smaller than the smallest subwatershed extracted. Thus the CSA values should be low enough to accomplish this objective. If the subwatersheds delineated in the preliminary process are larger than the smallest feature in the overlying layer which is used for aggregation, then those smaller features shall not be extracted instead they will be aggregated in accordance with the neighboring larger entity. At lower CSA and MSCL values the delineated network becomes exceedingly dense. This impedes the computation of the data as the processing become extremely sluggish. The enormously large number of subwatersheds and the dense network makes the process for the identification of coordinates and I.D.'s for node addition and aggregation of subwatersheds respectively a very difficult task. Another key question for which answer is sought is so as to what percentage of homogeneity can be achieved across the basin. The answer to this question eludes the generalization. The level of homogeneity achieved across the basin is largely a factor of the complexity of the overlying layer. If the overlying layer is complex in geometry then the level of homogeneity achieved will vary across the basin from 100% for few subwatersheds to 50% for others. Not to forget the fact that this is also a function of CSA and MSCL values. However for simple overlying layer with the combination of low CSA and MSCL values 100% homogenous subwatershed can be achieved for over 90% of the entire basin. That is to say up to 90% of the subwatersheds in a basin can achieve 100% homogeneity for a simple overlying layer and low CSA and MSCL values. No generalization can be made in this regard.

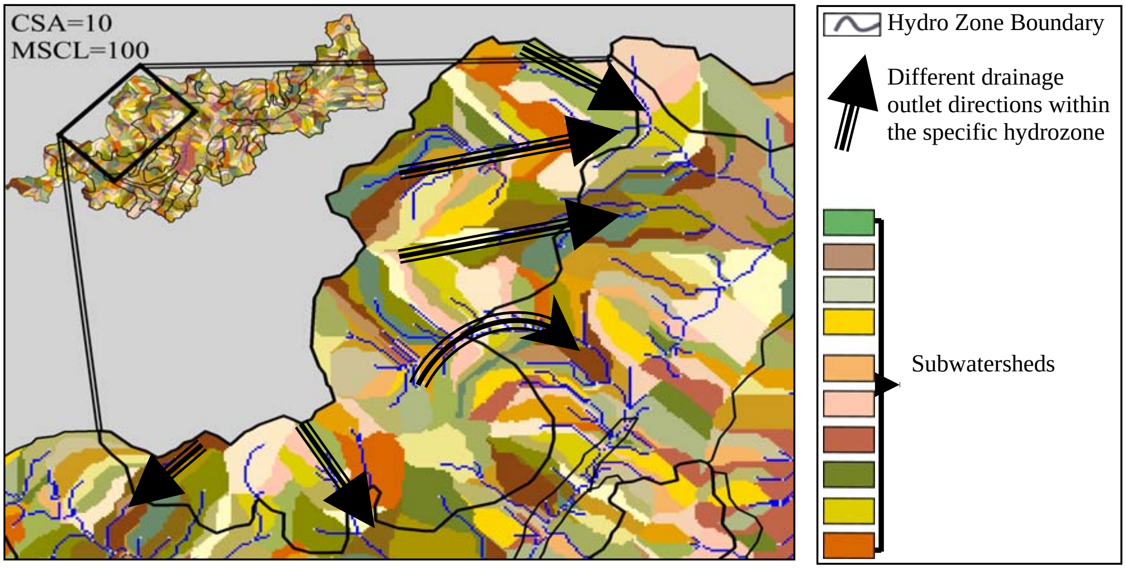

CSA=10
MSCL=100
Hydro Zone Boundary
Different drainage outlet directions within the specific hydrozone
Subwatersheds

Figure 5.10: One of the hydrozone for extracting homogenous subwatersheds.

Figure 5.10 shows the upper left portion of the Wolf Creek basin. This particular zone has flow in three distinctly different directions. I aggregated the subwatersheds for the main stream starting from right hand side as shown in Figure 5.11. Towards the lower left area of this zone the subwatersheds were having parallel flow hence could not be further aggregated.

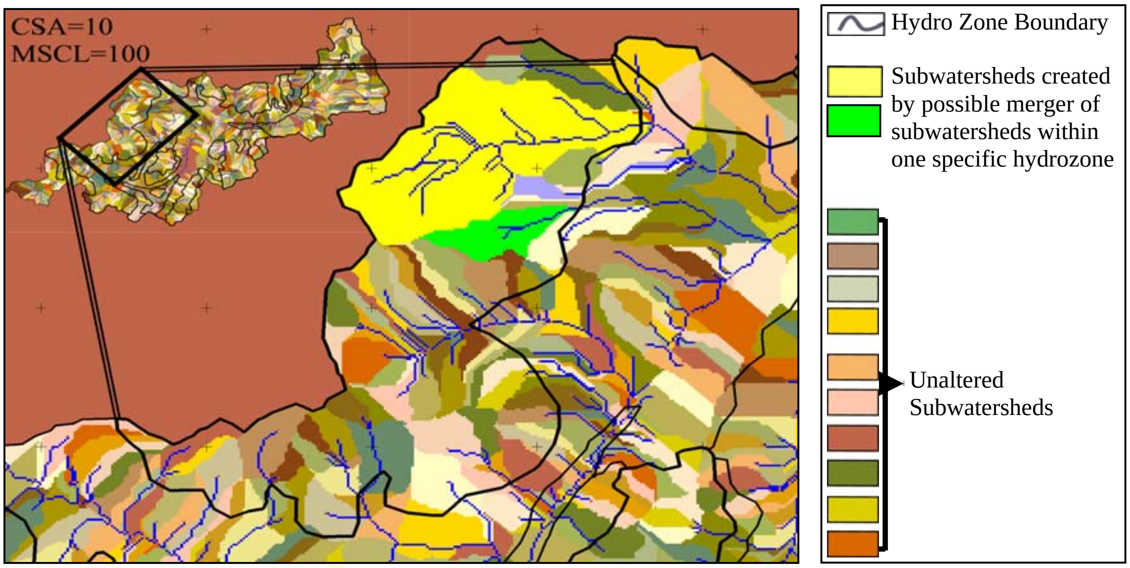

CSA=10
MSCL=100
Hydro Zone Boundary
Subwatersheds created by possible merger of subwatersheds within one specific hydrozone
Unaltered Subwatersheds

Figure 5.11: Aggregation of subwatersheds

The subwatersheds on the boundary of the overlying zone tend to have lesser degree of homogeneity as some of their areas fall in other zone. The solution to this problem is further fragmentation of these border subwatersheds then aggregating the smaller subwatersheds derived from this process in accordance with the boundary restriction. This would improve the homogeneity but some areas still would fall into two different zones, though very small areas. To do achieve this, MSCL should be low, which results in single cell subwatersheds and clustered network. This leads to the size of the clustered network more significant than their contributing areas. As shown in the following test runs, excluding the boundary areas result in more homogeneity. The decrease in the CSA value created smaller subwatersheds there by resulting in exclusion of smaller areas for improved homogeneity (Refer to Figure 5.12, 5.13 and 5.14).

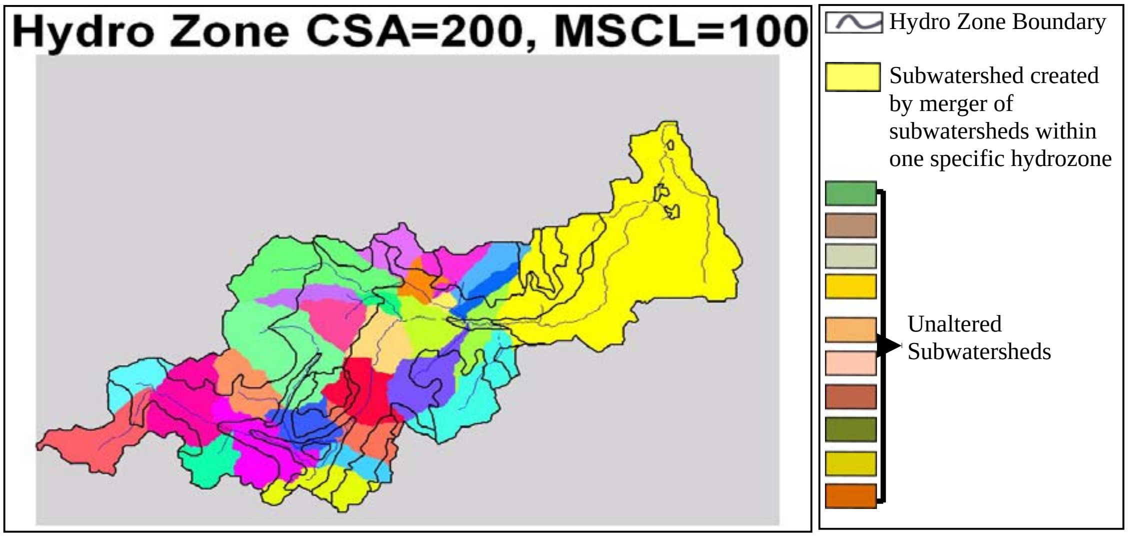

Hydro Zone CSA=200, MSCL=100
Hydro Zone Boundary
Subwatershed created by merger of subwatersheds within one specific hydrozone
Unaltered Subwatersheds

Figure 5.12: Aggregation at CSA=200 and MSCL=100 for extracting homogenous subwatersheds in accordance with hydrozone boundaries.

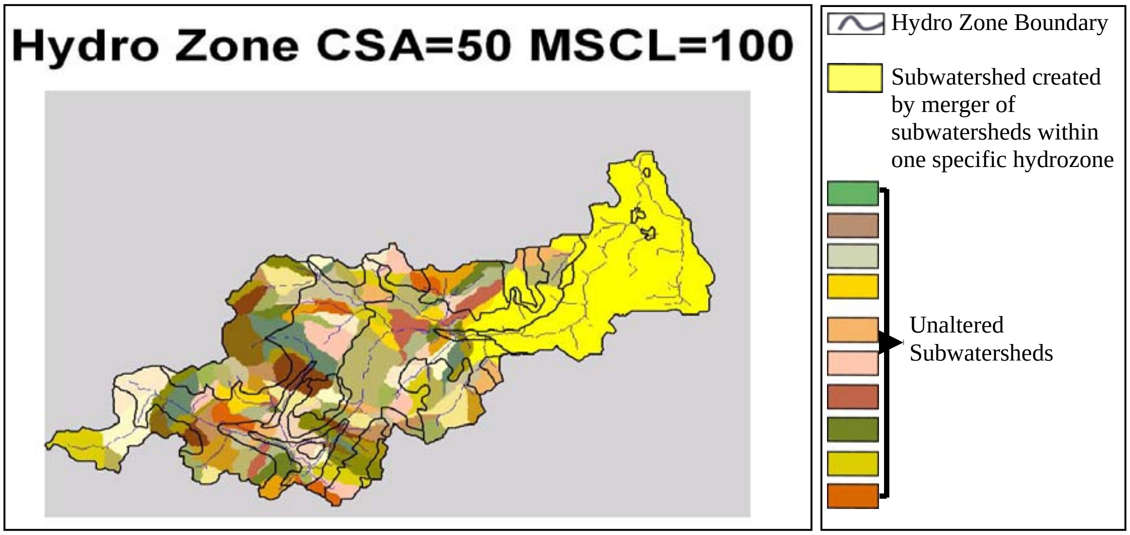

Hydro Zone CSA=50 MSCL=100
Hydro Zone Boundary
Subwatershed created by merger of subwatersheds within one specific hydrozone
Unaltered Subwatersheds

Figure 5.13: Aggregation at CSA=50 and MSCL=100 for extracting homogenous subwatersheds in accordance with hydrozone boundaries.

