UNIVERSITY OF CALIFORNIA,
IRVINE

A Framework for Optimization and Simulation of Reservoir Systems Using Advanced Optimization and Data Mining Tools

DISSERTATION

submitted in partial satisfaction of the requirements
for the degree of

DOCTOR OF PHILOSOPHY

in Civil Engineering

by

Matin Rahnamay Naeini

Dissertation Committee:
Professor Kuolin Hsu, Co-Chair
Professor Amir AghaKouchak, Co-Chair
Professor Soroosh Sorooshian

2019

DEDICATION

To
my lovely wife for her endless love and patience,
my greatest parents for their unconditional love and support throughout my life,
and my kind siblings for all their help along my journey.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ALGORITHMS

LIST OF ABBREVIATIONS

AMALGAM-SO	A Multialgorithm Genetically Adaptive Method for Single Objective Optimization
CART	Classification and Regression Tree
CCE	Competitive Complex Evolution
CONUS	Contiguous United States
DE	Differential Evolution
DP	Dynamic Programming
EA	Evolutionary Algorithm
EMP	Evolutionary Methods Performance
FL	Frog Leaping
GMT	Generalized Model Tree
GWO	Grey Wolf Optimizer
LHS	Latin Hypercube Sampling
LP	Linear Programming
MCCE	Modified Competitive Complex Evolution
MCMC	Markov Chain Monte Carlo
MFL	Modified Frog Leaping
MGWO	Modified Grey Wolf Optimizer
MOCOM-UA	Multi-Objective Complex evolution, University of Arizona
MOSCEM	Multi-Objective Shuffled Complex Evolution Metropolis

MT	Model Tree
NFL	No Free Lunch
NLP	NonLinear Programming
PCA	Principal Component Analysis
PSO	Particle Swarm Optimization
RF	Random Forest
SC	Splitting Criterion
SaDE	Self-adaptive Differential Evolution
SCE-UA	Shuffle Complex Evolution-developed at University of Arizona
SCEM-UA	Shuffled Complex Evolution Metropolis algorithm-developed at University of Arizona
SC-SAHEL	Shuffle Complex-Self Adaptive Hybrid EvoLution
SDR	Standard Deviation Reduction
SP-UCI	Shuffled Complex strategy with Principal component analysis-developed at University of California, Irvine
SSR	Sum Squared Residual
SSRM	Sum Squared Residual with respect to Mean of response variable
SSRML	Sum Squared Residual with Multiple Linear regression
URS	Uniform Random Sampling

ACKNOWLEDGMENTS

As my journey as Ph.D. student comes to an end, I would like to express my gratitude to a number of people for their invaluable support and help along the way. First, I would like to express my sincere appreciation to my adviser, Professor Kuolin Hsu, for being a great mentor, friend, and guide for me. His insightful tips and comments on my work taught me to look at the research problems from different angles. In fact, Professor Hsu taught me to think out of the box and explore different ways to tackle the research problems. His insight and broad knowledge about different areas of science has always astonished and inspired me. This work would have not been possible without his generous support, care, and encouragement.

I would like to express my special gratitude to Professor Sorooshian for his generous support, help and all his scientific guidance during my Ph.D. He has always been kind, supportive and caring to me and has been a great mentor not only for my research but also for my life. I would never forget his fatherly advice and I am always grateful for the lifelong lessons I learned from him. I would particularly like to thank Professor Amir AghaKouchak for being a great friend, mentor, guide, and teacher for me. I have leaned a lot from him since I started my journey as a graduate student at UCI. His passion and brilliance in research has always been a source of motivation for me to follow my academic journey.

I would also like to thank my friend, Professor Mojtaba Sadegh. He is one of the first people that I met at UCI, and since then he has always been a great friend and mentor for me. He taught me how to test and develop my research ideas and encouraged me to investigate them. I am always thankful for what I learned from him. I would like to thank Professor Tiantian Yang. He has been a huge source of motivation and confidence for me. He helped with my papers and taught me how to write manuscripts and articulate my research. I would like to acknowledge the role of Dr. Andrea Thorstensen in my research. She has introduced me to amazing people at UCAR and NCAR which opened me doors to collaboration. I am grateful to Dr. Arezoo RafieeiNasab for helping me to initiate collaboration with other research agencies. I would like to thank Dr. Ahmad A. Tavakoly for all his inputs, comments, and data he provided for my research. I would like to express my appreciation to my colleague Dr. Bita Analui for her help, comments, and suggestions on my research work. I would like to acknowledge Professor Qingyun Duan for his great comments on my first paper. I would like to thank Professor Lei Xiaohui for his suggestions and comments on my manuscript. Also, I would like to thank Dan Braithwaite for all his technical support and Diane Hohnbaum for her administrative assistance in every step of my Ph.D. I would like to express my special thank to April Heath for all her assistance and help during my graduate studies.

Moreover, I have also had the opportunity and privilege of working with many amazing people at the Center for Hydrometeorology and Remote Sensing (CHRS), who provided a nice and friendly research environment for me. Thank you Professor Phu Ngyun, Mohammad (Pouya) Faridzad, Ata Akbari Asanjan, Negin Hayatbini, Mohammed Omer Ombadi, Dr. Negar Karbalaee, Jinny Lee, Dr. Hoang Tran Viet, Baoxiang Pan, Dr. Raied Alharbi, Mojtaba Sadeghi, Vesta Afzali, Eric Shearer, Ai-Ling Jiang, and Dr. Yumeng Tao. I am always honored to work with them in the same group and looking forward to many years

of collaboration and friendship with my family at CHRS. I would like to thank all my great friends, who gave me a remarkable experience during my graduate studies. I am looking forward to many more years of friendship and many more great memories with them.
Most importantly I would like to thank my family. My beautiful, lovely wife, Parnian, for her patience, support, and love. My parents for their unconditional love, support, and kindness. My amazing siblings and in laws for their encouragement and help. I would like to especially thank my cousins, Behrooz, Geraldine, and Kevan, for their emotional and financial support to pursue my graduate studies. I would like to honor the memory of my late aunt, Forough, who has been a source of motivation and encouragement for me. Her memory will be with me forever.
This work is supported by U.S. Department of Energy (DOE Prime Award # DE-IA0000018), California Energy Commission (CEC Award # 300-15-005), NSF CyberSEES Project (Award CCF-1331915), NOAA/NESDIS/NCDC (Prime award NA09NES4400006 and NCSU CICS and subaward 2009-1380-01), and the U.S. Army Research Office (award W911NF-11-1-0422).

CURRICULUM VITAE

Matin Rahnamay Naeini

EDUCATION

Doctor of Philosophy in Civil Engineering **2019**
University of California, Irvine *Irvine, CA*

Master of Science in Civil Engineering **2016**
University of California, Irvine *Irvine, CA*

Bachelor of Science in Civil Engineering **2014**
Iran University of Science and Technology *Tehran, Iran*

RESEARCH EXPERIENCE

Graduate Research Assistant **2014–2019**
University of California, Irvine *Irvine, CA*

Undergraduate Research Assistant **Summer 2012**
Iran University of Science and Technology *Tehran, Iran*

TEACHING AND MENTORING EXPERIENCE

Teaching Assistant: Hydrology **2017–2018**
Teaching Assistant: Hydrologic Systems **Spring 2017**
Reader: Mathematical Methods in Engineering **Fall 2017**
Reader: Introduction to Environmental Chemistry **Winter 2016**
Teaching Assistant: Engineering Problem Solving **Spring 2015**
University of California, Irvine *Irvine, CA*

Teaching Assistant: Applied Hydrology **2012-2013**
Iran University of Science and Technology *Tehran, Iran*

REFEREED JOURNAL PUBLICATIONS

Matin Rahnamay Naeini, Bita Analui, Qingyun Duan, Hoshin V. Gupta, Soroosh Sorooshian, Three Decades of Shuffled Complex Evolution Optimization: Review and Application. **2019**
Under Preparation

Tiantian Yang, Matin Rahnamay Naeini, Soroosh Sorooshian, An enhanced Bayesian Model Averaging method for improving the hydropower discharge simulation using multiple regression tree models **2019**
Under Preparation

Matin Rahnamay Naeini, Tiantian Yang, Ahmad Tavakoly, Bita Analui, Amir AghaKouchak, Kuolin Hsu, Soroosh Sorooshian, Generalized Model Tree (GMT) framework for simulating rule-based hydrologic systems **2019**
Water Resources Research, Under Review

Matin Rahnamay Naeini, Tiantian Yang, Mojtaba Sadegh, Amir AghaKouchak, Kuolin Hsu, Soroosh Sorooshian, Qingyun Duan, Xiaohui Lei, Shuffled Complex-Self Adaptive Hybrid EvoLution (SC-SAHEL) optimization framework **2018**
Environmental Modelling & Software,104, 215-235.

Conferences

Matin Rahnamay Naeini, Tiantian Yang, Ahmad Tavakoly, Amir AghaKouchak, Kuolin Hsu, Soroosh Sorooshian, Developing a Generalized Model Tree (GMT) framework for simulating reservoir systems **2018**
American Geophysical Union

Matin Rahnamay Naeini, Tiantian Yang, Mojtaba Sadegh, Amir AghaKouchak, Kuolin Hsu, Soroosh Sorooshian, Developing a Shuffled Complex-Self Adaptive Hybrid Evolution (SC-SAHEL) Framework for Water Resources Management and Water-Energy System Optimization **2017**
American Geophysical Union

Matin Rahnamay Naeini, Jasper A. Vrugt, Mojtaba Sadegh, Guilherme J. C. Gomes, Scaling and Regionalization of Flow Duration Curve Across the Contiguous US **2015**
American Geophysical Union

Technical, Computational and Programming Skills

Programming: Proficient in MATLAB. Familiar with Python, FORTRAN, and R
Hydrologic Modeling: Worked with HEC-HMS. Familiar with WRF-Hydro and SWAT
GIS: Worked with ArcGIS
Text Editors: Proficient in Microsoft Office and LATEX
OS: Worked with Windows and Unix

Honors and Awards

Civil and Environmental Eng. Graduate Student of the Year **2018-2019**
CUAHSI Travel Grant Recipient for WRF-Hydro Workshop **Summer 2018**
Henry Samueli School of Eng. Research and Travel Grant **Winter 2018**
People Choice Speaker of the Urban Plan. and Eng. Section **Spring 2017**
University of California, Irvine *Irvine, CA*

ABSTRACT OF THE DISSERTATION

A Framework for Optimization and Simulation of Reservoir Systems Using Advanced Optimization and Data Mining Tools

By

Matin Rahnamay Naeini

Doctor of Philosophy in Civil Engineering

University of California, Irvine, 2019

Professor Kuolin Hsu, Co-Chair
Professor Amir AghaKouchak, Co-Chair

The simulation and optimization of reservoir systems has attracted a great deal of attention in the field of hydrology and water resources management. Although many advances have been made, the gap between theoretical and real-world operation of reservoir systems still exists. Here, multiple tools and algorithms are proposed to bridge the existing gap to some extent. These tools are developed to aid decision makers, engineers, and scientists to understand, simulate, and improve the operational rules of reservoir systems.

In this dissertation, I propose an optimization framework, titled shuffled complex-self adaptive hybrid evolution (SC-SAHEL), to optimize the controlled discharge from reservoirs. This new optimization tool can solve a wide range of optimization problems, using a self-adaptive search mechanism. The algorithm employs multiple search methods from different optimization tools and selects the most suitable method for the problem space. This process reveals the potential of each search mechanism during the course of the search and enhances the efficiency and effectiveness of the search. The SC-SAHEL framework is tested on multiple benchmark problems and showed superior performance in comparison to single search methods. The framework is applied to the Folsom reservoir to maximize hydropower generation by tuning the controlled discharge. The results showed that the SC-SAHEL algorithm is

superior to other single search method algorithms for finding near optimum solutions. The results also demonstrated the robustness of the SC-SAHEL framework for solving a wide range of reservoir optimization problems.

In addition to the optimization framework, a new data-mining algorithm, title generalized model tree (GMT), is proposed for simulating rule-based hydrologic systems. The new framework is developed based on the decision tree models and employs linear regression for prediction and model induction. The newly developed framework is tested on several benchmark datasets to compare its performance with other popular decision tree models. The framework can generate simple models to replicate different rule-based hydrologic systems. The simple structure of the inducted models makes them easy to implement and use in any programming languages and modeling framework. The framework is employed to simulate controlled discharge from multiple reservoirs across the Contiguous United States (CONUS). The results revealed the potential of the GMT framework for generating reservoir routing models. The models inducted by the GMT framework can be employed as a reservoir module within the hydrologic models, especially in large scale hydrologic modelling. In addition, the GMT framework model structure can reveal useful information about the underlying structure of the system. Hence, it can reveal information about importance of decision variables in the real operation of the reservoir systems.

The proposed tools benefit stakeholders to understand and improve their management practices of reservoir systems. The simulation method reveals the hydrologic response of natural systems to historical reservoir operation through reservoir routing, while shed light on the importance of decision variable in real operation of the system. The optimization algorithm finds the optimum operational rules for the reservoir systems for the specified goal and objective. The outcome of these two algorithms can be used for evaluating reservoir management practices, while considering the hydrologic behavior of the watersheds.

Chapter 1

Introduction

1.1 Reservoir Operation

Spatial and temporal variability of precipitation make reservoirs a vital component of water resources systems (Simonovic, 1992; Chang and Chang, 2001). Reservoirs play an essential role in providing resilience against flood and drought (Mehran et al., 2015) and provide a wide range of services including water supply, hydropower electricity, recreation and ecosystem protection (Simonovic, 1992) to our society. Due to the multi-user and multi-objective nature of these systems, reservoir operators need to address the conflicts among different targets (Ahmad et al., 2014). In other words, the goal of reservoir operation is to maximize benefits, minimize costs and meet various water demands considering the conditions of the system (Rani and Moreira, 2010).

In spite of the extensive studies in the field of reservoir operation (Hejazi et al., 2008), many researchers acknowledged the existing gap between the theory and real-world operation of these systems (Yeh and Becker, 1982; Simonovic, 1992; Hejazi et al., 2008; Yang et al., 2016). Also, many reservoirs fall short in producing the level of benefits projected for these systems

(Labadie, 2004; Azamathulla et al., 2008). Hence, shortcoming of reservoir operation and the existing gap between theory and real-world operation urge for further investigation into the real operation of the reservoir systems (Hejazi et al., 2008). Furthermore, increase in the frequency and severity of extreme events (Cheng and AghaKouchak, 2014) and changes in land-cover and land-use impose more stress and risk on the dams and reservoirs (Vahedifard et al., 2017; Mallakpour et al., 2019). Hence, more effective and efficient operation of reservoirs are required to mitigate the effect of these changes (Ahmad et al., 2014).

To achieve these goals, tools and algorithms are needed to understand the operational procedure of the reservoir systems and improve these operational rules. Nonetheless, modifying the reservoir operation can alter the downstream natural flow, which results in changes in hydrologic response of the natural systems. Hence, models are required to understand and simulate the real operation of the reservoir systems and replicate the controlled discharge from these systems. These models can aid decision makers to understand the consequence of their decision making or evaluate the current operational rules of the reservoir system. Also, these models shed light on the potentials for improving the existing reservoir operation. In addition to that, reservoir simulation models can significantly enhance the performance of hydrologic models in simulating streamflow (Tavakoly et al., 2017).

So, understanding and improving the operation of the reservoirs are coined with the simulation and optimization of these systems. Thus, in this dissertation I emphasize these two important aspects of reservoir operation. First, I focus on the optimization aspect of reservoir systems and introduce a framework for optimizing the controlled discharge from dams. Second, a data-driven framework will be introduced to mimic human decision making in reservoir systems and simulate the controlled discharge based on the historical data. Application of these two frameworks on reservoir systems are demonstrated through various case studies across CONUS.

In this chapter, a brief background on reservoir optimization and simulation is presented. Then, challenges and opportunities for optimization and simulation of the reservoir systems

are summarized. At last, the research objectives and approaches are listed at the end of this chapter.

1.2 Reservoir Optimization

Optimizing reservoir systems has been a popular area of research in water resources management (Hejazi et al., 2008). In fact, reservoir optimization is the source of motivation for development of many optimization algorithms (Li et al., 2010; Haddad et al., 2006; Afshar and Shahidi, 2009). Harvard Water Program (Maass et al., 1962) was the group that first introduced the systems method, and in particular the application of optimization approach in water resources systems modeling (Meier Jr and Beightler, 1967). Since then optimization techniques have found many applications in the water resources planning. In general, the optimization problems for reservoir systems are formulated with the goal to maximize benefits for stakeholders (Ahmad et al., 2014). All optimization problems have two common elements, objective functions and constraints (Belaineh et al., 1999). Hence, the general form of the optimization problems can be demonstrated as (Farmani and Wright, 2003; Marler and Arora, 2004):

$$\begin{aligned} \text{minimize } \mathbf{F} &= [F_1(\mathbf{X}), F_2(\mathbf{X}), \ldots, F_p(\mathbf{X})]^T \\ F_i(x) &= f(x_1, \ldots, x_n), \end{aligned} \tag{1.1}$$

subject to inequality constraints,

$$g_j(\mathbf{X}) \leq 0, (j = 1, \ldots, q), \tag{1.2}$$

and equality constraints,

$$h_j(\mathbf{X}) = 0, (j = q + 1, \ldots, m), \tag{1.3}$$

where x is the decision variable and F is the objective function. The objective functions and constraints address the management goals and physical constraints of the system (Belaineh et al., 1999). The optimization algorithms are usually selected according to the characteristics of these two elements of the optimization problems.

The evolution of application of optimization theory in reservoir operation started with linear programming (LP) and dynamic programming (DP) and to more advanced nonlinear tools in par with the improvement of computation power of computers. LP is concerned with solving problems in which all the relationships are linear among the variables in the constraints and objective function (Yeh, 1985). LP has been widely used for reservoir operation due to available solvers (Belaineh et al., 1999), flexibility in solving large scale reservoir problems, and convergence to global solutions (Rani and Moreira, 2010). For instance, Dorfman (1962) employed LP to optimize an economic objective function for a reservoir system by tuning the storage capacity and releases, Revelle et al. (1969) employed LP to find the optimum release during reservoir operation, Loucks and Dorfman (1975) applied LP to solve a chance-constraint reservoir model to optimize storage and release limits, and Needham et al. (2000) employed LP to analyze the optimal operation policy for reservoir systems. Although the LP algorithm is efficient in solving large scale problems, the algorithm can be used for linear and convex objective functions and constraints only (Rani and Moreira, 2010). However, most reservoir optimization problems deal with nonlinear objective functions and constraints (Ahmad et al., 2014; Rani and Moreira, 2010). In such problems, the objective functions and constraints can be approximated by linear functions (Mays, 1989; Needham et al., 2000; Garcia-Gonzalez and Castro, 2001) or successive LP (Rani and Moreira, 2010). However, nonlinear constraints and objective functions are more suitable for reservoir problems (Needham et al., 2000). Also, in addition to traditional objectives such as flood control, water supply, hydropower production, and navigation, operators are interested in other objectives and constraints such as water quality, riparian habitat, and recreation interests (Zagona et al., 2001) which increase the complexity and nonlinearity of the reservoir problems.

Nonlinear reservoir optimization problems can be solved by nonlinear programming (NLP) and DP. However, DP can handle complex nonlinear optimization problems by decomposing them into more simple problems (Meier Jr and Beightler, 1967). Hall and Buras (1961) were the first to employ dynamic programming for reservoir optimization problems (Meier Jr and Beightler, 1967). Thence, these type of algorithms have been used in reservoir optimization problems to maximize return from reservoir systems (Hall et al., 1968), maximize power generation over a specific period of time (Chu et al., 1979), and optimize water release for a reservoir network (Cervellera et al., 2006). However, these algorithms are not capable of solving large and sparse problems (Mays, 1989) and suffer from the curse of dimensionality (Yakowitz, 1982; Ahmad et al., 2014).

Although mathematical programming optimization tools attracted a great deal of attention in the field of water resources management and hydrology, they are incapable of solving many nonlinear, nonconvex, discontinuous, discrete, and multiobjective optimization problems (Rani and Moreira, 2010). Metaheuristic optimization algorithms have been introduced to overcome these shortcomings (Rani and Moreira, 2010) and have gained more application due to their flexibility in handling nonlinearity and uncertainty in problem (Rani and Moreira, 2010; Sadegh, 2015). In general, the metaheuristic algorithms can be divided into population-based and single point-based algorithms (Maier et al., 2014). The population-based algorithms such as Particle Swarm Optimization (PSO) (Kennedy, 2011), Genetic Algorithm (GA) (Goldberg and Holland, 1988; John, 1992), Differential Evolution (DE) (Storn and Price, 1997), and Shuffled Complex Evolution developed at University of Arizona (SCE-UA) (Duan et al., 1993), combine a sample of points from a population to generate a new solution which is superior to other points (Roeva et al., 2014). In contrast, single point-based methods such as Simulated Anealing (SA) (Kirkpatrick et al., 1983) and Tabu Search (TS) (Glover, 1989, 1990) improve each point by searching neighboring regions (Roeva et al., 2014). Many different algorithms have been proposed for these two categories.

Although numerous metaheuristic optimization algorithms have been proposed, the No Free

Lunch (NFL) theorem (Wolpert and Macready, 1997) implies that none of these algorithms are superior to others for all optimization problems (Maier et al., 2014). The NFL theorem has been the source of motivation for developing and improving the metaheuristic optimization algorithms (Mirjalili et al., 2014; Naeini et al., 2018). Due to the nature of the metaheuristic algorithms, their performance varies based on the characteristics of the problem space. Yet, studies on the performance of the algorithms revealed the problem-specific performance of particulate implementation of an algorithm (Kollat and Reed, 2006). This characteristic of metaheuristic optimization algorithms incites efforts for the development of hybrid and auto adaptive optimization algorithms (Maier et al., 2014) to tackle a broad class of optimization problems. In this regard, several algorithms have been developed which employ two or more evolutionary algorithms (Vrugt et al., 2009; Hadka and Reed, 2013), or tune their settings according to the problem (Qin and Suganthan, 2005). These self adaptive optimization algorithms are more flexible and can be used for a wide range of problems (Naeini et al., 2018).

This feature of self adaptive optimization algorithms encouraged me to investigate the development of the self adaptive concept for other optimization algorithms. To this end, the SCE-UA algorithm, which is one of the most popular optimization algorithms in the field of water resources management, is selected for the development of a self adaptive hybrid optimization algorithm. My research led to development of a new version of SCE-UA, titled Shuffled Complex-Self Adaptive Hybrid EvoLution (SC-SAHEL). The SC-SAHEL algorithm is detailed in chapter 2. Chapter 3 demonstrates the application of the SC-SAHEL framework for a reservoir optimization problem.

1.3 Reservoir Simulation

Although optimization tools play a significant role in improving the reservoir operation, simulation models are important tools for understanding reservoir systems (Rani and Moreira, 2010). In general, the reservoir simulation models can be categorize into two classes; the models which simulate the controlled discharge from reservoirs based on the operational rules of the system, so called pure simulation models (Rani and Moreira, 2010), and models which simulate controlled discharge or some objective function values based on the changes in the operational rules (Wang et al., 2005), so called simulation-optimization models (Rani and Moreira, 2010). The former is mostly used for reservoir routing, to incorporate the effect of lakes and reservoirs into hydrologic models, and to mimic human decision making in these systems (Giuliani and Herman, 2018). The latter is mostly used to derive the objective function values for decision variables obtained from optimization algorithms (Neelakantan and Pundarikanthan, 1999). The simulation-optimization tools such as RiverWare (Zagona et al., 2001), CalSim (Draper et al., 2004), and MODSIM (Labadie, 2006), are mostly designed for water management and planning (Rani and Moreira, 2010).

Here, I mainly focus on the pure simulation models for the purpose of understanding and replicating human decision making in reservoir systems. Pure simulation models can be physical-based or data-driven. The physical-based models employ the operational rules of the reservoir within a physical framework to simulate the controlled discharge. Many of these reservoir models, such as HEC-ResSim (USACE, 2013) and RiverWare (Zagona et al., 2001), are embedded within the river system modelling framework. The application of these models can be restricted due to their complexity (Draper et al., 2004; Yang et al., 2016) and require information about the operational rules. Also, in many cases the operators may deviate from these operational rules (Oliveira and Loucks, 1997; Yang et al., 2016). Thus, the application of the physical-based models can be limited, specially for the large scale hydrologic modeling. Therefore, data-driven models can be used to overcome these obstacles

in reservoir routing. Data-driven models can be trained on the historical reservoir releases to mimic the real reservoir operation.

Among data-driven models, decision trees have the advantage of providing interpretable models, which can provide information about the underlying system. The transparent structure of the decision tree models is easy to understand and employ in practice (Yang et al., 2016). In recent years, decision tree models have found many applications in the field of water resources management (Castelletti et al., 2010; Galelli and Castelletti, 2013b; Yang et al., 2016, 2017b; Han et al., 2018). A popular class of decision tree algorithms is model trees (MTs) which was developed for continuous spaces (Quinlan et al., 1992; Wang and Witten, 1997). MTs are suitable for representing continous spaces and have been widely used for representing water resource systems (Kompare et al., 1997; Solomatine and Dulal, 2003; Štravs and Brilly, 2007; Jothiprakash and Kote, 2011; Galelli and Castelletti, 2013a). In this dissertation, I focus on this class of decision tree algorithms for reservoir routing. Multiple decision tree and model tree algorithms are investigated for simulating the controlled discharge from reservoir systems. Also, a new MT framework titled, generalized model tree (GMT), is proposed to simulate the reservoir releases and capture the variability of the flow. To enhance the efficiency and robustness of the algorithm, several modules have been implemented into the GMT framework. The newly developed algorithm is applied to multiple benchmark data sets and compared to other popular decision trees. Application of the GMT framework for reservoir routing is demonstrated through multiple reservoir case studies across CONUS.

1.4 Research Motivations and Approaches

The motivations for this research stemmed from the need to understand, simulate and improve the current operational rules of the reservoir systems. The main objective is to provide

tools and algorithms for decision makers and modelers to optimize and simulate reservoir systems. The proposed frameworks are tested and evaluated on multiple case studies to show their potential application to real reservoir problems. The main objectives and approaches of this dissertation are as follows:

Objective 1:

The first objective of this dissertation is to provide a self adaptive optimization framework for optimizing reservoir problems. Due to the increasing number of optimization algorithms and the complexity of the optimization problems, a self adaptive optimization framework can solve a wide range of problems, without the need to find the best optimization algorithms and settings for the problem at hand. In general, my goal is to develop a flexible, robust, and efficient optimization algorithm for the reservoir systems.

Approach 1:

To achieve the first objective, I employ the structure of the Shuffled Complex Evolution developed at University of Arizona, as the cornerstone for developing a robust optimization algorithm. I implement several search methods into the SCE-UA framework, so the algorithm can automatically select the best search method for the problem space at each stage of the optimization process. The new hybrid algorithm can reveal the performance of the search methods during the course of the search.

Objective 2:

The next objective of this dissertation is to understand and simulate the real operation of the reservoir systems. My goal is to provide a tool and algorithm to mimic human decision making in the operation of dams and reservoirs. Hence, these systems can be simulated within the hydrologic models.

Approach 2:

To achieve the second objective, I employ decision tree algorithms. In contrast to other data-mining algorithms, decision trees can reveal information about the underlying process, while they represent the system. Here, I employ the MTs, which are suitable for continuous spaces, to simulate and understand the controlled discharge from reservoir systems. Also, a new framework is developed to overcome some of the shortcomings of the existing MTs for reservoir routing.

1.5 Scope of the Dissertation

The rest of this dissertation is organized as follows: Chapter 2 introduces a self adaptive hybrid evolution optimization framework. Since the previous investigations on optimization algorithms have revealed the advantage of the self adaptive optimization algorithms, here, I focus on the development of this type of optimization tools for reservoir systems. In chapter 2, I also test and evaluate the new optimization algorithm on various mathematical test functions. Chapter 3 verifies the application of the newly developed optimization framework on a conceptual reservoir case study. The reservoir optimization problem is defined based on the information obtained for the Folsom reservoir. In Chapter 4 a new data-mining framework is developed for the rule-based systems. The framework is tested on multiple continuous benchmark data-sets in this chapter. Chapter 5 is devoted to the application of the newly developed framework for simulating controlled discharge and reservoir routing. In this chapter multiple reservoir systems with different range of services are selected, and the new data-mining algorithm is trained and tested on the historical data. Chapter 6 summarize the findings, limitations and future directions.

Chapter 2

Developing a Shuffled Complex-Self Adaptive Hybrid EvoLution (SC-SAHEL) Optimization Framework

2.1 Introduction

Metaheuristic optimization algorithms have gained a great deal of attention in science and engineering (Blum and Roli, 2003; Lee and Geem, 2005; Nicklow et al., 2009; Reed et al., 2013; BoussaïD et al., 2013; Maier et al., 2014). Simplicity and flexibility of these algorithms, along with their robustness make them attractive tools for solving optimization problems (Lee and Geem, 2005; Coello et al., 2007). Many of the metaheuristic algorithms are inspired by a physical phenomenon, such as animals social and foraging behavior and natural selection. For example, Simulated Annealing (Kirkpatrick et al., 1983), Big Bang-Big Crunch (Erol and

Eksin, 2006), Gravitational Search Algorithm (Rashedi et al., 2009), and Charged System Search (Kaveh and Talatahari, 2010) are all inspired by various physical phenomena. Ant Colony Optimization (Dorigo et al., 1996), Particle Swarm Optimization (Kennedy, 2011), Bat-inspired Algorithm (Yang, 2010), Firefly Algorithm (Yang, 2009), Dolphin Echolocation (Kaveh and Farhoudi, 2013), Grey Wolf Optimizer (Mirjalili et al., 2014), Bacterial Foraging (Passino, 2002), Genetic Algorithm (Goldberg and Holland, 1988; John, 1992), and Differential Evolution (Storn and Price, 1997) are examples of algorithms inspired by animal's social and foraging behavior, and the natural selection mechanism of Darwin's evolution theorem. According to the No-Free-Lunch (NFL) (Wolpert and Macready, 1997) theorem, none of these algorithms are consistently superior to others over a variety of problems, although some of them may outperform others on a certain type of optimization problem.

The NFL theorem has been a source of motivation for developing optimization algorithms (Woodruff et al., 2013; Mirjalili et al., 2014). It has encouraged scientists and researchers to combine the strengths of different algorithms and devise more robust and efficient optimization algorithms that suit a broad class of problems (Qin and Suganthan, 2005; Vrugt and Robinson, 2007; Vrugt et al., 2009; Hadka and Reed, 2013; Sadegh et al., 2017). These efforts led to the emergence of multi-method and self-adaptive optimization algorithms such as Self-adaptive DE algorithm (SaDE) (Qin and Suganthan, 2005), A Multialgorithm Genetically Adaptive Method for Single Objective Optimization (AMALGAM-SO) (Vrugt and Robinson, 2007; Vrugt et al., 2009) and Borg (Hadka and Reed, 2013). They all regularly update the search mechanism during the course of optimization according to the information obtained from the response surface.

Here, I propose a new self-adaptive hybrid optimization framework, titled Shuffled Complex-Self Adaptive Hybrid EvoLution (SC-SAHEL). The SC-SAHEL framework employs multiple Evolutionary Algorithms (EAs) as search cores, and enables competition among different algorithms as optimization run progresses. The proposed framework differs from other multi-method algorithms as it grants independent evolution of the population by each EA. In this

framework, the population is partitioned into equally sized groups, so-called complexes, each assigned to different EAs. Number of complexes assigned to each EA is regularly updated according to their performance. In general, the newly developed framework has two main characteristics. First, all the EAs evolve population in a parallel structure. Second, each participating EA works independent of other EAs. The architecture of SC-SAHEL is inspired by the concept of the Shuffled Complex Evolution algorithm - University of Arizona (SCE-UA) (Duan et al., 1992). The SCE-UA algorithm is a population-evolution based algorithm (Madsen, 2003), which evolves individuals by partitioning population into different complexes. The complexes are evolved for a specific number of iterations independent of other complexes, and then are forced to shuffle.

