APPLICATION OF GOAL PROGRAMMING APPROACH IN HYDROECONOMIC OPTIMIZATION: A STUDY CASE OF THE CITARUM CASCADE RESERVOIRS OPERATION RULES

by
TIARAVANNI HERMAWAN

Master thesis report submitted in
fulfilment of the requirements for the MSc in
Hydroinformatics and Water Management of
The Joint Master Course by EuroAquae consortium

University of Nice-Sophia Antipolis – Brandenburg University of Technology Cottbus-Seftenberg – Newcastle University – Technical University of Catalonia – Budapest University of Technology and Economics

DELFT, AUGUST 2016

Academic tutor
Dr. Frank Molkenthin
Brandenburg University of
Technology Cottbus-Seftenberg

Institutional tutors
Prof Ir Eelco van Beek
Dr. Peter Gijsbers
Jorn Baayen
Stichting Deltares

AUTHOR'S DECLARATION

I hereby declare that this work was written independently without any unauthorized assistance or sources not given credit within the work. All words, phrases, passages, and data taken from other sources have been properly cited. No parts of this work in the same form or a similar form have ever been previously handed in to fulfil an examination.

[illegible]

SIGNED: Tiaravanni Hermawan DATE: Delft, 15.08.2016

EXECUTIVE SUMMARY

In recent years, the hydroeconomic approach to solve multi-objective problems in integrated river basin management receives growing attention. By combining the principles of economics and engineering, the hydroeconomic models transform the concept of fixed demand into the economic value of water defined through water rights. Unfortunately, the management schemes and the policy insight are less likely to be easily represented by a hydroeconomic objective function. In Deltares, the necessity to explicitly implementing priority ordered by the policy on water resources allocation to a conventional hydroeconomic model has been done by combining the particle SWARM (PS) optimization with the rule-based simulation tools (RIBASIM).

The main objective of the study is to compare the application of RTC-Tools 2.0 with the RIBASIM-PS approach. The new generation of RTC-Tools 2.0 is currently being developed by Deltares for real-time control and optimization of hydraulic systems. This study focuses on the reservoir operation strategies to determine the most promising water allocation under similar attainment targets by constructing various hydroeconomic optimization models for a study case in the RTC–Tools 2.0.

Similarly to RIBASIM-PS study, the case study of the Citarum cascade reservoirs in Indonesia was adopted to provide rounded, detailed illustrations of the policy-based-management in water resources. This study aims to provide useful insight to support the decision-making in reservoir operation including the trade-offs between the conflicting objectives whereas the current reservoir operation rule is derived by optimizing of a single objective. The improvement on the reservoir operation rules is expected to enhance the social and economic benefit of the reservoirs in the basin.

This study has been able to demonstrate, for the first time, the possibility to develop a similar model network as RIBASIM and to optimize a similar hydroeconomic objective as particle SWARM in the RTC-Tools 2.0. The principal theoretical implication of this study is that it is possible to transform the algorithm of the simulation model (RIBASIM) into the explicit hydrological sequences of objectives in the optimization model in RTC-Tools 2.0. The finding in this study, while preliminary, suggests that the optimization results from RTC-Tools 2.0 present more promising reservoir operation rules in comparison with the result from RIBASIM simulation and the RIBASIM-PS study.

The results from this study indicate that finding an appropriate approach and properly formulating the optimization problem are crucial steps in order to derive the most promising optimization results. The goal programming approach seems to provide a robust, easy-to-build and communicative method to achieve a transparent water allocation based on the policy insight. In this approach, specific numeric goals are derived based on the priorities to set the sequences of objective functions.

One of the more significant finding to emerge from this study is that constructing a hybrid hydroeconomic optimization model is a preferable approach to address the multi-objective problems in RTC-Tools 2.0. The goal programming approach assists the optimization algorithm in satisfying the sequences of objectives while it simultaneously searches for the highest economic benefit. The results from this optimization present the most promising reservoir operation rules that substantially enhance the social benefit with a slight reduction in economic benefit for the study case of Citarum cascade reservoirs.

Key words: integrated river basin management, multi-objective reservoir operation, hydroeconomic optimization, goal programming

ACKNOWLEDGEMENT

This master thesis was supported by Stichting Deltares, an independent institute for applied research in the field of water, which is one of institutional partners of The Joint Master Course by EuroAquae consortium. I would like to express my sincere gratitude to my daily institutional supervisor, Dr. Peter Gijsbers, for the continuous support, his immense knowledge and the thorough supervision on the contents in this research. I would also like to acknowledge the crucial role of Prof. Eelco van Beek, who provided me an opportunity to join their team as an intern. His insight and expertise have greatly assisted this research. Then, I would like to express my appreciation to Jorn Baayen, as the product manager of the innovative software used in this study, who has provided me a substantial help on the detailed, technical problem of this research.

Furthermore, I would like to express my gratitude towards Dr. Frank Molkenthin as the academic supervisor, who helped me continuously to coordinate my master thesis throughout the process of researching and report writing, including for his time spent on visiting Deltares. My thanks and appreciations also go to my colleagues, Laura Basco Carrera and Lőrinc Mészáros, who help me to markedly improve my academic writing skills, also to Tarasinta Perwitasari and Jesse van der Wees, for their time spent on discussing the script language in this study.

Finally, I would like to dedicate this thesis to my parents. Their accomplishments and independence in life inspire me to be a devoted person. They always motivate me to finish my master study the best I could.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

1. INTRODUCTION

Reservoir operation in many river basins have to deal with conflicting goals from downstream water users, hydropower and flood control. Finding the most promising operation is a multi-objective problem. The inter-linkages between these water users need to be addressed so that all water users reach sustainable benefits. Improved water use efficiency could be accomplished by implementing a set of allocation measures that might include priority setting and subsidies. To understand these complexities better, many conceptual modelling tools in integrated water resources management are being used. Several new modelling tools are becoming available, some of them with innovative methods and distinctive advantages.

1.1 Problem Description

At this moment, Deltares is applying the RIBASIM software simulation package to analyse water scarcity and water allocation problems, including reservoir operation. By simulating alternative strategies, the most suitable solution is identified manually. This is done in a kind of trial-and-error way which is time-consuming. Van der Vat [1] has applied a particle SWARM optimization technique in combination with RIBASIM that is able to find the optimal reservoir operating strategy directly. In that approach, the parameters of the operation strategy are computerized and optimized based on costs and benefits function. From this point, this study of Van der Vat [1] will be referred as the RIBASIM-PS study.

Similar to the RIBASIM-PS study, reservoir optimization might be also possible with the new generation of RTC-Tools 2.0. For water system control problems, Deltares is developing the RTC-Tools 2.0 package as a simulation and an optimization tool. The RTC-Tools is widely applied to real time control situations to optimize the operation of pumps and reservoirs. RTC-Tools 2.0 is and upgrade of the present RTC-Tools, providing more facilities and finding an optimum solution faster.

The objective of the study described in the document is to compare the application of RTC-Tools 2.0 with the RIBASIM-PS approach. This study mainly focuses on the reservoir operation strategies to determine the most promising water allocation under similar targets.

1.2 Objective of the study and research questions

The objective of this study is to look into the possibility of applying RTC-Tools 2.0 to determine the reservoir operation rules in comparison with the RIBASIM-PS study. To achieve this objective, the following research questions are formulated:

1. *Is RTC-Tools 2.0 able to model a similar network as RIBASIM does, using allocation rules based on demand priority and reservoir operation rules including hedging?*
2. *Is it possible to formulate a set of objectives and constraints in the RTC-Tools 2.0 that will result in optimized reservoir operating rules?*
3. *Are the calculations of optimal reservoir operation rules by RTC-Tools 2.0 different from the operation rules resulting from the RIBASIM-PS optimization and, if so, why?*
4. *How can we further improve the results of the optimization approaches in order to get a better applicability of the reservoir operation rules?*

1.3 Overview of the study

To answer the research questions, this research is addressing the following three key aspects; (i) the way of water allocation is being done in the network model, (ii) the practicalities of the optimization procedure and (iii) improvement of the RIBASIM-PS study. Figure 1.1 describes briefly the overview of this research while each component is further explained below this figure.