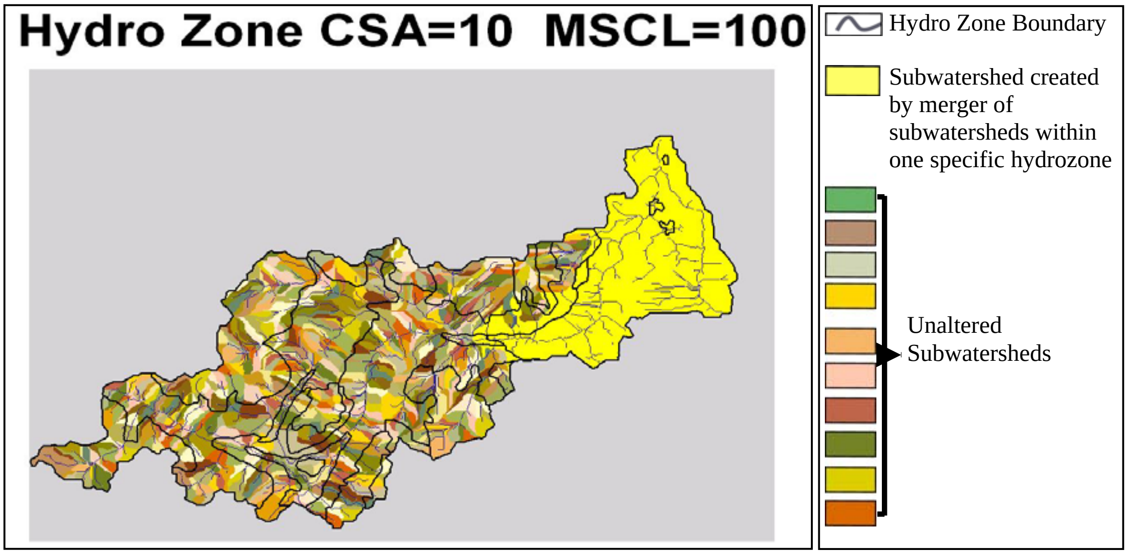

Hydro Zone CSA=10 MSCL=100
Hydro Zone Boundary
Subwatershed created by merger of subwatersheds within one specific hydrozone
Unaltered Subwatersheds

Figure 5.14: Aggregation at CSA=50 and MSCL=100 for extracting homogenous subwatersheds in accordance with hydrozone boundaries.

The following is the graphical representation (Figure 5.15) which shows the variation in homogeneity with CSA values for the runs (Figure 5.12, 5.13 and 5.14). The lower the CSA value, the better the homogeneity of the extracted watershed. Thus with TOPAZ-N, it is possible to extract features with high level of homogeneity. However same cannot be said for re-classification and aggregation of the entire subwatershed according to some overlying attribute layer, like hydrozone, soil type etc.

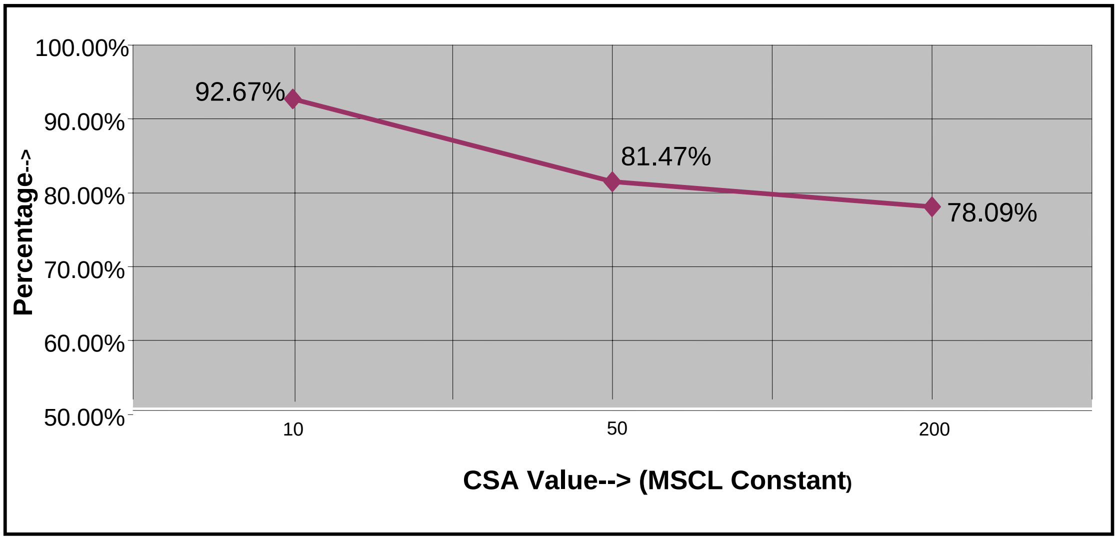


Figure 5.15– Graphical representation of variation of homogeneity of delineated subwatershed with CSA values.

CHAPTER-6

CONCLUSIONS

TOPAZ is a reliable software tool for automated delineation and parameterization in the raster-based modeling domain. The DEDNM module of TOPAZ provides the input for the TOPAZ-N software module. This ensured prevention of redundancy in the TOPAZ-N code by using the DEDNM processing capabilities.

The automated extraction of drainage network and subwatersheds has shortened the time involved in hydrological modeling without compromising the quality of results. TOPAZ-N allows for a higher level of modeling variability and capabilities. It also embarks on evaluating new and improved data modeling techniques for the raster-based hydrological data, and explores new data structure designs keeping in line with the special requirements of the spatial hydrological data. A new process model for this type of unique data was also put to evaluation under the TOPAZ-N module.

The results have indicated that features like lakes can be extracted and included in the raster-based hydrological modeling. They also have demonstrated that process models can be designed in such a fashion so as to retain the continuous nature of the raster data type while incorporating the ‘entity-manipulation’ type of discrete functionality in the raster-based hydrological modeling domain. Furthermore, results of test runs in an attempt to generate subwatersheds based on a certain attribute as a criterion have shown interesting results. They have demonstrated that although it is possible to generate a subwatershed with a high level of homogeneity within a basin, it is nevertheless highly unlikely to achieve the same level of homogeneity across the entire basin. This is due to the fact that the attribute criterion does not conform to the flow paths which are the primary criterion in the subwatershed initiation.

Also, the level of homogeneity is a function of CSA and MSCL values chosen. The smaller the CSA and MSCL values, the higher the degree of homogeneity. It is also worth noting that it is possible to extract highly homogenous subwatersheds. However, it is not possible to extract subwatersheds based on any complex overlay of any particular attributes (the boundaries cannot be mimicked in the subwatershed extraction), the flow path being the main criteria of the delineating process.

6.1 Limitations

TOPAZ-N depends on DEDNM module of TOPAZ and the INTER addendum of TOPAZ for its execution. Hence, it lacks independent execution. TOPAZ-N lacks its own graphical interface and overlay processing. Due to this, the identification of subwatersheds to be merged and the identification of coordinates for the node insertion makes use of external GIS solutions. Also, TOPAZ-N considerably slows down while processing DEM of a size larger than 1000 x 1000

6.2 Recommendations for Future Research

Further development of data models and data structures in the object-oriented domain is needed. Improvement of the Graphical User Interface (GUI) is also recommended, as it is not just a cosmetic feature, but rather has far reaching effects on the processing design. Development of TOPAZ-N is recommended to incorporate decision support system capabilities to the module. Inclusion of a ‘querying’ capability can also be included in the future research as one of the desired features. Research to establish standards for data model and process model design specifically for the hydrological domain is also recommended.

REFERENCES

Band, L. E., (1986). "Topographic Partition of Watersheds with Digital Elevation Models." Water Resources Research **Vol. 22**(NO.1): 15-24.

Band, L. E., (1989). "A terrain based watershed information system", Hydrol. Processes, Vol. 3: 151-162.

Band, L. E., and I.D.Moore, (1995). "Scale: Landscape Attributes and Geographical Information Systems." Hydrological Processes **Vol. 9**: 401-422.

Barney, Dennis, Process Definition Guidebook, January 1995.

Becker, A. and Serban, P. (1990). "Hydrological models for water resources system design and operation." WMO Operational Hydrological Report 34, World Meteorological Organization, Geneva.

Beven, K., (1985). "Distributed models. In: Anderson, M.G., Burt, T.P. (Eds)", Hydrological Forecasting. John Wiley & Sons, New York, pp. 405-435.

Bie, S.W. and Beckett, D.H.T., (1973). Comparison of four independent soil surveys by air-photo interpretation, Paphos area (Cyprus). Photogrammetria, 29, 189-202.

Blackie, J.R., Eeles, C.W.O., (1985). "Lumped catchment Models" In: Anderson, M.G.,Burt, T.P. (Eds), Hydrological Forecasting. John Wiley & Sons, New York, pp. 311-345

Blakemore, M. (1984). "Generalisation and error in spatial data bases" Cartographica 21, 131-9

Braun, P., T. Molnar, et al. (1997). "The Problem of Scaling in Grid-Related Hydrological Process Modelling." Hydrological Processes **Vol. 11**: 1219-1230.

Calkins, Hugh W., (1996). "Entity Relationship Modeling of Spatial Data for Geographic Information Systems", International Journal of Geographical Information Systems, January 1996.

Chorowicz, J., C. Ichoru, et al. (1992). "A Combined Algorithm for Automated Drainage Network Extraction." Water Resources Research **Vol. 28**(NO. 5): 1293-1302.

Clarke, R.T., (1973). "Mathematical Models in Hydrology." Food and Agriculture Organization of the United Nations, Irrigation and Drainage Paper No. 19, Rome

Coffman, D.M., Keller, E.A., and Melhorn, W.N., (1972). "New Topological relationship as an indicator of drainage network evolution", Water Resources Research, Vol. 8, 1497-1505

Costa-Cabral, M. C. and S. J., Burges, (1994). "Digital elevation model networks (DEMON): A model of flow over hillslopes for computation of contributing and dispersal areas." Water Resources Research **Vol. 30**(NO.6): 1681-1692.

Djokic , D. and Z. Ye, (1999). "DEM Preprocessing for Efficient Watershed Delineation." 1999 ESRI User Conference, San Diego, CA

Dooge, J.C.I., (1959), "A general theory of the unit hydrograph", J.Geophys. Res., 64(2)

Douglas, D.H.,(1986). "Experiments to locate ridges and channels to create a new type of digital elevation model." Cartographica, Vol. 23, No. 4, pp.29-61.

Doyle, F. J. (1978). "Digital Terrain Models: An Overview." Photogrammetric Engineering and Remote Sensing **Vol. 44**(No. 12): 1481-1485.

Fairchild, J., Leymarie, P., (1991). "Drainage networks from grid digital elevation models." Water Resources Research, Vol. 27(4),29-61

Fleming, G., (1975). "Computer Simulation techniques in hydrology" American Elsevier Publishing Co., New York 333 pp.

Fleming, G., (1979). "Deterministic Model in Hydrology." Food and Agriculture Organization (FAO) of the United Nations, Rome, 1-80

Francis, S., (1997). "Data integration and ecological stratification of Wolf Creek watershed, south-central Yukon. Report prepared for Indian and Northern Affairs Canada and Agriculture Canada." Applied Ecosystem Management Ltd., Whitehorse, Yukon, 23 pp

Franklin, W.R., (1984). "Cartographic errors symptomatic of underlying algebra problems." In Proc. Int. Symp. on Spatial Data handling, 20-24 Aug., Zurich, Switzerland, pp. 190-208

Freeman, H. (1974). Computer processing of line-drawing images. Computer Survey

G.C. Grender (1976). "TOPO III: A Fortran Program for Terrain Analysis." Computer and Geoscience **Vol. 2**: 195-209.

Garbrecht, J. and L. Martz (1992). "Numerical definition of drainage networks and subcatchment Areas from Digital Elevation Models." Computer and Geosciences, 18(6) 747-761, July 1992

Garbrecht, J. and L. Martz, (1993). "Case application of the automated extraction of drainage network and subwatershed characteristics from digital elevation models by DEDNM" In Harlin, J.M., Lanfear, K.J. (Eds), Proceedings of the Symposium on Geographic Information Systems and Water Resources, in Mobile, Alabama. American Water Resource Association, Bathesda, MD, 221-229

Garbrecht, J. and L. Martz (1994). "Grid Size Dependency of Parameters Extracted from Digital Elevation Models." Computer and Geoscience **Vol. 20**(No. 1): 85-87.