The SCE-UA framework employs Nelder-Mead simplex (Nelder and Mead, 1965) technique along with the concept of controlled random search (Price, 1987), clustering (Kan and Timmer, 1987), competitive evolution (John, 1992) and complex shuffling (Duan et al., 1993) to offer a global optimization strategy. By employing these techniques, the SCE-UA algorithm provides a robust optimization framework and has shown numerically to be competitive and efficient comparing to other algorithms, such as GA, for calibrating rainfall-runoff models (Gan and Biftu, 1996; Wagener et al., 2004; Wang et al., 2010; Beven, 2011). The SCE-UA algorithm has been widely used in water resources management (Sorooshian et al., 1993; Yapo et al., 1996; Madsen, 2000; Toth et al., 2000; Eckhardt and Arnold, 2001; Ajami et al., 2004; Liong and Atiquzzaman, 2004; Lin et al., 2006; Barati et al., 2014; Yang et al., 2015), as well as other fields of study, such as pyrolysis modeling (Ding et al., 2016; Hasalová et al., 2016) and Artificial Intelligence (Yang et al., 2017a).

Application of the SCE-UA is not limited to solving single objective optimization problems. The Multi-Objective Complex evolution, University of Arizona (MOCOM-UA), is an extension of the SCE-UA for solving multi-objective problems (Yapo et al., 1998; Boyle et al., 2000). Besides, the SCE-UA architecture has been used to develop Markov Chain Monte Carlo (MCMC) sampling, named Shuffled Complex Evolution Metropolis algorithm (SCEM-

UA) and the Multi-Objective Shuffled Complex Evolution Metropolis (MOSCEM) to infer posterior parameter distributions of hydrologic models (Vrugt et al., 2003a,b). The Metropolis scheme is used as the search kernel in the SCEM-UA and MOSCEM-UA (Vrugt et al., 2003a,b; Chu et al., 2010). There is also an enhanced version of SCE-UA, which is developed by Chu et al. (2011) entitled the Shuffled Complex strategy with Principle Component Analysis, developed at the University of California, Irvine (SP-UCI). Chu et al. (2011) found that the SCE-UA algorithm may not converge to the best solution on high-dimensional problems due to population degeneration phenomenon. The population degeneration refers to the situation when the search particles span a lower dimension space than the original search space (Chu et al., 2010), which causes the search algorithm to fail in finding the global optimum. To address this issue, the SP-UCI algorithm employs Principle Component Analysis (PCA) in order to find and restore the missing dimensions during the course of search (Chu et al., 2011).

Both SCE-UA and SP-UCI start the evolution process by generating a population within the feasible parameters space. Then, population is partitioned into different complexes, and each complex is evolved independently. Each member of the complex has the potential to contribute to offspring in the evolution process. In each evolution step, more than two parents may contribute to generating offspring. To make the evolution process competitive, a triangular probability function is used to select parents. As a result, the fittest individuals will have a higher chance of being selected. Each complex is evolved for a specific number of iterations, and then complexes are shuffled to globally share the information attained by individuals during the search.

The Competitive Complex Evolution (CCE) and Modified Competitive Complex Evolution (MCCE) are the search cores of the SCE-UA and SP-UCI algorithm, respectively. The CCE and MCCE evolutionary processes are developed based on Nelder-Mead (Nelder and Mead, 1965) method with some modification. The evolution process in the SCE-UA is not limited to these algorithms. In fact, several studies have incorporated different EAs into the structure

of the SCE-UA algorithm. For example, the Frog Leaping (FL) is developed by adapting Particle Swarm Optimization (PSO) algorithm to the SCE-UA structure for solving discrete problems (Eusuff and Lansey, 2003; Eusuff et al., 2006). Mariani et al. (2011) proposed an SCE-UA algorithm which employs DE for evolving the complexes. These studies revealed the flexibility of the SCE-UA in combination with other types of EAs; however, the potential of combining different algorithms into a hybrid shuffled complex scheme has not been investigated.

The unique structure of the SCE-UA algorithm along with the flexibility of the algorithm for using different EAs, motivated us to use the SCE-UA as the cornerstone of the SC-SAHEL framework. The SC-SAHEL algorithm employs multiple EAs for evolving the population in a similar structure as that of the SCE-UA, with the goal of selecting the most suitable search algorithm at each optimization step. On the one hand, some EAs are more capable of visiting the new regions of the search space and exploring the problem space, and hence are particularly suitable at the beginning of the optimization (Olorunda and Engelbrecht, 2008). On the other hand, some EAs are more capable of searching within the visited regions of the search space, and hence boosting the convergence process after finding the region of interest (Mirjalili and Hashim, 2010). Balancing between these two steps, which are referred to as exploration and exploitation (Moeini and Afshar, 2009), is a challenging task in stochastic optimization methods (Črepinšek et al., 2013). The SC-SAHEL algorithm maintains a balance between exploration and exploitation phases by evaluating the performance of participating EAs at each optimization step. EAs contribute to the population evolution according to their performance in previous steps. The algorithms' performance is evaluated by comparing the evolved complexes before and after evolution. In this process, the most suitable algorithm for the problem space become the dominant search core.

In this study, four different EAs are used as search cores in the proposed SC-SAHEL framework, including Modified Competitive Complex Evolution (MCCE) used in the SP-UCI algorithm, Modified Frog Leaping (MFL), Modified Grey Wolf Optimizer (MGWO), and

Differential Evolution (DE). To better illustrate the performance of the hybrid SC-SAHEL algorithm, the framework is benchmarked over 29 test functions and compared to SC-SAHEL with single EA. Among the 29 employed test functions, there are 23 classic test functions (Yao et al., 1999) and 6 composite test functions (Liang et al., 2005), which are commonly used as benchmarks in comparing optimization algorithms. All the problems presented here are minimization problems.

The rest of this chapter is organized as follows. In section 2.2, structure of the SC-SAHEL algorithm and details of four EAs are presented. Section 2.3 presents the test functions, settings of the experiments, and results obtained for each test function. Finally, in section 2.4, I draw conclusion, summarize some limitations about the newly introduced framework, and suggest some directions for future work.

2.2 Methodology

The SC-SAHEL algorithm is a parallel optimization framework, which is built based on the original SCE-UA architecture. SC-SAHEL, however, differs from the original SCE-UA algorithm by using multiple search mechanisms instead of only employing the Nelder-Mead simplex downhill method. In this section, I first introduce the main structure of SC-SAHEL. Then, I present four different EAs, which are employed as search cores in the SC-SAHEL framework. These algorithms are selected for illustrative purpose only and can be replaced by other evolutionary algorithms. Some modifications are made to the original form of these algorithms, to allow fair competition between EAs. These modifications are detailed in appendix A-D.

2.2.1 The SC-SAHEL Framework

The proposed SC-SAHEL optimization strategy starts with generating a population with a pre-defined sampling method within feasible parameters' range. The framework supports user-defined sampling methods, besides built-in Uniform Random Sampling (URS) and Latin Hypercube Sampling (LHS). The population is then partitioned into different complexes. The partitioning process warrants maintaining diversity of population in each complex. In doing so, population is first sorted according to (objective) function values. Then, sorted population is divided into NGS equally-sized groups (NGS being the number of complexes), ensuring that members of each group have similar objective function values. Each complex subsequently will randomly select a member from each of these groups. This procedure maintains diversity of the population within each complex. The complexes are then assigned to EAs and evolved. In contrast to the original concept of the SCE-UA, the complexes are evolved with different EAs rather than single search mechanism. At the beginning of the search, an equal number of complexes is assigned to each evolutionary method. For instance, if population is partitioned into 8 complexes and 4 different EAs are used, each algorithm will evolve 2 complexes independently (2-2-2-2). After evolving the complexes for pre-specified number of steps, the Evolutionary Method Performance (EMP) metric (Eq. (1)) will be calculated for each EA,

$$\text{EMP} = \frac{\text{mean}(F) - \text{mean}(F_N)}{\text{mean}(F)}, \tag{2.1}$$

in which, F and F_N are objective function values of individuals in each complex before and after evolution, respectively.

The EMP metric measures change in the mean objective function value of individuals in each complex in comparison to their previous state. A higher EMP value indicates a larger reduction in the mean objective function value obtained by the individuals in the complex. The performance of each evolutionary algorithm is then evaluated based on the mean value

of EMP calculated for each evolved complex. EAs are then ranked according to the EMP values. Ranks are in turn used to assign number of complexes to each evolutionary method for the next iteration. The highest ranked algorithm will be assigned an additional complex to evolve in the next shuffling step, while, the lowest ranked evolutionary algorithm will lose one complex for the next step. For instance, if all the EAs have 2 complexes to evolve (2-2-2-2 case), the number of complexes assigned to each EA can be updated to 3-2-2-1. In other words, this logic is an award and punishment process, in which the algorithm with best performances will be awarded with an additional complex to evolve in the next iteration, while the worst-performing algorithm will be punished by losing one complex.

It is worth mentioning that as some of the algorithms may have poor performance in the exploration phase, they might lose all their complexes during the adaptation process. This might be troublesome as these algorithms may be superior in the exploitation phase. If such algorithms are terminated in the exploration phase, they cannot be selected during the convergence steps. Hence, EAs termination is avoided to fully utilize the potential of EAs in all the optimization steps and balance the exploration and exploitation phases. The minimum number of complexes assigned to each evolutionary method is restricted to at least 1 complex in this case. If the lowest ranked EA has only 1 complex to evolve, it won't lose its last complex. If an algorithm outperforms others throughout the evolution of complexes, the number of complexes assigned to the superior EA will be equal to the total number of complexes minus the number of EAs plus one. In this case, all other algorithms are evolving one complex only. As all algorithms are evolving at least one complex, they have the chance to outperform other EAs and gain more complexes during the optimization process, and to potentially become the dominant search method as the search continues toward exploitation phase. Figure 2.1 briefly shows the flowchart of the SC-SAHEL algorithm, pseudo code of which is as follows:

0. Initialization. Select NGS > 1 and NPS (suggested NPS $> 2n+1$, where n is dimension

of the problem), where NGS is the number of complexes and NPS is the number of individuals in the complexes. NGS should be proportional to the number of evolutionary algorithms so that all the participating EAs have an equal number of complexes at the beginning of the search.

1. Sample NPT points in the feasible parameter space using a user-defined sampling method, where NPT equals to NGS $\times$ NPS. Compute objective function value for each point.
2. Rank and sort all individuals in the order of increasing objective function value.
3. Partition the entire population into complexes. Assign complexes to the participating EAs.
4. Monitor and restore population dimensionality using PCA algorithm (Optional).
5. Evolve each complex using the corresponding EA.
6. After evolving the complexes for a pre-defined number of iterations, calculate the mean EMP for each EA.
7. Rank the participating EAs according to the mean EMP value of each evolutionary method. The highest ranked method will get additional complex in the next iteration, while the worst evolutionary method will lose one.
8. Shuffle complexes and form a new population.
9. Check whether the convergence criteria are satisfied, otherwise go to step 3.

SC-SAHEL allows for different settings that can influence the performance of the algorithm. Careful consideration should be devoted to the selection of these settings, including number of complexes, number of individuals within each complex, number of evolution steps before each shuffling, and stopping criteria thresholds. Some of these settings are adopted

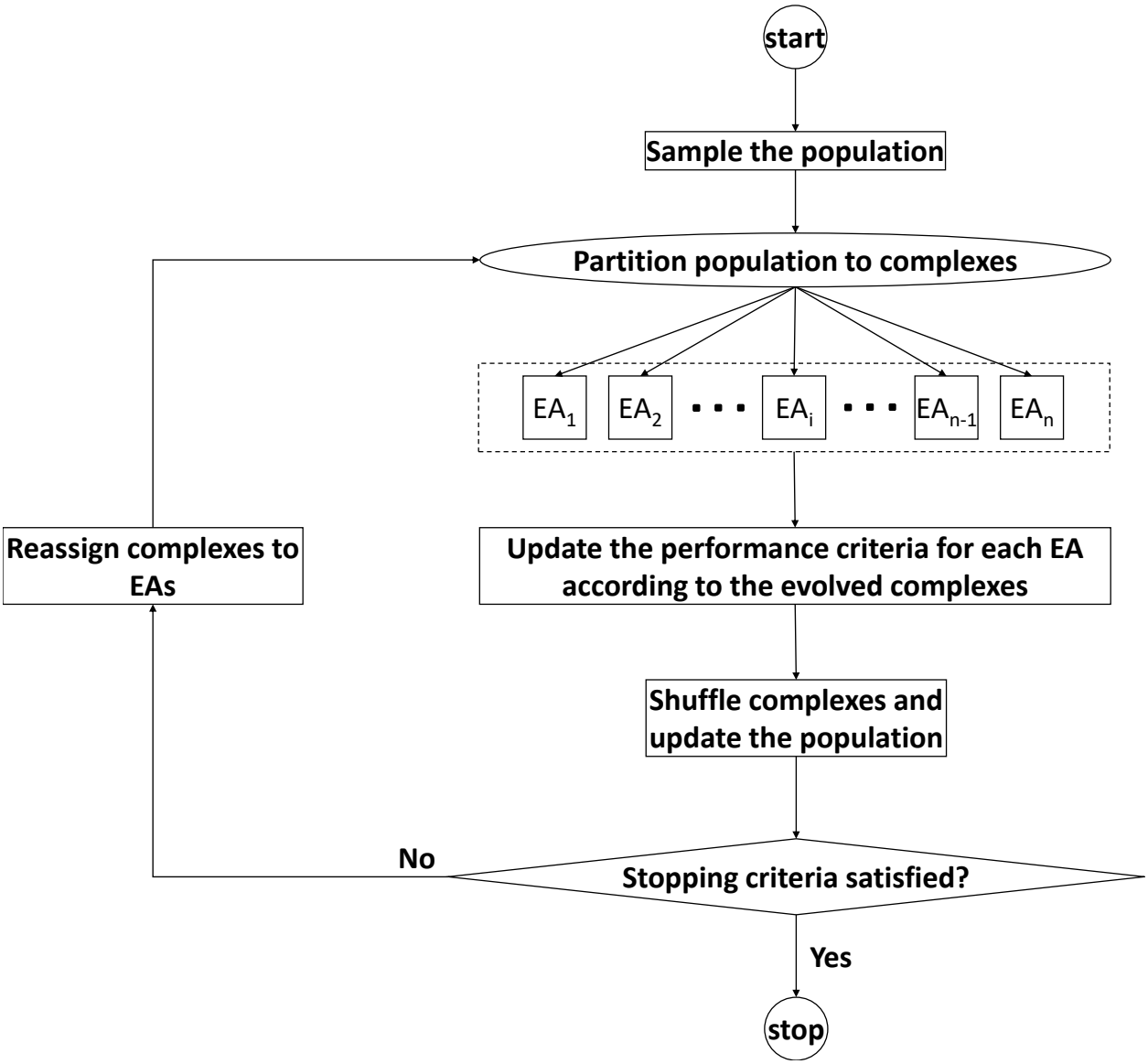


Figure 2.1: The SC-SAHEL framework flowchart.

from the suggested settings for the SCE-UA. For instance, the number of individuals within each complex is set to $2d + 1$, where d is dimension of the problem. However, some of the suggested settings cannot be applied to the SC-SAHEL framework due to use of different EAs. These settings can be changed according to the complexity of the problem and the EAs employed within the framework. For instance, the number of complexes, the number of points within each complex, and the number of evolution steps before each shuffling are problem dependent.

The SC-SAHEL framework employs three different stopping criteria which are adopted from SCE-UA and SP-UCI. These stopping criteria include number of function evaluations, range of samples that span the search space, and improvement in the objective function value in the last m shuffling steps. These criteria are compared to pre-defined thresholds, which can in turn be tuned according to the complexity of the problem. Improper selection of these thresholds may lead to early or delayed convergence.

2.2.2 Evolutionary Algorithms Employed Within SC-SAHEL

In this chapter, I employ four different EAs to illustrate the flexibility of the SC-SAHEL framework in adopting various EAs and show the algorithms competition. These algorithms are briefly presented here. The pseudo code and details of these algorithms can be found in Appendix A-D.

Modified Competitive Complex Evolution (MCCE)

The MCCE algorithm is an enhanced version of CCE algorithm used in the SCE-UA framework; which provides a robust, efficient, and effective EA for exploring and exploiting the search space. The MCCE algorithm is developed based on the Nelder-Mead algorithm, how-

ever, Chu et al. (2011) found that the shrink concept in the Nelder-Mead algorithm can cause premature convergence to a local optimum. Interested readers can refer to (Chu et al., 2010, 2011) for further details on MCCE algorithm. The pseudo code of the MCCE algorithm is detailed in Appendix A. SC-SAHEL has similar performance to SP-UCI, when the MCCE algorithm is used as the only search mechanism and PCA and resampling settings of SP-UCI are enabled. For simplification and comparison, SC-SAHEL with the MCCE algorithm as search core is referred as SP-UCI, hereafter.

Modified Frog Leaping (MFL)

The Frog Leaping (FL) algorithm uses adapted PSO algorithm as a local search mechanism within the SCE-UA framework (Eusuff and Lansey, 2003). FL has shown to be an efficient search algorithm for discrete optimization problems, and can find optimum solution much faster as compared to the GA algorithm (Eusuff et al., 2006). In order to adapt the FL algorithm to the SC-SAHEL parallel framework, I introduce a slightly modified version of FL algorithm entitled MFL. Further details and pseudo code of the MFL can be found in Appendix B. The original FL algorithm and the MFL have four main differences. First, the original FL is designed for discrete optimization problems, however, the MFL is modified for continuous domain. Second, the modified FL uses the best point in the subcomplex for generating new points, however, in the original FL framework new points are generated using the best point in the complex and the entire population. The reason for this modification is to avoid using any external information by participating EAs. In other words, the amount of information given to each EAs is limited to the complex assigned to the EAs. Third, as the MFL algorithm only uses the best point within the complex for generating the new generation, two different jump rates are used. The reason for different jump rates is to allow MFL to have a better exploration and exploitation ability during optimization process. These jump rates are selected by trial and error and may need further investigation to achieve a

better performance by MFL algorithm. Fourth, when the generated offspring is not better than the parents, a new point is randomly selected within the range of individuals in the subcomplex. This process, which is referred to as censorship step in the FL algorithm (Eusuff et al., 2006), is different from the original algorithm. The MFL algorithm uses the range of points in the complex rather than the whole feasible parameters range. Resampling within the whole parameter space can decrease the convergence speed of the FL algorithm. Hence, the resampling process is carried out only within the range of points in the complex. Hereafter, the SC-SAHEL with MFL algorithm as the only search core is referred as SC-MFL.

Modified Grey Wolf Optimizer (MGWO)

The Grey Wolf Optimizer is a metaheuristic algorithm inspired by the social hierarchy and hunting behavior of grey wolves (Mirjalili et al., 2014, 2016). Grey wolves hunting strategy has three main steps: first, chasing and approaching the prey; second, encircling and pursuing the prey, and finally attacking the prey (Mirjalili et al., 2014). The GWO process resembles the hunting strategy of the grey wolves. In this algorithm, the top three fittest individuals are selected and contribute to the evolution of population. Hence, the individuals in the population are navigated toward the best solution. The GWO algorithm has shown to be effective and efficient in many test functions and engineering problems. Furthermore, performance of GWO is comparable to other popular optimization algorithms, such as GA and PSO (Mirjalili et al., 2014). GWO follows an adaptive process to update the jump rates, to maintain balance between exploration and exploitation phases. The adaptive jump rate of the GWO is removed here and 3 different jump rates are used instead. The reason for this modification is that the information given to each EA is limited to its assigned complex. Similar to MFL algorithm, the modified GWO (MGWO) algorithm uses the range of parameters to resample individuals, when the generated offspring are not superior to their parents. Details and pseudo code of the MGWO algorithm can be found in Appendix C.

Hereafter, the SC-SAHEL with MGWO algorithm as the only search core is referred as SC-MGWO.

Differential Evolution (DE)

The DE algorithm is a powerful but simple heuristic population-based optimization algorithm (Qin and Suganthan, 2005; Sadegh and Vrugt, 2014) proposed by Storn and Price (1997). In 2011, Mariani et al. (2011) integrated the DE algorithm into SCE-UA framework and showed that the new framework is able to provide more robust solutions for some optimization problems in comparison to the SCE-UA. Similar to the work by Mariani et al. (2011), I use a slightly modified DE algorithm based on the concepts from Qin and Suganthan (2005), in order to integrate the DE algorithm into the SC-SAHEL framework. As the DE algorithm has slower performance in comparison to other EAs used here, I have added multiple steps to the DE. Here, the DE algorithm uses three different mutation rates in three attempts. In the first attempt, the algorithm uses a larger mutation rate. This helps exploring the search space with larger jump rates. In the second attempt, the algorithm reduces the mutation rate to a quarter of the first attempt. This will enhance the exploitation capability of the EA. If none of these mutation rates could generate a better offspring than the parents, in the next attempt the mutation rate is set to half of the first attempt. Lastly, if none of these attempts generate a better offspring in comparison to the parents, a new point is randomly selected within the range of individuals in the complex. The pseudo code of the modified DE algorithm is detailed in Appendix D. The SC-SAHEL algorithm is referred to as SC-DE, when the DE algorithm is used as the only search algorithm.

2.3 Conceptual Test Functions

2.3.1 Test Functions

The SC-SAHEL framework is benchmarked over 29 mathematical test functions using single-method and multi-method search mechanisms. This includes 23 classic test functions obtained from Yao et al. (1999). The name and formulation of these functions along with their dimensionality and range of parameters are listed in Table 2.1. I selected these test functions as they are standard and popular benchmarks for evaluating new optimization algorithms (Mirjalili et al., 2014). The remaining 6 are composite test functions, cf_{1-6}, (Liang et al., 2005), which represent complex optimization problems. Details of the composite test functions can be found in the work of Liang et al. (2005) and Mirjalili et al. (2014). Classic test functions have dimensions in the range of 2 to 30, and all the composite test functions are 10 dimensional. Figure 2.2 and 2.3 show response surface of the test functions which can be shown in 2-dimension form. The SC-SAHEL settings used for optimizing these test functions are listed in Table 2.2 for each test function. Number of points in each complex and number of evolution steps for each complex are set to $2d+1$ and $\max(d+1, 10)$, respectively, where d is the dimension of the problem. The number of evolution steps is set to $\max(d+1, 10)$, to guarantee that EAs evolve the complexes for enough number of steps, before evaluating the EAs. In the high-dimension problems, the maximum number of function evaluation should be selected with careful consideration.

Several experiments were conducted to find an optimal set of parameters for the SC-SAHEL setting. These experiments revealed that a low number of evolutionary steps before shuffling the complexes, may not show the potential of the EAs. On the other hand, using a large value for the number of evolution steps may shrink the complex to a small space, which cannot span the whole search space (Duan et al., 1994). Maximum number of function evaluation is determined according to the complexity of the problem and is different for each

Function Number	Name	Function	Dim	Range	f_{min}
$f_1(x)$	Sphere Model	$f_x = \sum_{i=1}^{n} {x_i}^2$	30	[-100,100]	0
$f_2(x)$	Schwefel's Problem 2.22	$f(x) = \sum_{i=1}^{n} \lvert x_i \rvert + \prod_{i=1}^{n} \lvert x_i \rvert$	30	[-10,10]	0
$f_3(x)$	Schwefel's Problem 1.2	$f(x) = \sum_{i=1}^{n} (\sum_{j=1}^{i} x_j)^2$	30	[-100,100]	0
$f_4(x)$	Schwefel's Problem 2.21	$f(x) = max\{\lvert x_i \rvert, 1 \leq i \leq n\}$	30	[-100,100]	0
$f_5(x)$	Generalized Rosenbrock's Function	$f(x) = \sum_{i=1}^{n-1} \lvert 100(x_{i+1} - {x_i}^2) \rvert^2 + (x_i - 1)^2$	30	[-30,30]	0
$f_6(x)$	Step Function	$f(x) = \sum_{i=1}^{n} (\lfloor x_i + 0.5 \rfloor)^2$	30	[-100,100]	0
$f_7(x)$	Quartic Function	$f(x) = \sum_{i=1}^{n} i{x_i}^4 + rand[0,1)$	30	[-1.28,1.28]	0
$f_8(x)$	Generalized Schwefel's Problem2.26	$f(x) = \sum_{i=1}^{n} -x_i \sin \sqrt{\lvert x_i \rvert}$	30	[-500,500]	-12569.5
$f_9(x)$	Generalized Rastrigin's Function	$f(x) = sum_{i=1}^{n} [{x_i}^2 - 10\cos 2\pi x_i + 10]$	30	[-5.12,5.12]	0
$f_{10}(x)$	Ackley's Function	$f(x) = -20\exp(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n} {x_i}^2}) - \exp(\frac{1}{n}\sum_{i=1}^{n} \cos(2\pi x_i)) + 20 + e$	30	[-32,32]	0
$f_{11}(x)$	Generalized Griewank Function	$f(x) = \frac{1}{4000}\sum_{i=1}^{n} {x_i}^2 - \prod_{i=1}^{n} \cos(\frac{x_i}{\sqrt{i}}) + 1$	30	[-600,600]	0
$f_{12}(x)$	Generalized Penalized Functions	$f(x) = \frac{\pi}{n}(10\sin^2(\pi y) + \sum_{i=1}^{n-1}(y_i - 1)^2)[1 + 10sin^2(\pi y_{i+1})^2] + (y_n - 1)^2) + \sum_{i=1}^{n} u(x_i, 10, 100, 4), y_i = 1 + \frac{1}{4}(x_i + 1), u(x_i, a, k, m) = \begin{cases} k(x_i - a)^m, & \text{if } x_i > a. \\ 0, & -a \leq x_i \leq a. \\ k(-x_i - a)^m, & x_i < -a \end{cases}$	30	[-50,50]	0
$f_{13}(x)$	Generalized Penalized Functions	$f(x) = 0.1(\sin 3\pi x_1^2 + \sum_{i=1}^{n-1}(x_i - 1)^2[1 + \sin^2(3\pi x_{i+1})] + (x_n - 1)^2[1 + \sin^2(2\pi x_n)]) + \sum_{i=1}^{n} u(x_i, 5, 100, 4), u(x_i, a, k, m) = \text{same as} f_{12}(x)$	30	[-50,50]	0
$f_{14}(x)$	Shekel's Foxholes Function	$f(x) = [\frac{1}{500} + \sum_{j=1}^{25} \frac{1}{j + \sum_{i=1}^{2}(x_i - a_{ij})}]^{-1}$	2	[-65.536,65.536]	1
$f_{15}(x)$	Kowalik's Function	$f(x) = \sum_{i=1}^{11} [a_i - \frac{x_1({b_i}^2 + b_i x_2)}{{b_i}^2 + b_i x_3 + x_4}]^2$	4	[-5,5]	0.0003075
$f_{16}(x)$	Six-Hump Camel-Back Function	$f(x) = 4{x_1}^2 - 2.1{x_1}^4 + \frac{1}{3}{x_1}^6 + x_1 x_2 - 4{x_2}^2 + 4{x_2}^4$	2	[-5,5]	-1.0316
$f_{17}(x)$	Branin Function	$f(x) = (x_2 - \frac{5.1}{4\pi^2}{x_1}^2 + \frac{5}{\pi}x_1 - 6)^2 + 10(1 - \frac{1}{8\pi})cos(x_1) + 10$	2	[-5,10]×[0,15]	0.398
$f_{18}(x)$	Goldstein-Price Function	$f(x) = [1 + (x_1 + x_2 + 1)^2(19 - 14x_1 + 3{x_2}^2 - 14x_2 + 6x_1 x_2 + 3{x_2}^2)] \times [30 + (2x_1 - 3x_2)^2(18 - 32x_1 + 12{x_1}^2 + 48x_2 - 36x_1 x_2 + 27{x_2}^2)]$	2	[-2,2]	3
$f_{19}(x)$	Hartman's Family	$f(x) = -\sum_{i=1}^{4} c_i \exp(-\sum_{j=1}^{4} a_{ij}(x_j - p_{ij})^2)$	4	[0,1]	-3.86
$f_{20}(x)$	Hartman's Family	$f(x) = -\sum_{i=1}^{4} c_{i\exp(-\sum_{j=1}^{6} a_{ij}(x_j - p_{ij})^2)}$	6	[0,1]	-3.32
$f_{21}(x)$	Shekel's Family	$f(x) = -\sum_{i=1}^{5} [(x - a_i)(x - a_i)^T + c_i]^{-1}$	4	[0,10]	-10.1532
$f_{22}(x)$	Shekel's Family	$f(x) = -\sum_{i=1}^{7} [(x - a_i)(x - a_i)^T + c_i]^{-1}$	4	[0,10]	-10.4028
$f_{23}(x)$	Shekel's Family	$f(x) = -\sum_{i=1}^{10} [(x - a_i)(x - a_i)^T + c_i]^{-1}$	4	[0,10]	-10.5363

Table 2.1: The detailed information of 23 test functions from Yao et al. (1999), including mathematical expression, dimension, parameters range and global optimum value (f_{min}).

Function	NGS	NPS	I
f_1	8	61	100,000
f_2	8	61	100,000
f_3	8	61	300,000
f_4	8	61	300,000
f_5	8	61	500,000
f_6	8	61	100,000
f_7	8	61	200,000
f_8	8	61	200,000
f_9	8	61	200,000
f_{10}	8	61	200,000
f_{11}	8	61	200,000
f_{12}	8	61	300,000
f_{13}	8	61	400,000
f_{14}	8	10	100,000
f_{15}	8	10	100,000
f_{16}	8	10	100,000
f_{17}	8	10	100,000
f_{18}	8	10	100,000
f_{19}	8	10	100,000
f_{20}	8	13	100,000
f_{21}	8	10	100,000
f_{22}	8	10	100,000
f_{23}	8	10	100,000
cf_1	8	21	100,000
cf_2	8	21	100,000
cf_3	8	21	100,000
cf_4	8	21	100,000
cf_5	8	21	100,000
cf_6	8	21	100,000

Table 2.2: List of the settings for the SC-SAHEL algorithm for classic and composite test functions. NGS is the number of complexes, NPS denotes the number of points in each complex and I is the maximum number of function evaluation.

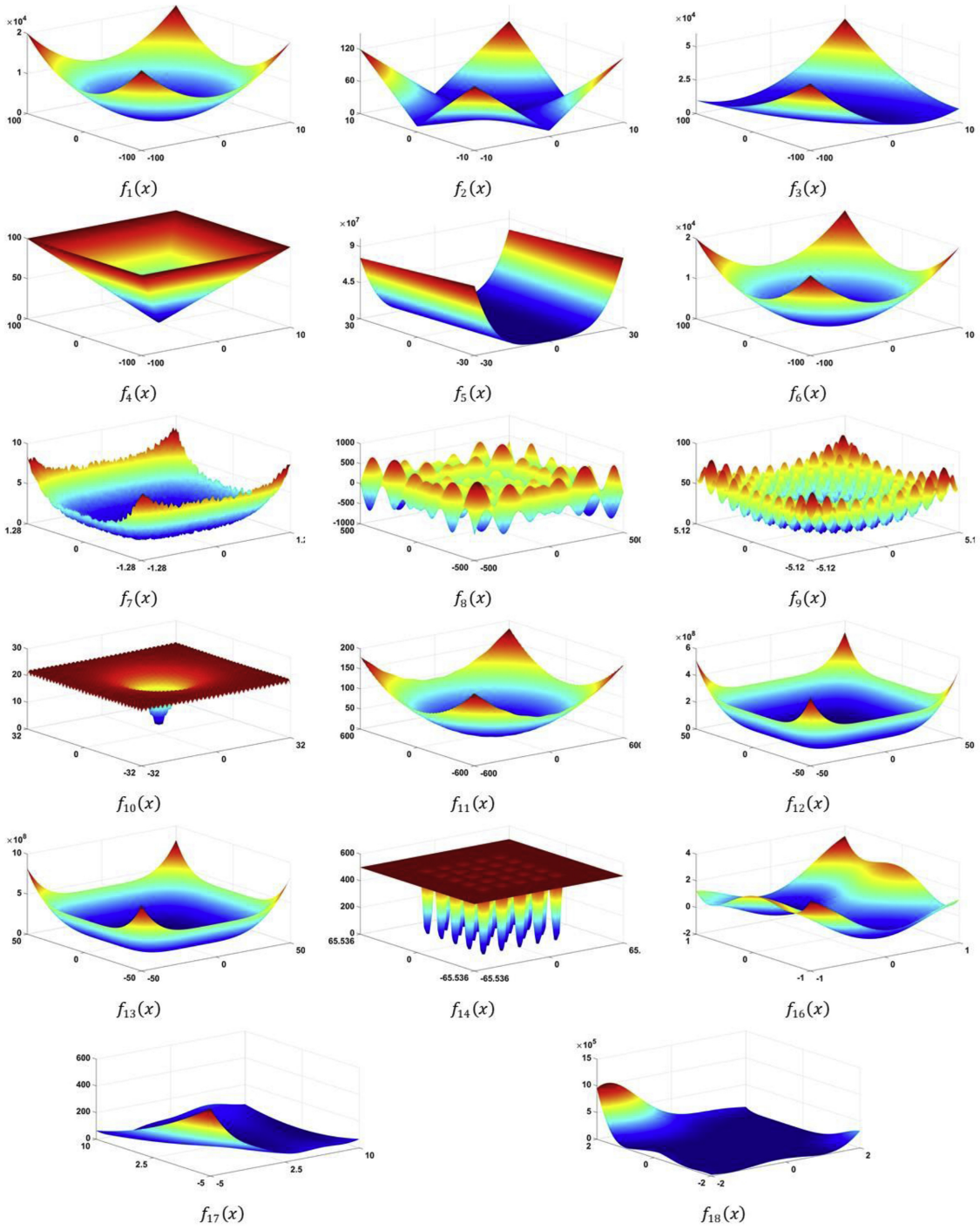


Figure 2.2: Classic test functions in 2-dimension form.