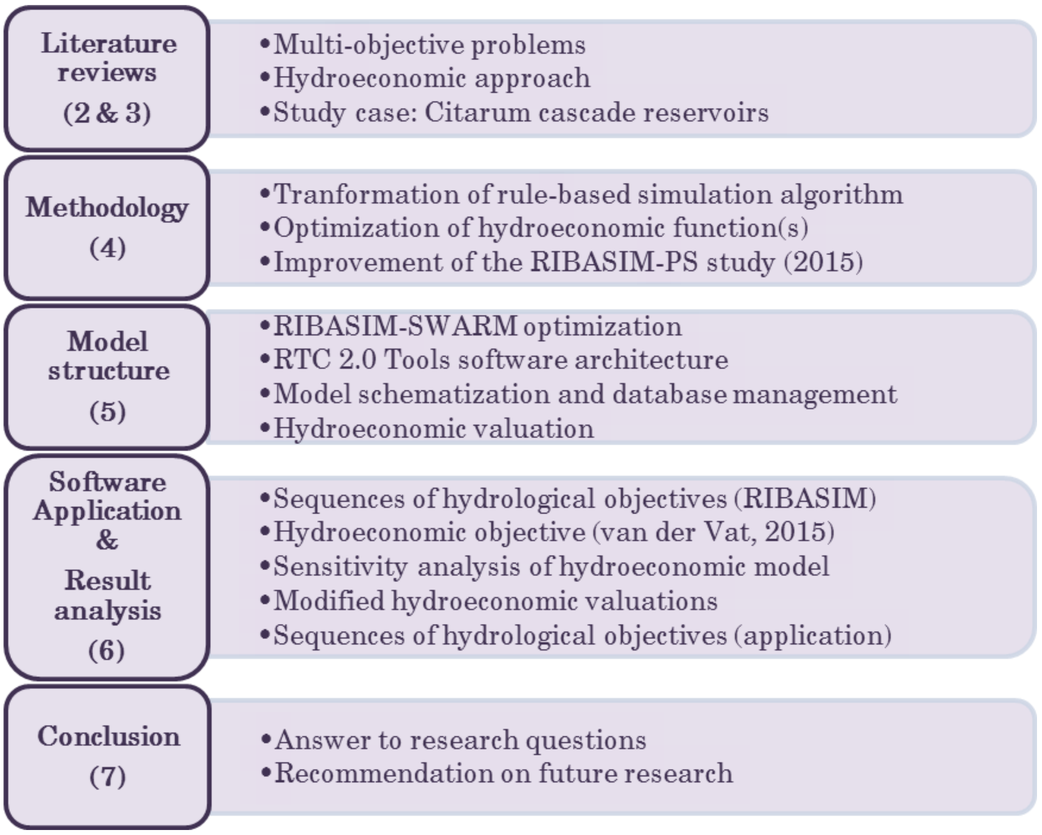

Literature reviews (2 & 3)
•Multi-objective problems
•Hydroeconomic approach
•Study case: Citarum cascade reservoirs
Methodology (4)
•Tranformation of rule-based simulation algorithm
•Optimization of hydroeconomic function(s)
•Improvement of the RIBASIM-PS study (2015)
Model structure (5)
•RIBASIM-SWARM optimization
•RTC 2.0 Tools software architecture
•Model schematization and database management
•Hydroeconomic valuation
Software Application & Result analysis (6)
•Sequences of hydrological objectives (RIBASIM)
•Hydroeconomic objective (van der Vat, 2015)
•Sensitivity analysis of hydroeconomic model
•Modified hydroeconomic valuations
•Sequences of hydrological objectives (application)
Conclusion (7)
•Answer to research questions
•Recommendation on future research

Figure 1.1 Overview of the study

Chapter 2 presents results of the literature on multi-objective problems solving. It begins with an extensive literature research on hydroeconomic studies and possible optimization approach in integrated river basin management. Chapter 3 describes the case study: the Citarum basin and the cascade reservoirs in that basin.

Chapter 4 explains the methodology that is applied to address each research question. This chapter includes the tools used, data collection and sequences of tasks. Chapter 5 describes the model structure and technical details of the optimization performed in the RTC-Tools 2.0. This includes the model schematization and database management. It also covers the assumptions taken and the governing equations in the tools. The hydroeconomic valuation analysis of the parameters in the optimization model is also defined in this chapter.

In Chapter 6, the different objective functions in the optimization models are explained. This chapter describes the software application in greater detail along with the practicalities of building various optimization models in RTC-Tools 2.0. The results and findings from each optimization models are discussed and analysed in this chapter. Finally, the last Chapter 7 concludes this study by answering research questions along with some future recommendations in the study area.

2. ADDRESSING MULTI-OBJECTIVE PROBLEMS

In recent years, there has been an increasing amount of literature on solving multi-objective problems in integrated river basin management. One of the current approaches is hydroeconomic analysis. In his publication, Harou *et al.* [2] reviewed the literature from the past 50 years and described that hydroeconomic models can play an important role in addressing the increased water scarcity and conflicts issues due to the future challenges. By combining the principles of economics and engineering, hydroeconomic models transform the concept of fixed demand into the economic value of water defined through water rights, priorities and future projections [3] [4] [2]. This new concept of economic water demand is optimized to generate the maximum net benefit by driving the water allocation and managing the existing supply-demand. In this way, a better operation system in the water management system is developed to avoid constructing new supply options [5]. Harou *et al.* [2] identified the design choices and options to construct a hydroeconomic model as model components, time representation and sub model integration. Table 2.1 provides an overview of these design choices. The highlighted components present the selected design to construct the hydroeconomic models in this study. Each component is further explained in the following section.

Table 2.1 Summary of hydroeconomic model design choices

Hydroeconomic model		
2.1	Model components	Simulation
		Optimization
		Combination between simulation and optimization
2.2	Time representation	Deterministic (various hydrological years and implicit stochastic optimization)
		Stochastic
		Dynamic
2.3	Submodel integration	Holistic
		Modular
2.4	Software implementation	Goal programming optimization (multi-objective)
		Linear deterministic optimization (a single objective)
		Hybrid between goal programming and linear optimization

2.1 Model components: Simulation and optimization model

Possible model components for a hydroeconomic model include simulation models, optimization models, or a combination between both models.

A simulation model runs a rule-based algorithm to reproduce the system complexities in integrated water resources management, planning and policies [6]. This model is mostly driven by a simple mass balance concept. The strategies are simulated with trial-and-error to identify the best feasible solution.

An optimization model runs based on the economic objectives and constraints that represent the system to identify the most promising operation rules. The objective can be formulated as linear or non-linear hydroeconomic functions. Optimization models are less suitable to simulate complex water networks and non-linear system dynamics [2]. On the other hand, linearity is considered to be too theoretical to represent the system in mathematical terms [7]. As finding a global optimum for non-linear objective functions is likely to be inconvenient, linearization is often applied to hydroeconomic modelling [2] [7]. Linearization can be taken in case the difference in results for both formulations is negligible and unimportant [8].

2.2 Time representation: Deterministic, stochastic and dynamic model

The time representation in the hydroeconomic model is classified as a deterministic model, a stochastic model, or a dynamic model. A dynamic model could be a deterministic model or a stochastic model depending on the input data.

A deterministic model uses historical or synthetically generated time series to obtain a single set of results. An appropriate operating rule under certain condition is mostly derived by representing hydrological conditions (wet, normal and dry period) as simplistic probabilistic events [9]. Furthermore, the deterministic method is often sophisticatedly considered as an implicit stochastic model if the hydrological time series data is long and representative enough [4]. Some authors have performed studies implementing linear optimization in the hydroeconomic model [4] [10]. This approach often termed as *linear programming*. Hydropower, irrigation, domestic demand, flood control, recreation, navigation and environmental flow are reliable to be represented in those studies since the deterministic model is relatively simple. Thus far, some studies have shown promising results from non-linear reservoir optimization for hydropower, irrigation and/or domestic water [11] [12]. Other studies even perform non-linearity in the objective functions with some additional constraints such as flood and/or environmental flows [13] [14].

A stochastic model explicitly incorporates the probabilistic character of model inputs to generate the probability results rather than a single, deterministic single set of results. In recent years, some authors have explored the stochastic-dynamic model in hydroeconomic applications [15] [16]. This method has been successfully applied to the reservoir optimization for irrigation and hydropower purpose in various river basins in the world. This advanced approach captures the complexity of economic phenomena, although it is considered as very computationally intensive. Besides this method often suffers from difficulties in representing the stochastic phenomena [4], it tends to create some implementation issues since it is harder to explain.

A dynamic model represents the time–dependent aspects of the model behaviour. Most existing models exemplify a static capture of a particular time while the reality is constantly changing over time [17]. In this specific context of the hydroeconomic study, a dynamic model or a time-varying economic optimization model considers that both benefit and cost are time-dependent. The objective function is further defined as a function of interest rate and time.