Garbrecht, J. and L. Martz (1996). "Comment on "Digital elevation model grid size, landscape representation, and hydrologic simulations" by Weihua Zang and David R. Montgomery." Water Resources Research **Vol. 32**(NO. 5): 1461-1462.

Garbrecht, J., Starks, P.J., Martz, L.W., (1996). "New digital landscape parameterization methodologies." In: Proceedings of the Symposium on GIS and Water Resources, American Water Resources Association, September 22-26, 1996, Fort Lauderdale, Florida, p. 357.

Gruenberger, F. (1984). Computer recreations . Scient. Am. 250(4), 10-14(April 1984)

Gyasi-Agyei, Y, Willgoose, G and de Troch, F P, (1995). "Effects of vertical resolution and map scale of digital elevation models on geomorphological parameters used in hydrology", Hydrological Process., **Vol. 9**, 363-382.

Holroyd, F. and S. Bell (1992). "Raster GIS: models of raster encoding." Computer and Geoscience **Vol. 18**(No. 4): 419-426.

Hardisty, J., Taylor, D.M. Metcalfe, S.E., (1993). Computation environmental modelling: a practical introduction using Excel. John Wiley & Sons, Toronto, 204 pp

Hutchinson, M. F. (1989). "A New Procedure for Gridding Elevation and Stream Line Data with Automatic Removal of Spurious Pits." Journal of Hydrology **106**: 211-232.

Ichoku, C. and J. Chorowicz (1994). "A numerical approach to the analysis and classification of channel network patterns." Water Resources Research **Vol. 30**(NO. 2): 161-174.

Ichoku, C., E. M. O'loughlin, et al. (1988). "A Contour -Based Topographic Model for Hydrological and Ecological Applications." Earth Surface Processes and Landforms **Vol. 13**: 305-320.

Jarvis, R.S., (1977). "Drainage Network Analysis", Progress in Physical Geography, v.1, 271-295.

Jenson, S.K., (1985). "Automated derivation of hydrologic basin characteristics from digital elevation data" Proc. Auto-Carto Digital Representations of Spatial Knowledge. Washington, D.C. p .310-310

Jenson, S.K., Trautwein, C.M., (1987). "Methods and applications in surface depression analysis." Proceedings of Auto-Carto, 8,137–144.

Jenson, S. K. and J. O. Domingue (1988). "Extracting Topographic Structure from digital Elevation Data for Geographic Information System Analysis." Photogrammetric Engineering and Remote Sensing **Vol. 54**(No. 11): 1593-1600.

Johnson, D. L. and A. C. Miller (1998). "A spatially distributed hydrologic model utilizing raster data structures." Computer and Geoscience.

Jones, N. L., S. G. Wright, et al. (1990). "Watershed Delineation with Triangle-Based Terrain Models." `Journal of Hydraulic Engineering **Vol. 116**(No. 10): 1232-1251.

Kessler, B.L. (1992). "Glossary of GIS terms." Journal of Forestry 90(11): 37-45

Kikuchi, L., J. A. Guevara, Mark, D. M., and D. F. Marble, (1982). "Rapid display of digital elevation models in a mini-computer environment. " Proceedings, ISPRS Commission IV Symposium 1982 (Auto-Carto V), Crystal City Virginia, August 22-28, pp. 297-307.

Kite, G.W., (1994). "Hydrologically modelling using remotely sensed data and geographic information systems." Trends in Hydrology ,1, 191-208

Kirkby, M.J., Naden, P.S. Burt, T.P., Butcher, D.P., 1993. "Computer Simulations in Physical Geography", 2nd edition. John Wiley & Sons, Toronto, 180 pp

Kouwen, N. et al., Grouping Response Units for Distributed Hydrologic Modelling, ASCE J. of Water Resources Management and Planning, 119(3), pp 289-305, 1993.

Lammers, R. B., and L. E. Band, (1990). "Automating object representation of drainage basins" , Computers and Geosciences, 16(6), 787–810, 1990.

Lee,D.T. & B.J. Schacter, "Two Algorithims for constructing a Delauney triangulation" International Journal of Computer and Information Sciences ,9(3) June 1980 , 219-242

Lee, J., Peter K. Snyder, et al. (1992). "Modeling the Effect of Data Errors on Feature Extraction from digital elevation Models." Photogrammetric Engineering and Remote Sensing **Vol. 58**(No. 10): 1461-1467.

Linsey, R.K. Jr., Kohler, M.A., Paulhus, J.L.H., (1986). Hydrology for Engineers, 3rd Editon. McGraw-Hill Book Company, Toronto, 508 pp.

Maguire, D., (1992). "The raster GIS design model -- a profile of ERDAS." Computer and Geoscience" **Vol. 18**(No. 4): 463-470.

Mark, D. M., (1983). Automated detection of drainage networks from digital elevation models. Proceedings, Sixth International Symposium on Automated Cartography (Auto-Carto VI), Volume 2, pp. 288-295

Marks, D., Dozier, J. and Frew, J., 1984. "Automated basin delineation from digital elavation data." Geo-Processing, 2, 299-311.

Martin, James and Clive Finkelstein. (1981). "Information Engineering", Technical Report, two volumes, Lancs, UK: Savant Institute, Carnforth.

Martz, L.W. and J. Garbrecht (1992) "Numerical Definition of drainage network and Subcatchment Areas from Digital Elevation Models." Computers and Geosciences,18(6):747-761, July 1992

Martz, L. W. and J. Garbrecht (1993). "Automated Extraction of Drainage Networks and Watershed Data From Digital Elevation Models." Water Resources Bulletin **Vol. 29**(No. 6): 901-908.

Martz, L. W. and J. Garbrecht, (1993). " DEDNM: A software system for the automated extraction of channel network and watershed data from raster digital elevation models." American Water Resources Association Symposium Proceedings "Geographic Information Systems and Water Resources".

Martz, L. W. and J. Garbrecht (1994). "Grid Sized Dependency of Parameters extracted from Digital Elavation Models". Computers and GeoSciences,20(1):85-87,1994

Martz, L. W. and J. Garbrecht (1995). comment on "Automated recognition of Valley lines and Drainage Networks from Grid Digital Elavation Models: A Review and a New Method" by A. Tribe . Journal of Hydrology 167(1):393-396,1995.

Martz, L. W. and J. Garbrecht, (1998). "The treatment of flat areas and depressions in automated drainage analysis of raster digital elevation models." Hydrological Processes **Vol. 12**: 843-855.

Maunder, J.C., (1999). "A automated method for constructing contour-based digital elevation models." Water Resources Research, Vol. 35(12), 3931-3940

Mayer, Richard J., et al., IDEF Family of Methods for Concurrent Engineering and Business Re-engineering Applications, Knowledge-Based Systems, Inc., 1992.

McCormack, J. E. and J. Hogg, (1996). "Virtual-memory tiling for spatial data handling in GIS."

Montgomery, D. R. and E. Foufoula-Georgiou, (1993), "Channel network source representation using digital elevation models," Water Resources Research, 29(12): 3925-3934.

Moore, I. D. and R. B. Grayson, (1991). "Terrain Based Catchment Partitioning and Runoff Prediction Using Vector Elevation Data." Water Resources Research **Vol. 27**(NO. 6): 1177-1191.

Moore, I.D., Grayson, R.B. and Ladson, A.R., (1991). "Digital Terrain modelling: A review of hydrological, geomorphological, and biological applications" Hydrological Processes, 5, 3-30

Morisawa, M. E., (1959). "Relation of quantitative geomorphology to stream flow in representative watersheds of the Appalachian Plateau Province", Tech. Rep. 20 , Department of Geology, Columbia Univ.,New York.

Nash J. E., (1957), "The form of the instantaneous unit hydrograph", Int. Assoc. Hydrol. Sci., Proc. of the Toronto General Assembly, publication, 45 (3-4), 114-121, 1957.

Nelson, E. J., N. L. Jones, et al., (1994). "Algorithm for Precise Drainage-Basin Delineation." Journal of Hydraulic Engineering **Vol. 120**(No. 3): 298-312.

O'Callaghan, J.F., Mark, D.M., (1984). The extraction of drainage networks from digital elavation data. Computer Vision, Graphic and Image Processing,4(3),375-387

Papo, H. B. and Gelbman, E. (1984). "Digital terrain models for slopes and curvatures." Photogrammetric Engineering and Remote Sensing, 50: 695-701.

Peddle, D. R. and S. E. Franklin (1991). "Image Texture Processing and Data Integration for Surface Pattern Descrimination." Photogrammetric Engineering and Remote Sensing **Vol. 57**(No. 4): 413-420.

Peucker, T.K.; Douglas, D.H. (1975) "Detection of surface specific points by local parallel processing of discrete terrain elevation data," Computer Graphics and Image Processing: vol. 4; pp 375-387.

Quinn, P., K. Beven, et al. (1991). "The Prediction of Hillslope Flow Paths for Distributed Hydrological Modelling using Digital Terrain Models." Hydrological Processes **Vol. 5**: 59-79.

Riazanoff, S., B. Cervelle, et al. (1988). "Ridge and valley line extraction from digital terrain models."

S. Riazanoff , B. Cervelle , J. Chorowicz (1990),” Parametrisable skeletonization of binary and multi-level images”, Pattern Recognition Letters, vol 11 n.1, p.25-33, January 1

Salome, A.I., van Dorsser, H.J., and Rieff, Ph.l. (1982). "A comparison of geomorphological mapping systems ". ITC J. 272-4

Saunders, W. K. (1999) "Preparation of DEMS for use in Environmental Modeling Analysis" In: Conference Proceedings: 1999 ESRI User Conference

Schetselaar, E., P. v. Dijk, et al. (1990). "Digital image processing of geophysical data using a raster based GIS." ITC Journal **Vol. 3**: 248-252.

Shaw, D.A., (2002), Thesis: "Preserving segment variability for flow routing in distributed hydrological models - An automated method” -Thesis (M.Sc.)--University of Saskatchewan, 2002

Singh, V.P, (1988). "Hydrologic Systems, Volume I: Rainfall-Runoff Modeling." Prentice-Hall Inc., Englewood Cliffs, New Jersey, 480 pp

Singh, V.P, (1989). "Hydrologic Systems, Volume II: Watershed Modeling." Prentice-Hall Inc, Englewood Cliffs, New Jersey, 320 pp.

Singh, V.P (Ed.), (1995). "Computer Models of Watershed Hydrology." Water Resources Publications, Colorado, USA, 1130 pp.

Sivapalan, M and J.D. Kalma, (1995). "Scale problems in hydrology: Contribution of the Robertson Workshop"

Skidmore, A. K. "Terrain position as mapped from a gridded digital elevation model." International Journal of GIS **Vol. 4**(No. 1): 33-49.

Speight, J.G., (1974). "A parametric approach to landform regions." In: Progress in Geomorphology. Institute of British Geographers special publ. No.7. Alden & Mowbray Ltd. at the Alden Press, Oxford, pp.213-230.

Speight, J.G., (1980). "The role of topography in controlling through flow generation: a discussion.", Earth Surface Processes and landforms, 5, 187-191

Tarboton, D. G. (1997). "A new method for the determination of flow directions and upslope areas in grid digital elevation models." Water Resources Research **Vol. 33**(NO. 2): 309-319.

Tribe, A. (1991). "Automated Recognition of Valley Heads From Digital Elevation Models." Earth Surface Processes and Landforms **Vol. 16**: 33-49.

Tribe, A. (1992). "Automated Recognition of valley lines and drainage networks from grid digital elevation models: a review and new method." Journal of Hydrology,139,263-293

Ullah, W. and W. T. Dickinson (1979). "Quantitative Description of Depression Storage Using a Digital Elevation Surface Model." Journal of Hydrology **Vol. 42**: 63-75.

V.D'Agostino, M.Stanghellini, et al. (1993). "A Fortran-77 Program for Preliminary Extraction of Drainage Networks Based on a DEM." Computer and Geoscience **Vol. 19**(No. 7): 981-1006.