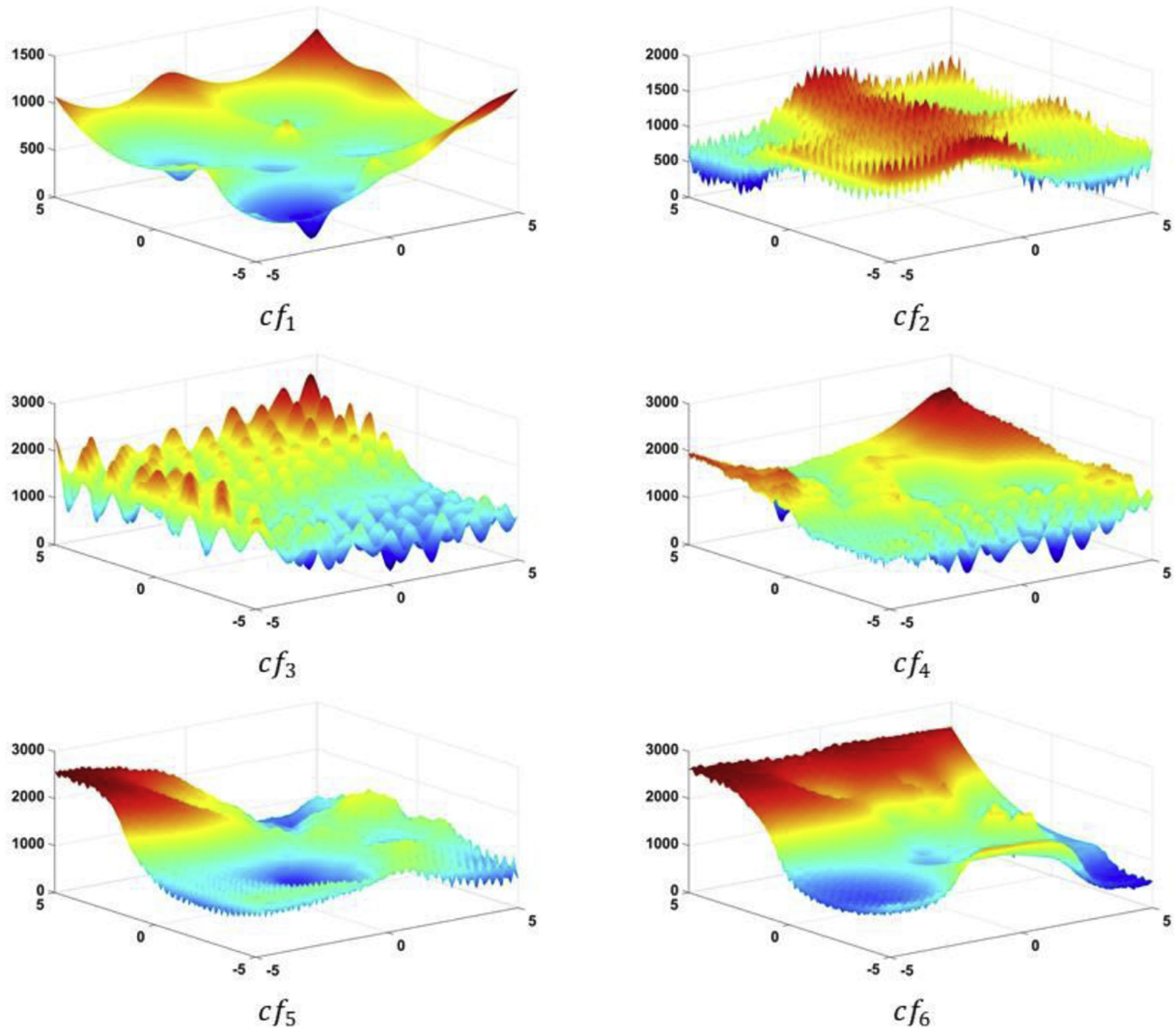


Figure 2.3: Composite test functions in 2-dimension form.

of the test cases. In addition to the maximum number of function evaluation, the range of the parameters in the population and the improvement in the objective function values are used as convergence criteria. The optimization run is terminated if the population range is smaller than $10^{-7}\%$ of the feasible range or the improvement in (objective) function value is smaller than 0.1% of the mean (objective) function value in the last 50 shuffling steps. The LHS mechanism is used as the sampling algorithm of SC-SAHEL for generating the initial population. The framework provides multiple settings for boundary handling, which can be selected by user. SC-SAHEL uses reflection as the default boundary handling method. Other initial sampling and boundary handling methods are also implemented in the SC-SAHEL framework. Sensitivity of the initial sampling and boundary handling on the performance of the SC-SAHEL algorithm is not studied in this chapter. The aforementioned settings can be applied to a wide range of problems.

2.3.2 Results and Discussion

Table 2.3 illustrates the statistics of the final function values at 30 independent runs on 29 test functions using the hybrid SC-SAHEL and individual EAs, with the goal to minimize the function values. The best mean function value obtained for each test function is expressed in bold in Table 2.3. Results show that the hybrid SC-SAHEL achieved the lowest function values in 15 out of 29 test functions, compared to the mean function values achieved by all individual algorithms. It is noteworthy that in 20 out of 29 test functions, the hybrid SC-SAHEL was among the top two optimization methods in finding the minimum function value. A two-sample t-test (with 5% significance level) also showed that the results generated with the SC-SAHEL algorithm is generally similar to the best performing algorithms. Comparing among single-method algorithms, in general, the statistics obtained by SP-UCI are superior to other participating EAs. In 12 out of 29 test functions, the SP-UCI algorithm achieved the lowest function value. SC-MFL, SC-MGWO, and SC-DE were superior to other

algorithms in 6, 10, and 11 out of 29 test functions, respectively. In test functions f_6, f_{16}, f_{17}, f_{18}, f_{19}, f_{20}, and f_{23}, the single-method and multi-method algorithms achieved same function values on average in most cases. In these cases, according to the statistics shown in Table 2.3, the SP-UCI and SC-SAHEL algorithms offer lower standard deviation values and show more consistent results as compared to other EAs. The low standard deviation values obtained by SP-UCI and SC-SAHEL indicate the robustness and consistency of these two algorithms in comparison to other algorithms. In the test functions that the hybrid SC-SAHEL algorithm was not able to produce the best mean function value, the achieved mean function values deviation from that of the best-performing algorithms are marginal. For instance, on the test functions f_2, f_4, f_{10}, and f_{22}, the statistics of the values obtained by SC-SAHEL are similar to that achieved by the best-performing methods, which are SP-UCI, and SC-MGWO. In general, the hybrid SC-SAHEL algorithm is superior to algorithms with individual EA on most of the test functions, although on some test functions, the SC-SAHEL algorithm is slightly inferior to the best-performing algorithm with only marginal differences. The performance of the SC-SAHEL in these test functions can be attributed to two main reasons. First, in the hybrid algorithm, all the EAs are involved in the evolution of the population. Hence, if one of the algorithms have poor performance in comparison to other EAs, it still evolves a portion of the population. As the complexes are evolved independently, the poor-performing EAs may devastate a part of the information in the evolving complex. On the other hand, when the algorithms are used individually in the SC-SAHEL framework, the EA utilizes the information in all the complexes and the whole population. In this case, better result will be achieved in comparison to the hybrid SC-SAHEL, if the EA is the fittest algorithm for the problem space. Second, some of the EAs are faster and more efficient in a specific optimization phase (exploration/exploitation) than others. However, they might not be as effective as other EAs for other optimization phases. Hence, dominance of these algorithm during the exploration or exploitation phases can mislead other EAs and cause early (and premature) convergence. Engagement of other algorithms in the evolution process

Function	SC-SAHEL (MCCE, MFL, MGWO, DE)		SP-UCI (SC-MCCE)		SC-MFL		SC-MGWO		SC-DE	
	Mean	Std	Mean	Std	Mean	Std	Mean	Std	Mean	Std
f_1	3.68E-11	1.60E-11	**1.68E-11**	1.18E-11	2.13E-06	2.98E-06	4.29E-11	1.01E-11	5.92E-05	5.51E-05
f_2	3.14E-06	3.92E-07	3.00E-06	5.94E-07	6.38E-04	5.26E-04	**2.35E-06**	2.75E-07	4.12E-03	1.27E-03
f_3	**2.11E-10**	6.08E-11	8.95E-10	4.37E-10	1.86E-09	1.48E-09	4.50E-10	9.15E-11	1.22E+03	2.16E+03
f_4	4.89E-06	7.88E-07	8.98E-05	4.60E-05	3.50E-01	2.35E-01	**3.65E-06**	5.43E-07	5.26E-06	5.59E-07
f_5	**7.81E-09**	3.15E-09	2.54E-08	1.52E-08	1.33	1.91	2.58E+01	2.85E-01	1.28E+01	1.85
f_6	**0**	0	**0**	0	6.33E-01	6.69E-01	**0**	0	3.33E-02	1.83E-01
f_7	1.09E-03	5.33E-04	**4.78E-04**	3.44E-04	2.08E-03	8.93E-04	1.37E-03	6.36E-04	1.34E-02	4.90E-03
f_8	**9.87E+03**	6.14E+02	5.09E+03	2.27E+02	9.75E+03	6.41E+02	4.36E+03	2.90E+02	4.91E+03	3.75E+02
f_9	8.29E-01	1.73	**3.32E-02**	1.82E-01	2.67E+01	4.57E+01	1.60E+01	9.78	2.01E+02	1.19E+01
f_{10}	1.49E-06	2.43E-07	**1.08E-06**	2.55E-07	1.42	4.98E-01	1.52E-06	2.00E-07	5.47E-06	5.34E-07
f_{11}	**8.05E-11**	2.08E-11	1.77E-10	5.19E-11	1.42E-02	1.51E-02	1.61E-04	8.81E-04	7.21E-03	1.15E-02
f_{12}	**1.58E-13**	5.02E-14	5.27E-13	3.38E-13	3.11E-02	7.77E-02	1.31E-01	8.80E-02	1.06E-12	1.80E-13
f_{13}	3.66E-04	2.01E-03	**2.55E-12**	8.69E-13	3.97E-03	6.59E-03	7.15E-02	8.94E-03	1.62E-11	3.31E-12
f_{14}	**9.98E-01**	1.40E-16	**9.98E-01**	1.27E-16	1.99	1.51	2.53	3.13	9.98E-01	2.16E-16
f_{15}	**3.07E-04**	5.61E-17	1.19E-03	3.80E-03	2.98E-03	6.93E-03	1.08E-03	3.68E-03	3.07E-04	8.87E-14
f_{16}	**1.03**	1.37E-15	**1.03**	7.61E-16	**1.03**	1.18E-15	**1.03**	6.28E-07	**1.03**	9.51E-15
f_{17}	**3.98E-01**	1.47E-15	**3.98E-01**	0	**3.98E-01**	0.00	**3.98E-01**	2.05E-04	**3.98E-01**	7.63E-15
f_{18}	**3.00**	2.20E-14	**3.00**	1.25E-14	**3.00**	1.72E-14	**3.00**	1.81E-05	**3.00**	7.30E-14
f_{19}	**3.86**	2.08E-15	**3.86**	2.12E-15	**3.86**	1.97E-15	**3.86**	5.46E-05	**3.86**	1.61E-15
f_{20}	**3.32**	2.17E-02	**3.32**	2.17E-02	3.31	4.11E-02	3.31	3.03E-02	3.25	5.92E-02
f_{21}	9.16	2.58	5.92	3.28	8.97	2.18	**9.69**	1.75	9.48	1.75
f_{22}	1.02E+01	9.63E-01	9.64	2.31	9.35	2.46	**1.04E+01**	4.56E-04	**1.04E+01**	5.05E-13
f_{23}	**1.05E+01**	1.93E-13	1.03E+01	1.22	9.64	2.35	**1.05E+01**	6.96E-06	**1.05E+01**	5.00E-13
cf_1	6.67	2.54E+01	3.33	1.83E+01	1.35E-11	5.66E-12	1.00E+01	3.05E+01	**9.41E-12**	3.42E-12
cf_2	**2.00E+01**	4.84E+01	1.23E+02	6.79E+01	3.14E+01	5.39E+01	7.76E+01	4.59E+01	3.94E+01	1.44E+01
cf_3	1.32E+02	9.33E+01	1.33E+02	8.22E+01	**1.28E+02**	3.83E+01	2.80E+02	3.16E+01	3.00E+02	4.21E+01
cf_4	2.71E+02	6.67E+01	2.93E+02	8.38E+01	**2.63E+02**	3.20E+01	3.46E+02	1.47E+01	3.30E+02	4.15E+01
cf_5	1.70E+01	3.77E+01	9.75E+01	1.83E+01	1.10E+01	3.05E+01	3.05E+01	4.33E+01	**3.37**	1.83E+01
cf_6	6.71E+02	2.00E+02	8.72E+02	6.59E+01	6.38E+02	1.86E+02	7.80E+02	1.85E+02	**5.40E+02**	1.23E+0

Table 2.3: The mean and standard deviation (Std) of function values for 30 independent runs on 29 test functions using the SC-SAHEL algorithm with single-method and multi-method search mechanisms.

may prevent early convergence in these cases. Generally, the performance criteria, EMP, is responsible for selecting the most suitable algorithm in each optimization step, however, the criteria used in the SC-SAHEL is not guaranteed to perform well in all problem spaces. The performance criteria are problem dependent and need further investigations based on the problem space and EAs. However, the EMP metric seems to be a suitable metric for a wide range of problems.

To further evaluate the performance of the hybrid SC-SAHEL algorithm, I present the success rate of the algorithms in Figure 2.4. The success rate is defined by setting target values for the function value for each test function. When the function value is smaller than the target value, the goal of optimization is reached, and therefore, the algorithm is considered successful. A higher success rate resembles a better performance. I use same target value for all algorithms in order to have a fair comparison. According to Figure 2.4, in 16 out of 29 test functions, the hybrid algorithm achieved 100% success rate. In other cases, the success rates achieved by the proposed hybrid algorithm are comparable to the best-performing algorithm with single EA. For instance, on the test function f_9, the SC-MGWO, SC-DE and SC-MFL are not successful in finding the optimum solution (success rates are 0%, 0%, and 10%, respectively). However, the hybrid SC-SAHEL algorithm has similar performance (80% success rate) to SP-UCI (97% success rate). On the test function f_{21}, the success rate of the hybrid SC-SAHEL algorithm (87%) is close to the SC-MGWO (93%), which is the most successful algorithm. The hybrid SC-SAHEL algorithm also achieved a higher success rate than SP-UCI algorithm (33%) in this test function. According to Fig. 4, the average success rate of SC-SAHEL is about 80% over all 29 test functions, and it is the highest compared to the average success rate of other EAs, i.e., 73%, 58%, 58%, and 54% for SP-UCI, SC-DE, SC-MGWO, and SC-MFL algorithm, respectively.

In some situations, the poor performing EAs may mislead other EAs and cause early (and premature) convergence. For instance, on the test function , the hybrid algorithm achieved 57% success rate, which is still better success rate than SP-UCI, SC-MFL and SC-MGWO,

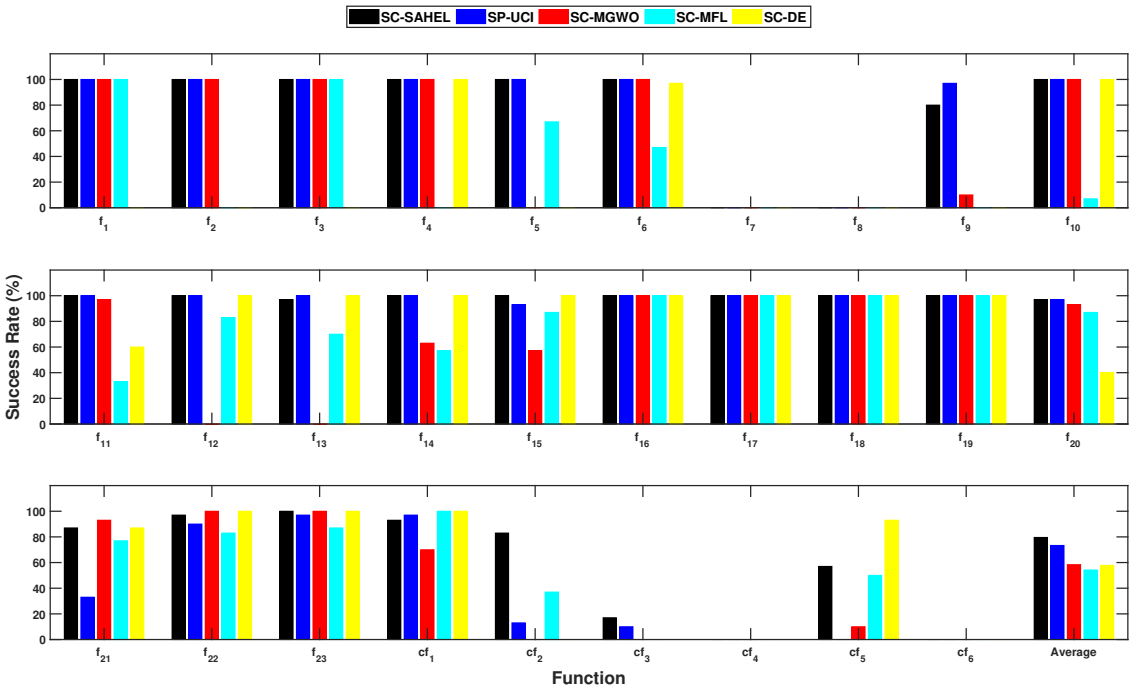


Figure 2.4: The success rate of the SC-SAHEL algorithm using multi-method and single-method search mechanism for 30 independent runs for 29 test functions

which are 0%, 10%, and 50%, respectively. On this test function (cf_5), the performance of the hybrid SC-SAHEL is less affected by the most successful algorithm (DE). This may be due to the low evolution speed of the DE algorithm, as the SC-SAHEL algorithm maintains both convergence speed and efficiency during the entire search. The hybrid SC-SAHEL presents promising performance on the test functions and . On test functions and , the success rate of hybrid SC-SAHEL is significantly higher than other EAs, most of which have 0% success rates. For test function , the SC-DE algorithm achieved the lowest objective function value and the highest success rate (37%) among single-method algorithms. However, when EAs are combined in the hybrid form, the objective function value and the success rate are significantly improved. This shows that SC-SAHEL has the capability of solving complex problems by utilizing the potentials and advantages of all participating algorithms and improving the search success rate.

In Table 2.4, I present the mean and standard deviation of the number of function evaluation, which indicates the speed of each algorithm. The lowest mean number of function

evaluation is expressed in bold in Table 2.4. As one of the stopping criteria in SC-SAHEL framework is the maximum number of function evaluation, some algorithms may terminate before they show their full potential. For instance, the SC-DE and the SC-MFL, usually reach the maximum number of function evaluations, while other algorithms satisfy other convergence criteria in much less number of function evaluations. In this case, the objective function value doesn't represent the potential of the slow algorithms. To give a better insight into this matter, the mean and standard deviation (Std) of the number of function evaluations are compared in Table 2.4. The goal is to compare the speed of the individual EAs and the hybrid optimization algorithm. According to Table 2.4, in most of the test cases, the SP-UCI algorithm has the least number of function evaluations, regardless of the objective function value achieved by the EAs.

Comparing the success rate and the number of function evaluation for different EAs shows that SP-UCI achieved 100% success rate with the lowest number of function evaluation, in 15 out of 29 test functions. The SC-MGWO algorithm only achieved 100% success rate with the lowest number of function evaluation in one test function. Although the hybrid SC-SAHEL algorithm is not the fastest algorithm, its speed is usually close to the fastest algorithm. This is due to the contribution of different EAs in the evolution process and the EAs behavior on different problem spaces. For instance, DE algorithm is slower in comparison to MCCE (SP-UCI) algorithm in most of the test functions. Hence, when the algorithms are working in a hybrid form, the hybrid algorithm will be slower than the situation when the MCCE (SP-UCI) algorithm is used individually.

Figure 2.5, 2.6, and 2.7 compare the average number of complexes assigned to each EA for the 29 employed test functions during the course of the search. The variation of the number of complexes assigned to each EA indicates the dominance of each EA during the course of the search. Hence, the performance of EAs at each optimization step can be monitored. In many test cases, MCCE (SP-UCI) algorithm has a relatively higher number of complexes than other EAs during the search. This shows that MCCE is a dominant search algorithm

Function	SC-SAHEL (MCCE, MFL, MGWO, DE)		SP-UCI (SC-MCCE)		SC-MFL		SC-MGWO		SC-DE	
	Mean	Std	Mean	Std	Mean	Std	Mean	Std	Mean	Std
f_1	32,816.33	723.0532	**26,877.93**	609.7676	100,199.9	129.6894	33,012.97	284.4416	100,325.5	144.6016
f_2	39,298.4	917.7627	**29,333.33**	674.156	100,193.7	126.3259	35,876.77	344.1018	100,307.9	168.2884
f_3	91,746.23	2288.806	**73,474.5**	5435.367	226,848.3	20,135.41	239,199.9	31,109.82	241,449.1	90,118.79
f_4	50,197.6	1761.589	82,183.67	19,213.58	300,252.9	121.2331	**37,987.4**	1170.002	227,316.4	5488.107
f_5	335,364.2	8102.236	401,124.8	14,599.33	439,093.2	47,901.23	**118,900.7**	38,671.69	500,310.9	163.5498
f_6	40,293.63	377.0177	**32,537.93**	151.8836	66,102.9	3708.332	43,063.63	1463.074	90,205.23	5773.92
f_7	**69,779.5**	23,763.53	69,823.27	24,314.74	78,895.43	24,205.89	81,421.53	22,877.8	117,468.4	33,083.6
f_8	71,834.83	9826.963	54,020	17,225.97	65,629.77	5201.429	**45,254.8**	15,104.68	62,555.83	21,579.07
f_9	59,710.57	12,460.93	**33,949.6**	996.5881	100,705.8	22,607.84	85,055.73	20,771.06	90,930.3	34,180.61
f_{10}	33,765.77	887.3708	**27,116.33**	379.9873	77,520.6	15,528.44	33,181.03	416.0297	165,489.4	4726.909
f_{11}	35,504.9	629.8192	**30,623.53**	860.8274	117,357.6	13,250.36	38,652.4	19,330.05	155,148.6	16,730.48
f_{12}	55,908.07	4735.601	**39,264.23**	3125.88	141,722.1	31,245.81	88,234.23	28,948.91	181,820.6	5132.966
f_{13}	54,148.7	3949.577	**32,262.23**	851.2965	123,903.4	20,354.81	72,334.73	22,345.94	170,930.5	3295.191
f_{14}	5216.333	443.942	5708.433	764.8273	4829.2	841.1355	14,986.77	3537.605	**4530.2**	322.0474
f_{15}	9059.8	551.1741	**6517.167**	2358.518	8144.667	1151.183	66,441.3	35,455.39	18,813.63	659.7369
f_{16}	3700.133	1115.033	**2746.3**	747.3937	3491.933	574.5392	8549.4	1494.076	3490.733	556.9577
f_{17}	3665.533	601.3615	**2910.633**	624.4692	3552.267	538.6385	11,453.13	2592.687	5115.367	2082.727
f_{18}	2837.933	308.7723	**2000.633**	151.8512	2899.567	299.5645	8405.933	1320.571	2833.5	111.3877
f_{19}	4225.733	424.8389	**2852.2**	95.83,045	4233.4	238.4404	13,183.77	519.0798	4983.9	179.5704
f_{20}	8915.833	1069.182	**5645.567**	268.4028	8858.967	300.0987	17,143.33	1316.025	12,691.67	1818.526
f_{21}	7455.033	1741.525	**7377.533**	3260.208	7471.4	1554.087	18,771.33	2996.925	10,755.57	1573.865
f_{22}	6370.5	869.9209	**4512.433**	1290.258	7541.433	1582.216	17,466.23	834.9485	8728.7	927.622
f_{23}	6200.133	614.6406	**4084.233**	464.9113	6823.7	327.7709	17,351.87	861.8541	8398.067	599.0103
cf_1	15,049.43	875.8969	**10,293.43**	233.9694	21,663.47	921.1805	74,089.17	17,803.2	28,321.6	678.283
cf_2	16,527.63	1432.21	**10,586.8**	464.8359	20,285.83	2346.093	36,617.1	13,118.85	30,686.4	9354.096
cf_3	25,991.03	8041.928	**16,021.5**	3833.203	23,801.2	3604.495	19,323.13	6052.813	29,496.9	9113.814
cf_4	22,873.87	4414.168	**16,510.13**	4052.642	21,121.93	2417.582	23,841.93	8026.638	35,134.33	14,468.3
cf_5	17,044.53	1350.845	**13,512.2**	746.6731	21,400.57	1759.215	53,551.43	23,577.95	39,200.77	4125.908
cf_6	13,779.33	2279.744	**10,518.1**	2977.194	14,967.5	2820.062	22,265.8	15,340.72	27,734.83	4606.317

Table 2.4: The mean and standard deviation (Std) of number of function evaluation for 30 independent runs on 29 test functions using the SC-SAHEL algorithm with single-method and multi-method search mechanisms.

on most of the test functions. However, in some other cases, MCCE is only dominant in a certain period of the search, while other EAs have demonstrated better efficiency during the entire search. For example, on test functions f_7 and f_{20}, MCCE algorithm appears to be dominant only during the beginning of the search. In the test function f_7, the exploration process starts with the dominance of the MCCE and shifts between MGWO and MFL after the first 20 shuffling steps. In some of the test functions, such as f_7, a more random fluctuation is observed in the number of complexes assigned to each EA. The reason for this behavior is the close competition of EAs in these shuffling steps. Due to the noisy response surface of the test function f_7, most of the EAs cannot significantly improve the (objective) function values during the exploitation phase. On test functions f_8 and f_{18}, the MFL and DE algorithms are the dominant search methods, respectively, during the beginning of the run, while MCCE algorithm becomes dominant only when the algorithm is in exploitation phase. Lastly, on test functions f_9, f_{22}, cf_1, and cf_4, the variations of the number of complexes and the precedence of different EAs as the most dominant search algorithm are observed.

It is worth mentioning that, Figure 2.5, 2.6, and 2.7 show the number of complexes assigned to each EA for a single optimization run. Our observation of each individual run results (not shown herein) shows variation of the number of complexes among different runs is similar to each other for most test cases. The observed variation for individual runs follows a specific pattern and is not random. The similarity of the EAs dominance pattern indicates that the selection of the EAs by the SC-SAHEL framework only depends on the characteristics of the problem space and the EAs employed. This also indicates that different EAs have pros and cons on different optimization problems.

As a summary of our experiments on the conceptual test functions (Table 2.3, and 2.4, and Figure 2.4, 2.5, 2.6, and 2.7), the main advantage of the SC-SAHEL algorithm over other optimization methods is its capability of revealing the trade-off among different EAs and illustrating the competition of participating EAs. Different optimization problems have different complexity, which introduces various challenges for each EA. By incorporating dif-

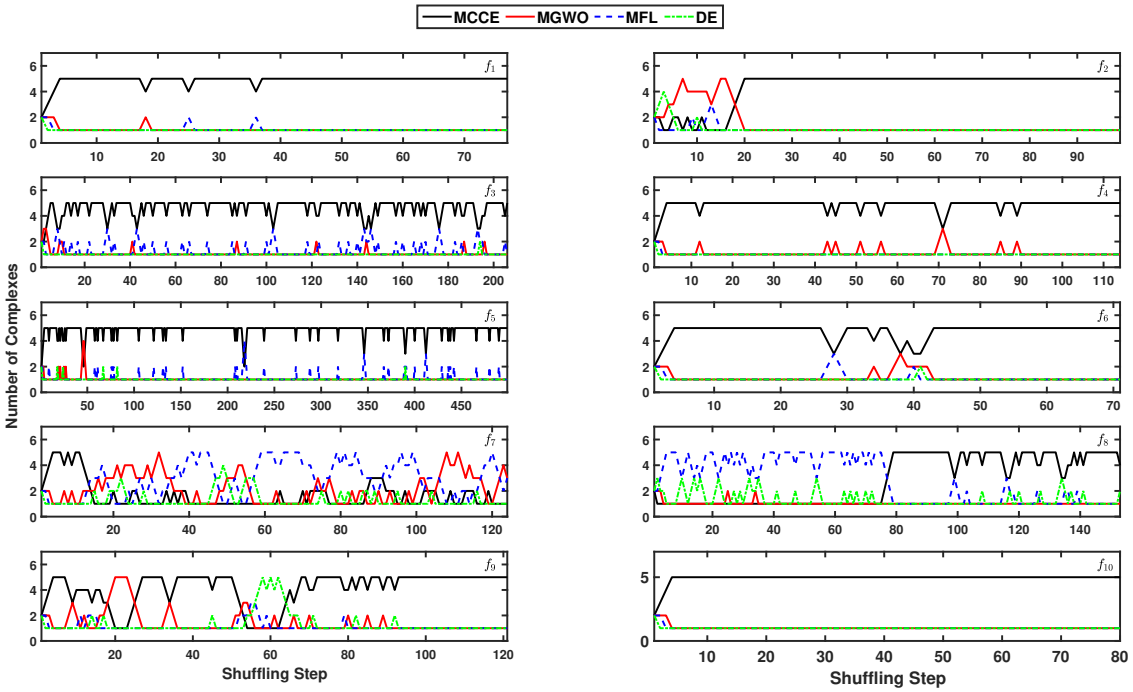


Figure 2.5: Number of complexes assigned to EAs during the entire optimization process for test functions f_1-f_{10}.

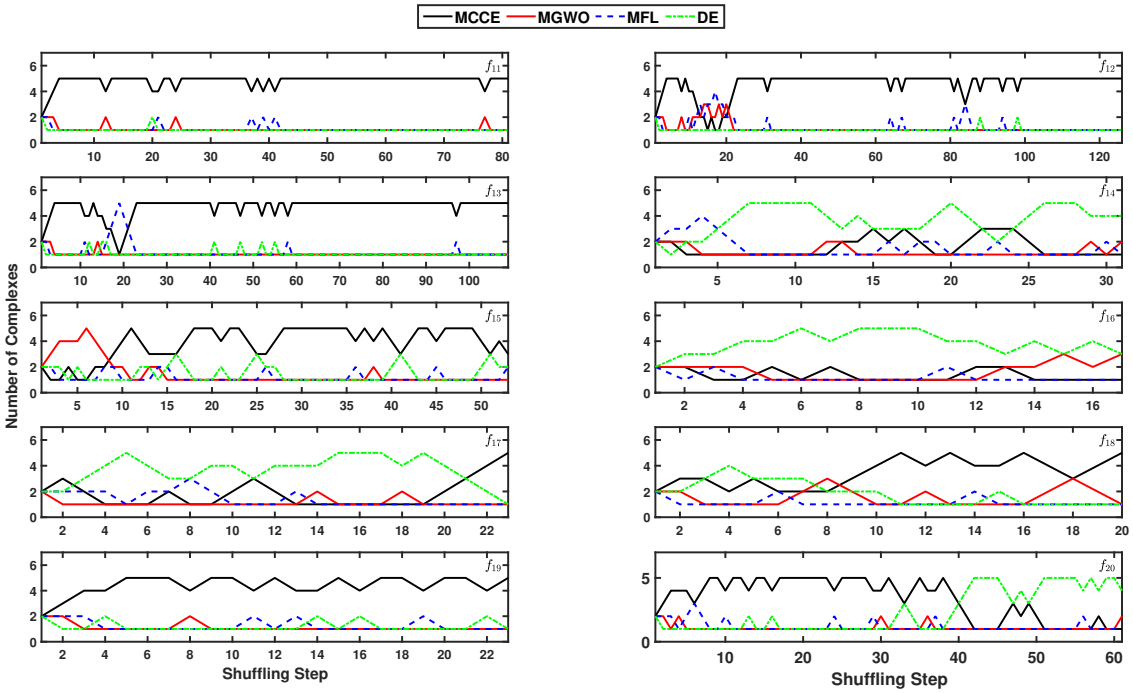


Figure 2.6: Number of complexes assigned to EAs during the entire optimization process for test functions f_{11}-f_{20}.