2.3 Sub model integration: Holistic and modular approach

The design choices of sub model integration in hydroeconomic model are a holistic approach or a modular approach. User preferences on sub model integration are based on the advantages offered by these different approaches.

In a holistic approach, all components are housed in a single model. It is considered as the best approach to identify the unique global optimal solution [18]. The main disadvantage of this approach is that it cannot be constructed on existing water planning models. The water resources and economic system needs to be recoded and simplified.

In a modular approach, the model components run separately. An advantage of using this approach is that it avoids the problem of deriving the operation rules from optimized release flows since it can be attached to the existing water planning model [18].

2.4 Software implementation

A seminal study in hydroeconomic area is the work of Lund & Ferreira [4] which covers most of the aspects of the system. They developed a linear deterministic hydroeconomic optimization model by assessing economic penalties for various conflicting water uses. Lund & Ferreira [4] points out that this simple linear deterministic approach provides valuable engineering functions to identify the promising, suitable operating rules. This study also concludes that the optimization quality strongly depends on the formulation of the objective functions, which are reported to be more influential than simplifications such as linearization taking a deterministic approach.

Van der Vat [1] has successfully applied this approach to optimize the deterministic linear hydroeconomic model. His study optimizes the rule curves of the cascade reservoirs in the Citarum River in Indonesia with benefit functions based on agricultural delivered demand, hydropower generation and flood risk reduction. This study did not include domestic water demand and environmental flow. His study implemented a fully user defined optimization formulation (Python scripted particle SWARM) in a combination with a simple mass-balanced rule-based simulation model (RIBASIM). This method is able to identify directly the rule curves instead of released flows with less computational efforts. The optimization results from this study are somehow limited by the impracticalities of the optimized rule curves. The reservoir operation rule curves change the reservoir operation rule curves up to 40 m monthly. This high monthly fluctuation may result in the inapplicable reservoir operation.

Harou, *et al.* [2] found that a major problem with this kind of hydroeconomic models is to explain the economic complexity to the stakeholders compared to existing water resources models. Also, achieving a suitable efficiency and transparency in water use will be more challenging. One of the reasons is that the management schemes and the policy insight are less likely to be easily represented by the benefit or penalty functions. In addition, hydroeconomic models often struggle with robustness at the local scale; small changes in model parameters could dramatically alter the water allocation in the system [2]. Thus, additional analysis such as a shadows values, the range of basis and a sensitivity analysis are mostly required to provide more information on the model behaviour.

The hydroeconomic concepts generally stand in distant from the stakeholders' perspective on priorities and infrastructure projects. In order to assembly these different perspectives, a *goal programming* approach can be chosen. A number of authors have considered that this method manages to bridge the gap between research and practical application. It avoids the problem of justifying the weighted values that define the priorities [19] [20]. In this approach, specific numeric goals are derived based on the priorities to set the series of objective functions. The lower and upper limit for each goal is restrained as soft constraints [21]. The optimized solution is obtained from the *minimum sum of deviations* of these objective functions. It implies that this approach allows violation on its own soft constraints; it creates a blurred boundary between the objectives and the soft constraints. On top of these soft constraints, the inviolable hard constraints can be also applied. This is one of the more practical ways considering the real implementation in the water resources planning.

Leavesley *et al.,* [22] presented a relevant study using this approach. He demonstrated a leading approach using goal programming for a multi-purpose reservoir study. His study has revealed the most striking advantages of this approach. It provides a new alternative to integrate explicitly the water system with the water-related policies. It also shows the flexibility on defining the future changes in the system. Their studies did not include an economic analysis. Another significant analysis and discussion on the goal programming approach were presented by Eschenbach *et al.*, [21]. Their study sets the river flow as a hydrology objective, the income goals as an economic objective and the amount of employment as a social objective. Altogether, this model captures more realistic phenomena when the inter-linkages and the trade-off between the conflicting goals are taken into account; it also explicitly reviews the impacts on water allocation changes if different strategies are implemented.

Concluding Remarks

The review of literature in this chapter has particularly concentrated in various hydroeconomic models to address the multi-objective problem. By combining the principles of economics and engineering, the hydroeconomic models transform the concept of fixed demand into the economic value of water defined through water rights. Unfortunately, the management schemes and the policy insight are less likely to be easily represented by a hydroeconomic objective function, notwithstanding a comprehensive analysis of economic valuations is accomplished. Collectively, these studies highlight the necessity to explicitly implementing the priority ordered by the policy on water resources allocation as an alternative to a conventional hydroeconomic model. It is possible, therefore, that applying the goal programming approach in the hydroeconomic model provides more robust, easy-to-build and communicative method to achieve a transparent water allocation based on the policy insight.

3. CASE STUDY: CITARUM CASCADE RESERVOIRS

Citarum is an intermountain basin located on the main island of Java, Indonesia. As one of the largest rivers in Java Island, its river drains an area of 6,080 m^2 [23]. With 225 kilometres in length, the Citarum River begins from south of Wayang Mountain, run through the provincial capital city of Bandung and out into the Java Sea. This literature review covers the characteristics of case study related to the data applied to the constructed hydroeconomic model. As an overview, Table 3.1 summaries of literature review on the Citarum cascade reservoirs. Each component is further explained in greater details this chapter.

Table 3.1 Overview of the case study applied

Study case	Citarum cascade reservoirs, Indonesia
Available model	Simplified model of Citarum cascade reservoirs, Indonesia
Modelling scale: space	Lumped
Modelling scale: time	Monthly
Rule-based simulation	RIBASIM schematization and algorithm (Dijkman, 2012)
Time series data (1920 - 2009)	Upstream inflow
Time series data (annual)	Evapotranspiration rate, water demand and energy firm demand
Reservoir bathymetry	Dead level, dam height, relation between water level, area and volume
Turbine characteristics	Capacity, friction losses and tail water level
Hydroeconomic optimization	Python-coded meta-heurictic SWARM particle (van der Vat, 2015)
Benefit	Agriculture (US\$/m^3), peak and rest hydropower generation (US\$/kWh)
Penalty	Flood damage at 320 m^3/s
Others	Hydropower peak fraction

The three segments of the Citarum river are: upper (25km, 750-3,000 m+MSL), middle (150km, 200-2,400 m+MSL) and the lower part (70km, >150 m+MSL) [24]. BBWS[1] stated that the basin slope decreases gradually from upstream to downstream [25]. The Citarum basin characteristics and schematic map are shown in Table 3.2 and Figure 3.1.

Table 3.2 The Citarum basin characteristics

Basin Characteristics	*Unit*	*Citarum*
Elevation	m+MSL	0 - 2,400
Rainfall	mm/year	1,500-4,000
Soil type		Loamy clay
Land use		
Agriculture	%	60
Forest	%	25
Human activities and others	%	15
Evaporation	mm/day	4.4
Agiculture	%	85
Domestic, industries	%	10
Others	%	5

[1] BBWS: Balai Besar Wilayah Sungai, the central government agency for riverbasin organization