Watson, D.F., (1981)Computing the N-dimensional Delauney Tessellation with application to Voronoi polytopes" ,The Computer journal,8(2),167-172

Wilson, J. P. and J. C. Gallant, (2000), Terrain Analysis: Principles and Applications, John Wiley and Sons, New York, 479 p.

Wisnosky, Dennis E., Allen W. Batteau, (1990) "IDEF in Principle and Practice," GATEWAY, May/June 1990, pp. 8-11.

Wise, S., (2000). Assessing the quality for hydrological applications of digital elevation models derived from contours. Hydrological Processes, 14, 1909-1929.

Wolock, D.M., and .J. McCabe. (1995). "Comparison of single and multiple flow direction algorithms for computing topographic parameters in TOPMODEL". Water Resources Research Vol. 31(No.5):1315-1324.

Wood, E.F. O'Connell, P.E., (1985). "Real time forecasting. In Anderson, M.G. Burt, T.P. (Eds)", Hydrological Forecasting. John Wiley & Sons, Toronto, pp. 505-558

Wood, E.F., Sivapalan, M., Beven, K., Band, L. (1988). "Effects of spatial variability and scal with implications to hydrologic modeling." Journal of Hydrology, 102.29-47

Yoeli, P. (1983). "Digital Terrain Models and their Cartographic and Cartometric Utilization." The Cartographic Journal **Vol. 20**(No. 1): 17-23.

Zang, W. and D. R. Montgomery (1994). "Digital elevation model grid size, landscape representation, and hydrologic simulations." Water Resources Research **Vol. 30**(NO. 4): 1019-1028.

Zevenbergen, L.W., and Thorne C.R, (1987). "Quantitative Analysis of Land Surface Topology." Earth Surface Processes and Landforms, VOL. 12,27-56