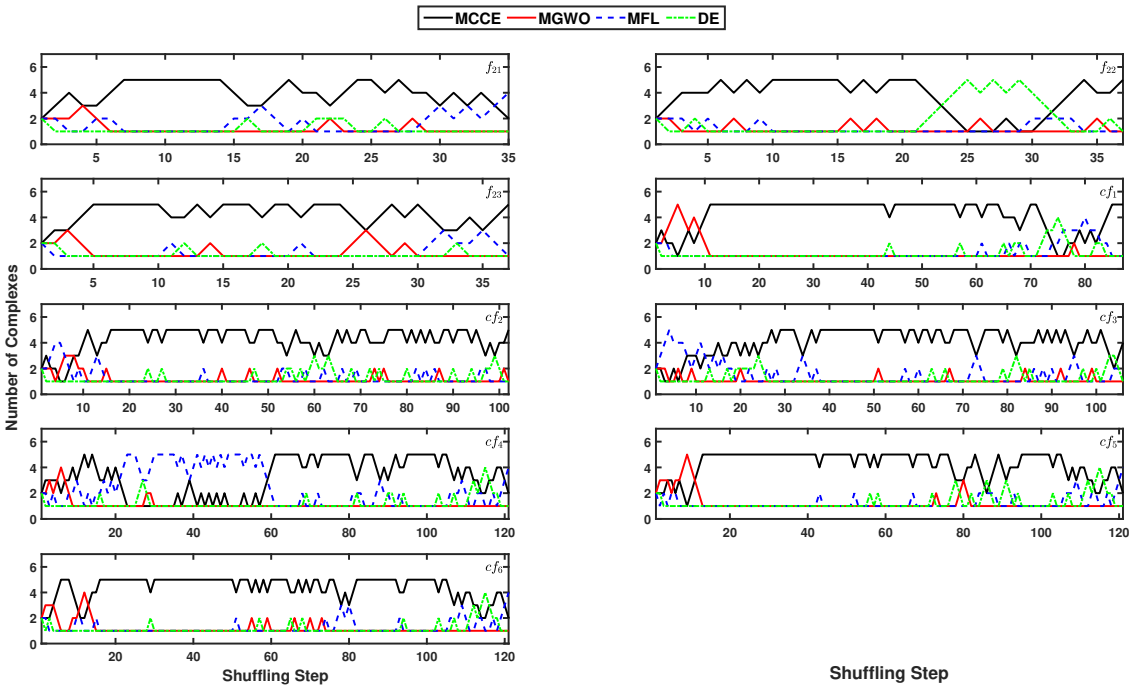


Figure 2.7: Number of complexes assigned to EAs during the entire optimization process for test functions f_{21}-f_{23} and cf_1-cf_6

ferent types of EAs in a parallel computing framework, and implementing an award and punishment logic, the newly developed SC-SAHEL framework not only provides an effective tool for global optimization but also gives the user insights about advantages and disadvantages of participating EAs on individual optimization tasks. This shows the potential of the SC-SAHEL framework for solving different class of problems with different level of complexity. Besides, the hybrid SC-SAHEL algorithm is superior to shuffled complex-based methods with single search mechanism, such as SP-UCI, in an absolute majority of the test functions.

2.4 Conclusions and Remarks

In this chapter a new hybrid optimization framework, named Shuffled Complex Self Adaptive Hybrid EvoLution (SC-SAHEL) is introduced. The new framework uses an award and

punishment logic in junction with various types of Evolutionary Algorithms (EAs), and selects the best EA that fits well to different optimization problems. The framework provides an arsenal of tools for testing, evaluating, and developing optimization algorithms. I compared the performance of the hybrid SC-SAHEL with single-method algorithms on 29 test functions. The results showed that the SC-SAHEL algorithm is superior to most of the single-method optimization algorithms and in general offers a more robust and efficient algorithm for optimizing various problems. Furthermore, the proposed algorithm is able to reveal the characteristics of different EAs during entire search period. The algorithm is also designed to work in a parallel framework which can take the advantage of available computation resources. The newly developed SC-SAHEL offers different advantages over conventional optimization tools. Some of the SC-SAHEL characteristics are:

- Intelligent evolutionary method adaptation during the optimization process
- Flexibility of the algorithm for using different evolutionary methods
- Flexibility of the algorithm for using initial sampling and boundary handling method
- Independent parallel evolution of complexes
- Population degeneration avoidance using PCA algorithm
- Robust and Fast optimization process
- Evolutionary algorithms comparison for different types of problems

Although the presented results support advantage of the hybrid SC-SAHEL to algorithms with individual EAs, there are multiple directions for further improvement of the framework. For example, EAs' performance metric for evaluating the search mechanism. In the current algorithm, the complex allocation to different EA is carried out by ranking the algorithm according to the EMP metric. The performance criteria can change the allocation process and

affect the performance of the algorithm. Depending on the application a more comprehensive performance criterion may be necessary for achieving the best performance. However, the current EMP criterion does not affect the conclusion and comparison of different EAs. In addition, the current SC-SAHEL framework is designed to solve single objective optimization problems. A multi-objective version can be developed to extend the scope of the application. This chapter serves as an introduction to the newly developed SC-SAHEL algorithm. I hope that more investigation on the interaction among different EAs, boundary handling schemes and response surface in different case studies and optimization problems reveal the advantages and limitations of SC-SAHEL. The SC-SAHEL framework can be downloaded from MathWork website.

Chapter 3

Application of the SC-SAHEL Framework to Reservoir Optimization

3.1 Introduction

In his chapter, the SC-SAHEL framework is tested for a conceptual hydropower model, which is built for the Folsom reservoir located in the northern California, USA. The objective is to maximize the hydropower generation, by finding the optimum discharge from the reservoir. The study period covers run-off season in California from April to June, in which reservoirs have the highest annual storage volume (Field and Lund, 2006a). Using the proposed framework, I compared different EAs' capability of finding a near-optimum solution for dry, wet, and below-normal scenarios. The results support that the proposed algorithm is not only competitive in terms of increasing power generation, but also is able to reveal the advantages and disadvantages of participating EAs.

The rest of this chapter is organized as follows. Section 3.2 details the conceptual reservoir model for the Folsom Reservoir. In this section, the setting, objectives, and constraints em-

ployed of the optimization problem is specified. At the end of this section, the results and performance of the SC-SAHEL optimization framework is detailed and compared with other optimization algorithms. At last, section 3.3 discuss the findings and concludes this chapter.

3.2 Reservoir Model

A conceptual model is set up based on the relationship between the hydropower generation, storage, water head and bathymetry of the Folsom reservoir. Daily releases from the reservoir in the study period are treated as the parameters of the model, which in turn determines the problem dimensionality. The model objective is to maximize the hydropower generation for a specific period. The total hydropower production is a function of the water head difference between forebay and tailwater and the turbine flow rate. The driving equation of the model is based on mass balance (water budget), which is formulated as,

$$S_t = S_{t-1} + I_t - R_t \pm M_t, \tag{3.1}$$

where S_t is storage at time step t, I_t and R_t signify total inflow and release from the reservoir at time t, respectively. M_t is total outflow/inflow error which is derived by setting up mass balance for daily observed data. The objective function employed here is,

$$\text{OF} = \sum_{t=1}^{N} 1 - \frac{P_t}{P_c}, \tag{3.2}$$

where P_c is total power plant capacity in MW and P_t is total power generated in day t in MW. Minimizing the objective function in this case is equivalent to maximizing hydropower generation. For each day P_t is derived as follow,

$$P_t = \eta \rho g Q_t H_t, \tag{3.3}$$

where η signifies turbine efficiency, ρ is water density (Kg/m3), g is gravity (9.81m/s2) and Q_t is discharge (m3/s) at time step t. H_t is hydraulic head (m) at time step t, which is defined as,

$$H_t = h_f - h_{tw}, \tag{3.4}$$

where h_f and h_{tw} are water elevation in forebay and tailwater, respectively. h_f and h_tw are derived by fitting a polynomial to reservoir bathymetry data.

In the reservoir model coined above, multiple constraints are considered for better representation of the real behavior of the system. These constraints include power generation capacity, storage level, spill capacity, and changes in the daily hydropower discharge. Total daily power generation is compared to maximum capacity of the hydropower plant. Also, rule curve is used to control reservoir storage level during the operation period. Besides, final simulated reservoir storage is constrained to $0.9 - 1.1$ of the observed storage. In another word, 10% variation from the observation data is allowed for the final simulated storage level. This constraint adds information from real reservoir operation into the optimization process and added as penalty to objective function. This constraint can be replaced by other operation rules for simulation purposes. The spill capacity of dam is calculated according to the water level in the forebay and compared to simulated spilled water. A quadratic function is fitted to the water level and spill capacity data, to derive the spill capacity at each time step. The change in daily hydropower release is also constrained to better represent actual hydropower discharge and avoid large variation in a daily release. The reservoir model used here is non-linear and continuous. The constraints of the model render finding the feasible solution a challenging task for all the EAs. The SC-SAHEL framework is used to maximize the hydropower generation by minimizing the objective function value. The settings used for the SC-SAHEL is similar to the settings used for the mathematical test functions. However, the maximum number of function evaluations is set to 10^6. Lower bound of the parame-

ters' range varies monthly due to the operational rules; however, upper bound is determined according to the hydraulic structure of the dam.

3.2.1 Study Basin

Folsom reservoir is located on the American river, in Northern California and near Sacramento, California. Folsom dam was built by US Army Corps of Engineers during 1948 to 1956, and is a multi-purpose facility. The main functions of the facility are flood control, water supply for irrigation, hydropower generation, maintaining environmental flow, water quality purposes, and providing recreational area. The reservoir has a capacity of 1,203,878,290 m^3 and the power plant has a total capacity of 198.7 MW. Three different periods are considered here. The first study period is April 1st, 2010 to June 30th, 2010. The year 2010 is categorized as below-normal period according to California Department of Water Resources. The same period is selected in 2011 and 2015, as former is categorized by California Department of Water Resources as wet, and latter is classified as critical dry year. The input and output from the reservoir are obtained from California Data Exchange Center `http://cdec.water.ca.gov/`. Note that demand is not included in the model because demand data was not available from a public data source.

3.2.2 Results and Discussion

The boxplot of the objective function values is shown in Figure 3.1 for the Folsom reservoir during the runoff season in 2015, 2010, and 2011, which are dry, below-normal, and wet years, respectively. The presented results are based on 30 independent optimization runs; however, infeasible objective function values are removed. The feasibility of the solution is evaluated according to the objective function values. Due to the large values returned by the penalty function considered for infeasible solutions, such solutions can be distinguished

from the feasible solutions. For wet year (2011) case, SC-MGWO, and SC-DE didn't find a feasible solution in 2, and 4 runs out of 30 independent runs, respectively. The hybrid SC-SAHEL found feasible solutions in all the cases; however, some of these solutions are not global optima. On average, the hybrid SC-SAHEL algorithm is able to achieve the lowest objective function value as compared to other algorithms during dry and below-normal period. During dry and below-normal periods, SC-SAHEL, SP-UCI, and SC-DE show similar performance. In the wet period, the SP-UCI algorithm achieved the lowest objective function value. The SC-SAHEL algorithm ranked second, comparing the mean objective function values. In this period, the results achieved by the SC-DE is also comparable to SC-SAHEL and SP-UCI. The results show that overall, the hybrid SC-SAHEL algorithm has similar or superior performance in comparison to the single-method algorithms. Also, the results achieved by SC-SAHEL and SP-UCI algorithms has less variability in comparison to other algorithms, which show the robustness of these algorithms. The worst performing algorithm is the SC-MGWO, which achieved the least mean objective function value in all the study periods. In Figure 3.2, boxplot of the number of function evaluations is presented for successful runs from the 30 independent runs during dry, below-normal and wet period years. Although the SC-MGWO algorithm satisfied convergence criteria in the least number of function evaluation, the SC-MGWO was not successful in achieving the optimum solution in many cases. The SP-UCI algorithm is the second fastest method among all the algorithms. The hybrid SC-SAHEL, SC-MFL, and SC-DE are the slowest algorithm for satisfying the convergence criteria, in almost all cases. The slow performance of the hybrid SC-SAHEL is due to the fact that 2 out of 4 (DE and MFL) participating EAs have very slow performance over the response surface. Figure 3.3 demonstrates the number of complexes assigned to each EA during the search, which indicates the dominance of the participating algorithms, and the award and punishment logic in the reservoir model. As seen in Figure 3.3, the MGWO algorithm is dominant in the beginning of the search; although, it is not capable of finding the optimum solution in most cases. The reason for the dominance of the MGWO

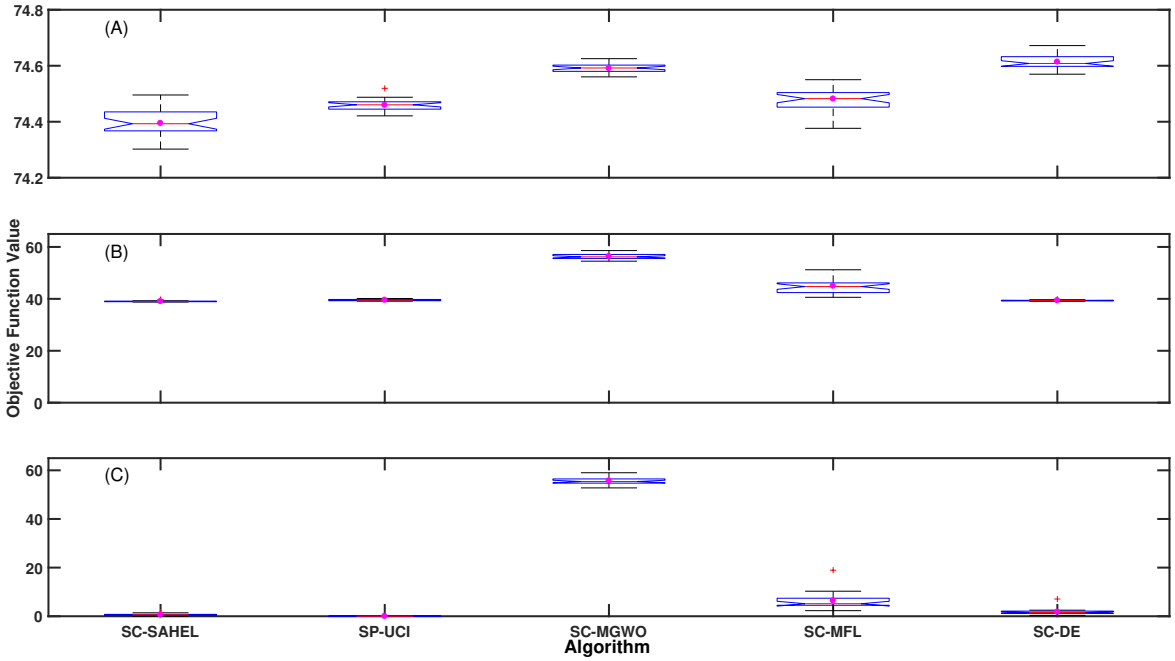


Figure 3.1: Boxplots of objective function values for successful runs among 30 independent runs, for dry (A), below-normal (B) and wet period (C). The mean of objective functions values is shown with pink marker.

is the speed of the algorithm in exploring the search space. MGWO is superior to other EAs in the beginning of the search, however, after a few iterations, the MCCE algorithm took the precedence and become the dominant algorithm over other EAs. MGWO and DE are less involved in the rest of the optimization process after the initial steps. However, competition between MCCE and MFL continues. Although contribution of MGWO and DE are at minimum in the rest of the optimization process, they are utilizing a part of information within the population. This can affect the speed and performance of the SC-SAHEL algorithm. In both the wet and below-normal cases, the hybrid SC-SAHEL algorithm is mostly terminated by reaching the maximum number of function evolution. However, the mean objective function value obtained by the hybrid SC-SAHEL is still superior to most of the algorithms. The performance of the SC-SAHEL can be affected by the settings of the algorithm. Different settings have been tested and evaluated for the reservoir model. The results show that the number of evolution steps before shuffling can influence the performance

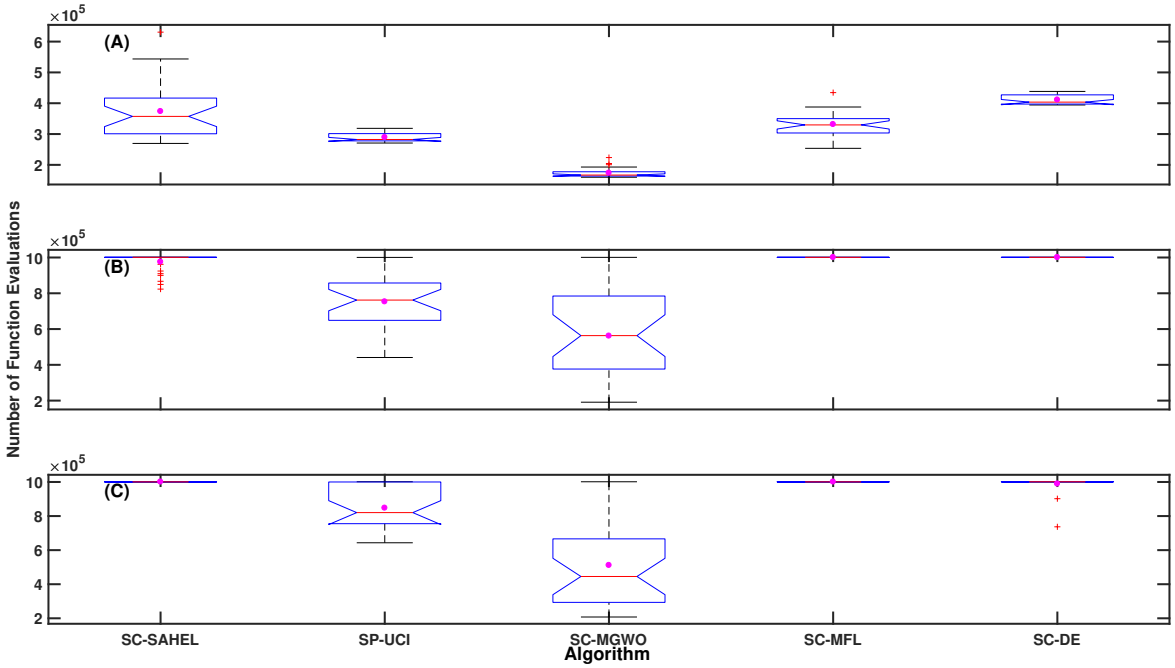


Figure 3.2: Boxplots of number of function evaluations for successful runs among 30 independent runs for dry (A), below-normal (B) and wet period (C). The mean number of function evaluation is shown with pink marker.

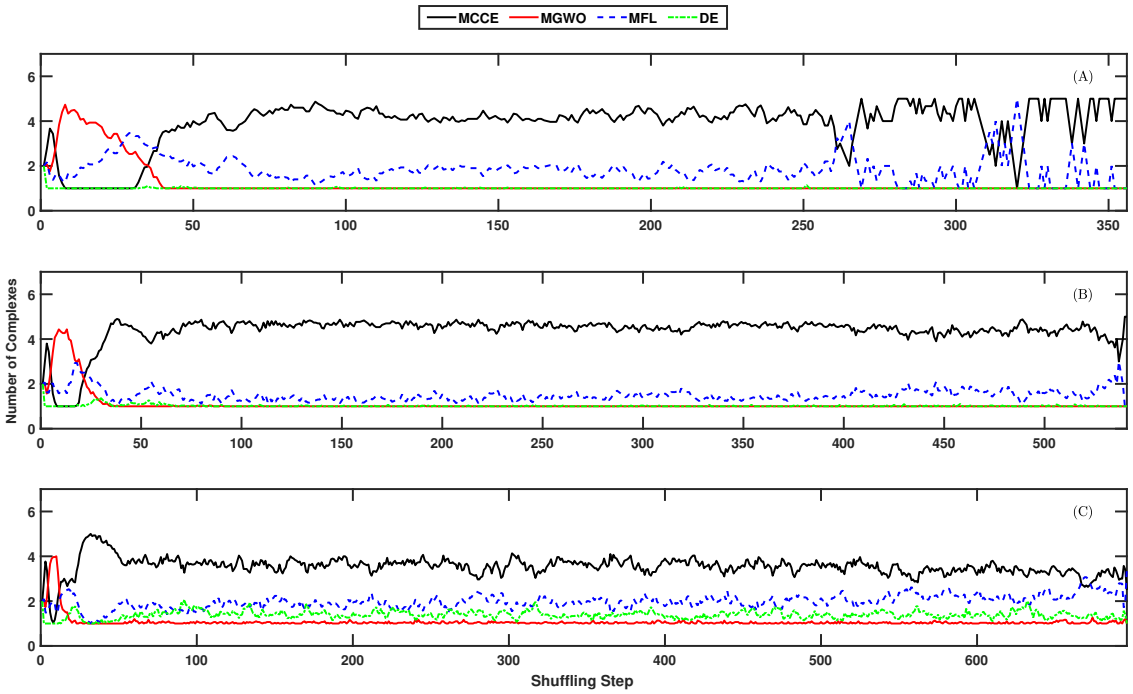


Figure 3.3: The average number of complexes assigned to each EA at each shuffling step for 30 independent runs for dry (A), below-normal (B), and wet (C) period.

of the hybrid SC-SAHEL algorithm. In the current setting, the number of evolution steps within each complex is set to $d + 1$ (d is dimension of the problem). Although this setting seems to provide acceptable performance for a wide range of problems, it may not be the optimum setting for all the problems spaces and EAs. In the reservoir model, as the study period has 91 days, the model evolves each complex for 92 steps. This number of evolution steps allows the algorithms to navigate the complexes toward local solutions and increase the total number of function evaluations without specific gain. Decreasing the number of evolution steps allows the algorithms to communicate more frequently, so they can use the information obtained by other EAs. Here, for demonstrative purposes, the same setting has been applied to all the problems. However, better performance is observed for the hybrid SC-SAHEL algorithm when the number of evolution steps are set to a value smaller than 92. The algorithm is less sensitive to other settings for the reservoir model, however they can still affect the performance of the algorithm.

In Figure 3.4, I present the simulated storage for different study periods achieved by different EAs. During the dry period, not only the SC-SAHEL algorithm achieved the lowest objective function value, but also the storage level is higher than the observed storage level in most of the period. This is due to the fact that, power generation is a function of water height, as well as discharge rate. During below-normal period, SC-SAHEL, SP-UCI, and SC-DE algorithms show a similar behavior in terms of the storage level. During wet period, storage level simulated by SP-UCI and SC-SAHEL algorithm is lower than all other algorithms. It is worth noting that, during wet period, SC-SAHEL and SP-UCI algorithms are able to find optimum solution (which objective function value is 0) in some of the runs. However, the simulated storage by these algorithms show some level of uncertainties (Figure 3.4). This shows equifinality in simulation, which means that same hydropower generation can be achieved by different sets of parameters (Feng et al., 2017). This equifinality can be due to deficiencies in the model structure, or the boundary conditions (Freer et al., 1996). The wet period seems to offer a more complex response surface for the reservoir model. During the wet

period, some algorithms, such as SC-DE, are not capable of finding a feasible solution in some of the runs. In this period, the large input volume and the rule curve added more complexity to the optimization problem. The results of the real-world application show the potential of the newly developed SC-SAHEL framework for solving high dimension problems. In general, the hybrid algorithm was more successful in finding a feasible solution in comparison to single-method algorithms. In some cases, the hybrid SC-SAHEL was terminated due to the large number of function evaluations. However, the performance of the hybrid SC-SAHEL is always comparable to the best performing method. This shows the potential of the SC-SAHEL for solving a broad class of optimization problems. Besides, the framework provides insight into the performance of the algorithms at different steps of the optimization process. This feature of the SC-SAHEL algorithm can aid user to select the best setting and EA for the problem.

3.3 Conclusions and Remarks

The results and performance of the SC-SAHEL framework on the reservoir problem shows the potential of the framework for solving real world optimization problems. In general, it has been observed that the hybrid optimization algorithm is superior to other single search method optimization algorithms in term of efficiency and effectiveness in finding the near optimum solution. Also, the performance of the algorithms in this section showed that the complexity of the optimization problems can change with the hydrologic condition of the reservoir system. For instance, during the wet season, the optimization algorithms required more function evaluation to find the optimum solution.

Although the SC-SAHEL framework showed superior result on this reservoir problem, there are some potential directions for future investigation. Most reservoir problems are many objective optimization problems. Hence, developing a many objective optimization algorithm

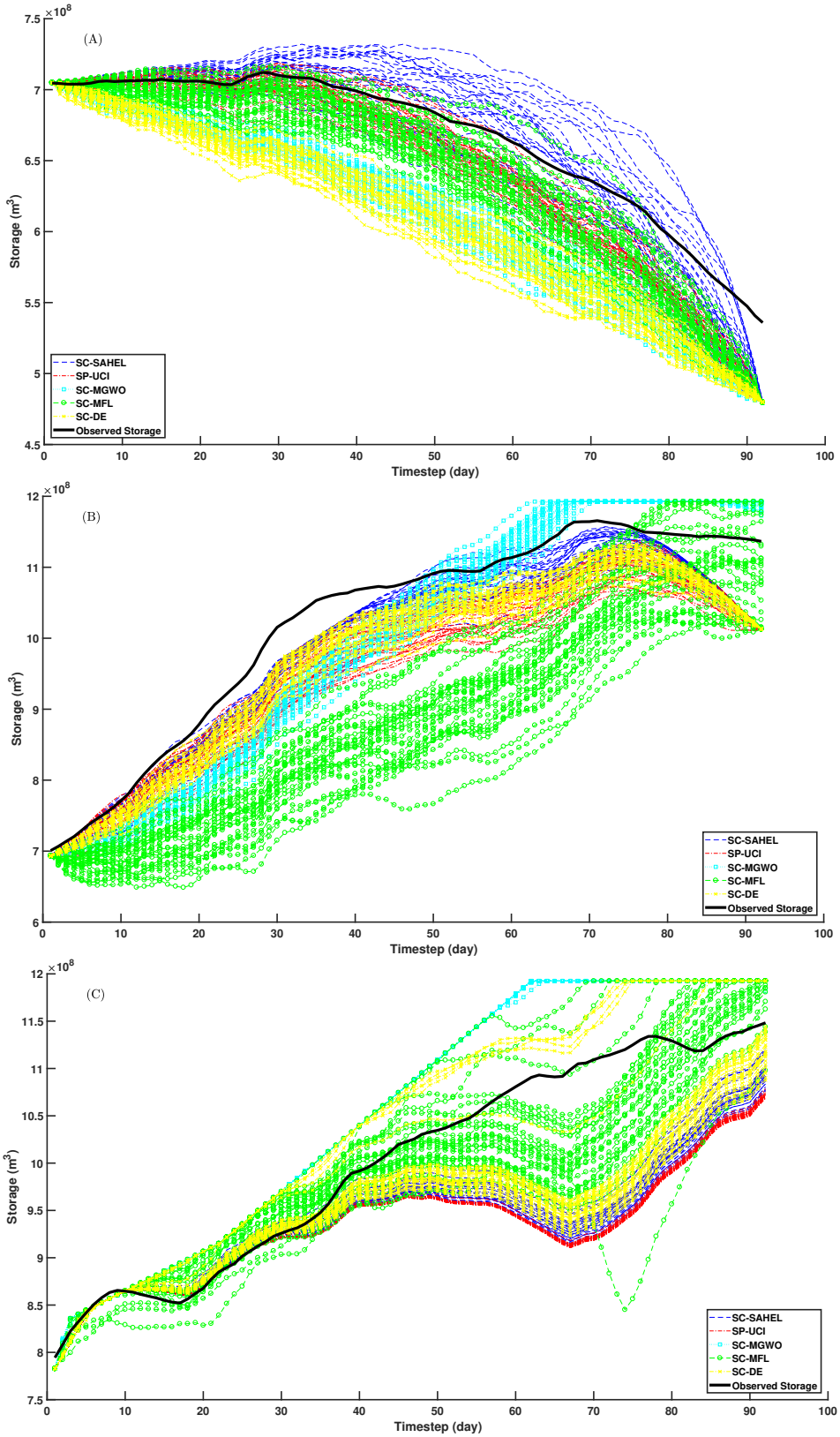


Figure 3.4: Simulated storage for dry (A), below-normal (B), and wet (C) period.

based on the SC-SAHEL framework can extend the application of the framework to a larger class of problems. Also, the SC-SAHEL framework is flexible in employing different methods and settings. Hence, the effect of these settings can be significant on different class of problems and needs further investigation.

Chapter 4

Developing a Generalized Model Tree (GMT) Framework for Simulating Rule-Based Hydrologic Systems

4.1 Introduction

Tree-based (decision tree) algorithms are transparent data-mining approaches (Jung et al., 2010; Etemad-Shahidi and Taghipour, 2012), which describe and present a response (dependent) variable by splitting the explanatory (independent) variables space into clusters of data (Ciampi, 1991; De'ath and Fabricius, 2000). Simplicity and accuracy of tree-based algorithms make them attractive tools among practitioners in different fields of study (Loh, 2014), including remote sensing (Friedl and Brodley, 1997; Huang and Townshend, 2003; Pal and Mather, 2003), water resources management (Castelletti et al., 2010; Yang et al., 2016, 2017b) and hydrology (Galelli and Castelletti, 2013b; Han et al., 2018). Although classic tree-based algorithms were more concerned with classification and discrete spaces (Wang

and Witten, 1997), application of tree induction methods have been extended to regression problems and continuous spaces (Loh, 2014). A wide range of algorithms have been proposed for regression tree induction, among which Classification And Regression Tree (CART) (Breiman et al., 1984), random forest (RF) (Breiman, 2001), and extremely randomized tree (Extra-Tree) (Geurts et al., 2006) found more applications in water resources management and reservoir studies (Galelli and Castelletti, 2013a,b; Yang et al., 2016; Goyal et al., 2013). An extension of the regression tree algorithms is model trees (MTs) (Malerba et al., 2004). MTs and regression tree models share similar structures, however, MTs are equipped with multiple linear regression as a predictive tools (Etemad-Shahidi and Mahjoobi, 2009). This feature of MT algorithms makes them effective tools for high dimensional continuous problems (Bhattacharya and Solomatine, 2005). M5 (Quinlan et al., 1992), and M5' (Wang and Witten, 1997) are among the popular MT algorithms in the field of hydrology and water resources management (Kompare et al., 1997; Solomatine and Dulal, 2003; Štravs and Brilly, 2007; Jothiprakash and Kote, 2011; Galelli and Castelletti, 2013a). Most MT algorithms follow a binary splitting mechanism. In these algorithms, an exhaustive search approach is employed to find the best attributes and split points (SP) to partition a set of instances (Parent: contains all the data) into two subsets (Child: partitioned data over a selected variable and split point). This partitioning process is carried out according to a predefined measure of goodness for split candidates (Murthy and Salzberg, 1995). In most MT algorithms, every single value of the attributes (independent variable) is treated as a split candidate. Hence, the algorithm evaluates all the values and attributes to find the split point. This exhaustive search mechanism can be biased in attribute selection when the number of possible split points are different for the attributes (Loh, 2002; Shih, 2004; Loh, 2014; Loh et al., 2015). Hence, the attributes with more split candidates have higher chance of being selected for partitioning. Also, this search mechanism is computationally inefficient (Witten et al., 2016) for finding the combinatorial effect of variables (Loh, 2014). These shortcomings motivated us to develop a new generalized model tree (GMT) framework for tree induction to reduce

the selection bias and computational burden of MTs, and also enhance the performance of these algorithms.

The newly developed GMT framework can describe continuous data and offer an interpretable structure for underlying system. To handle large datasets and high dimensional problems, GMT is equipped with various tools and settings. To enhance the efficiency of the algorithm, I employ principal component analysis (PCA), which is an effective tool for analyzing (Chu et al., 2010; Naeini et al., 2018) and reducing dimensions of data (Hsu et al., 2002; Shlens, 2014). In addition to PCA, GMT employs a k-fold cross-validation tree growing scheme to overcome bias in variable selection and reduce the computational burden of the algorithm. In this process, attributes and split points are selected by evaluating a specific number of split candidates over all attribute domain, rather than testing all the values in dataset. This can significantly reduce the number of split candidates, while maintaining the performance of the model. Also, the split candidates are evaluated by taking multiple samples of data, which reduces the bias in variable selection. As a result, the GMT algorithm is computationally more efficient and robust in comparison to conventional MTs.