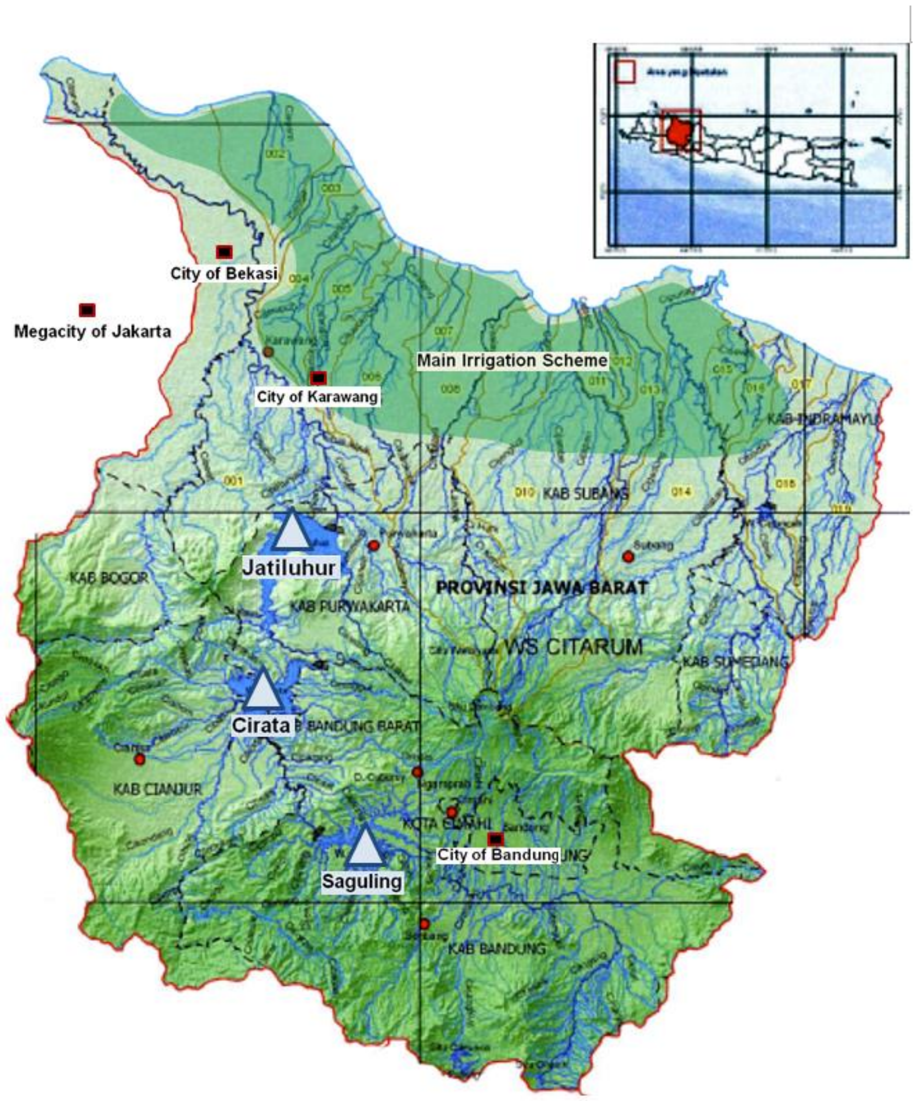


Figure 3.1 Schematic map of the Citarum River basin [26]

To date, several studies have reported that rainfall ranges spatially between 1500 and 4000 mm/year [27]. BPLHD [2] [28] identified that the basin mostly contains many volcanic products of loamy clay basin with the main cover of agricultural area (60%), forest-bushes (25%) with the remaining area used for human activities and water bodies. The average surface water availability provides an estimate of 44 m^3/s [29]. This inflow discharge is ranging throughout the year; it drops during the dry season from April to September.

In this basin, the average evapotranspiration reaches 4.5 mm/day. This value is derived from the high humidity between 80-92 g/m^3 and the high temperature between 15°C – 27°C. Both parameters are almost constant all over the year but spatially varied depending on the elevation. Rejekiningrum [30] identifies the water demands are divided into 85% for irrigation and the rest for domestic use of 15 million population, fisheries and industrial activities. The cascade reservoirs are installed in the heart of this basin to satisfy these target demands and to supply electricity for Java and Bali.

[2] BPLHD: Badan Pengelolaan Lingkungan Hidup Daerah, the regional environmental agency

MPW [29] analyses the most likely changes in the basin are the increased pressure in demand from economic development and population growth. Also, climate change is forecasted to change the rainfall pattern in the basin. In additional, the possible future changes in the Citarum basin are listed below [27]:

- In the upper Citarum, the basin struggles to control the land use change, specifically on the trend of primary forest conversion into the build-up land [31]. The trend that the forest has decreased until 40% in 15 years in the upper basin [32].
- In the middle Citarum, the daily domestic waste thrown into the river reaches 700 m^3. This amount of solid waste reduces the capacity of river and reservoirs while both already strive with the high level of sedimentation. It also aggravates the water quality problem that was previously created from the fisheries activities in the reservoir.
- In the lower Citarum, the risk of downstream flooding mostly happens in the city of Karawang and Bekasi is foreseen to be higher due to the land use change.

Realising the importance of the Citarum basin, the Jatiluhur reservoir was constructed as multi-purpose reservoirs in 1957 [33]. After nearly 25 years, another two upstream reservoirs, Saguling and Cirata were built mainly for hydropower generation. As shown in Table 3.3, the characteristics of each reservoir including the reservoir area, storage capacity and turbines capacity of each reservoir are compiled [34] [35] [36].

Table 3.3 The Citarum cascade reservoirs characteristics

Reservoir Characteristics	*Unit*	*Saguling*	*Cirata*	*Jatiluhur*
Purpose		Flood control, hydropower	Flood control, hydropower	Flood control, hydropower, irrigation and public water supply
Management		PT PLN State-owned company for electricity	PT PJB State-owned company for Jawa-Bali	Perum Jasa Tirta II State-owned enterprise for management of public water supply
Dam name		Saguling	Cirata	Djuanda
Elevation	m+MSL	643	220	106.89
Dam height	m	99	125	105
Reservoir area	km^2	30	50	70
Reservoir capacity	Mcm	800	3,200	2,600
Hydropower capacity	MW	700	1,008	150

Figure 3.2 presents the reservoir operation rules based on the trial and error in a spreadsheet model produced by NEDECO[3] [37]. In 2010, SPK-TPA [4] [38] generated the reservoirs rules from the similar spreadsheet with the automatic solver *goal-seek*. In March and May 2010, an exceptionally high inflow and unregulated Jatiluhur released discharge resulted in downstream flooding in the city of Karawang and surroundings. These repeated flooding events triggered the review the SOP[5], including the new annual operation procedures and the implementation rules [39]. Additionally, Perwitasari [40] reviewed the literature from the flooding events in 2010 and performed a study on a decision support tools on the daily basis.

[3] NEDECO: Netherland Engineering Consultants BV
[4] SPK-TPA: Secretariat of coordination committee on the administration of Citarum water
[5] SOP: Standard Operation Procedure

To implement RIBASIM rule curve into The SOP Citarum, it assumed that the firm curve represents the dry year and the target curve represents the normal year [39]. The RIBASIM has a curved rule on the flood control, while in the SOP it is defined as a specified constant level below the spilling level.