Digital References

http://interactive2.er.usgs.gov/faq/get_answer.asp?id=367

APPENDIX –A

TOPAZ-N CODE

```
///////////////////////////////////////////////////////////
//                    TOPAZ-N                            //
///////////////////////////////////////////////////////////
//                                                       //
//      Copyright: Naveen Mudgal                         //
//                 Dr. L.W. Martz                        //
//                 J.Garbchet                            //
//                 University of Saskatchewan            //
//                 Saskatoon,Canada.                     //
//                                                       //
///////////////////////////////////////////////////////////


//------ Preprocessor List ---------//

#include<iostream>
#include<fstream>
#include<vector>
#include<list>
#include<string>
#include<stddef.h>
#include<ctype.h>
#include<utility>
#include<iterator>
#include<iomanip>
//#include"vars.h"
# pragma warning (disable:4786)
using namespace std;
using namespace std::rel_ops;


long unsigned int
          trow=0,tcol=0,tempvalue,temprow,tempcol,tcell_id,
          counter;
vector<int> node_no;
vector<int> node_x;
vector<int> node_y;
vector<int> subcatch_id;
vector<int long unsigned> node_id;
vector<int long unsigned> cell_id;
vector<int long unsigned> finalr(10000000);
vector<int long unsigned> temphold_1;
vector<int long unsigned> temphold_11;
vector<int long unsigned> temphold_2;
vector<int long unsigned> temphold2;
vector<int long unsigned> ovf(10000000);
vector<bool> cs;
vector<bool> wss;
vector<bool> ns;
vector<int>cc;
long unsigned int xl;
long unsigned int yl;
int size;
```

```
//----------------CLASS CELL--------------------------//
class Cell{
private:
          int cell_id;
          int flowrow;
          int flowcol;
public:

  ifstream infile;

  int colfunct(long unsigned int totalcol,long unsigned int
          cell_id);


  int rowfunct(long unsigned int totalcol,long unsigned int
          cell_id);


  int long unsigned cellid(long unsigned int totalcol,long
          unsigned int totalrow,int col,int row);

  void getFlowDir(long unsigned int,long unsigned int);

  void getFinalFlowDirExtraction(long unsigned int, long
          unsigned int);

  void getOutletInfo(long unsigned int, long unsigned int);

  void getWatershedStatus();

};//cell


void Cell::getFlowDir(long unsigned int trow, long unsigned
          int tcol){

   ifstream infile;
               infile.open("FLOVEC.DAT");
               if(!infile)
               {
                    cerr<<"Unable to open \"Flovec.dat\"
          file"
                         <<"......bailing out!\n";
                    exit(-1);
               }
               else
               {
                    cout<<"\nDefining Flow directions in
          the map";
                    ovf.push_back(0);
                    //Declare Temporary Variables Used in
          the following "For Loop"
                    int cell_id,flowrow,flowcol;
```

```
     for(counter=1;counter<=(trow*tcol);counter++
)
          {
               cell_id=counter;
               infile>>tempvalue;
               temprow=rowfunct(tcol,cell_id);
               tempcol=colfunct(tcol,cell_id);
               //***********CASE-1 First Row
First Column-Different Situations
               if((temprow==1)&&(tempcol==1))
               {

     if((tempvalue==1)||(tempvalue==2)||(tempvalu
e==3)||(tempvalue==4)||(tempvalue==7))
                    {
                         flowrow=0;
                         flowcol=0;
                         tcell_id=0;
                         ovf[counter]=tcell_id;
                    }
                    else if(tempvalue==6)
                    {
                         flowrow=temprow;
                         flowcol=tempcol+1;

     tcell_id=cellid(tcol,trow,flowcol,flowrow);
                         ovf[counter]=tcell_id;
                    }
                    else if(tempvalue==8)
                    {
                         flowrow=temprow+1;
                         flowcol=tempcol;

     tcell_id=cellid(tcol,trow,flowcol,flowrow);
                         ovf[counter]=tcell_id;
                    }
                    else if(tempvalue==9)
                    {
                         flowrow=temprow+1;
                         flowcol=tempcol+1;

     tcell_id=cellid(tcol,trow,flowcol,flowrow);
                         ovf[counter]=tcell_id;
                    }
               }

               //***********CASE-2 First Row,
Neither First Column, Nor Last Column-Different
Situations
               else
if((temprow==1)&&(tempcol!=1)&&(tempcol!=tcol))
               {
```

```
     if((tempvalue==1)||(tempvalue==2)||(tempvalu
e==3))
                    {
                         flowrow=0;
                         flowcol=0;
                         tcell_id=0;
                         ovf[counter]=tcell_id;
                    }
                    else if(tempvalue==4)
                    {
                         flowrow=temprow;
                         flowcol=tempcol-1;

     tcell_id=cellid(tcol,trow,flowcol,flowrow);
                         ovf[counter]=tcell_id;
                    }
                    else if(tempvalue==6)
                    {
                         flowrow=temprow;
                         flowcol=tempcol+1;

     tcell_id=cellid(tcol,trow,flowcol,flowrow);
                         ovf[counter]=tcell_id;
                    }
                    else if(tempvalue==7)
                    {
                         flowrow=temprow+1;
                         flowcol=tempcol-1;

     tcell_id=cellid(tcol,trow,flowcol,flowrow);
                         ovf[counter]=tcell_id;
                    }
                    else if(tempvalue==8)
                    {
                         flowrow=temprow+1;
                         flowcol=tempcol;

     tcell_id=cellid(tcol,trow,flowcol,flowrow);
                         ovf[counter]=tcell_id;
                    }
                    else if(tempvalue==9)
                    {
                         flowrow=temprow+1;
                         flowcol=tempcol;

     tcell_id=cellid(tcol,trow,flowcol,flowrow);
                         ovf[counter]=tcell_id;
                    }
               }
               //***********CASE-3 First Row and
Last Column Different Situations
               else
if((temprow==1)&&(tempcol==tcol))
```

```
               {

     if((tempvalue==1)||(tempvalue==2)||(tempvalu
e==3)||(tempvalue==6)||(tempvalue==9))
                    {
                         flowrow=0;
                         flowcol=0;
                         tcell_id=0;
                         ovf[counter]=tcell_id;
                    }
                    else if(tempvalue==4)
                    {
                         flowrow=temprow;
                         flowcol=tempcol-1;

     tcell_id=cellid(tcol,trow,flowcol,flowrow);
                         ovf[counter]=tcell_id;
                    }
                    else if(tempvalue==7)
                    {
                         flowrow=temprow+1;
                         flowcol=tempcol-1;

     tcell_id=cellid(tcol,trow,flowcol,flowrow);
                         ovf[counter]=tcell_id;
                    }
                    else if(tempvalue==8)
                    {
                         flowrow=temprow+1;
                         flowcol=tempcol;

     tcell_id=cellid(tcol,trow,flowcol,flowrow);
                         ovf[counter]=tcell_id;
                    }
               }
               //***********CASE-4 First Column
but NOT First and Last Row-Different Situations
               else
if((temprow!=1)&&(temprow!=trow)&&(tempcol==1))
               {

     if((tempvalue==1)||(tempvalue==4)||(tempvalu
e==7))
                    {
                         flowrow=0;
                         flowcol=0;
                         tcell_id=0;
                         ovf[counter]=tcell_id;
                    }
                    else if(tempvalue==2)
                    {
                         flowrow=temprow-1;
                         flowcol=tempcol;
     tcell_id=cellid(tcol,trow,flowcol,flowrow);
                         ovf[counter]=tcell_id;
                    }
                    else if(tempvalue==3)
                    {
                         flowrow=temprow-1;
                         flowcol=tempcol+1;

     tcell_id=cellid(tcol,trow,flowcol,flowrow);
                         ovf[counter]=tcell_id;
                    }
                    else if(tempvalue==6)
                    {
                         flowrow=temprow;
                         flowcol=tempcol+1;

     tcell_id=cellid(tcol,trow,flowcol,flowrow);
                         ovf[counter]=tcell_id;
                    }
                    else if(tempvalue==8)
                    {
                         flowrow=temprow+1;
                         flowcol=tempcol;

     tcell_id=cellid(tcol,trow,flowcol,flowrow);
                         ovf[counter]=tcell_id;
                    }
                    else if(tempvalue==9)
                    {
                         flowrow=temprow+1;
                         flowcol=tempcol+1;

     tcell_id=cellid(tcol,trow,flowcol,flowrow);
                         ovf[counter]=tcell_id;
                    }

               }
               //***********CASE-5 Last Column
but NOT the First and Last Row-Different
Situations
               else if
((temprow!=1)&&(temprow!=trow)&&(tempcol==tcol))
               {

     if((tempvalue==3)||(tempvalue==6)||(tempvalu
e==9))
                    {
                         flowrow=0;
                         flowcol=0;
                         tcell_id=0;
                         ovf[counter]=tcell_id;
                    }
                    else if(tempvalue==1)
                    {
                         flowrow=temprow-1;
                         flowcol=tempcol-1;

     tcell_id=cellid(tcol,trow,flowcol,flowrow);
                         ovf[counter]=tcell_id;
                    }
                    else if(tempvalue==2)
                    {
                         flowrow=temprow-1;
                         flowcol=tempcol;

     tcell_id=cellid(tcol,trow,flowcol,flowrow);
                         ovf[counter]=tcell_id;
                    }
                    else if(tempvalue==4)
                    {
                         flowrow=temprow;
                         flowcol=tempcol-1;

     tcell_id=cellid(tcol,trow,flowcol,flowrow);
                         ovf[counter]=tcell_id;
                    }
                    else if(tempvalue==7)
                    {
                         flowrow=temprow+1;
                         flowcol=tempcol-1;

     tcell_id=cellid(tcol,trow,flowcol,flowrow);
                         ovf[counter]=tcell_id;
                    }
                    else if(tempvalue==8)
                    {
                         flowrow=temprow+1;
                         flowcol=tempcol;

     tcell_id=cellid(tcol,trow,flowcol,flowrow);
                         ovf[counter]=tcell_id;
                    }
               }
               //***********CASE-6 Last Row and
First Column-Different Situations
               else
if((temprow==trow)&&(tempcol==1))
               {

     if((tempvalue==1)||(tempvalue==4)||(tempvalu
e==7)||(tempvalue==8)||(tempvalue==9))
                    {
                         flowrow=0;
                         flowcol=0;
                         tcell_id=0;
                         ovf[counter]=tcell_id;
                    }
                    else if(tempvalue==2)
                    {
                         flowrow=temprow-1;
                         flowcol=tempcol;

     tcell_id=cellid(tcol,trow,flowcol,flowrow);
                         ovf[counter]=tcell_id;
                    }
                    else if(tempvalue==3)
                    {
                         flowrow=temprow-1;
                         flowcol=tempcol+1;

     tcell_id=cellid(tcol,trow,flowcol,flowrow);
                         ovf[counter]=tcell_id;
                    }
                    else if(tempvalue==6)
                    {
                         flowrow=temprow;
                         flowcol=tempcol+1;

     tcell_id=cellid(tcol,trow,flowcol,flowrow);
                         ovf[counter]=tcell_id;
                    }
               }

               //***********CASE-7 Last Row
Neither the First Column Nor the Last Column-
Different Situations
               else if
((temprow==trow)&&(tempcol!=1)&&(tempcol!=tcol))
               {

     if((tempvalue==7)||(tempvalue==8)||(tempvalu
e==9))
                    {
                         flowrow=0;
                         flowcol=0;
                         tcell_id=0;
                         ovf[counter]=tcell_id;
                    }
                    else if(tempvalue==1)
                    {
                         flowrow=temprow-1;
                         flowcol=tempcol-1;

     tcell_id=cellid(tcol,trow,flowcol,flowrow);
                         ovf[counter]=tcell_id;
                    }
                    else if(tempvalue==2)
                    {
                         flowrow=temprow-1;
                         flowcol=tempcol;
     tcell_id=cellid(tcol,trow,flowcol,flowrow);
                         ovf[counter]=tcell_id;
                    }
                    else if(tempvalue==3)
                    {
                         flowrow=temprow-1;
                         flowcol=tempcol+1;

     tcell_id=cellid(tcol,trow,flowcol,flowrow);
                         ovf[counter]=tcell_id;
                    }
                    else if(tempvalue==4)
                    {
                         flowrow=temprow;
                         flowcol=tempcol-1;

     tcell_id=cellid(tcol,trow,flowcol,flowrow);
                         ovf[counter]=tcell_id;
                    }
                    else if(tempvalue==6)
                    {
                         flowrow=temprow;
                         flowcol=tempcol+1;

     tcell_id=cellid(tcol,trow,flowcol,flowrow);
                         ovf[counter]=tcell_id;
                    }
               }
               //***********CASE-8 Last Row and
the Last Column-Different Situations
               else if
((temprow==trow)&&(tempcol==tcol))
               {

     if((tempvalue==3)||(tempvalue==6)||(tempvalu
e==7)||(tempvalue==8)||(tempvalue==9))
                    {
                         flowrow=0;
                         flowcol=0;
                         tcell_id=0;
                         ovf[counter]=tcell_id;
                    }
                    else if(tempvalue==1)
                    {
                         flowrow=temprow-1;
                         flowcol=tempcol-1;

     tcell_id=cellid(tcol,trow,flowcol,flowrow);
                         ovf[counter]=tcell_id;
                    }
                    else if(tempvalue==2)
                    {
                         flowrow=temprow-1;
                         flowcol=tempcol;

     tcell_id=cellid(tcol,trow,flowcol,flowrow);
                         ovf[counter]=tcell_id;
                    }
                    else if(tempvalue==4)
                    {
                         flowrow=temprow;
                         flowcol=tempcol-1;

     tcell_id=cellid(tcol,trow,flowcol,flowrow);
                         ovf[counter]=tcell_id;
                    }
               }
               //***********CASE-9 Main Map Body
Section
               else if
((temprow!=1)&&(temprow!=trow)&&(tempcol!=1)&&(te
mpcol!=tcol))
               {
                    if(tempvalue==1)
                    {
                         flowrow=temprow-1;
                         flowcol=tempcol-1;

     tcell_id=cellid(tcol,trow,flowcol,flowrow);
                         ovf[counter]=tcell_id;
                    }
                    else if(tempvalue==2)
                    {