Similar to many tree-based algorithms, GMT framework follows a top-down binary tree induction scheme. The top-down binary tree splitting mechanism is an inseparable component of many regression tree algorithms (Breiman et al., 1984; Quinlan et al., 1992; Murthy and Salzberg, 1995; Wang and Witten, 1997; Hothorn et al., 2006; Quinlan, 2014). In these algorithms, the tree induction process follows a successive partitioning of a set of explanatory variables into smaller clusters of data until no further partitioning is possible. The resulting tree-structured model has a hierarchical if-then form and consists of three different types of nodes. The node at the top level of the tree is the *root* node, which contains all the instances of the dataset. The lowest nodes of the tree are terminal nodes (leaf nodes), which predict the response variable. All the nodes between the top node and lowest nodes are non-terminal (decision) nodes. The selected attributes and split points shape the decision rules in the non-terminal nodes. The non-terminal nodes navigate the instances through

branches of the tree towards the terminal (leaf) nodes. Simulation is then made through the terminal nodes.

Predictive mechanisms in the terminal nodes are different for regression tree algorithms and MTs. CART algorithm reports the mean of response variable in the terminal nodes as the prediction value (Breiman et al., 1984). However, MT algorithms, such as M5 and M5', use linear regression as predictor (Quinlan et al., 1992; Wang and Witten, 1997). Similar to M5 and M5' algorithms, the GMT framework employs linear regression in terminal nodes. However, the GMT framework employs a backward elimination process to drop linear regression terms and simplify the regressed line. Hence, the resulting model employs less variables in terminal nodes to represent data.

The GMT framework is tested on multiple datasets. The performance of the framework is evaluated as follows. First, seven benchmark datasets are obtained from the University of California, Irvine (UCI) data repository (Asuncion and Newman, 2007). These datasets have been widely used for developing and testing different machine learning and data-mining tools (Yang et al., 2017a). The effect of the split criteria (SC) and the algorithm settings are investigated through employing multiple combinations of settings for the GMT framework. The GMT framework is also compared against other tree-based algorithms. These algorithms include CART, M5', which are single tree algorithms, and random forest (RF), which is an ensemble tree algorithm.

Second, performance of the GMT framework is assessed for simulating discharges from eight reservoir systems across the Contiguous United States (CONUS) in the next chapter. Due to the importance and complexity of the reservoir simulation in hydrologic routing (Tavakoly et al., 2017; Zhang et al., 2018), data-mining algorithms can be employed as effective tools to derive and reproduce the controlled outflow from reservoir systems (Yang et al., 2016). Among data-mining techniques, tree-based algorithms have the advantage of producing interpretable models. So, the rules extracted with tree-based algorithms can be monitored and further enhanced (Bessler et al., 2003). Here, the proposed GMT algorithm is employed to

simulate and extract the operational rules of multiple reservoirs with different services and purposes. Historical reservoir discharge data, along with the information about hydrologic conditions of the system are employed to train these models. Performance of the GMT algorithm is compared against CART, M5' and RF algorithms on these reservoir cases.

The rest of this chapter is organized as follows. Section 4.2 introduces the newly developed GMT framework and explains the algorithm settings. In section 4.3, the GMT framework is tested on different benchmark datasets obtained from the UCI repository. Performance of the GMT algorithm is further compared against CART, M5', and RF in this section. Section 4.4 concludes the chapter and elaborates on GMT performance, limitations and future works.

4.2 Generalized Model Tree

The GMT framework follows two main steps to generate tree-based models. First, it follows a top-down tree induction mechanism to shape the structure of the trees. Second, multiple linear regressions are used to explain the response variable for each cluster of data. Figure 4.1 shows the schematic representation of the tree induction process in the GMT framework. In this process, parent nodes are recursively partitioned into *two* child nodes, until no further splitting is possible. Then a multiple linear regression model is fitted to each terminal node. The partitioning process in GMT framework is carried over according to a predefined SC. The pseudo code of the GMT framework is as follows:

The GMT framework is equipped with multiple SC to evaluate and select attributes and split points for partitioning data. Employing different SC may lead to different model structures, which can affect the performance of the resulting model. In general, most regression tree algorithms employ a least square approach for finding the split point. For instance, partitioning of data in CART algorithm is carried over such that the sum squared residual

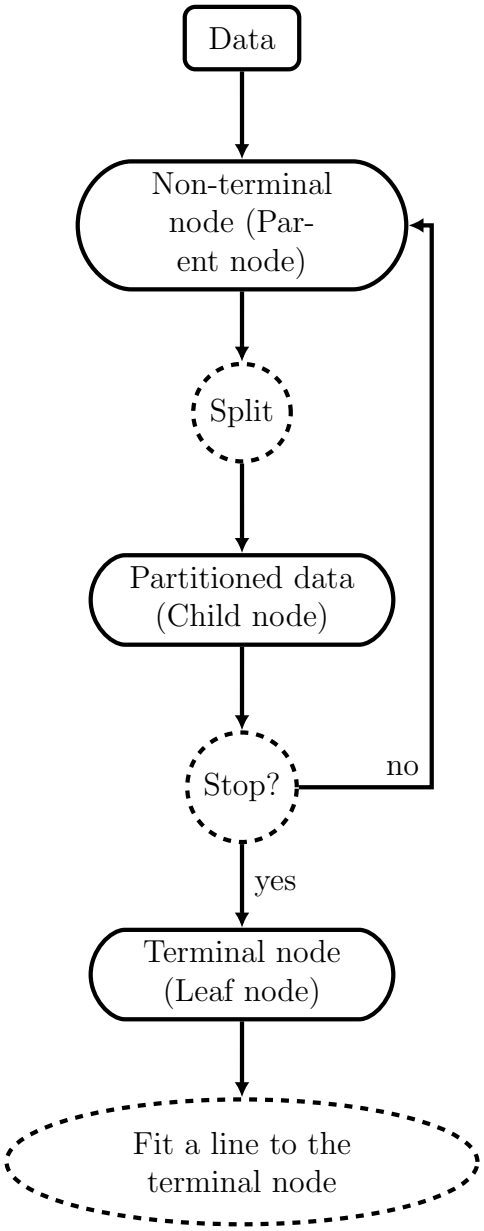


Figure 4.1: The schematic representation of the GMT framework.

(SSR) with respect to the mean of response variable is minimized for all the subset regions (Breiman et al., 1984; Huang and Townshend, 2003). The SSR with respect to the mean of response variable (SSRM) is defined as,

$$\text{SSRM} = \sum_{n=1}^{N} (y_n - \bar{y})^2, \tag{4.2}$$

Algorithm 4.1 GMT($\cdot$) Framework

Step 0: Set the problem dependent parameters: (a) minimum leaf size ι and (b) error threshold ε. (Tune additional algorithm settings if necessary.)
Step 1: Split $\mathbf{X}_p$ and $\mathbf{Y}_p$ into $\mathbf{X}_l$, $\mathbf{Y}_l$ and $\mathbf{X}_r$, $\mathbf{Y}_r$ according to split criteria SC (Equ. 4.2-4.4). ($\mathbf{X}$: explanatory variables; $\mathbf{Y}$: response variable; p: parent node; l: left child node; r: right child node)
Step 2: Evaluate the stopping criteria for both child nodes l and r.

$$\text{If} \quad \text{SC}_l + \text{SC}_r \leq \varepsilon \ \text{SC}_p \ \vee \ \min|\mathbf{X}_{l(\cdot D)}| \leq \iota \ \vee \ \min|\mathbf{X}_{r(\cdot D)}| \leq \iota \tag{4.1}$$

go to Step 4, else proceed. (D is the number of independent variables)
Step 3-1: $\mathbf{X}_p \leftarrow \mathbf{X}_r$ and go to Step 1.
Step 3-2: $\mathbf{X}_p \leftarrow \mathbf{X}_l$ and go to Step 1.
Step 4: Fit a linear model to each child node. $\mathbf{Y}_l := f(\Theta_l, \mathbf{X}_l)$ and $\mathbf{Y}_r := f(\Theta_r, \mathbf{X}_r)$
Step 5: Simplify the fitted line employing backward elimination using Bayesian Information Criterion (BIC) (Equ. 4.6).

where y_n is the observed response variable, $\bar{y}$ is the mean of the observed response variables in the subset, and N is the number of instances in the subset region. CART calculates SSRM for all the subsets and selects the split candidate which offers the lowest SSRM for resulted subsets. In contrast to CART algorithm, M5 and M5' algorithms treat standard deviation as a measure of error (Quinlan et al., 1992; Wang and Witten, 1997). The attributes and split points which maximize the standard deviation reduction (SDR) will be selected for generating the subsets. The SDR is defined as,

$$\text{SDR} = sd(\mathbf{Y}) - \sum_{m=1}^{M} \frac{|\mathbf{Y}_m|}{|\mathbf{Y}|} \times sd(\mathbf{Y}_m), \tag{4.3}$$

where $sd()$ is the standard deviation operator, $\mathbf{Y}$ is the vector of target values, $\mathbf{Y}_m$ is the vector of target values in the subset m, M is the number of subsets, which is two in the binary splitting algorithms (i.e. $M = r, l$) and $|\cdot|$ determines the size of the vectors. Huang and Townshend (2003) proposed a stepwise regression tree approach which clusters data by minimizing distance of the data points from a fitted line for each subset. In this method,

the SSR using multiple linear regression (SSRML) is derived as,

$$\text{SSRML} = \sum_{n=1}^{N} \left(y_n - f\left(\boldsymbol{\theta}, \mathbf{x}_n\right)\right)^2, \tag{4.4}$$

where $f(\cdot,\cdot)$ is the regressed linear function, and $\mathbf{x}_n$ and $\boldsymbol{\theta}$ signify the vector of attributes and coefficients employed in the linear regression, respectively. Huang and Townshend (2003) employed stepwise linear regression (SLR) to remove the effect of multi-collinearity of the attributes.

The GMT framework is equipped with these three SC mentioned above, however, the framework employs SSRML as the default SC. As the GMT framework employs multiple linear regression in leaf nodes as predictor, the SSRML works better with the fitted lines in the terminal nodes. Trees generated by SSRML criteria is expected to have fewer branches, and demonstrate superior performance. Figure 4.2 shows an example to demonstrate the superiority of SSRML to SSRM. In this example, a conceptual dataset with a single response variable and two explanatory variables are simulated with CART and GMT framework. The left plot (A) shows the model generated by CART algorithm (which employs SSRM), and the right plot (B) shows the model generated by GMT algorithm using SSRML and multiple linear regression models in the terminal nodes. In this figure, the regressed and split hyperplanes are shown in green and gray, respectively. The figure shows the effectiveness of the SSRML in combination with multiple linear regression in presenting a dataset with a linear pattern. GMT used a single split plane and two regression planes to describe data, while CART algorithm employed three split planes and four predictive planes. Also, the regressed planes offer a better representation of the dependent variable than the average values, employed by CART algorithm. Hence, SSRML offers a more simplified structure for representing the data, while capturing the linear pattern in the dataset.

Employing SSRML as SC requires evaluating all split candidates for each explanatory variables. The conventional exhaustive search mechanism embedded in most tree-based al-

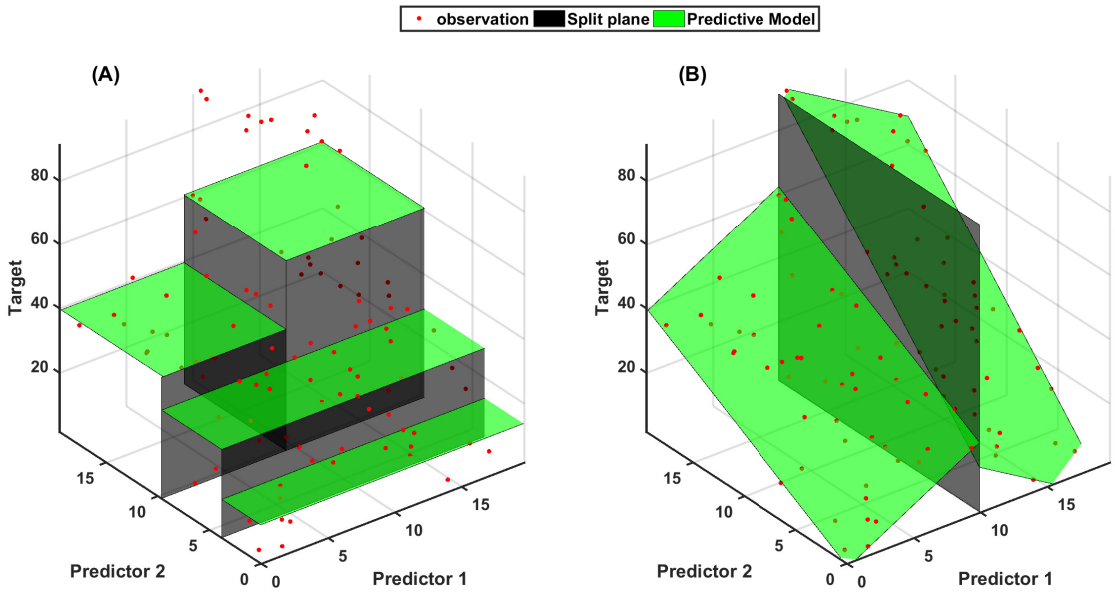


Figure 4.2: In comparison to Classification And Regression Tree (CART; plot A) algorithm, Generalized Model Tree (GMT) framework with Sum Squared Residual with Multiple Linear (SSRML) split criteria (plot B) offers less decision rules and better representation of linear pattern in dataset.

gorithms requires evaluating all possible split candidates. In other words, in these algorithms every single instance in the dataset will be evaluated for every dimension of data. Hence, employing SSRML as criteria requires fitting a multivariate linear model to all possible clusters of data, resulting from each split candidates. This evaluation mechanism is inefficient and computationally intensive. Also, this exhaustive search method can be biased towards attributes with more split candidates (Loh, 2002). To overcome these shortcomings, the GMT framework is loaded with three different modules. **First**, the algorithm searches over pre-specified number of equidistant quantiles (λ) of data as split candidates. In this process, the quantiles are derived empirically for the sorted values of each explanatory variables using their plotting position. The cumulative probability of each value is calculated using Hazen plotting position as,

$$F_X(x) = \frac{r - 0.5}{N}, \tag{4.5}$$

where r is the rank of the sorted values and N is the number of data points. Then equidistant quantiles of data are selected according to these probabilities.

This approach can significantly reduce the number of split candidates while covering the entire domain for each explanatory variable. Considering a node with D explanatory variables and N instances, the size of data will be $D \times N$. If the minimum leaf size is set to ι, the number of split candidates in an exhaustive search process will be $(N - 2 \times \iota - 1) \times D$ for the node. In most cases, $\iota \ll N$, and the nodes closer to the root node have more instances. Hence, the exhaustive search mechanism evaluates all the $(N - 2 \times \iota - 1) \times D$ candidates. Therefore induction time has a direct relationship with the dimension of data. The equidistant quantile sampling scheme can significantly reduces the number of split candidates at each node. The number of split candidates in this process is $\lambda \times D$, where λ is the number of candidates for each dimension. The default value for λ is 100, which can be tuned by the user. Applying the default setting of the algorithm, GMT evaluates $100 \times D$ candidate at each node. $100 \times D \ll (N - 2 \times \iota - 1) \times D$ when the size of data is large. In addition to the induction speed, the equidistant quantile sampling can reduce bias in variable selection, as all the explanatory variables have an equal number of split candidates.

Second, GMT employs k-fold cross-validation mechanism during the tree growing process to select the best attributes and split candidates. In this process, at each level of the tree, the parent node is partitioned into almost equally sized folds. Then GMT performs a cross validation for each of the folds. Each fold is selected at a time as the testing data and the remaining folds are used for training. The training folds are used for partitioning the parent node and fitting a multiple linear model. Then the test fold is used to evaluate the split candidate according to the pre-specified SC (here, SSRML). This process is repeated for all the folds, and the SSRML values are averaged for each split candidates. This process tests each split candidate on multiple samples of data. Although k-fold cross-validation increases the number of line fittings and evaluations, the limited number of split candidates, make this process computationally feasible for large datasets. GMT employs 10-fold as default for cross-

validation, which is recommended by many researchers (Arlot et al., 2010; Zhang and Yang, 2015), knowing that increasing the number of folds will increase the computation time for the algorithm. Considering the default setting of the algorithm, each split candidate will be evaluated 10 times. Hence, at each partitioning step, $1000 \times D$ evaluations will be conducted to find the best split candidates using the default settings. $1000 \times D \ll (N - 2 \times \iota - 1) \times D$ will hold when the size of the dataset is large.

Third, GMT leverages it's computational efficiency to the principal component analysis (PCA). GMT employs PCA to perform principal components (PCs) transformations of explanatory variables for linear regression. This guarantees a well-posed regression problem, while reduces the number of dimension of the independent variables (Hsu et al., 2002). GMT selects the PCs which describe 99.99% of the variability of the independent variables. This step can remove the redundancy and noise in the data to some extent (Shlens, 2014). Although PCA can remove some of the informative variables and increase the error of linear regression (Jolliffe, 1982), it is employed solely for improving the induction speed of GMT. Reducing the number of explanatory variables during regression will reduce the computational burden of the regression process. Interested readers can refer to (Hsu et al., 2002) for further detail on PC regression.

In addition to these modules, GMT employs different mechanisms to control the size of the resulting models and avoid over fitting data. It employs two criteria to stop partitioning dataset and control the number of branches. These stopping criteria are employed to avoid over-fitting training data and to reduce the depth of the tree, which is the longest path between the root node and any terminal node. The depth of the tree resembles the complexity of the inducted model. In most cases, deeper trees have more branches and complexity. Another stopping criteria in GMT is the amount of improvement in prediction resulting from splitting the data. For each testing fold, the SSRML is used to compare the performance of child nodes with the parent node in terms of prediction accuracy. Partitioning will only occur, if the prediction accuracy of the child nodes are superior to the parent node by a

specific amount. GMT uses 5% as the default value for the minimum improvement resulting from splitting the parent node. This means that partitioning will only occur if splitting will improve the average prediction accuracy of that node by at least 5%.

Many tree induction mechanisms employ a pruning algorithm after the tree growing process to simplify the tree. In this process, some of the non-terminal nodes are replaced with terminal nodes to reduce bias, simplify the structure of the model and to avoid over fitting the data. Although GMT supports employing pruning algorithms, the current framework doesn't have any pruning method built-in. However, the implemented stopping criteria in the algorithm, works as a pre-pruning process, which controls the size of the tree.

After shaping the structure of the trees, GMT fits multivariate lines to each terminal node by least squared residual regression using the original independent variables (not the PC transformed ones). The regressed line is further simplified by employing a backward elimination process. In this process, a variable is removed in an stepwise procedure at a time and Bayesian Information Criterion (BIC) (Schwarz et al., 1978; Broman and Speed, 2002) is calculated for the lines fitted to the rest of variables. The set of variables with the lowest BIC is selected at each step. This process is repeated until removing more variables does not reduce BIC. For convenience, residuals are assumed to be independent and identically distributed with a normal distribution (Diks and Vrugt, 2010; Sadegh et al., 2017). Hence, the BIC equation can be stated as,

$$\mathrm{BIC} = D\mathtt{ln}(n) + n\mathtt{ln}(\frac{\mathrm{SSR}}{n}), \tag{4.6}$$

where D is the dimension of the problem, `ln` is the natural logarithm operator, and n is the number of instances in the terminal node. BIC penalizes the number of parameters employed in the model (Nasta et al., 2013). Hence, it simplifies the regressed model in the terminal nodes. This can reveal the importance of independent variables in describing the response variables in each partition of data.

Table 4.1: GMT settings

Setting / GMT No.	split Criteria	Dimension Reduction (PCA)	k-fold ($k = 10$)
GMT(1)	SSRML	✓	✓
GMT(2)	SSRML		✓
GMT(3)	SSRM		✓
GMT(4)	SSRML	✓	
GMT(5)	SSRML		
GMT(6)	SSRM		

The setting implemented in the GMT framework can be tuned by user according to the dataset. Here, I employ six different combination of settings for the GMT framework to investigate the influence of these modules on the performance of the resulting models. Table 4.1 lists all the settings employed in this study. The first three settings (GMT(1), GMT(2), and GMT(3)) employ the k-fold cross-validation scheme with the equidistant quantile sampling method. GMT(1) employs all the default settings of the GMT framework including the dimension reduction and cross-validation induction settings. GMT(2) is similar to GMT(1), however dimension reduction is disabled in this case. The effect of SC is investigated by employing SSRM in GMT(3). The rest of the settings GMT(4), GMT(5), and GMT(6) are similar to the first three settings, however, exhaustive search mechanism is used to find the split points and attributes. The error improvement threshold ε, the number of candidates for splitting λ, and the number of folds for k-fold cross-validation are set to default in all these cases. In the following sections, these settings are applied to different cases, and are compared against CART, M5' (Jekabsons, 2016), and RF. The M5' code is obtained from `http://www.cs.rtu.lv/jekabsons/`.

4.3 Benchmark Datasets

4.3.1 Datasets

In this section, the GMT framework is benchmarked on multiple datasets obtained from the UCI machine-learning repository (`https://archive.ics.uci.edu/ml/datasets.html`) (Asuncion and Newman, 2007). The selected datasets are different in size and number of attributes. These datasets have been extensively used for evaluating and comparing models and algorithms (Yang et al., 2017a). The selected datasets are used to evaluate the GMT family before applying the algorithm to large real-world case studies. Detailed information on the selected datasets are listed in Table 4.2. The minimum leaf size ι for each dataset is also shown in Table 4.2. The minimum leaf size for each case study is selected according to the number of instances and the overall performance of the algorithms. The selected minimum leaf size is not optimized for any of the algorithms, and is selected for comparison purposes only.

Table 4.2: Selected UCI machine learning repository data details with minimum leaf size

Case	Dataset Name	No. Features (D)	No. Instances ($\mathbf{N}$)	Minimum Leaf Size (ι)
a	Airfoil self-noise	5	1503	150
b	Auto MPG	7	398	25
c	Combined cycle power plant	4	9568	500
d	Concrete Compressive Strength	8	1030	100
e	Energy efficiency (heating load)	8	768	100
f	Energy efficiency (cooling load)	8	768	100
g	Istanbul stock exchange	7	536	100

4.3.2 Experiment Design

In addition to the selected settings for GMT, CART, M5' and RF algorithms are also tested and evaluated on the benchmark datasets. Minimum leaf size for CART and M5' are the same as the GMT framework, and the minimum leaf size for RF is set to 5, which is the default setting for the algorithm. Number of trees in the RF algorithm is set to 50 for all the test cases. Pruning is enabled for the M5' algorithm. Comparisons for all the algorithms are carried out using k-fold cross-validation. The number of folds is set to 30 for all the datasets. The performance of the algorithms are evaluated on the testing and training folds subsequently. The performance of the algorithms are compared using boxplots and notches are provided to evaluate the significance of the differences. Notches show the confidence interval for the median (Krzywinski and Altman, 2014), that can be derived as,

$$\mathrm{CI} = \mathrm{m} \pm \frac{\mathrm{IQR}}{\sqrt{N_s}} \tag{4.7}$$

where, CI is confidence interval, m is the sample median, IQR is the interquartile range ($Q_{75} - Q_{25}$), and N_s is the sample size. The IQR shows the consistency and robustness of the model performance.

4.3.3 Results

Figure 4.3 and Figure 4.4 show the boxplots for the normalized root mean squared error (RMSE) values for the training and testing folds, respectively. The results for the training data (Figure 4.3) shows that RF algorithm is superior to all other algorithms, as expected. On average the accuracy of the best single-tree algorithm fall 10% below ensemble methods (Loh, 2014). Hence, the single tree algorithms are not expected to outperform RF. Among the single tree methods, the CART algorithm has the highest RMSE in all cases. The poor

performance of the CART algorithm can be explained by the prediction method employed in the terminal nodes. Since the CART algorithm reports the average value of the dependent variable in each cluster of data, it cannot explain the variability of data within each cluster. GMT and M5' are equipped with linear regression models in terminal nodes, which justifies their better performance in the continuous spaces. However, the GMT family are superior to M5' in all the cases. To shed light on the effect of linear regression in the terminal nodes, CART and GMT(6) are further compared here. GMT(6) and CART algorithm share the same model structure, however, GMT(6) employs linear regression in the terminal nodes to present data. Performance of GMT(6) algorithm shows that employing linear regression in terminal nodes can significantly improve the performance of the regression trees (Karalic, 1992).

Among other GMT algorithms, GMT(1), GMT(2), GMT(4), and GMT(5) offer lower RMSE in most cases. These GMTs employ SSRML as SC, which works well with the linear regression employed in the terminal nodes. This shows the effectiveness of the SSRML SC on enhancing the performance of the MTs. It is worth noting that although GMT(1) and GMT(2) algorithms evaluate predefined number of split candidates for splitting, their performance is similar to GMT(4) and GMT(5) in most cases. If the size of the data is small (similar to case **b**) the k-fold induction process may show inferior performance due to the small size of the folds. In general, the results show the effectiveness of the k-fold induction mechanism and the equidistant quantile sampling. These results also show that the MTs are less sensitive to the cut point, and models with different structures can achieve similar performance. Among other single tree methods, M5' offers close performance to GMT algorithms in dataset **c** and **g**. However, in other cases, all the GMTs are superior to M5'.

Furthermore, robustness and consistency of the performance for each algorithm is evaluated according to the interquartile ranges (IQR). The RF algorithm offers the least IQR in all the datasets. In almost all the test cases, the GMT family models show more consistent results in comparison to other single tree methods. Among GMT settings, the algorithms with

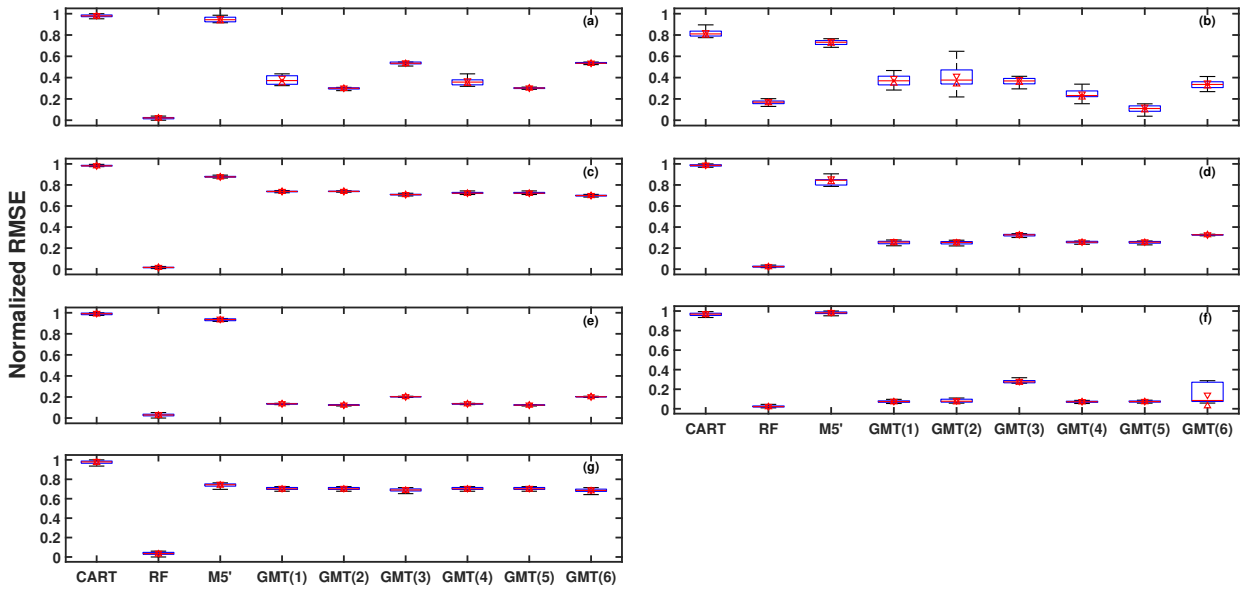


Figure 4.3: Boxplots of the RMSE values for 30-fold cross-validation on the airfoil self-noise(a), auto MPG (b), combined cycle power plant (c), concrete compressive strength (d), energy efficiency-heating load (e), energy efficiency-cooling load (f), Istanbul stock exchange (g) training datasets. The notches are shown with red triangles.

SSRM as SC (GMT(3) and GMT(6)) offer a larger IQR for dataset **f**, however, in general the GMTs show similar performance in term of IQR. In some of the presented datasets, PCA has slightly increased RMSE. Although PCA is employed for speed enhancement in GMT, the removed dimension of data can increase regression error during the tree induction process. As dimension reduction is solely based on the variance of each dimension, some of the informative dimensions can be removed during this process (Jolliffe, 1982). Any changes in the regression error during the induction process can affect the structure of the tree. Hence, it can change the performance of the model as can be seen in dataset **a**. The default setting of the GMT algorithm, GMT(1), employs PCA to reduce induction speed. However, this setting is recommended for large datasets with a large number of attributes.

Figure 3.4 shows the boxplots for the the testing data. These results illustrate the performance of the algorithm on *unseen* data. Similar to the training dataset, RF is the best performing algorithms in almost all cases. The IQR for RF also shows that the algorithm offers more consistent performance on different samples of data. Among single tree algorithms,

CART has the highest median for RMSE. M5' shows close performance to the CART algorithm in most cases. Among GMTs, the algorithms which employed SSRML as SC show superior performance in all cases. Comparing the notches also shows that performance of these algorithms is significantly superior to GMTs with SSRM (Specifically in dataset **a**, **f**, and **g**). The result also shows that the k-fold induction scheme with a fixed number of equidistant quantile samples has very close performance (in some cases k-fold has slightly improved performance such as dataset **d**). Similar to the training data, the dimension reduction process has slightly increased the RMSE of the model. However, in most cases, the accuracy of the model is less influenced by this setting. The IQR value of the GMTs are lower than other single tree methods in most cases. In general, these results show that models generated with the default setting of the GMT algorithm, GMT(1), have close performance to the best performing setting of the GMTs in most cases. Details on the induction speed of the algorithm are left for the next section. The size of the dataset employed in this section cannot reveal the full potential of the algorithms on large datasets.

Performance of the algorithms is further compared by evaluating the correlation between simulated and observed response variables. The correlation values show the capability of the algorithms in describing the variability of data. The average values of the correlation are presented in Figure 4.5. Correlation for the training data and testing data are shown in blue circles and triangles, respectively. Figure 4.5 shows that RF achieved the highest correlation in almost all the test cases in both training and testing data. However, in dataset **g** correlation for RF dropped for the test data. In this case, the GMT algorithms outperform RF in terms of correlation. In the rest of the cases, most GMT algorithms are superior to other single tree methods in all the cases. However, the GMTs which employed SSRML as split criteria show more consistent performance and higher correlation for the the training and testing data. This can be observed in the dataset **a**, **b**, **d** and **g**. Also, comparing the testing and training correlation in dataset **b** shows that more consistent performance can be achieved using the k-fold cross-validation induction process. This also reveals that the new

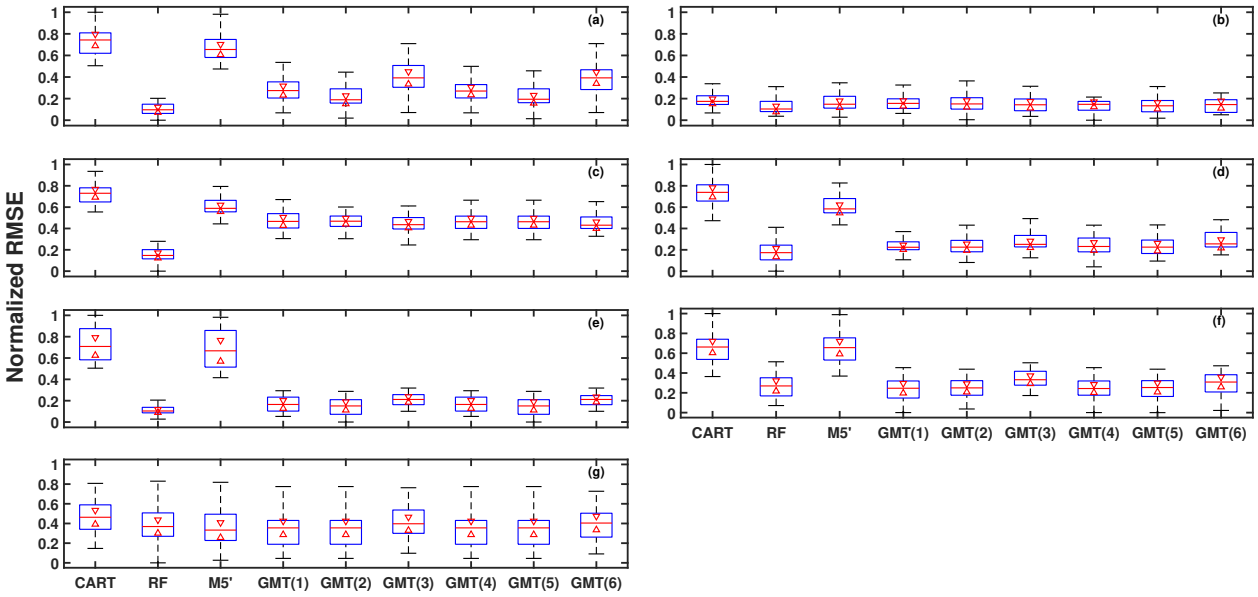


Figure 4.4: Boxplots of the RMSE values for 30-fold cross-validation on the airfoil self-noise(a), auto MPG (b), combined cycle power plant (c), concrete compressive strength (d), energy efficiency-heating load (e), energy efficiency-cooling load (f), Istanbul stock exchange (g) testing datasets. The notches are shown with red triangles.

induction process reduces bias in variable selection; hence the performance of the model is more consistent for training and testing data.