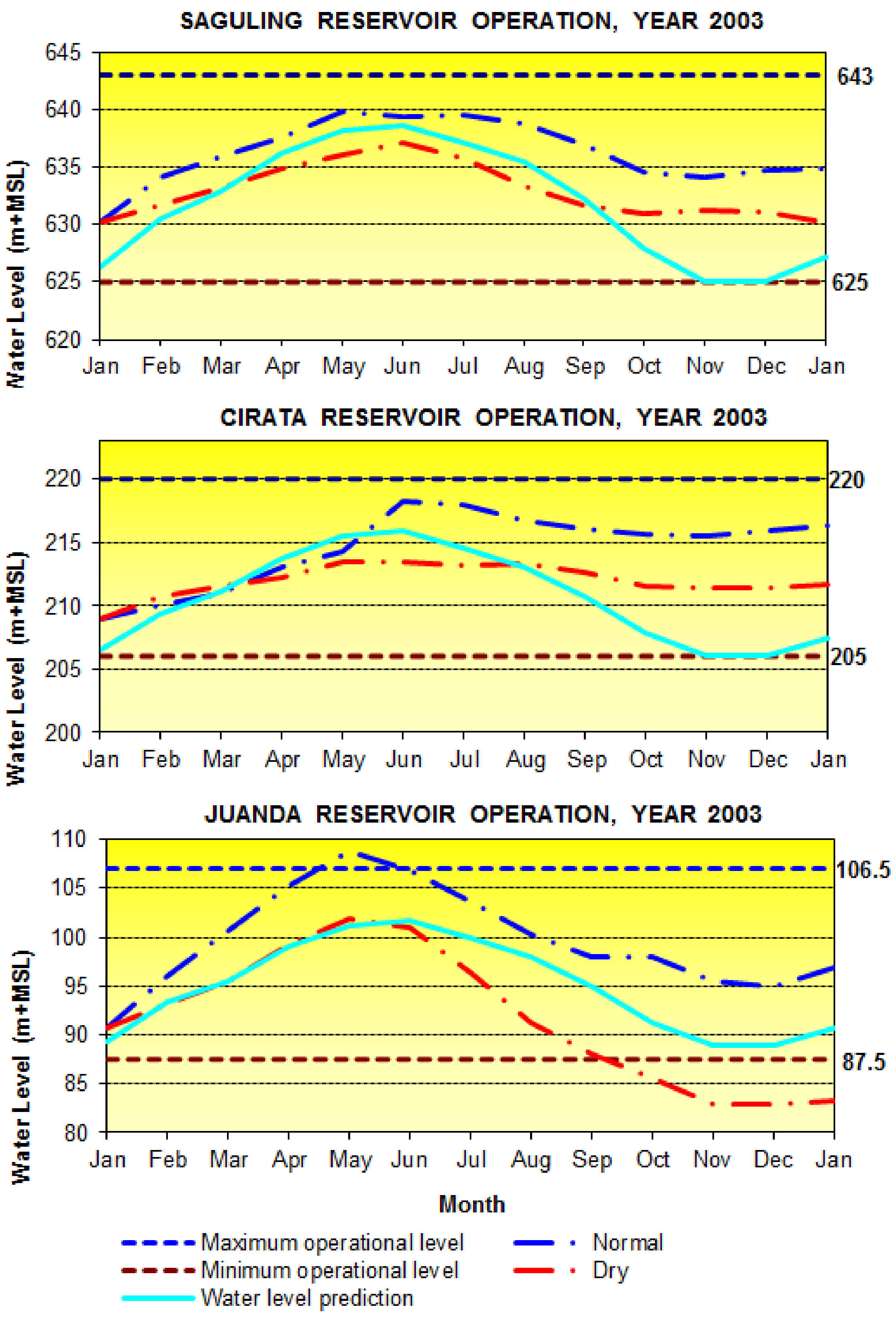


Figure 3.2 The Citarum cascade reservoir operation rules [37]

A simplified Citarum cascade reservoir schematization in RIBASIM by Dijkman*, et al.* [39] is a favourable reference in this case study. This existing rule-based simulation model is a lumped model with the monthly time step, but sufficient for the purpose of this study. In this model, the land use, cropping pattern, climate, water demand and energy demand are presumed to be similar in this period.

Most data required in the optimization model in this study are extracted from this RIBASIM model. The upstream inflow into reservoir time series data is exported from the period 1920-2009, ranging from 600 to 2400 m^3/s annually. The monthly agricultural water demand (80-200 $m^3/month$) is assumed to be similar each year. The reservoir evaporation rate is ranging between 3 mm/day and 5 mm/day. The reservoir bathymetries including its turbine characteristics are taken directly from the built reservoirs model in RIBASIM. These provided figures are comparable to the literature review in this study.

In the hydroeconomic optimization model in this study, the benefit and penalty cost are mostly taken from the RIBASIM-PS study This covers agricultural benefit, flood damage cost, the fraction of peak and rest hydropower generation including their distinctive benefits. These hydroeconomic valuations, which provide a promising optimization result in RIBASIM-PS study, will be modified in this study.

4. METHODOLOGY

To date, various methods have been developed and introduced to assess water allocation in a river basin. Each method has its advantages and drawbacks. Traditionally, the optimum water allocation has been assessed by simulating various strategies in a rule-based simulation tool such as RIBASIM. As this method is time-consuming and not necessarily leads to the most promising result, recent advances in optimization techniques have facilitated the possibility to find better results. The reservoir optimization is expected to become a possibility in the extended version of Deltares software package RTC-Tools 2.0. To address this issue, four research questions have been formulated. The tasks allocated to answers these questions are summarised from Figure 4.1 to Figure 4.4.

The development of the methodology for this study is based on a study conducted by Van der Vat [1] entitled *Optimizing reservoir operation for flood storage, hydropower and irrigation using a hydroeconomic model for the Citarum River, West-Java, Indonesia.* He implemented fully user defined optimization formulation (Python-scripted particle SWARM) in combination with simple mass-balanced water allocation model (RIBASIM). The study of Van der Vat [1] will be referred as the RIBASIM-PS study.

The detailed technical user manuals of RIBASIM provide the guidelines to further analyse the mechanism of water allocations and the assumptions taken [41] [42]. By applying hydroeconomic objective functions, the optimum reservoir rule curves are attained from the particle SWARM optimization [43] [44]. For additional information, a number of important reviews of the standard operation procedures of the Citarum cascade reservoirs are presented in a project report prepared for Indonesian Ministry of Public Works [39].

RIBASIM (https://www.deltares.nl/en/software/ribasim/) is a software package by Deltares [45] that provides sources of analysis in water distribution pattern together with water quality and sedimentation for integrated river basin management. This software has been applied in more than 20 countries since 1985 in many river basins in the world mainly to evaluate alternative measures in infrastructure as well as for operational and demand management questions. The RIBASIM technical details can be retrieved from its user manual [42].

RTC-Tools (https://www.deltares.nl/en/software/rtc-tools/) are an open-source toolbox for hydraulic systems developed by Deltares [46]. The RTC-Tools 1.0 Tools, which was published in 2007, are widely applied to optimization problem by coupling the tools with a rule-based simulation tool. It is widely applied to real-time control operation of hydraulic structures such as weirs, reservoirs, and pump stations. The tools used in this study will make use of the new generation of RTC-Tools 2.0.

RTC-Tools 2.0 tools offer various approaches to address optimization problems, ranging from single to multi-objective problems. The new RTC-Tools 2.0 employs Modelica user interface inside their software architecture. Modelica is an open source object-oriented language developed by Modelica Association [47]. It is widely applied to cyber-physical systems modelling in automotive or aerospace industries. Modelica is primarily designed for simulation while its application for optimization is noticeable.

A case-study approach was adopted to provide rounded, detailed illustrations of the policy-based-management in water resources. The case study chosen is a simplified water network of the Citarum basin in West Java, Indonesia. As many types of research have been conducted in this basin, Deltares could provide the detailed information needed for this study.

Research question 1: *Is RTC-Tools 2.0 able to model a similar network as RIBASIM does, using allocation rules based on demand priority and reservoir operation rules including hedging?*

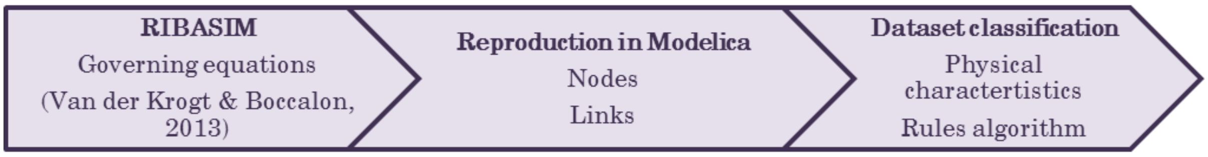


Figure 4.1 Network development in the RTC-Tools 2.0

The first step in this study is to fully comprehend the formulation of the nodes and links in RIBASIM. This has been done by developing a spreadsheet model based on the RIBASIM user manual as the main guidelines. Once the water allocation rules in RIBASIM have been evaluated, similar technical details and visualisation are reproduced in Modelica. Some differences are expected in certain components, such as surface water reservoirs.

The datasets used for constructing a new optimization model in RTC-Tools 2.0 is extracted from the existing RIBASIM model without any additional data collection. Prior to importing this dataset, the data are classified whether they are the parameters of physical characteristics or the rules in the model algorithm. In addition to the physical characteristics of infrastructure, the physical data also cover time series input, such as the upstream inflow into three reservoirs and evaporation rate. The rules in the model algorithm include the water demand and the firm energy demand. Other important information that must be highlighted are the reservoir operation rules and allocation rules based on demand priority setting.

Research question 2: *Is it possible to formulate a set of objectives and constraints in the RTC-Tools 2.0 that will result in optimized reservoir operating rules?*

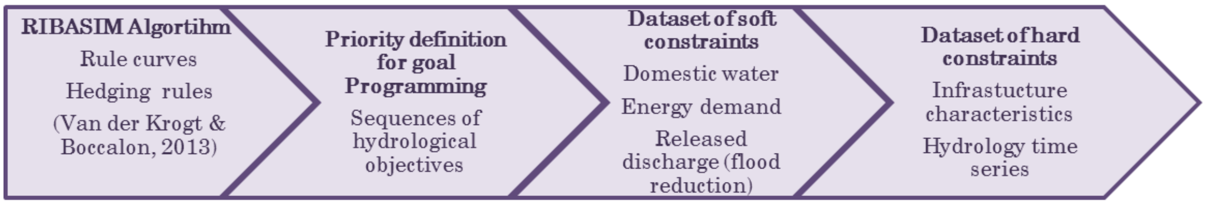

RIBASIM Algortihm
Rule curves
Hedging rules
(Van der Krogt & Boccalon, 2013)
Priority definition for goal Programming
Sequences of hydrological objectives
Dataset of soft constraints
Domestic water
Energy demand
Released discharge (flood reduction)
Dataset of hard constraints
Infrastucture characteristics
Hydrology time series

Figure 4.2 Sequences of objectives definition in the RTC-Tools 2.0 (RIBASIM proxy)

As pointed before, the goal programming approach requires the sequences of objectives to optimize the reservoir operation rules. These sequences of objectives are defined based on the demand priority setting and reservoir operation including rule curves and hedging rules in RIBASIM model. The datasets of rules in the model algorithm are defined as soft constraints. These soft constraints refer to the lower and upper limit for each objective in the form of time series taken from the RIBASIM datasets. The data management and the result analysis are performed in the spreadsheet.