                         flowrow=temprow-1;
                         flowcol=tempcol;

     tcell_id=cellid(tcol,trow,flowcol,flowrow);
                         ovf[counter]=tcell_id;
                    }
                    else if(tempvalue==3)
                    {
                         flowrow=temprow-1;
                         flowcol=tempcol+1;

     tcell_id=cellid(tcol,trow,flowcol,flowrow);
                         ovf[counter]=tcell_id;
                    }
                    else if(tempvalue==4)
                    {
                         flowrow=temprow;
                         flowcol=tempcol-1;

     tcell_id=cellid(tcol,trow,flowcol,flowrow);
                         ovf[counter]=tcell_id;
                    }
                    else if(tempvalue==6)
                    {
```

```
                                   flowrow=temprow;
                                   flowcol=tempcol+1;

               tcell_id=cellid(tcol,trow,flowcol,flowrow);
                                   ovf[counter]=tcell_id;
                              }
                              else if(tempvalue==7)
                              {
                                   flowrow=temprow+1;
                                   flowcol=tempcol-1;

               tcell_id=cellid(tcol,trow,flowcol,flowrow);
                                   ovf[counter]=tcell_id;
                              }
                              else if(tempvalue==8)
                              {
                                   flowrow=temprow+1;
                                   flowcol=tempcol;

               tcell_id=cellid(tcol,trow,flowcol,flowrow);
                                   ovf[counter]=tcell_id;
                              }
                              else if(tempvalue==9)
                              {
                                   flowrow=temprow+1;
                                   flowcol=tempcol+1;

               tcell_id=cellid(tcol,trow,flowcol,flowrow);
                                   ovf[counter]=tcell_id;
                              }
                         }
                    }

               cout<<"..done(Elements:"<<counter<<")"<<endl
          ;

               cout<<"_____________________________________
          ____________________"<<endl;
               }
               infile.close();
}


//#########################################################
          ##
//
          02...............................................
          .........
int Cell:: colfunct(long unsigned int totalcol,long
          unsigned int cell_id)
{
          int current_col;
          if((cell_id%totalcol)==0)
          {
               current_col=totalcol;
          }
          else
          {
               current_col=(cell_id%totalcol);
          }
          return current_col;
}

//#########################################################
          #####
//
          03...............................................
          ............
int Cell:: rowfunct(long unsigned int totalcol,long
          unsigned int cell_id)
{
          int current_row;
          if((cell_id%totalcol)==0)
          {
               current_row=(cell_id/totalcol);
          }
          else
          {
               current_row=((cell_id/totalcol)+1);
          }
          return current_row;
}

//#########################################################
          #####
//
          04...............................................
          ............
int long unsigned Cell::cellid(long unsigned int
          totalcol,long unsigned int totalrow,int col,int
          row)
{
          int cell_id;
          if((col%totalcol)==0)
          {
               cell_id=row*totalcol;
          }
          else
          {
               cell_id=(((row-1)*totalcol)+(col%totalcol));
          }
          return cell_id;
}

void Cell::getFinalFlowDirExtraction(long unsigned int
          trow, long unsigned int tcol){
```

```
/////////////////////////////////////////////////
///////////////////////////////////////////////
     //Procedure for Final Flow Direction
Extraction
     cout<<"Entering the  Loop to get final flow
direction values";
     vector<int long unsigned> holden;
     bool fflag=true;
     int si,tval,hval,can;

     for(long unsigned int ck=1;
ck<=(trow*tcol);ck++)
     {
          si=ck;
          fflag=true;
          if(wss[ck]!=0)
          {
               while(fflag)
               {
                    if ((wss[ck]!=0) &&
(cs[ck]!=1))
                    {
                         holden.push_back(ck);
                         ck=ovf[ck];
                    }
                    else if (cs[ck]==1)
                    {
                         tval=ck;
                         for (int
pk=0;pk<holden.size();pk++)
                         {
                              hval=holden[pk];
                              finalr[hval]=tval;
                         }

     while(holden.empty()!=true)
                         {
                              holden.pop_back();
                         }
                         ck=si;
                         fflag=false;
                    }
               }
          }

     }

     ////////////////////////////////////////////
/////////////////////////////////////////////////
//////////
     cout<<"..done(Elements
processed:"<<counter<<")"<<endl;
```

```
               cout<<"_____________________________________
          ____________________"<<endl;
               cout<<"All data loaded Successfully"<<endl;

               cout<<"_____________________________________
          ____________________"<<endl;


}


void Cell::getWatershedStatus(){
infile.open("SUBWTA.DAT");
               if(!infile)
               {
                    cerr<<"Unable to open \"subwta.dat\"
          file"
                         <<"......bailing out!\n";
                    exit(-1);
               }
               else
               {
                    wss.push_back(false);
                    cout<<"Defining Watershed Status of
          Cells";

               for(counter=1;counter<=(trow*tcol);counter++
          )
                    {
                         infile>>tempvalue;
                         if(tempvalue>0)
                         {
                              wss.push_back(true);
                         }
                         else if(tempvalue==0)
                         {
                              wss.push_back(false);
                         }
                    }
               }
               infile.close();
               cout<<"..done(Elements
          Loaded:"<<wss.size()<<")"<<endl;

               cout<<"_____________________________________
          ____________________"<<endl;


}


//----------------CLASS CHANNEL-----------------------//
class Channel:public Cell{
```

```
public:

 ifstream infile;

 void getChannelstatus();


};//channel


void Channel::getChannelstatus(){
          //Define channel Status
          ifstream infile;
               infile.open("Netw.dat");
               if(!infile)
               {
                    cerr<<"Unable to open \"Netw.dat\"
          file"
                         <<"......bailing out!\n";
                    exit(-1);
               }
               else
               {
                    cout<<"Defining Channel Status of
          Cells";
                    cs.push_back(false);
                    for
          (counter=1;counter<=(trow*tcol);counter++)
                    {
                         infile>>tempvalue;
                         if(tempvalue==0)
                         {
                              cs.push_back(tempvalue);
                         }
                         else
                         {
                              tempvalue=1;
                              cs.push_back(tempvalue);
                         }

                    }
                    cout<<"..done(Elements
          Loaded:"<<cs.size()<<")"<<endl;

               cout<<"_____________________________________
          ____________________"<<endl;

               }
               infile.close();


               //xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
          xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
               //Define channel Status
               infile.open("Netw.dat");
               if(!infile)
               {
                    cerr<<"Unable to open \"Netw.dat\"
          file"
                         <<"......bailing out!\n";
                    exit(-1);
               }
               else
               {
                    cout<<"Defining Channel Status of
          Cells";
                    cc.push_back(0);
                    for
          (counter=1;counter<=(trow*tcol);counter++)
                    {
                         infile>>tempvalue;
                         cc.push_back(tempvalue);

                    }
                    cout<<"..done(Elements
          Loaded:"<<cc.size()<<")"<<endl;

               cout<<"_____________________________________
          ____________________"<<endl;

               }
               infile.close();

               //xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
          xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

}
//----------------CLASS NODE-------------------------//
class Node:public Channel{
public:
 void setNodeNumber(int value);
 void setNodeX(int outcol);
 void setNodeY(int outrow);
 void setNodeID(int long unsigned outcellid);
 void nsidfill(vector<int long unsigned> & node_id,
          vector<int> node_x, vector<int> node_y,int long
          unsigned const trow,int long unsigned const
          tcol);
 void printNodeInfo();
 void setNodeStatus();
 void addNode();
 void validate_Add();
 int getdsnode(int node,vector<int long unsigned>
          nodeid,vector<int>nodeno,vector<bool>cs,vector<bo
          ol>ns,vector<int long unsigned>ovf);
 int getusnode(int node,vector<int long unsigned>
          nodeid,vector<int>nodeno,vector<bool>cs,vector<in
          t long unsigned>ovf,vector<int>subcatch_id,int
          long unsigned trow,int long unsigned tcol);
 void load_dim(int long unsigned &trow,int long unsigned
          &tcol,vector<int> & node_no, vector<int> &
          node_x, vector<int> & node_y,long unsigned int &
          xl, long unsigned int & yl, int & size);
 void Pre_load_dim(int long unsigned &trow,int long
          unsigned &tcol,vector<int> & node_no, vector<int>
          & node_x, vector<int> & node_y,long unsigned int
          & xl, long unsigned int & yl, int & size);

};//node


void Node::setNodeNumber(int value){
  node_no.push_back(value);
}

void Node::setNodeX(int outcol){
  node_x.push_back(outcol);
}


void Node::setNodeY(int outrow){
  node_y.push_back(outrow);
}


void Node::setNodeID(int long unsigned outcellid){
  node_id.push_back(outcellid);
}


void Node:: nsidfill(vector<int long unsigned> & node_id,
          vector<int> node_x, vector<int> node_y, int long
          unsigned const trow,int long unsigned const tcol
          )
{
          int tempx,tempy;
          int long unsigned hold;
          //cout<<"Size of node_x:"<<node_x.size();
          for(int i=0; i<node_x.size();i++)
          {
               tempx=node_x[i];
               tempy=node_y[i];
               //cout<<"\n X/Y are: "<<tempx<<"/"<<tempy;
               hold=cellid(tcol,trow,tempx,tempy);
               //cout<<" Cell ID is :"<<hold;
               node_id.push_back(hold);
          }
}


void Node::setNodeStatus(){
```

```
//Define Node Status
               cout<<"Defining Node Status of Cells";

               for(counter=0;counter<=(trow*tcol);counter++
          )
               {
                    ns.push_back(false);
               }

               for(counter=0;counter<node_x.size();counter+
          +)
               {
                    temprow=node_y[counter];
                    tempcol=node_x[counter];

               tcell_id=cellid(tcol,trow,tempcol,temprow);
                    ns[tcell_id]=true;
               }
               cout<<"..done(Elements
          Loaded:"<<ns.size()<<")"<<endl;

               cout<<"_____________________________________
          ____________________"<<endl;


}



void Node::addNode(){
ifstream  addx("add_x.txt");
               ifstream  addy("add_y.txt");
               vector<int> addxx;
               vector<int> addyy;
               int xadd;
               int yadd;
               long unsigned int idadd;
               vector<long unsigned int> addid;
               //Read all the nodes to be added
               cout<<"\nReading File to add nodes\n";
               while((addx.peek())!=EOF)
               {
                    cout<<".";
                    addx>>xadd;
                    addy>>yadd;
                    addxx.push_back(xadd);
                    addyy.push_back(yadd);
                    idadd= cellid(tcol,trow,xadd,yadd);
                    addid.push_back(idadd);
               }

               //@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
          @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
               cout<<endl;
```

```
     cout<<"Checking the validity & addition of
new nodes \n";
     for(int ni=0;ni<addid.size();ni++)
     {
          idadd=addid[ni];
          /////////////////////////////////
          if(cs[idadd]==true)
          {
               if(ns[idadd]==false)
               {
                    int usnode,node;
                    node=subcatch_id[idadd];
//This gives the Node Number of US Node (to the
new node)
                    //change all nodes>node
                    for(int
i=1;i<=(trow*tcol);i++)
                    {
                         if(subcatch_id[i]>=node)

     subcatch_id[i]=subcatch_id[i]+1;
                    }
                    for(i=0;i<node_no.size();i++)
                    {
                         if(node_no[i]>=node)

     node_no[i]=node_no[i]+1;
                    }
                    node_no.push_back(node);
                    node_id.push_back(idadd);

     node_x.push_back(colfunct(tcol,idadd));

     node_y.push_back(rowfunct(tcol,idadd));
                    ns[idadd]=true;
                    // Now change the Subcatch_ID
of Cells draining from node+1's tcell_ID to
tcell_ID
                    temphold_1.push_back(idadd);
                    bool condition=true;
                    while(condition)
                    {
                         int get=1;
                         if(get>(trow*tcol))

     cout<<"\nOverstreching\n";

                         idadd=ovf[idadd];
                         if((cs[idadd]==true) &&
(ns[idadd]==false))
                         {

     temphold_1.push_back(idadd);
                         }
```

```
                                   else
          if((cs[idadd]==true) && (ns[idadd]==true))
                                   {
                                        condition=false;
                                   }
                              }
                              //Now  I have the list of
          channel cell from old US to the Added Node
                              int long unsigned changer;

               for(i=0;i<temphold_1.size();i++)
                              {
                                   changer=temphold_1[i];
                                   for(int
          j=1;j<=(trow*tcol);j++)
                                   {

               if(finalr[j]==changer)

               temphold_2.push_back(j);
                                   }
                              }
                              for(int
          j=0;j<temphold_2.size();j++)
                              {

               subcatch_id[temphold_2[j]]=node;
                              }

               temphold_2.erase(temphold_2.begin(),temphold
          _2.end());

               temphold_1.erase(temphold_1.begin(),temphold
          _1.end());


                         }

                    }
                    /////////////////////////////////