Figure 4.5 also shows the depth of each tree by red squares. The tree-depth shows the maximum number of decision nodes an instance should pass to reach a terminal node. Shallow trees are expected to have simpler structure and fewer decision nodes. On average, M5' offers the most simple representation of the underlying system. In most case studies, trees inducted by the M5' algorithm have depths of 3 or less. The reason for this is the pruning method employed in the M5' algorithm Wang and Witten (1997). M5' prunes the resulted model tree by estimating the nodes error for *unseen* data. This estimation is derived by multiplying the training data error by a factor, as training data error underestimates the error for test data. This process leads to a more simplified structure for the model. Among the other algorithms, CART, GMT(3), and GMT(6) generate the largest trees. The large depth of trees in these algorithms is attributed to the SSRM criteria employed for evaluating the split candidates.

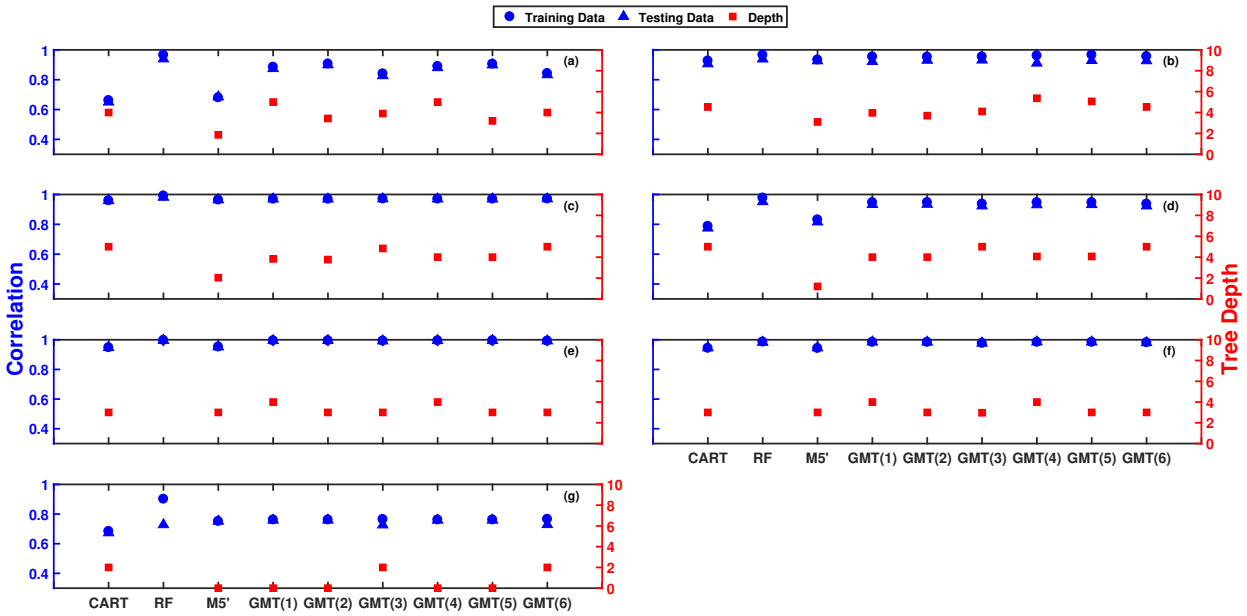


Figure 4.5: The average correlation between simulation and observed data and also the average depth of the trees generated by each algorithm for 30-fold cross-validation on the airfoil self-noise(a), auto MPG (b), combined cycle power plant (c), concrete compressive strength (d), energy efficiency-heating load (e), energy efficiency-cooling load (f), Istanbul stock exchange (g) datasets. Correlation for training data and testing data are shown in blue circles and triangles, respectively. Tree-depths are shown in red squares.

Among GMT settings, GMT(1), GMT(2), GMT(4), and GMT(5) generate smaller trees, however, GMT(2), and GMT(5) have the least depth. The dimension reduction process in the GMT algorithm seems to affect the attribute selection in GMT(1), and GMT(4), hence, the depths of the trees are larger in these cases. As PCA reduces dimension solely on the variance, it can remove informative dimensions during the regression process. Removing the informative dimensions can introduce error into the regression process. This can change the structure of the trees. On the other hand, the larger error during the regression process affects the minimum improvement stopping criteria. Hence, the algorithm may partition data into more clusters to further reduce the error. In general, the effect of dimension reduction on data depends on the characteristics of data.

The effect of the k-fold cross-validation induction mechanism on tree depth is evident comparing the depth of the tree for GMT families. In general, the new induction scheme employed in GMT(1), GMT(2), and GMT(3) controls the depth of the trees more rigorously than other

GMT settings. In this process, the k-fold cross-validation controls the error improvement to stop the partitioning process. Hence, the error is the average error on multiple samples of data, rather than the whole data, so it tends to be smaller and less biased. Therefore, the k-fold cross-validation setting within GMT reduces the depth of the trees and offers a more simplified representation of the underlying system. In the following section, the GMT framework is tested on larger datasets to investigate the potential of the framework for reservoir routing.

4.4 Conclusions and Remarks

In this chapter a new data-mining framework, titled Generalized Model Tree (GMT) is introduced for simulating rule-based systems. The proposed algorithm provides a flexible tool for generating tree-based models. The newly developed framework allows employing different split criteria for tree induction, which result in trees with different structure. The GMT framework also benefits from multiple modules to enhance the efficiency and robustness of the resulted models. The algorithm employs a dimension reduction scheme to improve the induction speed of the trees, while maintain the accuracy of the models. Also, the exhaustive search mechanism of the conventional tree induction algorithms is replaced with a equidistant quantile sampling method. This new tree induction approach is combined with a k-fold cross-validation induction scheme to generate more robust model trees. This process can reduce bias in variable selection while increases the induction speed of the algorithm. The growing process is also controlled with the k-fold cross-validation scheme, which resulted in more simplified trees. The presented results support the effectiveness of the implemented modules for tree induction process. To better understand the effect of these modules, combination of settings for GMT has been tested on multiple test cases. The presented results

showed that the default settings of the GMT algorithm (GMT(1) and GMT(2)) offer more accurate models with less computational burden. However, the dimension reduction mechanism (employed in GMT(1)) may introduce some error during the induction process. Hence, the dimension reduction module (GMT(1)) is recommended for larger datasets. In other cases GMT(2) can produce robust and simplified trees.

The experiments also revealed useful information about the effect of split criteria. As MTs employ multiple linear regression in terminal nodes, considering linearity during the partitioning process led to more accurate and efficient models. The default setting of the GMT algorithm consider linearity during the tree induction process. Besides, the GMT framework leverage its simplicity to the multiple linear regression models employed in the terminal nodes. Hence, the generated models can be easily used and combined with other models. This feature of the algorithm allows implementing the inducted trees into any programming languages without the need for external libraries and complex functions.

This chapter of the dissertation serves as the introduction to the GMT framework. However, there are multiple directions for improving the performance of the algorithm. The PCA dimension reduction method has been observed to introduce some error to the model. Hence, employing other dimension reduction procedures can enhance the performance of the GMT algorithm, specially on larger datasets. Another potential improvement for the GMT algorithm is to employ the framework with an ensemble algorithm. The robust performance of the algorithm can be further improved by using the algorithm in the form of forest rather than single tree. However, this approach will reduce the transparency of the GMT framework.

Chapter 5

Application of the GMT Framework for Reservoir Routing

5.1 Introduction

Simulating reservoir routing in hydrologic models is coined with multiple hardships. In practice, the storage-release relationships, so called rule curves, are used for operating reservoir systems (Chang et al., 2005; Yang et al., 2016). However, the operators deviate from these rules according to the constraints and condition of the reservoir (Oliveira and Loucks, 1997). Also, many reservoir models require detailed information about the operational rules and constrains of the system. Hence, application of these models can be restricted by their complexity due to the existing details (Draper et al., 2004). On the other hand, recent improvements in the computational efficiency of high-resolution river flow modeling have changed the river flow modeling paradigm from local to high-resolution continental scale (e. g. Tavakoly et al. (2012); Yamazaki et al. (2013); David et al. (2013); Tavakoly et al. (2017)). The implementation of continental-scale modeling system offers unprecedented in-

sight into river system dynamics (Lin et al., 2018; Salas et al., 2018; Snow et al., 2016). However, modeling surface water is a real challenge without the inclusion of reservoirs and can result in a non-realistic representation of actual surface water flows (David et al., 2015). Tavakoly et al. (2017) showed that, based on the reservoir storage capacity, location, and magnitude of the controlled discharge, the reservoirs can potentially gain $12-150\%$ accuracy in streamflow simulation of the river basin. The GMT framework can incorporate the effect of reservoir systems in continental scale river routing to enhance the accuracy of streamflow simulation. The algorithm can generate models for reservoir routing based on available historical data, and without using information about operational rules and constraints of the system. Of course, availability of this information can enhance the performance of the model. At the same time, the transparent structure of the models generated by the GMT framework provide a better insight into the underlying system. Here, I test the GMT framework on multiple reservoir systems to investigate the potential application of the framework for reservoir routing.

The rest of this chapter is organized as follows. In section 5.2 details of the selected reservoirs and their location are presented. Section 5.3 details the experiment and the settings employed for each model tree. Section 5.4 presents the results and compares the performance of the GMT framework with other popular tree-based algorithms. At last, section 5.5 concludes this chapter and mentions the potential directions for future investigations.

5.2 Reservoir Case Study

In this section the GMT framework is employed to simulate daily discharge from eight different reservoirs to evaluate the performance of the algorithms on rule-based hydrologic systems. Figure 5.1 shows the location of these reservoirs across CONUS. The selected reservoirs provide different ranges of services including flood control, water supply, recreation,

Figure 5.1: Location of the selected reservoirs

and hydropower. Four out of the eight selected reservoirs provide hydropower, among which three of them are located in California. For most selected reservoirs, the study period covers more than 10 years, however, nine years of data are used for the Coralville dam, due to availability of data. The minimum leaf size for the reservoir case studies are larger than the ones used for benchmark dataset in the previous chapter. The value of the minimum leaf size for each reservoir is selected according to the number of instances and performance of the algorithms. The optimum minimum leaf size for each reservoir and algorithms needs further investigation. Here, the same leaf size is employed to compare the performance of the models with similar settings and complexity. Table 5.1 lists the details of the selected reservoirs and the minimum leaf size for each case.

5.3 Experiment Design

The settings used for GMT are similar to the settings used for the benchmark dataset in the previous chapter. These settings are compared against the CART, M5', and RF algo-

Table 5.1: The selected reservoirs and the minimum leaf size.

Case	Reservoir	State	City	Services	Study Period	Min. leaf size
a	Arkabutla	MS	Tunica	Flood Control, Recreation	1999-2016	500
b	Coralville	IA	Iowa City	Flood Control, Recreation	2005-2013	500
c	Dale Hollow	TN	Celina	Hydropower, Flood Control, Recreation	2000-2012	500
d	El Dorado	KS	El Dorado	Flood Control, Water Supply, Recreation	2000-2016	1000
e	Folsom	CA	Folsom	Hydropower, Flood Control, Water Supply, Recreation	2001-2017	500
f	Prado	CA	Corona	Flood Control	2000-2016	500
g	Shasta	CA	Shasta Lake	Hydropower, Flood Control, Water Supply, Recreation	2004-2016	500
h	Trinity	CA	Lewiston	Hydropower, Flood Control, Water Supply, Recreation	2000-2017	1000

rithms. Here, I employ storage and inflow data as input for the algorithms. The advantage of employing storage and inflow information is the availability data. As inflow data can be simulated by hydrologic models, and storage data can be obtained by mass balance, this information can be obtained within most hydrologic models. Hence, trees inducted by this information can be employed for reservoir routing in hydrologic models. Although, lagged discharge data is suggested as indicators for reservoir releases (Hejazi et al., 2008), discharges at previous time steps are not considered here. The high autocorrelation of discharge data can deteriorate time series simulations. Figure 5.2 shows autocorrelation for the discharge data for the selected reservoir cases. For almost all the selected reservoirs, the autocorrelation remains above 0.5 after seven lags (days). Due to the high autocorrelation, the MT algorithms tend to employ lagged data for regression in the terminal nodes. However, employing lagged discharge can propagate the simulation error, and increase error in long term simulations. Hence, storage and inflow information are used for all the reservoirs rather than for previous reservoir releases. Using lagged storage and inflow data is also avoided as discharge can be derived from these data using linear regression. Yet, lagged data can provide information about the previous state of the reservoir and seasonality. So, to provide this information to the algorithm, average storage and inflow data in the last 7, 14, and 30 days is used instead of lagged data. Due to availability of data for the Corallvile reservoir, only inflow data is employed for training the models in this case study. It is worth noting that, the storage volume for some reservoirs was not available, so the storage level is employed in these cases. To evaluate the performance of the algorithms on reservoir datasets, a k-fold cross-validation scheme is employed here. In the cross-validation process each *year* of data is considered as a fold. Each fold of data is first removed, and the remaining folds are used for training the model. Then the trained model is tested on the removed fold. This process is repeated for every single year (fold) in the data, and a comparison is carried out according to the performance of the algorithms on all the folds. Hereafter, the removed year is referred to as testing data, and the remaining data is the training data.

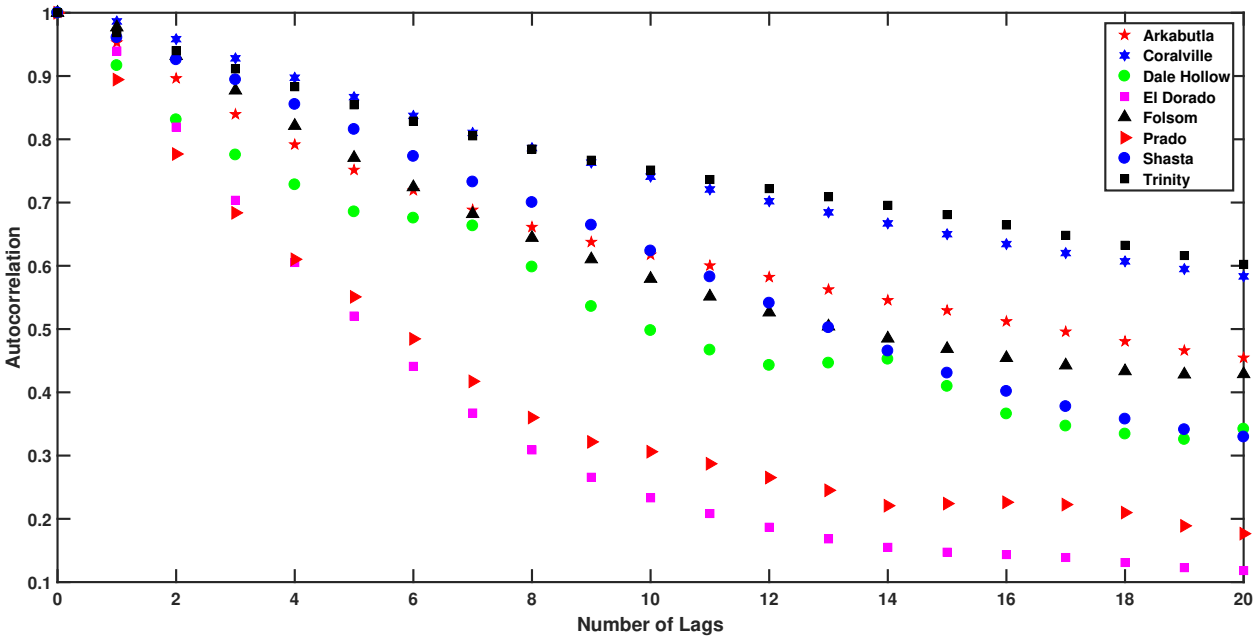


Figure 5.2: Autocorrelation of reservoir discharge for different lags for the selected reservoirs.

5.4 Results

Figures 5.3 and 5.4 show the boxplots for the normalized RMSE values for the training and testing data, respectively. The notches are shown with red triangles. According to the training data results (Figure 5.3) RF has the best performance among the algorithms as expected. Among the single tree algorithms, CART has the highest median for RMSE in almost all reservoirs, except reservoir **d** (El Dorado), **e** (Folsom), and **g** (Shasta). In these reservoirs, CART has similar performance to the M5' algorithm. Comparing CART and GMT(6) reveals the significant effect of the linear regression in terminal nodes. Although these two algorithms generate models with same structures, GMT(6) which employs linear model has better performance. GMT(6) is superior to CART in all the cases, however, the algorithm is slightly inferior to other GMTs. This shows the effect of the SC on the performance of the model trees. In most cases, the GMTs which employed SSRML as SC offer lower median for RMSEs, in comparison to GMTs which employed SSRM. The significant effect of SSRML criteria can be observed in reservoir **a** (Arkabutla). In the rest

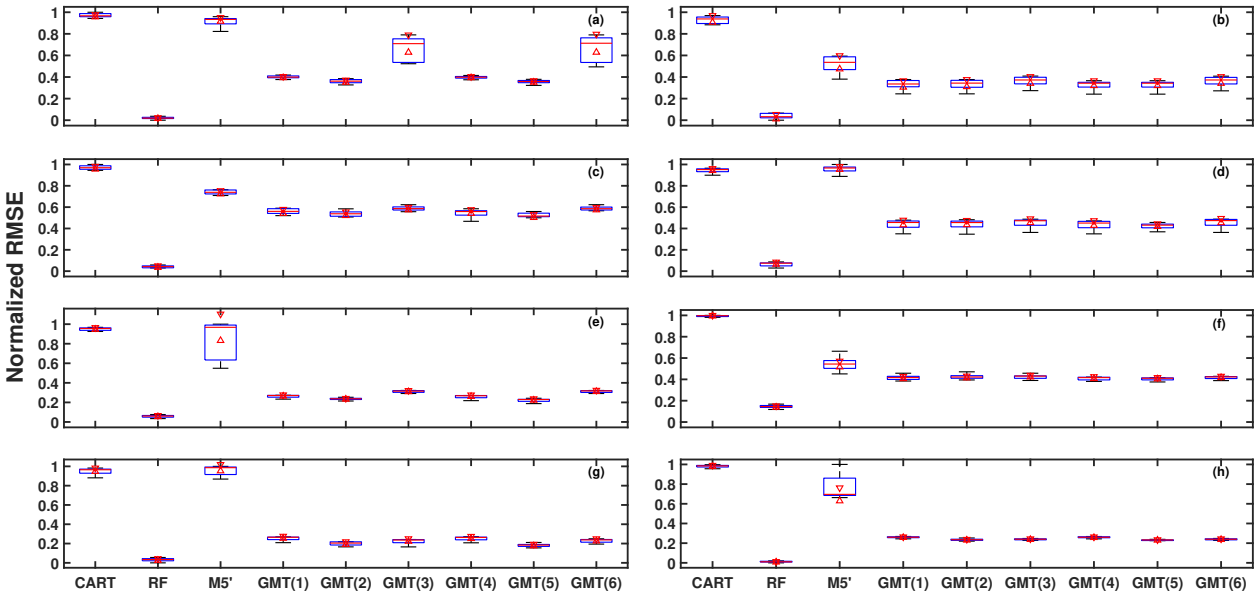


Figure 5.3: Boxplots of the normalized RMSE values for k-fold cross-validation on Arkabutla(a), Coralville (b), Dale Hollow (c), El Dorado (d), Folsom (e), Prado (f), Shasta (g), and Trinity (h) reservoir training datasets. The notches are shown with red triangles.

of the cases, the algorithms have similar performance on the training dataset. Among the rest of GMTs, GMT(1), GMT(2), and GMT(3) only employed a fixed number of splitting candidates. However, the performance of these algorithms are close to the methods which employed an exhaustive search approach. This shows that the MTs are less sensitive to the split point.

The IQR values are compared to show the consistency of the algorithms in their performance. RF offer the most stable performance on the datasets. The algorithm offer the least IQR on different years of data. Among single tree methods, CART and M5' show similar variability in the RMSE values. These two algorithms have the highest median and IQR in all the cases. However, GMT shows performance similar to RF in most cases. The effect of the splitting criteria can also be observed by comparing IQR values for different GMTs. In reservoir **a** (Arkabutla), the algorithms with SSRM as splitting criteria offer larger IQR. This shows that the models generated by SSRML are more robust and consistent in their performance. In the rest of the cases, the algorithms have similar performance.

Figure 5.4 shows the boxplots for the normalized RMSE values for the *unseen* data. Con-

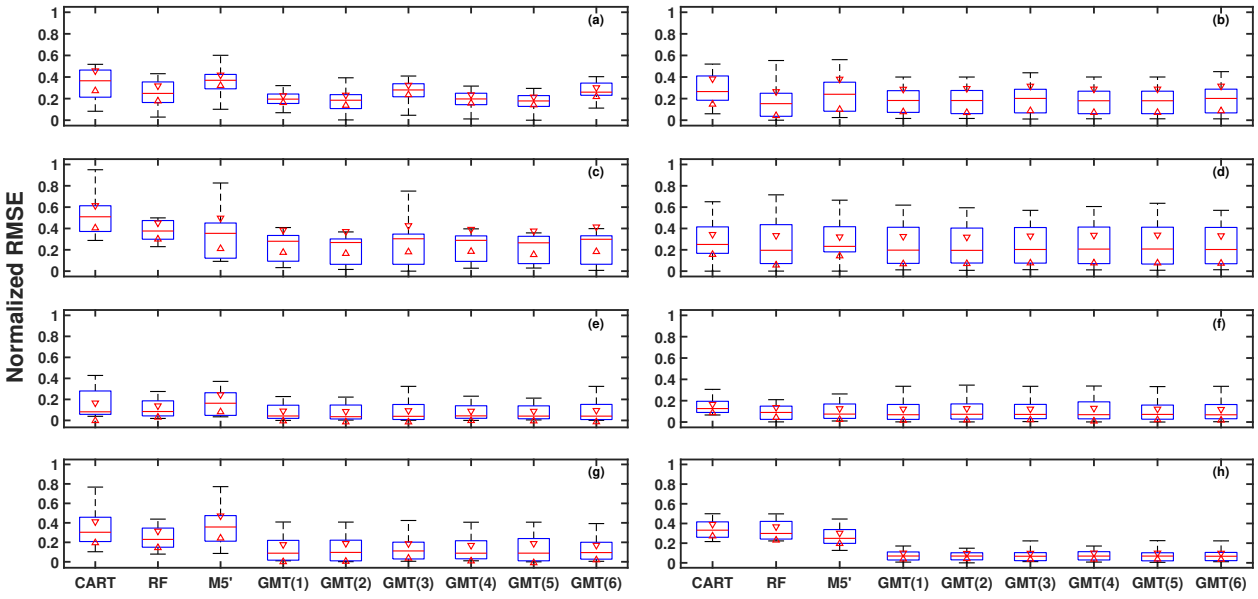


Figure 5.4: Boxplots of the normalized RMSE values for k-fold cross-validation on Arkabutla(a), Coralville (b), Dale Hollow (c), El Dorado (d), Folsom (e), Prado (f), Shasta (g), and Trinity (h) testing datasets. The notches are shown with red triangles.

trary to the training folds, the RF algorithm is inferior to other algorithms in almost all the cases. In most cases, RF is outperformed by GMT family algorithms. This shows that RF over-fitted the training data. Among single tree methods, CART is the worst performing algorithm followed by M5'. In cases **a**, **d**, and **f** these two algorithms show very close performance. The GMT family algorithms are superior to other algorithms in most cases. Among the GMT settings, the GMTs which employed SSRML are superior to ones that used SSRM. This can be observed in the Arkabutla and Dale Hollow (case **a** and case **c**) reservoir. In other reservoirs, the difference between GMTs are not significant. In the testing folds, the algorithms which employed k-fold cross-validation scheme shows slightly better performance in terms of the median of RMSEs. In the rest of the cases, the GMTs have close performance and no significant difference is observed. The GMTs have close behavior in term of IQR. In almost all the cases, they offer lower IQR in comparison to other single tree methods. These results show that the speed enhancement modules in the GMT framework maintained the performance of the algorithm.

In addition, the average correlation for each of the models is presented in Figure 5.5. In this

figure, correlation for training and testing folds are shown with blue circles and triangles, respectively. Among the algorithms, RF achieved the highest correlation for the training folds. However, the RF algorithm shows the highest reduction in correlation with regards to the training and testing data. This shows that RF is over-fitting the training data. Comparing single tree methods, CART and M5' offer the lowest correlation among all the algorithms for the training and testing period. However, this reduction is less than that observed for RF. Although RF outperforms GMT family comparing the correlation for training data, the GMT algorithm is superior to RF in the test cases. Among GMT algorithms, the algorithms with SSRML as splitting criteria are superior to ones with SSRM, in all the cases. Hence, the variability of data is better presented by these algorithms. This can be seen in reservoir cases **a**, **c**, **d**, and **f**. The effect of k-fold cross-validation scheme is well depicted in cases **d**, **f**, and **g**. In these cases the new induction method has reduced the difference between correlation for the training and testing data. This shows that the model generated by the k-fold cross-validation is more robust and less bias in variable selection. The rest of the settings offer similar correlation in most cases.

Figure 5.5 also shows the depth of the inducted trees. As RF is an ensemble method, no depth is reported for the algorithm and comparison is made among the single tree methods. On average, the M5' algorithm generated models with the least number of branches. This is due to the pruning method employed in the M5' algorithm Wang and Witten (1997). CART algorithms and GMT(6) have the same depth as both algorithms employ the same SC. Among other GMT algorithms, the algorithms with k-fold cross-validation induction scheme have simplified the structure of the models in most cases. The k-fold cross-validation scheme evaluates the stopping criteria on multiple samples of data. Hence, the depth of the trees in these cases are monitored and controlled more rigorously on multiple samples of data. This can be observed in reservoir **b**, **d**, **f**, and **h**. On the other hand, the dimension reduction scheme has slightly increased the depth of the trees. As discussed in the previous section, the dimension reduction mechanism can introduce some error into the regression

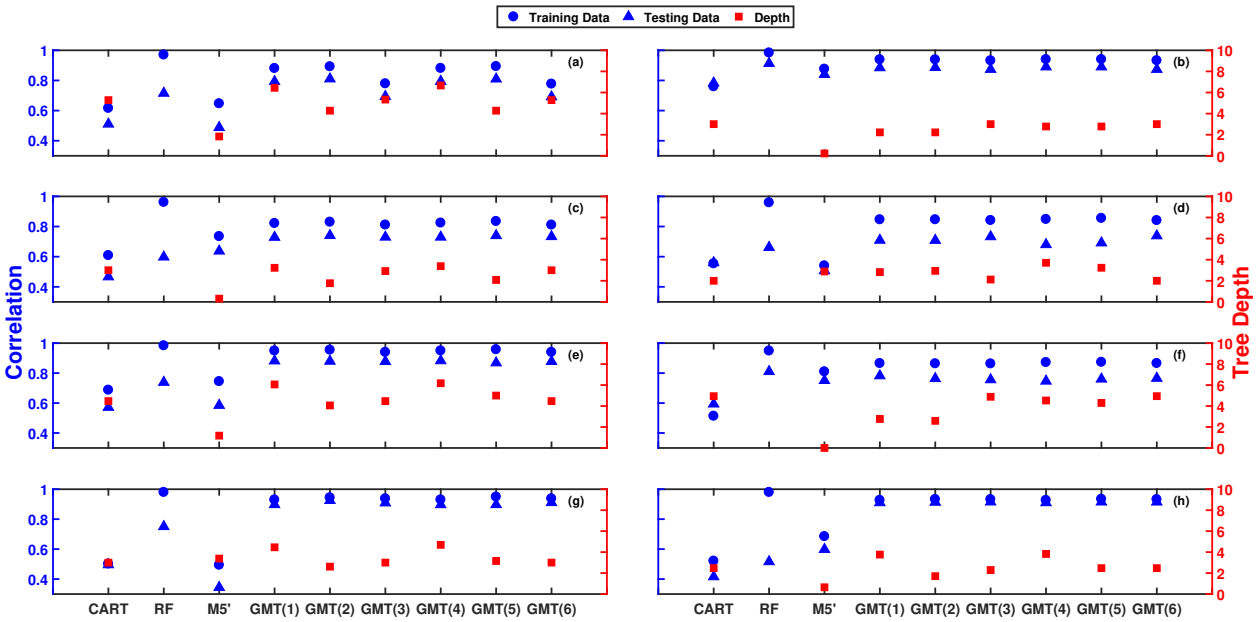


Figure 5.5: The average correlation between simulation and observed data and also the average depth of the trees generated by each algorithm for k-fold cross-validation on Arkabutla(a), Coralville (b), Dale Hollow (c), El Dorado (d), Folsom (e), Prado (f), Shasta (g), and Trinity (h) datasets. Correlation for training data and testing data are shown in blue circles and triangles, respectively. Tree-depths are shown in red squares.

process. This can affect the stopping criteria during the tree induction process. However, dimension reduction is solely employed for speed enhancement of the GMT algorithm.

The setting of the GMT algorithm can significantly affect the induction time of the algorithm. Evaluating a fixed number of splitting candidates, besides the dimension reduction mechanism, has significantly affected the induction time of the algorithm in the presented cases. Monitoring the induction speed of the algorithms revealed that on average $100-300\%$ and $25-200\%$ speed enhancement can be observed from the new induction mechanism and dimension reduction, respectively. Among the presented settings of the GMT framework, GMT(1) has the lowest induction time in most datasets. The induction speed enhancement is mostly observed in datasets with more than 10000 instances and 4 explanatory variables. Of course, the significance of these improvement will be more compelling in larger datasets. To further investigate the structure of the models generated by GMTs, two out of the eight presented reservoirs are selected for more detailed study. Prado and Folsom are selected as

storage volume data is available for both of these cases. Hence, the observed storage volume can be compared with the one simulated by mass balance and GMT model. Here, another experiment is conducted by using the first 90% of data period for training the model and the remaining 10% as the test period. Figure 5.6 shows the simulated storage (plot (a)) and simulated discharge (plot (b)) for the GMT(1), GMT(2), GMT(3), M5', CART, and RF algorithms for Prado reservoir for the testing period. The first 3 GMTs are only selected here for demonstration purposes, as these GMTs have shown better or similar performance to other GMTs. The results show that the simulated storage and discharge are better represented by GMTs. This shows the effect of linear regression in terminal nodes. Among GMT algorithms, the algorithms which employed SSRML as SC have better performance (GMT(1) and GMT(2)). GMT(1) and GMT(2) captured the variability and peak flows better than other algorithms. GMT(3) shows a close performance to GMT(1) and GMT(2), but the algorithm underestimates a few peaks and generates negative discharges for some days. The CART and M5 algorithms are inferior to other algorithms in most of the simulations. In many situations, the storage simulated by these algorithms is negative. This is due to the overestimation of the reservoir releases.