As the deterministic approach is applied in this study, a note of caution is due to the presumption of the perfect knowledge of the future events. However, in the application, the upstream inflows into the reservoir and the water demand should be considered as the major sources of uncertainty.

Research question 3: *Are the calculations of optimal reservoir operation rules by RTC-Tools 2.0 different from the operation rules resulting from the RIBASIM-PS optimization and, if so, why?*

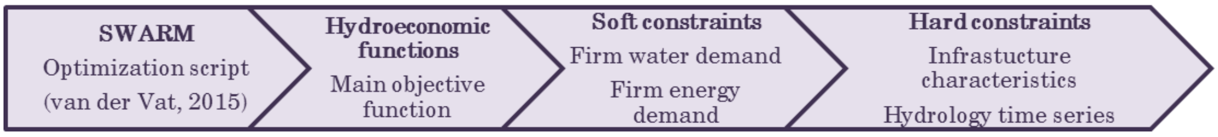


Figure 4.3 Hydroeconomic optimization approach in the RTC-Tools 2.0 (RIBASIM-PS)

In order to gain the comparable results with the particle SWARM optimization, a similar hydroeconomic objective function is applied in this study. This requires a deep analysis on the particle SWARM optimization and the RIBASIM algorithm that may impact the optimization results. The maximum benefit as hydroeconomic objective function is set in RTC-Tools 2.0. Other hydroeconomic formulations that build this main objective function are declared in Modelica. In the next step, the downstream target demand in RIBASIM algorithm is set as soft constraints in RTC-Tools 2.0. The target demand consists of the firm energy generation and the agricultural water demand. As RTC-Tools 2.0 has a more distinctive optimization solver than the particle SWARM optimization, a comparison of the two results could reveal the applicability of the reservoir operating rules based on the case study applied. The optimization results from both tools are presented as the actual water level of reservoirs and the fulfilment of downstream target demands.

As hydroeconomic models often struggle with robustness at the local scale, small changes in economic valuation may dramatically alter the water allocation in the system [2]. Thus, the optimization results strongly depend on the objective functions, which is likely to be the most expensive and time-consuming part of the hydroeconomic study. In particular, the analysis of economic valuation in this study is limited considering that it should be conducted by an expert with a strong economic background. To address this issue, the sensitivity analyses are performed to identify the changes in the water allocation if the economic valuation differs from what was previously assumed.

Research question 4: *How can we further improve the results of the optimization approaches in order to get a better applicability of the reservoir operation rules?*

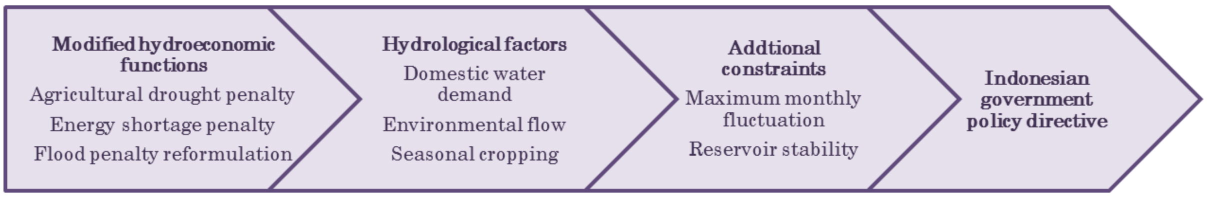

Modified hydroeconomic functions
Agricultural drought penalty
Energy shortage penalty
Flood penalty reformulation
Hydrological factors
Domestic water demand
Environmental flow
Seasonal cropping
Addtional constraints
Maximum monthly fluctuation
Reservoir stability
Indonesian government policy directive

Figure 4.4 Reservoir operation rules improvement

The RIBASIM-PS study of the Citarum cascade reservoirs optimization has not dealt with detailed constraints and hydroeconomic valuations since it mainly focused on the coupling and optimization process. That can be improved by conducting the hydroeconomic analysis by reanalysing the economic valuations and formulation, such as an additional penalty functions. This could be partly linked to the results of the sensitivity analysis on the economic valuation.

Comprising more hydrological factors into account could improve the RTC-2.0 Tools optimization model. To assess agricultural water demand better, the seasonal cropping of paddy and nuts are further analysed. Other prioritized water demands, such as domestic water demand and environmental flow are also included in the optimization model although they are not quantified economically. More constraints related to the applicability of reservoir operation rules are added to find a more suitable solution in the case study. A minimum water level should be set to be higher than the dead level to ensure the reservoir stability. The maximum water level fluctuation in a month is also taken into account. Improving the hydroeconomic model can be also achieved by following the operational procedure for Citarum cascade reservoirs by the operators. This requires some knowledge on the current Indonesian government policy directive for cascade reservoirs.

The results of the optimization models are compared by data post-processing to obtain the quantity of annual benefit. The economic benefit is calculated based on the economic valuation by Van der Vat [1]. The social benefit is based on the number of events (in the monthly time step) per year.

5. MODEL STRUCTURE

To reproduce the RIBASIM-PS study in the RTC-Tools 2.0, identical datasets are used to obtain comparable results. The schematic presentation of the optimization model constructed in this study is depicted in Figure 5.1. The technical details of the RIBASIM-PS study are briefly explained in Section 5.1. Section 5.2 elaborates the software architecture and features of the RTC-Tools 2.0. Next, Section 5.3 describes the model schematization and database management in greater detail. This section also covers the application of the hard constraints which consist of the parameters of physical infrastructures in the model schematization. At last, Section 5.4 delineates the derivation of the hydroeconomic valuations in this study.

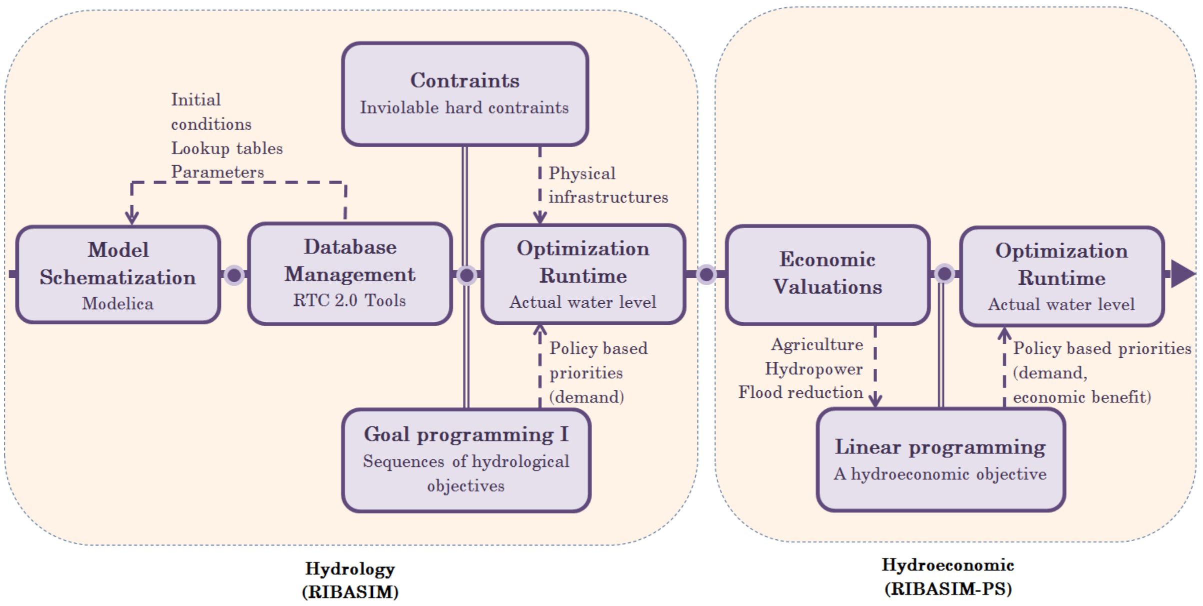


Figure 5.1 Schematic presentation of modular optimization approach

5.1 RIBASIM Rule Based Algorithm and particle SWARM Optimization

To obtain comparable results to the RIBASIM-PS study, the nodes and links of the RIBASIM model are replicated in the RTC-Tools 2.0. The RIBASIM technical details are retrieved from its user manual [42]. The governing equations of the surface water reservoir in RIBASIM are shown in (eq. 1) to (eq. 6).