               }

}


void Node:: load_dim(int long unsigned &trow,int long
          unsigned &tcol,vector<int> & node_no, vector<int>
          & node_x, vector<int> & node_y,long unsigned int
          & xl, long unsigned int & yl, int & size)
{
          char line[300];
          int holder;
          int numNode;
```

```
char comma;
ifstream din("CHANNEL.INT");
ifstream pin("Dnmcnt.inp");
string sometext4;
for(int pip=1;pip<=88;pip++)
{
     if(pip==34)
     {
          pin>>xl;
          //cout<<"\nXL is:"<<xl;
     }
     if(pip==41)
     {
          pin>>yl;
          //cout<<"\nInitial YL:"<<yl;
     }

     if(pip==48)
     {
          pin>>trow;
          //cout<<"\nRow:"<<trow;
     }
     if(pip==55)
     {
          pin>>tcol;
          //cout<<"\nCol:"<<tcol;
     }
     if(pip==80)
     {
          pin>>size;
          //cout<<"\nSize:"<<size;
          yl=(yl)-((trow)*size);
          //cout<<"\nYL is:"<<yl;
     }
     else
     {
          pin.getline(line,300,'\n');
     }
}
for(int k1=0;k1<3;k1++)
{
     string sometext3;
     getline(din, sometext3);
}
din>>numNode;
int num=(int)(numNode);
for(int i=0;i<numNode;i++)
{
     din>>holder;
     node_no.push_back(holder);
     din.get(comma);
}
// skip line, now it does not matter how big this
line is
```

```
          string sometext;
          getline(din, sometext);
          //reads the y coordinate
          for(int j=0;j<numNode;j++)
          {
               din>>holder;
               node_y.push_back(holder);
               din.get(comma);
          }
          // skip line
          //string sometext2;
          //getline(din, sometext2);
          //reads the x coordinate
          for(int k=0;k<numNode;k++)
          {
               din>>holder;
               node_x.push_back(holder);
               din.get(comma);
          }
          din.close();

};


void Node::Pre_load_dim(int long unsigned &trow,int long
          unsigned &tcol,vector<int> & node_no, vector<int>
          & node_x, vector<int> & node_y,long unsigned int
          & xl, long unsigned int & yl, int & size)
{
          char pline[300];
          int pholder;
          int pnumNode;
          int nodeno_t,nodex_t,nodey_t;
          long unsigned int nodeid_t;
          char pcomma;
          ifstream pdin1("int_node_no.int");
          ifstream pdin2("int_node_id.int");
          ifstream pdin3("int_node_x.int");
          ifstream pdin4("int_node_y.int");
          ifstream ppin("Dnmcnt.inp");
          string psometext4;
          for(int ppip=1;ppip<=88;ppip++)
          {
               if(ppip==34)
               {
                    ppin>>xl;
                    //cout<<"\nXL is:"<<xl;
               }
               if(ppip==41)
               {
                    ppin>>yl;
                    //cout<<"\nInitial YL:"<<yl;
               }
               if(ppip==48)
               {
                    ppin>>trow;
                    //cout<<"\nRow:"<<trow;
               }
               if(ppip==55)
               {
                    ppin>>tcol;
                    //cout<<"\nCol:"<<tcol;
               }
               if(ppip==80)
               {
                    ppin>>size;
                    //cout<<"\nSize:"<<size;
                    yl=(yl)-((trow)*size);
                    //cout<<"\nYL is:"<<yl;
               }
               else
               {
                    ppin.getline(pline,300,'\n');
               }
          }
          int limit;
          pdin1>>limit;
          for(int i=0;i<limit;i++)
          {
               pdin1>>nodeno_t;
               pdin3>>nodex_t;
               pdin4>>nodey_t;
               pdin2>>nodeid_t;
               node_no.push_back(nodeno_t);
               node_id.push_back(nodeid_t);
               node_x.push_back(nodex_t);
               node_y.push_back(nodey_t);

          }
}

//##########################################################
          #######

void Node::validate_Add(){

//ifstream merge("merger.txt");
               ifstream  addx("add_x.txt");
               ifstream  addy("add_y.txt");
               vector<int> addxx;
               vector<int> addyy;
               int xadd;
               int yadd;
               int merge_1,merge_2;
               long unsigned int idadd;
               vector<long unsigned int> addid;
               //Read all the nodes to be added
```

```
     cout<<"\nReading File to add nodes\n";
     while((addx.peek())!=EOF)
     {
          cout<<".";
          addx>>xadd;
          addy>>yadd;
          addxx.push_back(xadd);
          addyy.push_back(yadd);
          idadd= cellid(tcol,trow,xadd,yadd);
          addid.push_back(idadd);
     }//while

     //@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
     cout<<endl;
     cout<<"Checking the validity & addition of
new nodes \n";
     for(int ni=0;ni<addid.size();ni++)
     {
          idadd=addid[ni];
          /////////////////////////////////
          if(cs[idadd]==true)
          {
               if(ns[idadd]==false)
               {
                    int usnode,node;
               node=subcatch_id[idadd]; //This
gives the Node Number of US Node (to the new
node)

                    //change all nodes>node
                    for(int
i=1;i<=(trow*tcol);i++)
                    {
                         if(subcatch_id[i]>=node)

     subcatch_id[i]=subcatch_id[i]+1;
                    }//for
                    for(i=0;i<node_no.size();i++)
                    {
                         if(node_no[i]>=node)

     node_no[i]=node_no[i]+1;
                    }//for
                    node_no.push_back(node);
                    node_id.push_back(idadd);

     node_x.push_back(colfunct(tcol,idadd));

     node_y.push_back(rowfunct(tcol,idadd));
                    ns[idadd]=true;
                    // Now change the Subcatch_ID
of Cells draining from node+1's tcell_ID to
tcell_ID
```

```
                    temphold_1.push_back(idadd);
                    bool condition=true;
                    while(condition)
                    {
                         int get=1;
                         if(get>(trow*tcol))

     cout<<"\nOverstreching\n";

                         idadd=ovf[idadd];
                         if((cs[idadd]==true) &&
(ns[idadd]==false))
                         {

     temphold_1.push_back(idadd);
                         }//if
                         else
if((cs[idadd]==true) && (ns[idadd]==true))
                         {
                              condition=false;
                         }//else
                    }//while
                    //Now  I have the list of
channel cell from old US to the Added Node
                    int long unsigned changer;

     for(i=0;i<temphold_1.size();i++)
                    {
                         changer=temphold_1[i];
                         for(int
j=1;j<=(trow*tcol);j++)
                         {

     if(finalr[j]==changer)

     temphold_2.push_back(j);
                         }//for
                    }//for
                    for(int
j=0;j<temphold_2.size();j++)
                    {

     subcatch_id[temphold_2[j]]=node;
                    }//for

     temphold_2.erase(temphold_2.begin(),temphold
_2.end());

     temphold_1.erase(temphold_1.begin(),temphold
_1.end());


               }//if
```

```
                    }//if
                    /////////////////////////////////

}//for
}//function
void Node::printNodeInfo(){
              cout<<"Total Row:"<<trow<<endl;
               cout<<"Total cols:"<<tcol<<endl;
               cout<<"Xllcorner:"<<xl<<endl;
               cout<<"Yllcorner:"<<yl<<endl;
               cout<<"Size of Grid:"<<size<<" units"<<endl;
               cout<<"\nTotal Number of
          nodes:"<<node_no.size()<<endl;

               cout<<"_____________________________________
          ____________________"<<endl;


}



int Node::getdsnode(int node,vector<int long unsigned>
          nodeid,vector<int>nodeno,vector<bool>cs,vector<bo
          ol>ns,vector<int long unsigned>ovf)
{
          int long unsigned tcellid=0, holder;
          bool flag=true;
          int dsn=0;
          for(int i=0;i<nodeno.size();i++)
          {
               if(nodeno[i]==node)
               {
                    tcellid=nodeid[i];
                    cout<<"The Cell_id of Node: "<<node<<"
          is: "<<tcellid<<endl;
               }
          }
          if(tcellid==0)
          {
               cout<<"There is no Node with the node number
          \" "<<node<<"\" in the entire drainage
          network"<<endl;
               flag=false;
          }

          while(flag)
          {
               tcellid=ovf[tcellid];
               if(ns[tcellid]==true)
               {
                    for(int i=0;i<nodeid.size();i++)
                    {
                         if(tcellid==nodeid[i])
```

```
                              tcellid=i;
                    }
                    dsn=nodeno[tcellid];
                    flag=false;
               }
          }
          return dsn;
}

//#########################################################
          ##########


int Node::getusnode(int node,vector<int long unsigned>
          nodeid,vector<int>nodeno,vector<bool>cs,vector<in
          t long unsigned>ovf,vector<int>subcatch_id,int
          long unsigned trow,int long unsigned tcol)
{
          int holder=0;
          int long unsigned tcellid=0;
          for(int i=0;i<nodeno.size();i++)
          {
               if(nodeno[i]==node)
               {
                    tcellid=nodeid[i];
               }
          }
          if(tcellid!=0)
          {
               for(i=1;i<=(trow*tcol);i++)
               {
                    if((cs[i]==true)&& (ovf[i]==tcellid))
                    {
                         holder=subcatch_id[i];
                         break;
                    }
               }
               if(holder==0)
                    cout<<"\nThe Given Node is the Source
          Node hence has NO Upstream Node\n";
          }
          return holder;
}
//#########################################################
          ############

//----------------CLASS SUBCATCHMENT-----------------//
class Subcatchment:public Cell{
public:
          void getSubID();
          void setID(int);
    void MergeSheds(Node);
};//subcatchment
```

```
void Subcatchment::setID(int i){
subcatch_id.push_back(i);
}

void Subcatchment::getSubID(){
          //Define Subcatchment ID's
               infile.open("SUBWTA.DAT");
               if(!infile)
               {
                    cerr<<"Unable to open \"subwta.dat\"
          file"
                         <<"......bailing out!\n";
                    exit(-1);
               }
               else
               {    subcatch_id.push_back(0);
               cout<<"Defining Subcatchment ID Status of
          Cells";

               for(counter=1;counter<=(trow*tcol);counter++
          )
               {
                    infile>>tempvalue;
                    if(tempvalue>0)
                    {

               subcatch_id.push_back((tempvalue/10));
                    }
                    else if(tempvalue==0)
                    {
                         subcatch_id.push_back(0);
                    }
               }
               }

               infile.close();
               cout<<"..done(Elements
          Loaded:"<<subcatch_id.size()<<")"<<endl;

               cout<<"_____________________________________
          ____________________"<<endl;


//---------------------------------------------------------
          ----------
               infile.open("SUBWTA.DAT");
               if(!infile)
               {
                    cerr<<"Unable to open \"subwta.dat\"
          file"
                         <<"......bailing out!\n";
                    exit(-1);
               }
               else
               {
                    wss.push_back(false);
                    cout<<"Defining Watershed Status of
          Cells";

               for(counter=1;counter<=(trow*tcol);counter++
          )
                    {
                         infile>>tempvalue;
                         if(tempvalue>0)
                         {
                              wss.push_back(true);
                         }
                         else if(tempvalue==0)
                         {
                              wss.push_back(false);
                         }
                    }
               }
               infile.close();
               cout<<"..done(Elements
          Loaded:"<<wss.size()<<")"<<endl;

               cout<<"_____________________________________
          ____________________"<<endl;
}


void Subcatchment::MergeSheds(Node node){

int merge_1, merge_2,adopt;
int i=0;
               cout<<"Please provide the ID of the
          Subwatersheds to be merged:"
                    <<"First Subwatershed ID:";
               cin>>merge_1;
               cout<<"Second Watershed ID:";
               cin>>merge_2;
               //check if they are adjacent
               cout<<"\nVarifying the Hydrologic Adjacency
          of Subwatershed "<<merge_1
                    <<" and Subwatershed "<<merge_2<<endl;
               int cc_merge1,cc_merge2;
               long unsigned int merge_ad1,merge_ad2;
               for( i=0;i<node_no.size();i++)
               {
                    if(merge_1==node_no[i])
                    {
                         merge_ad1=node_id[i];
                    }
                    else if (merge_2==node_no[i])
                    {
                         merge_ad2=node_id[i];
                    }
               }
               cc_merge1=cc[merge_ad1];
               cc_merge2=cc[merge_ad2];
               ////////////////////////////////////
               /*{
               ofstream myfile("subcatch.asc");

               for(counter=1;counter<=(trow*tcol);counter++
          )
               myfile<<subcatch_id[counter]<<endl;
               }*/
               ////////////////////////////////////

               if((merge_2==node.getdsnode(merge_1,node_id,
          node_no,cs,ns,ovf))||(merge_1==node.getdsnode(mer
          ge_2,node_id,node_no,cs,ns,ovf))||(node.getdsnode
          (merge_1,node_id,node_no,cs,ns,ovf)==node.getdsno
          de(merge_2,node_id,node_no,cs,ns,ovf)))
               {
                    cout<<"Hydrologic adjacency of the
          given Subwatersheds varified\n";
                    for(int long unsigned
          i=1;i<=(trow*tcol);i++)
                    {
                         if(cc_merge1>cc_merge2)
                         {
                              if(subcatch_id[i]==merge_2)
                              {
                                   subcatch_id[i]=merge_1;
                              }
                         }
                         else if (cc_merge2>cc_merge1)
                         {
                              if(subcatch_id[i]==merge_1)
                              {
                                   subcatch_id[i]=merge_2;
                              }
                         }
                         else if (cc_merge2==cc_merge1)
                         {
                              if(merge_1>merge_2)
                              {

               if(subcatch_id[i]==merge_2)
                                   {

               subcatch_id[i]==merge_1;
                                   }
                              }
                              else if(merge_2>merge_1)
                              {

               if(subcatch_id[i]==merge_1)
```

```
                         {

     subcatch_id[i]==merge_2;
                         }
                    }
               }
          }
          int retained;
          if(cc_merge1>cc_merge2)
          {
               adopt=merge_2;
               retained=merge_1;
          }
          else if(cc_merge2>cc_merge1)
          {
               adopt=merge_1;
               retained=merge_2;
          }

          else if(cc_merge2==cc_merge1)
          {