Figure 5.7 shows the structure of the trees generated by each of the GMT algorithms. The models generated by different GMTs have different structure and complexity. The GMTs which employed SSRML as SC (GMT(1) and GMT(2)) seem to offer a more simplified structure. In the trees generated by GMT(1), the root node uses x_5, which is the storage in the previous day, as the decision rule, however, GMT(2) employs x_1, which is the inflow in previous day, as splitting point at root node. The tree generated by GMT(1) and GMT(2) are mostly using the storage related information for splitting, however, GMT(3) employs rules combining information from storage and inflow. Although GMT(1) employs x_5 as the decision rules in all the nodes, all terminal nodes employ previous day inflow as a variable in the regressed lines. However, the storage information is mostly employed in the regressed line. Similar case is observed in the model generated by GMT(2) and GMT(3). The result

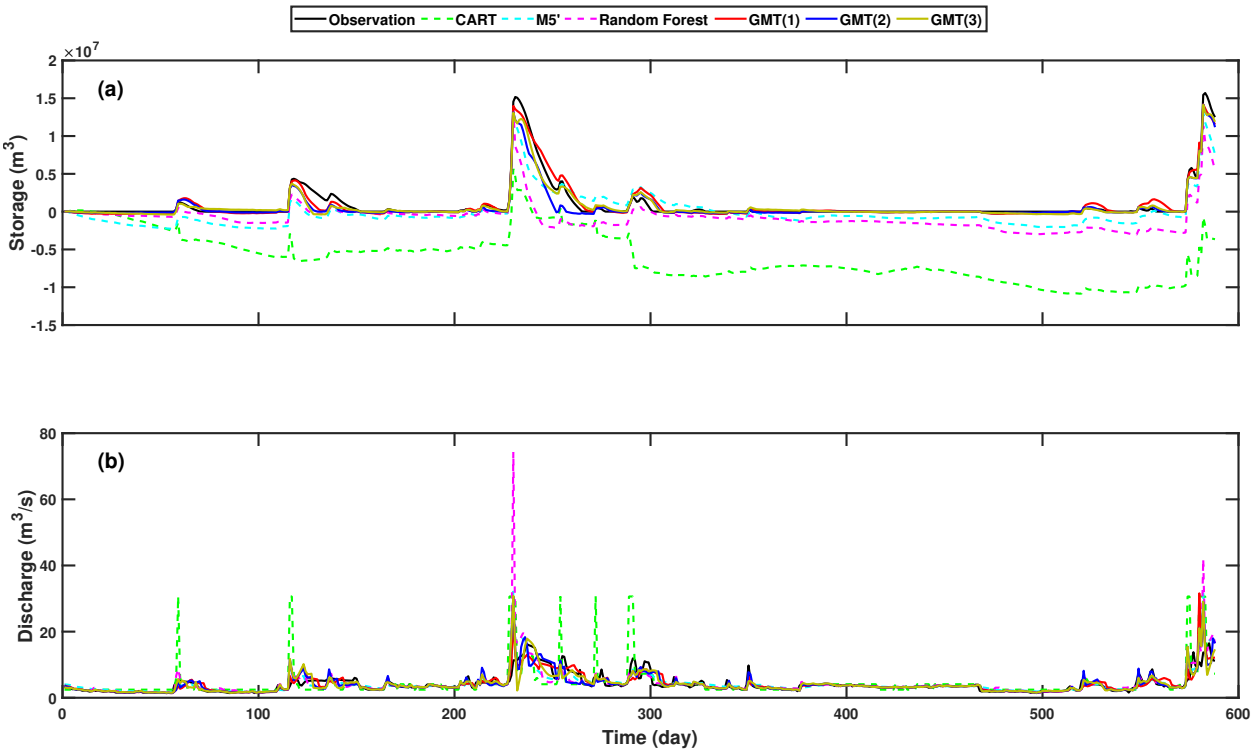
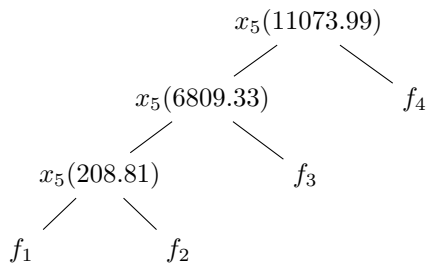


Figure 5.6: Simulated storage (a) and discharge (b) for Prado dam.

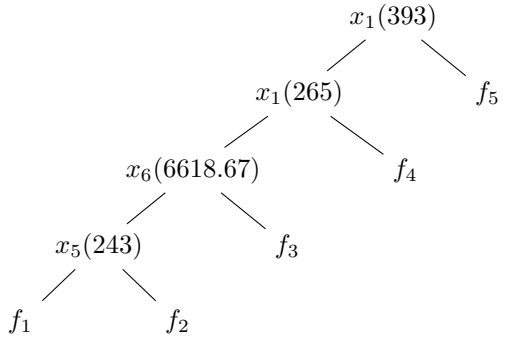
from this section shows that there is not a unique tree structure for the model. Also, trees with different structures can have very close performance.

A similar experiment is conducted for the Folsom reservoir, which provides a wide range of services in California. These services include: flood control, hydropower, recreation, environmental protection, water supply and drought management (Yao and Georgakakos, 2001; Field and Lund, 2006b; Zhu et al., 2007). The operational rules and services that the Folsom reservoir provides make it a more complex system. Hence, the GMT framework is tested on the Folsom reservoir to evaluate its performance on a multi-purpose system. Figure 5.8 shows the storage and discharge simulation for the Folsom reservoir. In this case study, I used the first 90% of the data as training period and the rest as testing period. The reservoir discharges and storage simulated by the GMT family shows superior results in comparison to other algorithms. The RF algorithm has captured the variability of the releases, however, GMTs show better performance in simulating the peak flows. The GMTs perform similarly on the Folsom reservoir. Peak flows are better simulated by GMT(2) in comparison to

(a)



(b)



(c)

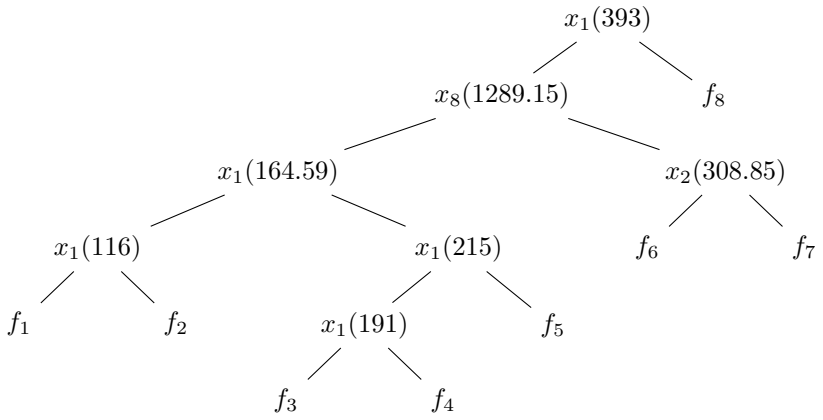


Figure 5.7: The tree structure for GMT(1) (plot a), GMT(2) (plot b), and GMT(3) (plot c) for the Prado dam. The splitting points are shown in paranthesis. The left branches corresponds to smaller than ($<$) and right branches are correspond to greater or equalt to ($\geq$). In these models, x_1 is the inflow at the previous day, x_2 is the average inflow in the last 30 days, x_5 is the storage for the previous day,x_6 is the average storage in last 30 days, and x_8 is the average storage in last 7 days.

other GMTs. Worth noting that as the Folsom reservoir provide different services, providing information about these service, such as demand for hydropower, can further enhance the performance of the GMT algorithms.

The structure of models inducted by GMTs are shown in Figure 5.9. All the GMTs employed a combination of different variables for splitting the data. The close performance of the GMT algorithms, and the differences in the structures of the models show that there is not a unique model structure. These differences partially stem from the high correlation of the variables employed here and error in observed data. This characteristic of the MTs are similar to equifinality condition in model calibration. However, it should be noted that the setting of the GMT algorithm can change the structure of the model as observed in the presented cases. Also, performance of the resulting models is a function of the employed settings. The combination of the settings presented in this section revealed the effect of some of these settings. However, the GMT algorithm has few other settings which need further investigation. Among the GMT settings, the k value in k-fold cross-validation induction mechanism can affect the model structure. Selecting very small or a very large k value for the model may lead to an unstable model structure. Although the effect of this instability may not be significant in the performance of the resulting model, the splitting variables and points can change. This behavior of the algorithm depends on the quality and characteristics of the data. Nonetheless, the recommended value of 10 for k offers a more stable result for a large range of problems.

5.5 Discussion and Conclusion

In this chapter, I employed the GMT framework for simulating the controlled discharge from multiple reservoir systems across the CONUS. The selected reservoirs provide a wide range

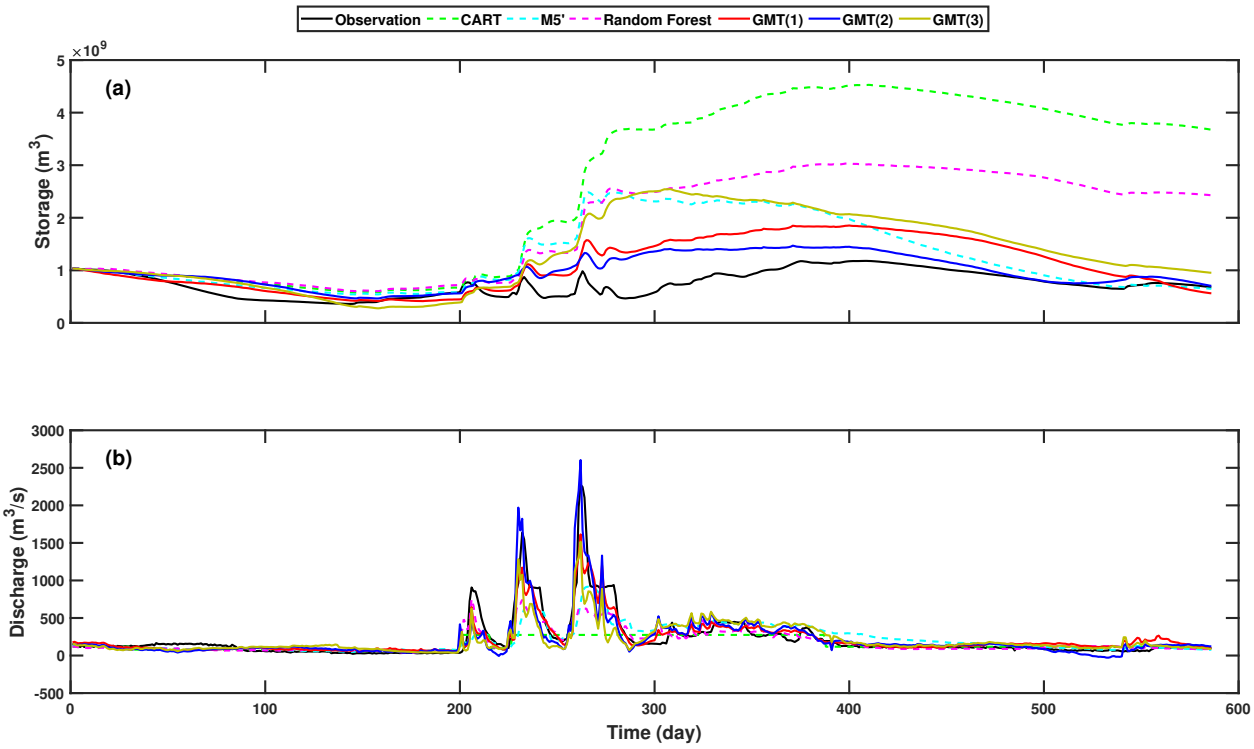
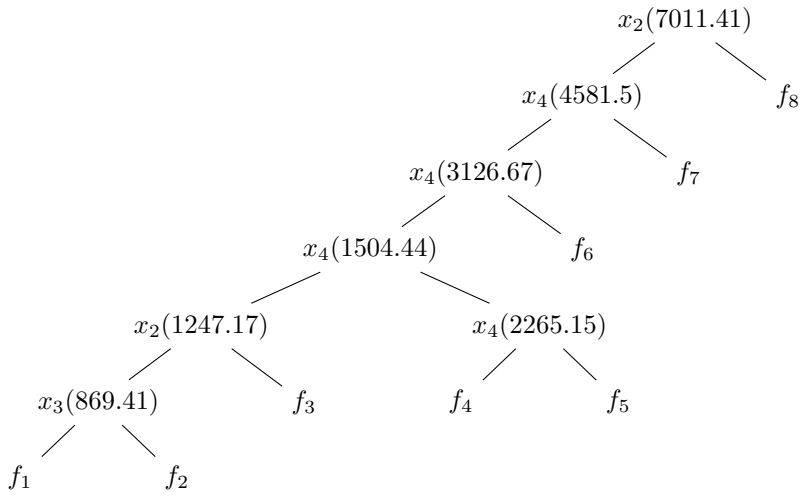


Figure 5.8: Simulated storage (a) and discharge (b) for Folsom reservoir.

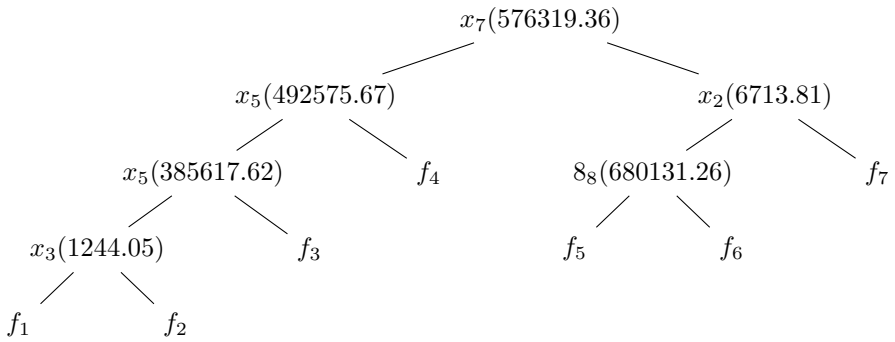
of services and represent different class of these systems. The transparent structure of the GMT framework, along with the algorithm performance make it a suitable choice for representing reservoir systems. The performance of the GMT on the reservoir case study supports the application of GMT for reservoir routing. The trees generated by GMT framework can also provide useful information about the underlying process. This allows understanding, auditing, and modifying these models. Although, information about the structure of the inducted trees are not discussed here, the structure of the trees can provide useful information about the real operation of the reservoirs.

This chapter showed the potential application of the GMT framework for reservoir routing. Although the presented results support the performance of the algorithm on different reservoir systems, there are multiple direction for future investigations. In most hydrologic systems, constraints can play significant role in the performance of the models. These constraints can be combined with the inducted trees to further enhance the performance of the models. In addition to that, the potential application of the GMT framework in hydrologic

(a)



(b)



(c)

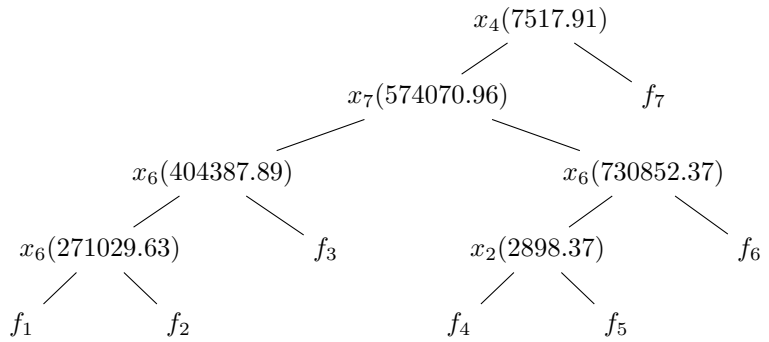


Figure 5.9: The tree structure for GMT(1) (plot a), GMT(2) (plot b), and GMT(3) (plot c) for Folsom reservoir. The splitting points are shown in paranthesis. The left branches corresponds to smaller than ($<$) and right branches are correspond to greater or equal to ($\geq$). x_2 is the average inflow in last 30 days, x_3 is the average inflow in the last 14 days, x_4 is the average inflow in the last 7 days, x_5 is the storage for the previous day, x_6 is the average storage in last 30 days, and x_7 is the average storage in the last 14 days.

models need further investigation. The output of the GMT framework can be integrated within large scale hydrologic models for reservoir routing to address the effect of reservoir operation in the river systems and to improve the modeling results. Moreover, the application of the GMT framework is not limited to reservoir systems and can be used for a wide range of systems. The GMT framework can be employed for simulating and understanding continuous spaces. At last, the structure of the GMT framework needs further investigation for understanding the represented system.

Chapter 6

Conclusions and Future Directions

With the advances in the algorithms and tools for decision making in reservoir operation, the gap between theoretical and real reservoir operation still exists (Hejazi et al., 2008). In this dissertation, tools and algorithms are offered to bridge the existing gap to some extent. My main focus in this dissertation is to develop tools and algorithms to aid decision makers and modelers in the field of reservoir operation, to understand, simulate, and improve the operation of reservoir systems. Hence, two algorithms are proposed for optimization and simulation of reservoirs.

In chapter 2 of this dissertation, a new self-adaptive hybrid optimization framework titled SC-SAHEL is introduced. The increasing number of optimization algorithms and the NFL theorem encouraged me to propose a new optimization framework for solving a wide range of optimization problems. The SC-SAHEL algorithm employs multiple search mechanisms as search cores and evaluates the performance of these search mechanisms on the problem space. During the course of the search, the algorithm allocates more resources and information (individuals from population) to methods that are performing better. This "award and punishment" process promotes the superior search methods for the problem space, hence, the overall performance of the algorithm will be enhanced. The SC-SAHEL framework is

also equipped with multiple tools and modules for initial sampling and boundary handling. Moreover, the SC-SAHEL framework reveals the performance of the search methods at each optimization step.

Chapter 3, demonstrated the application of the SC-SAHEL framework on reservoir optimization problems. In this chapter, a conceptual reservoir model is employed to optimize the controlled discharge from the Folsom reservoir. The SC-SAHEL framework is employed to tune the controlled discharge to maximize hydropower generation over a three month period. In this problem, several constraints are implemented into the model to increase the complexity of the problem space and represent the real condition of the reservoir system. The results demonstrated that the SC-SAHEL framework can be successfully used for solving real-world optimization problems. The SC-SAHEL framework outperformed the single search methods and exhibited robust performance on the reservoir optimization problem.

In addition to the optimization tool, this dissertation also offers a new framework for simulating reservoir systems. Although there are physical models for simulating the operation of reservoir systems, these models require detailed information about the real operation of reservoir. Also, in many cases the operators may deviate from these operational rules (Yang et al., 2016), so the physical models cannot represent the real operation of the reservoir systems. Hence, application of the physical models can be limited, especially in the large scale hydrolgic modeling. Hereupon, I proposed a generalized model tree (GMT) framework, to simulate the real operation of the reservoir systems. I specifically employed model trees, which is a special form of decision tree, to mimic human decision making in reservoir systems. In this dissertation, several modules are implemented to enhance the computational efficiency of the model trees, while enhancing the performance of these models.

In chapter 4, first the GMT algorithm is employed to simulate multiple benchmark datasets. The results showed that the GMT framework can outperform many existing algorithms in terms of accuracy and performance. The framework is then applied to 8 reservoir across CONUS and compared with other popular algorithms such as CART and M5' for simulating

the controlled discharge. The result showed that the algorithm outperforms other regression tree models. These results revealed the potential application of the algorithm for reservoir routing. The simple structure of the models inducted by the GMT framework make them easy to use and implement in any programming language and hydrologic model. Also, the transparent structure of the algorithm reveals useful information about the underlying system.

Although the proposed algorithms for optimization and simulation of the reservoir systems showed superior results for multiple case studies, there are multiple directions for future investigations. In general this dissertation serves as the introduction to these algorithms which have the potential for improvements and extended applicability of these methods. Some of these potentials directions are summarized as follows:

- Extend the application of the SC-SAHEL framework to many-objective optimization problems
- Implement more evolutionary algorithms into the SC-SAHEL framework to enhance the performance of the algorithm
- Investigate the effect of initial sampling and boundary handling on the performance of the SC-SAHEL framework
- Incorporate the opinion of the reservoir operators into the reservoir optimization models
- Enhance the dimension restoration mechanism employed within the SC-SAHEL framework
- Employ other algorithms for dimension reduction within the GMT framework
- Investigate the potential application of the GMT framework for reservoir routing within the hydrologic models

- Employ physical constraints into the GMT framework to improve the performance of the model
- Extend the application of the GMT framework to other rule-based hydrologic systems.
- Employ the GMT framework within an ensemble framework to further enhance the performance of the model
- Investigate the structure of the models inducted by the GMT framework to better understand the underlying system
- Train the GMT framework on optimized reservoir discharges to extract optimum operational rules for the reservoir systems

Bibliography

Afshar, M. and Shahidi, M. (2009). Optimal solution of large-scale reservoir-operation problems: Cellular-automata versus heuristic-search methods. *Engineering optimization*, 41(3):275–293.

Ahmad, A., El-Shafie, A., Razali, S. F. M., and Mohamad, Z. S. (2014). Reservoir optimization in water resources: a review. *Water resources management*, 28(11):3391–3405.

Ajami, N. K., Gupta, H., Wagener, T., and Sorooshian, S. (2004). Calibration of a semi-distributed hydrologic model for streamflow estimation along a river system. *Journal of Hydrology*, 298(1-4):112–135.

Arlot, S., Celisse, A., et al. (2010). A survey of cross-validation procedures for model selection. *Statistics surveys*, 4:40–79.

Asuncion, A. and Newman, D. (2007). Uci machine learning repository.

Azamathulla, H. M., Wu, F.-C., Ab Ghani, A., Narulkar, S. M., Zakaria, N. A., and Chang, C. K. (2008). Comparison between genetic algorithm and linear programming approach for real time operation. *Journal of Hydro-environment Research*, 2(3):172–181.

Barati, R., Neyshabouri, S., and Ahmadi, G. (2014). Sphere drag revisited using shuffled complex evolution algorithm. In *River flow*, pages 345–353.

Belaineh, G., Peralta, R. C., and Hughes, T. C. (1999). Simulation/optimization modeling for water resources management. *Journal of water resources planning and management*, 125(3):154–161.

Bessler, F. T., Savic, D. A., and Walters, G. A. (2003). Water reservoir control with data mining. *Journal of water resources planning and management*, 129(1):26–34.

Beven, K. J. (2011). *Rainfall-runoff modelling: the primer*. John Wiley & Sons.

Bhattacharya, B. and Solomatine, D. P. (2005). Neural networks and m5 model trees in modelling water level–discharge relationship. *Neurocomputing*, 63:381–396.

Blum, C. and Roli, A. (2003). Metaheuristics in combinatorial optimization: Overview and conceptual comparison. *ACM computing surveys (CSUR)*, 35(3):268–308.

BoussaïD, I., Lepagnot, J., and Siarry, P. (2013). A survey on optimization metaheuristics. *Information Sciences*, 237:82–117.

Boyle, D. P., Gupta, H. V., and Sorooshian, S. (2000). Toward improved calibration of hydrologic models: Combining the strengths of manual and automatic methods. *Water Resources Research*, 36(12):3663–3674.

Breiman, L. (2001). Random forests. *Machine learning*, 45(1):5–32.

Breiman, L., Friedman, J., Stone, C. J., and Olshen, R. A. (1984). *Classification and regression trees*. CRC press.

Broman, K. W. and Speed, T. P. (2002). A model selection approach for the identification of quantitative trait loci in experimental crosses. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 64(4):641–656.

Castelletti, A., Galelli, S., Restelli, M., and Soncini-Sessa, R. (2010). Tree-based reinforcement learning for optimal water reservoir operation. *Water Resources Research*, 46(9).

Cervellera, C., Chen, V. C., and Wen, A. (2006). Optimization of a large-scale water reservoir network by stochastic dynamic programming with efficient state space discretization. *European journal of operational research*, 171(3):1139–1151.

Chang, F.-J., Chen, L., and Chang, L.-C. (2005). Optimizing the reservoir operating rule curves by genetic algorithms. *Hydrological processes*, 19(11):2277–2289.

Chang, L.-C. and Chang, F.-J. (2001). Intelligent control for modelling of real-time reservoir operation. *Hydrological processes*, 15(9):1621–1634.

Cheng, L. and AghaKouchak, A. (2014). Nonstationary precipitation intensity-duration-frequency curves for infrastructure design in a changing climate. *Scientific reports*, 4:7093.

Chu, W., Gao, X., and Sorooshian, S. (2010). Improving the shuffled complex evolution scheme for optimization of complex nonlinear hydrological systems: Application to the calibration of the sacramento soil-moisture accounting model. *Water Resources Research*, 46(9).

Chu, W., Gao, X., and Sorooshian, S. (2011). A new evolutionary search strategy for global optimization of high-dimensional problems. *Information Sciences*, 181(22):4909–4927.

Chu, W., Yeh, W. W.-G., and Rossman, L. A. (1979). A nonlinear programming algorithm for real-time hourly reservoir operations. *JAWRA Journal of the American Water Resources Association*, 15(4):1178–1180.

Ciampi, A. (1991). Generalized regression trees. *Computational Statistics & Data Analysis*, 12(1):57–78.

Coello, C. A. C., Lamont, G. B., Van Veldhuizen, D. A., et al. (2007). *Evolutionary algorithms for solving multi-objective problems*, volume 5. Springer.

Črepinšek, M., Liu, S.-H., and Mernik, M. (2013). Exploration and exploitation in evolutionary algorithms: A survey. *ACM Computing Surveys (CSUR)*, 45(3):35.

David, C. H., Famiglietti, J. S., Yang, Z.-L., and Eijkhout, V. (2015). Enhanced fixed-size parallel speedup with the muskingum method using a trans-boundary approach and a large subbasins approximation. *Water Resources Research*, 51(9):7547–7571.

David, C. H., Yang, Z.-L., and Famiglietti, J. S. (2013). Quantification of the upstream-to-downstream influence in the muskingum method and implications for speedup in parallel computations of river flow. *Water Resources Research*, 49(5):2783–2800.

De'ath, G. and Fabricius, K. E. (2000). Classification and regression trees: a powerful yet simple technique for ecological data analysis. *Ecology*, 81(11):3178–3192.

Diks, C. G. and Vrugt, J. A. (2010). Comparison of point forecast accuracy of model averaging methods in hydrologic applications. *Stochastic Environmental Research and Risk Assessment*, 24(6):809–820.

Ding, Y., Wang, C., Chaos, M., Chen, R., and Lu, S. (2016). Estimation of beech pyrolysis kinetic parameters by shuffled complex evolution. *Bioresource technology*, 200:658–665.

Dorfman, R. (1962). *Mathematical models: The multistructure approach*.

Dorigo, M., Maniezzo, V., and Colorni, A. (1996). Ant system: optimization by a colony of cooperating agents. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 26(1):29–41.

Draper, A. J., Munévar, A., Arora, S. K., Reyes, E., Parker, N. L., Chung, F. I., and Peterson, L. E. (2004). Calsim: Generalized model for reservoir system analysis. *Journal of Water Resources Planning and Management*, 130(6):480–489.

Duan, Q., Gupta, V. K., and Sorooshian, S. (1993). Shuffled complex evolution approach for effective and efficient global minimization. *Journal of optimization theory and applications*, 76(3):501–521.

Duan, Q., Sorooshian, S., and Gupta, V. (1992). Effective and efficient global optimization for conceptual rainfall-runoff models. *Water resources research*, 28(4):1015–1031.

Duan, Q., Sorooshian, S., and Gupta, V. K. (1994). Optimal use of the sce-ua global optimization method for calibrating watershed models. *Journal of hydrology*, 158(3-4):265–284.

Eckhardt, K. and Arnold, J. (2001). Automatic calibration of a distributed catchment model. *Journal of hydrology*, 251(1-2):103–109.

Erol, O. K. and Eksin, I. (2006). A new optimization method: big bang–big crunch. *Advances in Engineering Software*, 37(2):106–111.

Etemad-Shahidi, A. and Mahjoobi, J. (2009). Comparison between m5 model tree and neural networks for prediction of significant wave height in lake superior. *Ocean Engineering*, 36(15-16):1175–1181.

Etemad-Shahidi, A. and Taghipour, M. (2012). Predicting longitudinal dispersion coefficient in natural streams using m5 model tree. *Journal of hydraulic engineering*, 138(6):542–554.

Eusuff, M., Lansey, K., and Pasha, F. (2006). Shuffled frog-leaping algorithm: a memetic meta-heuristic for discrete optimization. *Engineering optimization*, 38(2):129–154.

Eusuff, M. M. and Lansey, K. E. (2003). Optimization of water distribution network design using the shuffled frog leaping algorithm. *Journal of Water Resources planning and management*, 129(3):210–225.

Farmani, R. and Wright, J. A. (2003). Self-adaptive fitness formulation for constrained optimization. *IEEE Transactions on Evolutionary Computation*, 7(5):445–455.

Feng, M., Liu, P., Guo, S., Shi, L., Deng, C., and Ming, B. (2017). Deriving adaptive operating rules of hydropower reservoirs using time-varying parameters generated by the e n kf. *Water Resources Research*, 53(8):6885–6907.

Field, R. and Lund, J. R. (2006a). *Multi-Objective Optimization of Folsom Reservoir Operation*, pages 205–214.

Field, R. and Lund, J. R. (2006b). Multi-objective optimization of folsom reservoir operation. In *Operating Reservoirs in Changing Conditions*, pages 205–214.

Freer, J., Beven, K., and Ambroise, B. (1996). Bayesian estimation of uncertainty in runoff prediction and the value of data: An application of the glue approach. *Water Resources Research*, 32(7):2161–2173.

Friedl, M. A. and Brodley, C. E. (1997). Decision tree classification of land cover from remotely sensed data. *Remote sensing of environment*, 61(3):399–409.

Galelli, S. and Castelletti, A. (2013a). Assessing the predictive capability of randomized tree-based ensembles in streamflow modelling. *Hydrology and Earth System Sciences*, 17(7):2669.

Galelli, S. and Castelletti, A. (2013b). Tree-based iterative input variable selection for hydrological modeling. *Water Resources Research*, 49(7):4295–4310.

Gan, T. Y. and Biftu, G. F. (1996). Automatic calibration of conceptual rainfall-runoff models: Optimization algorithms, catchment conditions, and model structure. *Water resources research*, 32(12):3513–3524.

Garcia-Gonzalez, J. and Castro, G. A. (2001). Short-term hydro scheduling with cascaded and head-dependent reservoirs based on mixed-integer linear programming. In *Power Tech Proceedings, 2001 IEEE Porto*, volume 3, pages 6–pp. IEEE.

Geurts, P., Ernst, D., and Wehenkel, L. (2006). Extremely randomized trees. *Machine learning*, 63(1):3–42.

Giuliani, M. and Herman, J. D. (2018). Modeling the behavior of water reservoir operators via eigenbehavior analysis. *Advances in Water Resources*, 122:228–237.

Glover, F. (1989). Tabu searchpart i. *ORSA Journal on computing*, 1(3):190–206.

Glover, F. (1990). Tabu searchpart ii. *ORSA Journal on computing*, 2(1):4–32.

Goldberg, D. E. and Holland, J. H. (1988). Genetic algorithms and machine learning. *Machine learning*, 3(2):95–99.

Goyal, M. K., Ojha, C., Singh, R., Swamee, P., Nema, R., et al. (2013). Application of ann, fuzzy logic and decision tree algorithms for the development of reservoir operating rules. *Water resources management*, 27(3):911–925.

Haddad, O. B., Afshar, A., and Mariño, M. A. (2006). Honey-bees mating optimization (hbmo) algorithm: a new heuristic approach for water resources optimization. *water resources management*, 20(5):661–680.

Hadka, D. and Reed, P. (2013). Borg: An auto-adaptive many-objective evolutionary computing framework. *Evolutionary computation*, 21(2):231–259.

Hall, W. A. and Buras, N. (1961). The dynamic programming approach to water-resources development. *Journal of Geophysical Research*, 66(2):517–520.

Hall, W. A., Butcher, W. S., and Esogbue, A. (1968). Optimization of the operation of a multiple-purpose reservoir by dynamic programming. *Water Resources Research*, 4(3):471–477.

Han, J., Mao, K., Xu, T., Guo, J., Zuo, Z., and Gao, C. (2018). A soil moisture estimation framework based on the cart algorithm and its application in china. *Journal of Hydrology*.

Hasalová, L., Ira, J., and Jahoda, M. (2016). Practical observations on the use of shuffled complex evolution (sce) algorithm for kinetic parameters estimation in pyrolysis modeling. *Fire safety journal*, 80:71–82.

Hejazi, M. I., Cai, X., and Ruddell, B. L. (2008). The role of hydrologic information in reservoir operation–learning from historical releases. *Advances in water resources*, 31(12):1636–1650.