The target demand $Q_{\,target\,[t]}$) defines the quantity of downstream water demands ($Q_{o[t]}$) and the minimum discharge ($Q_{Efirm[h_{t-1},h_t]}$)) to generate firm energy demand in the dry period.

$$\boldsymbol{Q_{target\,[t]} = Max\,(Q_{o[t]}, Q_{Efirm[h_{t-1},h_t]})} \qquad \text{(eq. 1)}$$

The maximum energy generation or maximum turbine flow is calculated by iteration since it has implicit relation of the net head $(h_{net[h_{[t-1]},h_{[t]}]})$. It is formulated from the intake level (h_{intake}), the friction losses $(h_{loss[h_{[t-1]},h_{[t]}]})$ and the tail level $(h_{tail[h_{[t-1]},h_{[t]}]}))$. The power capacity $(P_{[h_{[t-1]},h_{[t]}]})$, efficiency (e), plant load factor (f), gravity (g) and water density (ρ) are taken into account. The conversion factors (c_1, c_2) are constant values based on the units of the parameters.

$$\boldsymbol{Q_{maxtur[t]} = c_1 * \frac{f * P_{[h_{[t-1]},h_{[t]}]}}{e * g * \rho * (h_{intake} - h_{tail[h_{[t-1]},h_{[t]}]} - h_{loss[h_{[t-1]},h_{[t]}]})}} \qquad \text{(eq. 2)}$$

The energy generated depends on the released discharge $(Q_{o[t]})$ and the net head. The maximum energy generated from the reservoir is capped by the maximum turbine flow. The auxiliary energy consumption (a) is considered in this calculation.

$$\boldsymbol{Q_{tur[t]} = \max(Q_{maxtur[t]}, Q_{o[t]})} \qquad \text{(eq. 3)}$$

$$\boldsymbol{E_{[t]} = c_2 * a * e * Q_{tur[t]} * h_{net[h_{[t-1]},h_{[t]}]}} \qquad \text{(eq. 4)}$$

The spilling from the main gate $(Q_{main[t]})$ happens if the actual released discharge is bigger than the maximum turbine flow. *The spilling from the top* of the reservoir only happens if the water level is higher than the reservoir height.

$$\boldsymbol{Q_{main[t]} = Q_{o[t]} - Q_{maxtur[t]}} \qquad \text{(eq. 5)}$$

Whereas the target released discharge is driven by the downstream demands, the quantity of released discharge $(Q_{o[t]})$ is determined by the rule curves. The *provisional water level* pinpoints a particular zone which determines a specific action for the next time step. It is necessary to clarify exactly what RIBASIM specified as the provisional water level. It is defined as an actual water level in the reservoir after calculating the actual inflows $(I_{[t]})$, all release targets $(Q_{o[t]})$, rainfall $(P_{[t]})$, evaporation $E(_{[t]})$ and other losses. By changing the released discharge, the concept of mass balance between the reservoir storage at the current $(S_{[t]})$ and the next time step $S_{[t+1]}$ is satisfied by an iterative procedure.

$$\boldsymbol{S_{[t+1]} = S_{[t]} + I_{[t]} + P_{[t]} - E_{[t]} - Q_{o[t]}} \qquad \text{(eq. 6)}$$

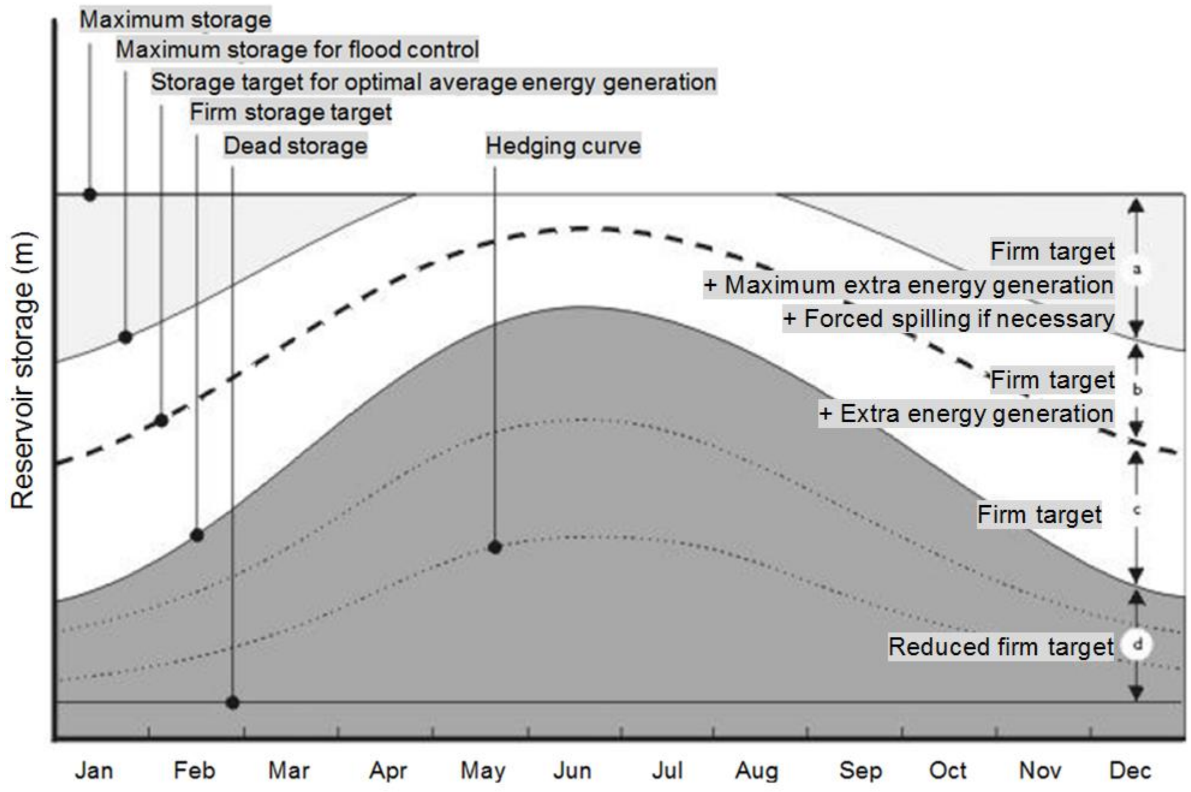

Maximum storage
Maximum storage for flood control
Storage target for optimal average energy generation
Firm storage target
Dead storage
Hedging curve
Reservoir storage (m)
Firm target
+ Maximum extra energy generation
+ Forced spilling if necessary
a
Firm target
+ Extra energy generation
b
Firm target
c
Reduced firm target
d
Jan
Feb
Mar
Apr
May
Jun
Jul
Aug
Sep
Oct
Nov
Dec

Figure 5.2 RIBASIM reservoir operation rule curves

Figure 5.2 illustrates the zones that determine the reservoir operation rules. The set of rules for each zone are:

The flood control curve indicates the maximum water level in the reservoir to provide some storing capacity in the case of a flooding as the consequences of high upstream inflow. If the provisional water level is above this curve, the firm target discharge and the maximum discharge for an extra energy generation are released first from the turbine gates. If the water level still higher than the flood control curve, extra water is forced to spill in order to reach the flood control curve at the end of time step.

The target curve reflects the reservoir water level that generates the maximum hydropower generation. If the provisional water level lies between the flood control curve and the target curve, the firm target discharge and the discharge for an extra energy generation are released until the provisional water level reaches the target curve at the end of time step.

The firm curve reflects the minimum level required to fully supply the firm target discharge. If the provisional water level is between the target curve and the firm curve, the firm target discharge is released.