               if(merge_1>merge_2)
               {
                    adopt=merge_2;
                    retained=merge_1;
               }
               else
               {
                    adopt=merge_1;
                    retained=merge_2;
               }
          }
          for(i=0;i<node_no.size();i++)
          {
               if(node_no[i]==adopt)
               {
                    ns[node_id[i]]=false;

     node_no.erase(node_no.begin()+(i));

     node_x.erase(node_x.begin()+(i));

     node_y.erase(node_y.begin()+(i));

     node_id.erase(node_id.begin()+(i));
               }
          }
          ////////////////////////////////////
     }
     else
     {
          cout<<"Sorry, the given subwatersheds
cannot be merged as they are not hydrologically
adjacent\n"
```

```
                    <<"Please provide another set of
          subwatersheds for merging";
               }

}




//----------------CLASS FILEMANAGER------------------//

class FileManager{
public:
          void output_11D( vector<int> vec, int long
          unsigned & trow,int long unsigned &
          tcol,vector<bool>cs, vector<bool>ns,char
          file[30],long unsigned int xl, long unsigned int
          yl, int size);

    void output_1D( vector<int> vec,int long unsigned &
          trow,int long unsigned & tcol,vector<bool>cs,
          vector<bool>ns,char file[30],long unsigned int
          xl, long unsigned int yl, int size);

          void output_1D( vector<bool> vec, int long
          unsigned & trow,int long unsigned &
          tcol,vector<bool>cs, vector<bool>ns,char
          file[30],long unsigned int xl, long unsigned int
          yl, int size);

    void output_1D( vector<int long unsigned> vec,int long
          unsigned & trow,int long unsigned & tcol,char
          file[30],long unsigned int xl, long unsigned int
          yl, int size);

    void output_1D( vector<bool> vec, int long unsigned &
          trow,int long unsigned & tcol,char file[30],long
          unsigned int xl, long unsigned int yl, int size);

};



void FileManager::output_11D( vector<int> vec, int long
          unsigned & trow,int long unsigned &
          tcol,vector<bool>cs, vector<bool>ns,char
          file[30],long unsigned int xl, long unsigned int
          yl, int size)
{
          ofstream outfile(file);
          outfile<<"ncols"<<" "<<tcol<<endl;
          outfile<<"nrows"<<" "<<trow<<endl;
          outfile<<"xllcorner"<<" "<<xl<<endl;
          outfile<<"yllcorner"<<" "<<yl<<endl;
          outfile<<"cellsize"<<" "<<size<<endl;
          outfile<<"nodata_value"<<" "<<"-32768"<<endl;
          for(int i=1;i<=(trow*tcol);i++)
          {
                    outfile<<vec[i]<<endl;
          }

}
//##########################################################
          ##########


void FileManager::output_1D( vector<int> vec, int long
          unsigned & trow,int long unsigned &
          tcol,vector<bool>cs, vector<bool>ns,char
          file[30],long unsigned int xl, long unsigned int
          yl, int size)
{
          ofstream outfile(file);
          outfile<<"ncols"<<" "<<tcol<<endl;
          outfile<<"nrows"<<" "<<trow<<endl;
          outfile<<"xllcorner"<<" "<<xl<<endl;
          outfile<<"yllcorner"<<" "<<yl<<endl;
          outfile<<"cellsize"<<" "<<size<<endl;
          outfile<<"nodata_value"<<" "<<"-32768"<<endl;
          for(int i=1;i<=(trow*tcol);i++)
          {
               if((cs[i]==true)&&(ns[i]==false))
               {
                    outfile<<1<<endl;
               }
               else if((cs[i]==true)&&(ns[i]==true))
               {
                    outfile<<0<<endl;
               }
               else
               {
                    outfile<<vec[i]<<endl;
               }
          }

}
//##########################################################
          #########


void FileManager::output_1D( vector<bool> vec, int long
          unsigned & trow,int long unsigned &
          tcol,vector<bool>cs, vector<bool>ns,char
          file[30],long unsigned int xl, long unsigned int
          yl, int size)
{
          ofstream outfile(file);
          outfile<<"ncols"<<" "<<tcol<<endl;
          outfile<<"nrows"<<" "<<trow<<endl;
          outfile<<"xllcorner"<<" "<<xl<<endl;
          outfile<<"yllcorner"<<" "<<yl<<endl;
          outfile<<"cellsize"<<" "<<size<<endl;
          outfile<<"nodata_value"<<" "<<"-32768"<<endl;
          for(int i=1;i<=(trow*tcol);i++)
          {
               if((cs[i]==true)&&(ns[i]==false))
               {
                    outfile<<1<<endl;
               }
               else if((cs[i]==true)&&(ns[i]==true))
               {
                    outfile<<2<<endl;
               }
               else
               {
                    outfile<<vec[i]<<endl;
               }
          }

}
//#########################################################
          #########


void FileManager::output_1D( vector<int long unsigned> vec,
          int long unsigned & trow,int long unsigned &
          tcol,char file[30],long unsigned int xl, long
          unsigned int yl, int size)
{
          ofstream outfile(file);
          outfile<<"ncols"<<" "<<tcol<<endl;
          outfile<<"nrows"<<" "<<trow<<endl;
          outfile<<"xllcorner"<<" "<<xl<<endl;
          outfile<<"yllcorner"<<" "<<yl<<endl;
          outfile<<"cellsize"<<" "<<size<<endl;
          outfile<<"nodata_value"<<" "<<"-32768"<<endl;
          for(int i=1;i<=(trow*tcol);i++)
               outfile<<vec[i]<<endl;
}

//#########################################################
          #########


void FileManager::output_1D( vector<bool> vec, int long
          unsigned & trow,int long unsigned & tcol,char
          file[30],long unsigned int xl, long unsigned int
          yl, int size)
{
          ofstream outfile(file);
          outfile<<"ncols"<<" "<<tcol<<endl;
          outfile<<"nrows"<<" "<<trow<<endl;
          outfile<<"xllcorner"<<" "<<xl<<endl;
          outfile<<"yllcorner"<<" "<<yl<<endl;
          outfile<<"cellsize"<<" "<<size<<endl;
          outfile<<"nodata_value"<<" "<<"-32768"<<endl;
          for(int i=1;i<=(trow*tcol);i++)
               outfile<<vec[i]<<endl;
}
//#########################################################
          #########




//----------------CLASS TOPAZN-----------------------//
class TopazN{
 public:
          char choice;
          bool flag_staff;
          int outrow,outcol,value;
          int long unsigned outcellid;
          char file[30];
          char ver;
          Node node;
          Cell cell;
          Channel channel;
          Subcatchment sub;
          FileManager fm;
    void printMsg();
          void readInputInfo();
    void mainmenu();
          Node getNode();
          void setNode(Node);
          Cell getCell();
          void setCell(Cell);
          void setFlagstaff(bool);
          void setChar(char);
          void addNode();
          void mergeWatersheds(Node node);
};// TOPAZN


void TopazN:: printMsg(){
    cout<<"\nWelcome to the TOPAZ-N Version-1.0
          \nCopyrigts:Dr. L.W.Martz, Jurgen
          Garbrecht,Naveen Mudgal\n \n";
          cout<<"Please press:('P' or 'p') to proceed:";

}



void TopazN::mainmenu()
```

```
{
          cout<<"\n________________________________________
          ___"
               <<"\nPlease choose from the following
          options:\n"
               <<"\n   Press \"A\" or 'a' to ADD NEW
          NODES\n"
               <<"\n   Press \"M\" or 'm' to Merge
          Watersheds\n"
               <<"\n   Press \"F\" or 'f' to Find\n"
               <<"\n   Press \"E\" or 'e' to Exit this
          Menu\n"
               <<"\n   For intermediate output,Press \"O\"
          or 'o'\n"
               /*<<"\n   To Save current settings ,Press
          \"S\" or 's'\n"*/
               <<"\n   To Add a list of New Nodes please
          press \"L\" or 'l' \n"

               <<"_________________________________________
          _"
               <<"\n   Please select your option now :";
}



void TopazN::readInputInfo(){
   cin>>ver;
   TopazN tn;
   Cell cell;
   Subcatchment sub;
   Node node;
   FileManager fm;
   flag_staff=true;
   outrow,outcol,value=0;


          if((ver=='P') ||(ver=='p'))
          {
               cout<<"Please Specifiy the Co-ordinates of
          the Outlet:"
                    <<"\nRow:";
               cin>>outrow;
               cout<<"Column:";
               cin>>outcol;

               node.load_dim(trow,tcol,node_no,node_x,node_
          y,xl,yl,size);

               node.nsidfill(node_id,node_x,node_y,trow,tco
          l);
```

```
       outcellid=cell.cellid(tcol,trow,outcol,outro
  w);

       node.setNodeNumber(value);
       node.setNodeX(outcol);
       node.setNodeY(outrow);
       node.setNodeID(outcellid);
       cout<<"\nThank You
  \n_______________________________________________
  ________\n"<<endl;

       node.printNodeInfo();

       node.setNodeStatus();
       system("cls");
       cout<<"Welcome to the TOPAZ-N Version-1.0
  \nCopyrigts:Dr. L.W.Martz, Jurgen
  Garbrecht,Naveen Mudgal\n \n";
       cout<<"Total Row:"<<trow<<endl;
       cout<<"Total cols:"<<tcol<<endl;
       cout<<"Xllcorner:"<<xl<<endl;
       cout<<"Yllcorner:"<<yl<<endl;
       cout<<"Size of Grid:"<<size<<" units"<<endl;
       cout<<"Row & Column for Outlet are:"<<outrow
  <<" & "<<outcol<<" respectively"<<endl;

       ////////////////////////////////////////////
  ////////////////////////////////
      //Defining Flow directions in the map";
      cell.getFlowDir( trow, tcol);
       //Define channel Status
channel.getChannelstatus();
       //Define Node Status
       node.setNodeStatus();
       //define watershed status of cells
       cell.getWatershedStatus();
        //Procedure for Final Flow Direction
  Extraction
cell.getFinalFlowDirExtraction(trow,tcol);

  }//if

  else if((ver=='X')||(ver=='x'))
  {

       node.Pre_load_dim(trow,tcol,node_no,node_x,n
  ode_y,xl,yl,size);

       node.nsidfill(node_id,node_x,node_y,trow,tco
  l);
       cout<<"\nTotal Number of
  nodes:"<<node_no.size()<<endl;
       node.printNodeInfo();
       cout<<"Proceeding to load other
  parameters\n";
       ifstream reader("inter_subwta.tpz");
       int holder;
       sub.setID(0);
       while((reader.peek())!=EOF)
       {
            reader>>holder;
            sub.setID(holder);
       }

       //Defining Flow directions in the map";
         cell.getFlowDir( trow, tcol);
       //Define channel Status
         channel.getChannelstatus();
       //Define Node Status
        node.setNodeStatus();
       //define watershed status of cells
        cell.getWatershedStatus();
        //Procedure for Final Flow Direction
  Extraction
 cell.getFinalFlowDirExtraction(trow,tcol);

       ////////////////////////////////////////////
  /////////////////////////////////////////////////
  ///



  }
  while(flag_staff==true)
  {   tn.mainmenu();
  cin>>choice;
  //@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
  @@@@@@
  if(choice=='A' || choice=='a')
  {
       node.addNode();

  }

  else if(choice=='M' || choice=='m')
  { sub.MergeSheds(node);

       char cho1;
       cout<<"\nDo you wish to have an intermediate
  output? (Press 'Y' for Yes & 'N' for No)";
       cin>>cho1;
       char extn1[30];
       if((cho1=='Y')||(cho1=='y'))
       {
            char file_I1[50]="_Inter_CNNS.Dat";
            cout<<"\nPlease provide a \"Prefix\"
  for the intermediate files:\n";
```

```
          cin>>extn1;
          strcat(extn1,file_I1);

     fm.output_1D(subcatch_id,trow,tcol,cs,ns,ext
n1,xl,yl,size);
     }
}
/////////////////////////////////////////////////
/////////////////////////////////////////////
else if(choice=='E' || choice=='e')
{
     cout<<"\nBefore Exiting, Please provide a
\"Prefix\" for the output files:\n";
     cin>>file;
     flag_staff=false;
}
else if(choice=='o' || choice=='O')
{
     char cho3;
     cout<<"\nDo you wish to have an intermediate
output? (Press 'Y' for Yes & 'N' for No)";
     cin>>cho3;
     char extn3[30];
     char extn4[30];
     if((cho3=='Y')||(cho3=='y'))
     {
          int pep;
          char file_I3[50]="_Inter_CNNS.Dat";
          char
file_I4[50]="_Inter_Subwatersheds.Dat";
          cout<<"\nPlease provide a \"Prefix\"
for the intermediate files:\n";
          cin>>extn3;
          strcpy(extn4,extn3);
          strcat(extn3,file_I3);
          cout<<"Size:"<<size;
          cout<<"\nThe size of
Subcatch_id:"<<(subcatch_id.size()-1);
          cin>>pep;

     fm.output_1D(subcatch_id,trow,tcol,cs,ns,ext
n3,xl,yl,size);
          strcat(extn4,file_I4);

     fm.output_11D(subcatch_id,trow,tcol,cs,ns,ex
tn4,xl,yl,size);
     }
}
else if(choice=='F' || choice=='f')
{
     int holders, nodenoe;
     cout<<"Please provide the node no:";
     cin>>nodenoe;
     holders=node.getdsnode(nodenoe,node_id,node_
no,cs,ns,ovf);
     cout<<"\nThe D/S node of "<<nodenoe<<" is
:"<<holders;

     holders=node.getusnode(nodenoe,node_id,node_
no,cs,ovf,subcatch_id,trow,tcol);
     if(holders>0)
     {
          cout<<"\nThe U/S node of "<<nodenoe<<"
is :"<<holders<<endl<<endl;
     }

}
else if(choice=='S' || choice=='s')
{
     ofstream out1("int_node_no.int");
     ofstream out2("int_node_id.int");
     ofstream out3("int_node_x.int");
     ofstream out4("int_node_y.int");
     out1<<node_no.size()<<endl;
     for(int i=0;i<node_no.size();i++)
     {
          out1<<node_no[i]<<endl;
          out3<<node_x[i]<<endl;
          out4<<node_y[i]<<endl;
          out2<<node_id[i]<<endl;
     }
     cout<<"The Size of Node_No
is:"<<node_no.size()<<endl;
     ofstream out5("inter_subwta.tpz");
     for(i=1;i<=(trow*tcol);i++)
     {
          out5<<subcatch_id[i]<<endl;
     }

}
else if (choice=='L' || choice=='l')
{
      node.validate_Add();

     }

     //@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@

     char extn11[30];
     char proceed;
     char file_I11[50]="_Inter_CNNS.Dat";
     cout<<"\nPlease provide a \"Prefix\" for the
intermediate files:\n";
     cin>>extn11;
     strcat(extn11,file_I11);
```

```
               fm.output_1D(subcatch_id,trow,tcol,cs,ns,ext
          n11,xl,yl,size);

               cout<<"\nPlease Press Proceed to Continue
          with the rest of the process\n";
               cin>>proceed;
               cout<<"\nThank You\n";
          ////////////////////////////////////////////////
          ////////////////////////////////////////
}//while

          //output CN-Channel Network
          char file_O[30];
          strcpy(file_O,file);
          char file2[30]="_CN.Dat"; //CN-Channel Network
          File
          char file3[30]="_CNN.Dat";//CNN-Channel Network &
          Node File
          char file4[30]="_CNNS.Dat";//CNNS-Channel
          Network, Node & Subcatchment File
          char file5[30]="_FFD.Dat";//FFD-Final Flow
          Direction File
          strcat(file,file2);
          fm.output_1D(cs,trow,tcol,file,xl,yl,size);
          strcpy(file,file_O);
          strcat(file,file3);
          fm.output_1D(cs,trow,tcol,cs,ns,file,xl,yl,size);
          strcpy(file,file_O);
          strcat(file,file4);
          fm.output_1D(subcatch_id,trow,tcol,cs,ns,file,xl,
          yl,size);
          strcpy(file,file_O);
          strcat(file,file5);
          fm.output_1D(finalr,trow,tcol,file,xl,yl,size);
          cout<<"\nThank You...can shut the program now\n";


}

Node TopazN::getNode(){
  return node;
}

void TopazN::setNode(Node node){
  Node node1;
  node1= node;
}

Cell TopazN::getCell(){
  return cell;
}
```

```
void TopazN::setCell(Cell cell){
 Cell cell1;
 cell1=cell;
}

void TopazN::setFlagstaff(bool t){
 flag_staff=t;
}

void TopazN::setChar(char c){
 ver =c ;
}

void TopazN::addNode(){
 node.addNode();
}

void TopazN::mergeWatersheds(Node node){
  sub.MergeSheds( node);
}
// ------------------ MAIN FUNCTION ----------------- //
//######################################################
void main()
{
 TopazN tn;
 tn.readInputInfo();

}
//######################################################
```