Hothorn, T., Hornik, K., and Zeileis, A. (2006). Unbiased recursive partitioning: A conditional inference framework. *Journal of Computational and Graphical statistics*, 15(3):651–674.

Hsu, K.-l., Gupta, H. V., Gao, X., Sorooshian, S., and Imam, B. (2002). Self-organizing linear output map (solo): An artificial neural network suitable for hydrologic modeling and analysis. *Water Resources Research*, 38(12).

Huang, C. and Townshend, J. (2003). A stepwise regression tree for nonlinear approximation: applications to estimating subpixel land cover. *International Journal of Remote Sensing*, 24(1):75–90.

Jekabsons, G. (2016). M5'regression tree, model tree, and tree ensemble toolbox for matlab/octave. *Octave ver*, 1(0).

John, H. (1992). Holland, adaptation in natural and artificial systems: An introductory analysis with applications to biology, control and artificial intelligence.

Jolliffe, I. T. (1982). A note on the use of principal components in regression. *Applied Statistics*, pages 300–303.

Jothiprakash, V. and Kote, A. S. (2011). Effect of pruning and smoothing while using m5 model tree technique for reservoir inflow prediction. *Journal of Hydrologic Engineering*, 16(7):563–574.

Jung, N.-C., Popescu, I., Kelderman, P., Solomatine, D. P., and Price, R. K. (2010). Application of model trees and other machine learning techniques for algal growth prediction in yongdam reservoir, republic of korea. *Journal of Hydroinformatics*, 12(3):262–274.

Kan, A. R. and Timmer, G. T. (1987). Stochastic global optimization methods part i: Clustering methods. *Mathematical programming*, 39(1):27–56.

Karalic, A. (1992). Linear regression in regression tree leaves. In *In Proceedings of ECAI-92*. Citeseer.

Kaveh, A. and Farhoudi, N. (2013). A new optimization method: dolphin echolocation. *Advances in Engineering Software*, 59:53–70.

Kaveh, A. and Talatahari, S. (2010). A novel heuristic optimization method: charged system search. *Acta Mechanica*, 213(3-4):267–289.

Kennedy, J. (2011). Particle swarm optimization. In *Encyclopedia of machine learning*, pages 760–766. Springer.

Kirkpatrick, S., Gelatt, C. D., and Vecchi, M. P. (1983). Optimization by simulated annealing. *science*, 220(4598):671–680.

Kollat, J. B. and Reed, P. M. (2006). Comparing state-of-the-art evolutionary multi-objective algorithms for long-term groundwater monitoring design. *Advances in Water Resources*, 29(6):792–807.

Kompare, B., Steinman, F., Cerar, U., and Dzeroski, S. (1997). Prediction of rainfall runoff from catchment by intelligent data analysis with machine learning tools within the artificial intelligence tools. *Acta hydrotechnica*, 16(17):79–94.

Krzywinski, M. and Altman, N. (2014). Points of significance: visualizing samples with box plots.

Labadie, J. W. (2004). Optimal operation of multireservoir systems: state-of-the-art review. *Journal of water resources planning and management*, 130(2):93–111.

Labadie, J. W. (2006). Modsim: decision support system for integrated river basin management.

Lee, K. S. and Geem, Z. W. (2005). A new meta-heuristic algorithm for continuous engineering optimization: harmony search theory and practice. *Computer methods in applied mechanics and engineering*, 194(36-38):3902–3933.

Li, Y., Zhou, J., Zhang, Y., Qin, H., and Liu, L. (2010). Novel multiobjective shuffled frog leaping algorithm with application to reservoir flood control operation. *Journal of Water Resources Planning and Management*, 136(2):217–226.

Liang, J.-J., Suganthan, P. N., and Deb, K. (2005). Novel composition test functions for numerical global optimization. In *Swarm Intelligence Symposium, 2005. SIS 2005. Proceedings 2005 IEEE*, pages 68–75. IEEE.

Lin, J.-Y., Cheng, C.-T., and Chau, K.-W. (2006). Using support vector machines for long-term discharge prediction. *Hydrological Sciences Journal*, 51(4):599–612.

Lin, P., Yang, Z.-L., Gochis, D. J., Yu, W., Maidment, D. R., Somos-Valenzuela, M. A., and David, C. H. (2018). Implementation of a vector-based river network routing scheme in the community wrf-hydro modeling framework for flood discharge simulation. *Environmental Modelling & Software*, 107:1–11.

Liong, S.-Y. and Atiquzzaman, M. (2004). Optimal design of water distribution network using shuffled complex evolution. *Journal of The Institution of Engineers, Singapore*, 44(1):93–107.

Loh, W.-Y. (2002). Regression tress with unbiased variable selection and interaction detection. *Statistica Sinica*, pages 361–386.

Loh, W.-Y. (2014). Fifty years of classification and regression trees. *International Statistical Review*, 82(3):329–348.

Loh, W.-Y., He, X., and Man, M. (2015). A regression tree approach to identifying subgroups with differential treatment effects. *Statistics in medicine*, 34(11):1818–1833.

Loucks, D. P. and Dorfman, P. J. (1975). An evaluation of some linear decision rules in chance-constrained models for reservoir planning and operation. *Water Resources Research*, 11(6):777–782.

Maass, A., Hufschmidt, M. M., Dorfman, R., Thomas, H. A., Marglin, S. A., Fair, G. M., Bower, B. T., Reedy, W. W., Manzer, D. F., Barnett, M. P., et al. (1962). Design of water-resource systems.

Madsen, H. (2000). Automatic calibration of a conceptual rainfall–runoff model using multiple objectives. *Journal of hydrology*, 235(3-4):276–288.

Madsen, H. (2003). Parameter estimation in distributed hydrological catchment modelling using automatic calibration with multiple objectives. *Advances in water resources*, 26(2):205–216.

Maier, H. R., Kapelan, Z., Kasprzyk, J., Kollat, J., Matott, L. S., Cunha, M. C., Dandy, G. C., Gibbs, M. S., Keedwell, E., Marchi, A., et al. (2014). Evolutionary algorithms and other metaheuristics in water resources: Current status, research challenges and future directions. *Environmental Modelling & Software*, 62:271–299.

Malerba, D., Esposito, F., Ceci, M., and Appice, A. (2004). Top-down induction of model trees with regression and splitting nodes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(5):612–625.

Mallakpour, I., AghaKouchak, A., and Sadegh, M. (2019). Climate-induced changes in the risk of hydrological failure of major dams in california. *Geophysical Research Letters*.

Mariani, V. C., Luvizotto, L. G. J., Guerra, F. A., and dos Santos Coelho, L. (2011). A hybrid shuffled complex evolution approach based on differential evolution for unconstrained optimization. *Applied Mathematics and Computation*, 217(12):5822–5829.

Marler, R. T. and Arora, J. S. (2004). Survey of multi-objective optimization methods for engineering. *Structural and multidisciplinary optimization*, 26(6):369–395.

Mays, L. W. (1989). Hydrosystems engineering simulation vs. optimization: why not both.

Mehran, A., Mazdiyasni, O., and AghaKouchak, A. (2015). A hybrid framework for assessing socioeconomic drought: Linking climate variability, local resilience, and demand. *Journal of Geophysical Research: Atmospheres*, 120(15):7520–7533.

Meier Jr, W. and Beightler, C. (1967). An optimization method for branching multistage water resource systems. *Water Resources Research*, 3(3):645–652.

Mirjalili, S. and Hashim, S. Z. M. (2010). A new hybrid psogsa algorithm for function optimization. In *Computer and information application (ICCIA), 2010 international conference on*, pages 374–377. IEEE.

Mirjalili, S., Mirjalili, S. M., and Lewis, A. (2014). Grey wolf optimizer. *Advances in engineering software*, 69:46–61.

Mirjalili, S., Saremi, S., Mirjalili, S. M., and Coelho, L. d. S. (2016). Multi-objective grey wolf optimizer: a novel algorithm for multi-criterion optimization. *Expert Systems with Applications*, 47:106–119.

Moeini, R. and Afshar, M. (2009). Application of an ant colony optimization algorithm for optimal operation of reservoirs: a comparative study of three proposed formulations.

Murthy, S. K. and Salzberg, S. (1995). Decision tree induction: How effective is the greedy heuristic? In *KDD*, pages 222–227.

Naeini, M. R., Yang, T., Sadegh, M., AghaKouchak, A., Hsu, K.-l., Sorooshian, S., Duan, Q., and Lei, X. (2018). Shuffled complex-self adaptive hybrid evolution (sc-sahel) optimization framework. *Environmental Modelling & Software*, 104:215–235.

Nasta, P., Vrugt, J. A., and Romano, N. (2013). Prediction of the saturated hydraulic conductivity from brooks and corey's water retention parameters. *Water Resources Research*, 49(5):2918–2925.

Needham, J. T., Watkins Jr, D. W., Lund, J. R., and Nanda, S. (2000). Linear programming for flood control in the iowa and des moines rivers. *Journal of Water Resources Planning and Management*, 126(3):118–127.

Neelakantan, T. and Pundarikanthan, N. (1999). Hedging rule optimisation for water supply reservoirs system. *Water resources management*, 13(6):409–426.

Nelder, J. A. and Mead, R. (1965). A simplex method for function minimization. *The computer journal*, 7(4):308–313.

Nicklow, J., Reed, P., Savic, D., Dessalegne, T., Harrell, L., Chan-Hilton, A., Karamouz, M., Minsker, B., Ostfeld, A., Singh, A., et al. (2009). State of the art for genetic algorithms and beyond in water resources planning and management. *Journal of Water Resources Planning and Management*, 136(4):412–432.

Oliveira, R. and Loucks, D. P. (1997). Operating rules for multireservoir systems. *Water resources research*, 33(4):839–852.

Olorunda, O. and Engelbrecht, A. P. (2008). Measuring exploration/exploitation in particle swarms using swarm diversity. In *Evolutionary Computation, 2008. CEC 2008.(IEEE World Congress on Computational Intelligence). IEEE Congress on*, pages 1128–1134. IEEE.

Pal, M. and Mather, P. M. (2003). An assessment of the effectiveness of decision tree methods for land cover classification. *Remote sensing of environment*, 86(4):554–565.

Passino, K. M. (2002). Biomimicry of bacterial foraging for distributed optimization and control. *IEEE control systems*, 22(3):52–67.

Price, W. (1987). Global optimization algorithms for a cad workstation. *Journal of Optimization Theory and Applications*, 55(1):133–146.

Qin, A. K. and Suganthan, P. N. (2005). Self-adaptive differential evolution algorithm for numerical optimization. In *Evolutionary Computation, 2005. The 2005 IEEE Congress on*, volume 2, pages 1785–1791. IEEE.

Quinlan, J. R. (2014). *C4. 5: programs for machine learning*. Elsevier.

Quinlan, J. R. et al. (1992). Learning with continuous classes. In *5th Australian joint conference on artificial intelligence*, volume 92, pages 343–348. Singapore.

Rani, D. and Moreira, M. M. (2010). Simulation–optimization modeling: a survey and potential application in reservoir systems operation. *Water resources management*, 24(6):1107–1138.

Rashedi, E., Nezamabadi-Pour, H., and Saryazdi, S. (2009). Gsa: a gravitational search algorithm. *Information sciences*, 179(13):2232–2248.

Reed, P. M., Hadka, D., Herman, J. D., Kasprzyk, J. R., and Kollat, J. B. (2013). Evolutionary multiobjective optimization in water resources: The past, present, and future. *Advances in water resources*, 51:438–456.

Revelle, C., Joeres, E., and Kirby, W. (1969). The linear decision rule in reservoir management and design: 1, development of the stochastic model. *Water Resources Research*, 5(4):767–777.

Roeva, O., Slavov, T., and Fidanova, S. (2014). Population-based vs. single point search meta-heuristics for a pid controller tuning. In *Handbook of Research on Novel Soft Computing Intelligent Algorithms: Theory and Practical Applications*, pages 200–233. IGI Global.

Sadegh, M. (2015). *Towards Improved Model Evaluation: Diagnostic, Likelihood-free, Bayesian Inference*. PhD thesis, UC Irvine.

Sadegh, M., Ragno, E., and AghaKouchak, A. (2017). Multivariate c opula a nalysis t oolbox (mvcat): Describing dependence and underlying uncertainty using a b ayesian framework. *Water Resources Research*, 53(6):5166–5183.

Sadegh, M. and Vrugt, J. A. (2014). Approximate bayesian computation using markov chain monte carlo simulation: Dream (abc). *Water Resources Research*, 50(8):6767–6787.

Salas, F. R., Somos-Valenzuela, M. A., Dugger, A., Maidment, D. R., Gochis, D. J., David, C. H., Yu, W., Ding, D., Clark, E. P., and Noman, N. (2018). Towards real-time continental scale streamflow simulation in continuous and discrete space. *JAWRA Journal of the American Water Resources Association*, 54(1):7–27.

Schwarz, G. et al. (1978). Estimating the dimension of a model. *The annals of statistics*, 6(2):461–464.

Shih, Y.-S. (2004). A note on split selection bias in classification trees. *Computational statistics & data analysis*, 45(3):457–466.

Shlens, J. (2014). A tutorial on principal component analysis. *arXiv preprint arXiv:1404.1100*.

Simonovic, S. P. (1992). Reservoir systems analysis: closing gap between theory and practice. *Journal of water resources planning and management*, 118(3):262–280.

Snow, A. D., Christensen, S. D., Swain, N. R., Nelson, E. J., Ames, D. P., Jones, N. L., Ding, D., Noman, N. S., David, C. H., Pappenberger, F., et al. (2016). A high-resolution national-scale hydrologic forecast system from a global ensemble land surface model. *JAWRA Journal of the American Water Resources Association*, 52(4):950–964.

Solomatine, D. P. and Dulal, K. N. (2003). Model trees as an alternative to neural networks in rainfallrunoff modelling. *Hydrological Sciences Journal*, 48(3):399–411.

Sorooshian, S., Duan, Q., and Gupta, V. K. (1993). Calibration of rainfall-runoff models: Application of global optimization to the sacramento soil moisture accounting model. *Water resources research*, 29(4):1185–1194.

Storn, R. and Price, K. (1997). Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces. *Journal of global optimization*, 11(4):341–359.

Štravs, L. and Brilly, M. (2007). Development of a low-flow forecasting model using the m5 machine learning method. *Hydrological sciences journal*, 52(3):466–477.

Tavakoly, A., David, C., Maidment, D., Yang, Z., and Cai, X. (2012). An upscaling process for large-scale vector-based river networks using the nhdplus dataset. In *Proceedings of the American Water Resources Association Spring Specialty Conference on Geographic Information Systems and Water Resources*.

Tavakoly, A. A., Snow, A. D., David, C. H., Follum, M. L., Maidment, D. R., and Yang, Z.-L. (2017). Continental-scale river flow modeling of the mississippi river basin using high-resolution nhdplus dataset. *JAWRA Journal of the American Water Resources Association*, 53(2):258–279.

Toth, E., Brath, A., and Montanari, A. (2000). Comparison of short-term rainfall prediction models for real-time flood forecasting. *Journal of hydrology*, 239(1-4):132–147.

USACE (2013). Hec-ressim reservoir system simulation, version 3.1: Users manual.

Vahedifard, F., AghaKouchak, A., Ragno, E., Shahrokhabadi, S., and Mallakpour, I. (2017). Lessons from the oroville dam. *Science*, 355(6330):1139–1140.

Vrugt, J. A., Gupta, H. V., Bastidas, L. A., Bouten, W., and Sorooshian, S. (2003a). Effective and efficient algorithm for multiobjective optimization of hydrologic models. *Water Resources Research*, 39(8).

Vrugt, J. A., Gupta, H. V., Bouten, W., and Sorooshian, S. (2003b). A shuffled complex evolution metropolis algorithm for optimization and uncertainty assessment of hydrologic model parameters. *Water resources research*, 39(8).

Vrugt, J. A. and Robinson, B. A. (2007). Improved evolutionary optimization from genetically adaptive multimethod search. *Proceedings of the National Academy of Sciences*, 104(3):708–711.

Vrugt, J. A., Robinson, B. A., and Hyman, J. M. (2009). Self-adaptive multimethod search for global optimization in real-parameter spaces. *IEEE Transactions on Evolutionary Computation*, 13(2):243–259.

Wagener, T., Wheater, H., and Gupta, H. V. (2004). *Rainfall-runoff modelling in gauged and ungauged catchments*. World Scientific.

Wang, Y. and Witten, I. H. (1997). Inducing model trees for continuous classes. In *Proceedings of the Ninth European Conference on Machine Learning*, pages 128–137.

Wang, Y., Yoshitani, J., and Fukami, K. (2005). Stochastic multiobjective optimization of reservoirs in parallel. *Hydrological Processes: An International Journal*, 19(18):3551–3567.

Wang, Y.-C., Yu, P.-S., and Yang, T.-C. (2010). Comparison of genetic algorithms and shuffled complex evolution approach for calibrating distributed rainfall–runoff model. *Hydrological Processes: An International Journal*, 24(8):1015–1026.

Witten, I. H., Frank, E., Hall, M. A., and Pal, C. J. (2016). *Data Mining: Practical machine learning tools and techniques*. Morgan Kaufmann.

Wolpert, D. H. and Macready, W. G. (1997). No free lunch theorems for optimization. *IEEE transactions on evolutionary computation*, 1(1):67–82.

Woodruff, M. J., Reed, P. M., and Simpson, T. W. (2013). Many objective visual analytics: rethinking the design of complex engineered systems. *Structural and Multidisciplinary Optimization*, 48(1):201–219.

Yakowitz, S. (1982). Dynamic programming applications in water resources. *Water resources research*, 18(4):673–696.

Yamazaki, D., Almeida, G. A., and Bates, P. D. (2013). Improving computational efficiency in global river models by implementing the local inertial flow equation and a vector-based river network map. *Water Resources Research*, 49(11):7221–7235.

Yang, T., Asanjan, A. A., Faridzad, M., Hayatbini, N., Gao, X., and Sorooshian, S. (2017a). An enhanced artificial neural network with a shuffled complex evolutionary global optimization with principal component analysis. *Information Sciences*, 418:302–316.

Yang, T., Asanjan, A. A., Welles, E., Gao, X., Sorooshian, S., and Liu, X. (2017b). Developing reservoir monthly inflow forecasts using artificial intelligence and climate phenomenon information. *Water Resources Research*, 53(4):2786–2812.

Yang, T., Gao, X., Sellars, S. L., and Sorooshian, S. (2015). Improving the multi-objective evolutionary optimization algorithm for hydropower reservoir operations in the california oroville–thermalito complex. *Environmental Modelling & Software*, 69:262–279.

Yang, T., Gao, X., Sorooshian, S., and Li, X. (2016). Simulating california reservoir operation using the classification and regression-tree algorithm combined with a shuffled cross-validation scheme. *Water Resources Research*, 52(3):1626–1651.

Yang, X.-S. (2009). Firefly algorithms for multimodal optimization. In *International symposium on stochastic algorithms*, pages 169–178. Springer.

Yang, X.-S. (2010). A new metaheuristic bat-inspired algorithm. In *Nature inspired cooperative strategies for optimization (NICSO 2010)*, pages 65–74. Springer.

Yao, H. and Georgakakos, A. (2001). Assessment of folsom lake response to historical and potential future climate scenarios: 2. reservoir management. *Journal of Hydrology*, 249(1-4):176–196.

Yao, X., Liu, Y., and Lin, G. (1999). Evolutionary programming made faster. *IEEE Transactions on Evolutionary computation*, 3(2):82–102.

Yapo, P. O., Gupta, H. V., and Sorooshian, S. (1996). Automatic calibration of conceptual rainfall-runoff models: sensitivity to calibration data. *Journal of Hydrology*, 181(1-4):23–48.

Yapo, P. O., Gupta, H. V., and Sorooshian, S. (1998). Multi-objective global optimization for hydrologic models. *Journal of hydrology*, 204(1-4):83–97.

Yeh, W. W.-G. (1985). Reservoir management and operations models: A state-of-the-art review. *Water resources research*, 21(12):1797–1818.

Yeh, W. W.-G. and Becker, L. (1982). Multiobjective analysis of multireservoir operations. *Water resources research*, 18(5):1326–1336.

Zagona, E. A., Fulp, T. J., Shane, R., Magee, T., and Goranflo, H. M. (2001). Riverware: A generalized tool for complex reservoir system modeling 1. *JAWRA Journal of the American Water Resources Association*, 37(4):913–929.

Zhang, D., Lin, J., Peng, Q., Wang, D., Yang, T., Sorooshian, S., Liu, X., and Zhuang, J. (2018). Modeling and simulating of reservoir operation using the artificial neural network, support vector regression, deep learning algorithm. *Journal of Hydrology*.

Zhang, Y. and Yang, Y. (2015). Cross-validation for selecting a model selection procedure. *Journal of Econometrics*, 187(1):95–112.

Zhu, T., Lund, J. R., Jenkins, M. W., Marques, G. F., and Ritzema, R. S. (2007). Climate change, urbanization, and optimal long-term floodplain protection. *Water Resources Research*, 43(6).

Appendices

Appendix A

MCCE algorithm pseudo code is as follows:

0. Initialize $i = 1$, and get maximum number of iteration allowed, I.

1. Sort individuals in order of increasing objective function value. Assign individuals a triangular probability (except for the fittest point) according to:

$$p = \frac{2(\mathrm{NPS} + 1 - n)}{\mathrm{NPS}(\mathrm{NPS} + 1)} \tag{A.1}$$

   where NPS is the number of individuals in the complex and is the rank of the sorted individuals.

2. Select $d+1$ individuals (d is problem dimension) from the complex including the fittest individual in the complex.

3. The selected individuals are then stored in **S**, forming a simplex. Generate offspring according to following steps.

   (a) Sort individuals in **S** according to their objective function value. Find centroid, $\vec{c}$, of the first d individuals.

(b) Reflection: Reflect the worst individual in $\mathbf{S}$, $\vec{w}$, across the centroid to generate a new point, $\vec{r}$, according to following equation:

$$\vec{r} = 2\vec{c} - \vec{w}. \tag{A.2}$$

Evaluate objective function for the new point, f_r. If $f_1 < f_r < f_d$ set offspring, $\vec{o} = \vec{r}$, and go to (g).

(c) Expansion: If $f_r < f_1$, reflect $\vec{c}$ across $\vec{r}$ and generate $\vec{e}$,

$$\vec{e} = 2\vec{r} - \vec{c}. \tag{A.3}$$

Evaluate objective function for the new point, f_e. If $f_e < f_r$, set $\vec{o} = \vec{e}$ and go to (VII); otherwise, and go to (g).

(d) Outside contraction: If $f_d \leq f_r < f_w$, calculate the outside contraction point,

$$\vec{oc} = \vec{c} + 0.5(\vec{r} - \vec{c}). \tag{A.4}$$

Evaluate the outside contraction point, f_{oc}. If $f_{oc} < f_r$ set $\vec{o} = \vec{oc}$ and go to (g); otherwise, $\vec{o} = \vec{r}$ and go to (g).

(e) Inside contraction: If $f_w < f_r$ calculate inside contraction point,

$$\vec{ic} = \vec{c} + 0.5(\vec{w} - \vec{c}). \tag{A.5}$$

Evaluate inside contraction point, f_{ic}. If $f_{ic} < f_r$ set $\vec{o} = \vec{ic}$ and go to (g); otherwise continue to (f).

(f) Multinormal sampling: If the steps above, did not generate a better offspring, an individual will be drawn with a multinormal distribution defined by simplex and replace the worst individual in the simplex, regardless of objective function value.

The multinormal sampling is as follow,

i. Calculate the covariance matrix, $\mathbf{R}$, for the simplex and store diagonal of matrix in $\vec{D}$.

ii. Modify $\vec{D}$ as follow,

$$\vec{D_m} = 2(\vec{D} + mean(\vec{D})). \tag{A.6}$$

iii. Generate a new covariance matrix $\mathbf{R}'$, with $\vec{D_m}$ as diagonal and zeroes everywhere else.

iv. Sample a point with multinormal distribution with mean of $\vec{c}$ and covariance of $\mathbf{R}'$ and store in $\vec{o}$.

(g) Replace the worst individual in the complex with $\vec{o}$. Let $i = i + 1$ If $i \leq I$, go to (Step 1); otherwise sort the points in the complex and return the evolved complex.

Appendix B

Modified FL (MFL) algorithm is as follows:

0. Initialize $i = 1$, and get maximum number of iteration allowed, I.

1. Sort individuals in order of increasing objective function value. Assign individuals a triangular probability (except for the fittest point) according to:

$$p = \frac{2(\mathrm{NPS} + 1 - n)}{\mathrm{NPS}(\mathrm{NPS} + 1)} \tag{A.7}$$

where NPS is the number of individuals in the complex and is the rank of the sorted individuals.

2. Select $d+1$ individuals (d is problem dimension) from the complex including the fittest individual in the complex.

3. The selected individuals are then stored in **S**, forming a simplex. Generate offspring according to following steps.

   (a) Generate a new point with the worst point in **S**, $\vec{w}$ and best point $\vec{b}$ in the subcomplex, as follows,

   $$\vec{n_b} = \vec{w} + (0.5R + 1.5)(\vec{b} - \vec{w}) \tag{A.8}$$

   where R is a random number in the range of [0,1]. Evaluate objective function for the new point and get f_b. If $f_b < f_w$; set $\vec{o} = \vec{n_b}$ and go to (d).

   (b) If $f_w < f_b$, generate a new point with the worst point in **S**, $\vec{w}$ and best point $\vec{b}$ in the subcomplex, as follows,

   $$\vec{n_B} = 0.5R(\vec{b} - \vec{w}) \tag{A.9}$$

   Evaluate objective function for the new point and get f_B. If $f_B < f_w$ set the offspring set $\vec{o} = \vec{n_B}$ and go to (d).

   (c) Censorship step: If $f_w < f_B$, randomly generate the offspring, $\vec{o}$ by sampling within the range of individuals in the subcomplex.

   (d) Replace the worst individual in the complex with the offspring, $\vec{o}$. Let $i = i + 1$. If $i \leq I$, go to (Step 1); otherwise sort the points in the complex and return the evolved complex.

Appendix C

Modified Grey Wolf Optimizer is as follows:

0. Initialize $i = 1$, and get maximum number of iteration allowed, I.

1. Sort individuals in order of increasing objective function value. Assign individuals a triangular probability (except for the fittest point) according to:

$$p = \frac{2(\mathrm{NPS} + 1 - n)}{\mathrm{NPS}(\mathrm{NPS} + 1)} \tag{A.10}$$

   where NPS is the number of individuals in the complex and is the rank of the sorted individuals.

2. Select $d + 1$ individuals (d is problem dimension) from the complex, with triangular probability, including the fittest point in the complex and store them in **S**.

3. Select the best three points in the **S** and store them in $\vec{\alpha}$, $\vec{\beta}$ and $\vec{\gamma}$, respectively. The worst point in the S, is stored in $\vec{w}$.

4. For each of $\vec{alpha}$, $\vec{beta}$ and $\vec{gamma}$, evolve individuals according to the following procedure,

   (a) Derive $\vec{A}$ and $\vec{C}$ as follow for $\vec{\alpha}$, $\vec{\beta}$ and $\vec{\gamma}$,

$$\vec{A} = 4 \times \vec{r_1} - 2, \tag{A.11}$$

$$\vec{C} = 2 \times \vec{r_2}, \tag{A.12}$$

where r_1, r_2 are two independent random vectors, which have d dimensions and values in range of (0,1).

(b) Derive $\vec{D}$, for $\vec{\alpha}$, $\vec{\beta}$ and $\vec{\gamma}$ as follows,

$$\vec{D_\alpha} = |\vec{C_\alpha} \times \vec{X_\alpha} - \vec{w}|, \vec{D_\beta} = |\vec{C_\beta} \times \vec{X_\beta} - \vec{w}|, \vec{D_\gamma} = |\vec{C_\gamma} \times \vec{X_\gamma} - \vec{w}|. \tag{A.13}$$

(c) Derive $\vec{Z}$, for $\vec{\alpha}$, $\vec{\beta}$ and $\vec{\gamma}$ as follow,

$$\vec{Z_\alpha} = \vec{X_\alpha} - \vec{A_\alpha}.\vec{D_\alpha}, \vec{Z_\beta} = \vec{X_\beta} - \vec{A_\beta}.\vec{D_\beta}, \vec{Z_\gamma} = \vec{X_\gamma} - \vec{A_\gamma}.\vec{D_\gamma}, \tag{A.14}$$

(d) Generate a new point by finding the centroid of $\vec{Z_\alpha}$, $\vec{Z_\beta}$ and $\vec{Z_\gamma}$,

$$\vec{C} = \frac{\vec{Z_\alpha} + \vec{Z_\beta} + \vec{Z_\gamma}}{3}. \tag{A.15}$$

(e) Calculate and store objective function value for the new point, f_C. If the new point is better than the worst point among the selected points, $f_C < f_w$, set $\vec{o} = \vec{C}$, go to step 7.

5. If $f_C > f_w$, go to step 4, and use a smaller range for $\vec{A}$. In this step, $\vec{A}$ is calculated as follows:

$$\vec{A} = 2 \times \vec{r_1} - 1, \tag{A.16}$$

6. If the newly generated individual is worse than the worst individuals in subcomplex, generate a new point with uniform random sampling within the range of individuals in the complex. Store the new point in $\vec{o}$.

7. Replace the worst individual among selected points in the complex with the offspring, $\vec{o}$. Let $i = i + 1$. If $i < I$, go to (Step 1); otherwise sort the points in the complex and

return the evolved complex.

Appendix D

Modified differential evolution algorithm is as follows:

0. Initialize $i = 1$, and get maximum number of iteration allowed, I.

1. Sort individuals in order of increasing objective function value. Assign individuals a triangular probability (except for the fittest point) according to:

$$p = \frac{2(\mathrm{NPS} + 1 - n)}{\mathrm{NPS}(\mathrm{NPS} + 1)} \tag{A.17}$$

   where NPS is the number of individuals in the complex and is the rank of the sorted individuals.

2. Select $d + 1$ individuals (d is problem dimension) from the complex, with triangular probability, including the fittest point in the complex and store them in **S**.

3. The selected individuals are sorted and stored in **S**, forming a subcomplex. Generate offspring according to following steps.

   (a) Generate a new point with the worst point in **S**, $\vec{w}$ and using the top three individuals in the subcomplex,

$$\vec{V} = \vec{w} + 2f(\vec{s_1} - \vec{w}) + 2f(\vec{s_2} - \vec{s_3}), \tag{A.18}$$

   where $\vec{w}$ is the worst point in the **S**, $\vec{s_1}$, $\vec{s_2}$, and $\vec{s_3}$ are three selected individuals. Then mutation, and crossover operator is applied to $\vec{w}$ and $\vec{V_1}$ to generate $\vec{V_{n1}}$.

The objective function value for the new point is calculated and stored in f_{n1}. If $f_{n1} < f_w$, set $\vec{o} = \vec{V_{n1}}$ and go to (e).

(b) If $f_w < f_{n1}$, generate a new point with the worst point in $\mathbf{S}$, $\vec{w}$ and using the top three points in the subcomplex as follow,

$$\vec{V_2} = \vec{w} + 0.5f(\vec{s_1} - \vec{w}) + 0.5f(\vec{s_2} - \vec{s_3}), \tag{A.19}$$

After mutation, crossover operator is applied to the $\vec{w}$ and $\vec{V_2}$ to generate $\vec{V_{n2}}$. Then, the objective function for the new point is derived and stored in f_{n2}. If $f_{n2} < f_w$; set $\vec{o} = \vec{V_{n2}}$ and go to (e).

(c) If $f_w < f_{n2}$, generate a new point with the worst point in $\mathbf{S}$, $\vec{w}$ and using the top three points in the subcomplex as follow,

$$\vec{V_3} = \vec{w} + f(\vec{s_1} - \vec{w}) + f(\vec{s_2} - \vec{s_3}), \tag{A.20}$$

After mutation, crossover operator is applied to the $\vec{w}$ and $\vec{V_3}$ to generate $\vec{V_{n3}}$. The objective function value is calculated and stored in f_{n3}. If $f_{n3} < f_w$; set $\vec{o} = \vec{V_{n3}}$ and go to (e).

(d) If the newly generated point is worse than the worst point in subcomplex, generate a new point from uniform random distribution within the range of points in the complex. Store the new point in $\vec{o}$.

(e) Replace the worst point in the complex with the offspring, $\vec{o}$. Let $i = i + 1$. If $i \leq I$, go to (Step 1); otherwise sort the points in the complex and return the evolved complex.