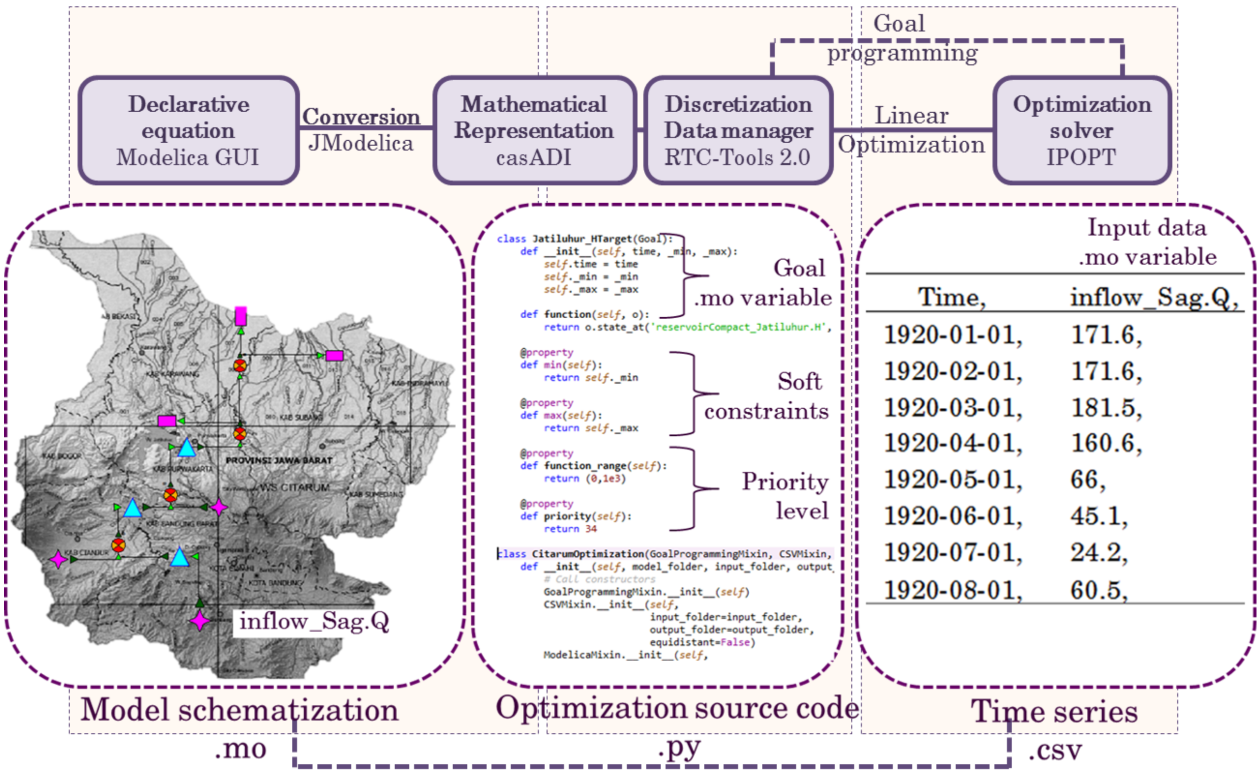
The hedging curves indicate the zone between the firm curve and the dead storage. In the RIBASIM algorithm, the area between the firm curve and the dead storage are divided into 5 zones. These zones determine what percentage of the firm target discharge is released if the provisional water level drops into this zone.

The online adjusted gate option is available for the cascade reservoirs simulation. This option, which is employed in the RIBASIM-PS study, forces the upstream reservoir to release a higher released discharge if the water level of the downstream reservoir is below the firm curve.

In the RIBASIM-PS study, a global optimization approach termed the particle SWARM is combined with the rule-based simulation tools (RIBASIM). This recent approach identifies the optimized parameters by initializing them by random values that generate the whole range of possible solutions [43]. Those parameters are later updated to reach the detected best solutions. Inspired by the synchronized move of the bird flocks, these parameters move to search the best solution. The best neighbouring solutions attract the other particles by adjusting the velocity vector of these particles. The optimization converges if most updated values of the parameters already generate the approximate best solutions. In this way, the particle SWARM optimization is able to cope with the local minima from non-linearity and non-convexity of the objective function. It is also independent from the initial values given as long as the particles cover the entire solutions.

5.2 RTC-Tools 2.0 Software architecture

A modular optimization model has been set up to optimize the sequences of objectives. Figure 5.3 depicts the architecture of the software along with their interaction during the optimization runtime. Each component is further explained in this section. This figure also captures the graphical user interface of the software.



```
class Jatiluhur_HTarget(Goal):
    def __init__(self, time, _min, _max):
        self.time = time
        self._min = _min
        self._max = _max

    def function(self, o):
        return o.state_at('reservoirCompact_Jatiluhur.H',

    @property
    def min(self):
        return self._min

    @property
    def max(self):
        return self._max

    @property
    def function_range(self):
        return (0,1e3)

    @property
    def priority(self):
        return 34

class CitarumOptimization(GoalProgrammingMixin, CSVMixin,
    def __init__(self, model_folder, input_folder, output
        # Call constructors
        GoalProgrammingMixin.__init__(self)
        CSVMixin.__init__(self,
                          input_folder=input_folder,
                          output_folder=output_folder,
                          equidistant=False)
        ModelicaMixin.__init__(self,
```

Time,	inflow_Sag.Q,
1920-01-01,	171.6,
1920-02-01,	171.6,
1920-03-01,	181.5,
1920-04-01,	160.6,
1920-05-01,	66,
1920-06-01,	45.1,
1920-07-01,	24.2,
1920-08-01,	60.5,

Figure 5.3 The architecture and graphical user interface of RTC-Tools 2.0

Modelica

The reproduced network of the Citarum cascade reservoirs is defined in Modelica. In this study, the use of Modelica is limited to the visualization purpose and the equation declaration. It can also store the physical parameters of defined objects. In Modelica, a physical infrastructure is defined as an object which stores some equations. Modelica also opens a possibility to define the detailed components inside an object. This can be illustrated briefly by representing two turbines, a main gate and a spilling gate as sub-model components inside a reservoir object. Then, this object can be connected to the other objects to create an integrated network.

One of the key features of Modelica is the declarative equation-based language. This language eliminates the step in implementing the algorithms explicitly [47]. It leads to a shorter, more understandable code which directly corresponds to the mathematical logic. Some examples of the declarative equation can be seen in Section 5.3. Following is a brief explanation how this declarative equation is used to solve an optimization problem. The declarative equations in Modelica language are compiled by a compiler called *JModelica*. This compiler converts Modelica models into a symbolic mathematical representation that is accessible in the *Python language* using a framework called CasADi. RTC-Tools 2.0 then discretizes these equations, injects time series and lookup tables, and interfaces the resulting optimization problem with a non-linear programming solver called IPOPT.

RTC-Tools 2.0

The new generation of the *RTC-Tools 2.0* is a toolbox for the control and optimization of environmental systems. RTC-Tools 2.0 is the modular optimization tools in the *Python* language [46]. The Python language is known as an effective programming language since the type of variable is implicitly defined. The RTC-Tools 2.0 is responsible for the data management and the linkages between the software during the optimization runtime. It mainly handles the non-physical input data such as lookup tables, initial conditions and the time series input. The constant physical parameters can be also managed.

The conventional linear programming approach is applied by the substantial assistance from the RTC-Tools 2.0. As shown in the (eq. 7) this optimization model searches a local minimum of an objective function $f(x)$. Therefore, applying an appropriate initial condition and a formulating a suitable objective function are the most crucial steps in this type of optimization.

$$\frac{df(x)}{dx} = 0 \qquad \text{(eq. 7)}$$

In the goal programming approach, several goals can be set together although they have different scale and unit. The goals are numbered depending on their priorities; a smaller number defines a higher priority while several goals can be set as the same priority level. Each goal in the sequences of objectives is directly linked to the variables in Modelica. The possible range of the optimized value for each goal is described in the time series of the upper and lower bounds. These bounding values are further termed as the soft constraints. They compel the optimization search space to shrink in the lower number of priority goal.

In the RTC-Tools 2.0, the algorithm satisfies the highest priority goals first on all-time series. The lower priority goals are solved afterward. As shown in the (eq. 8), the optimum solution is attained when the sum of deviation ($f_k(x)$) from all variable in all-time series reaches the minimum value. Moreover, the optimum solution must be inside the bounds of the inviolable hard constraints($g(x)$). It is also important to note that the total deviations of the higher goals $\left(f_i(x_{opt,i}) \right)$ must remain constant or smaller after the lower goals ($f_i(x)$) are solved.

$$\boldsymbol{min\, f_k(x)} \quad \textbf{subject to}$$
$$\begin{cases} \boldsymbol{g(x) \leq 0} \\ \boldsymbol{f_i(x) = \varepsilon_i \; \forall i < k} \qquad \boldsymbol{\varepsilon_i = f_i(x_{opt,i})} \end{cases} \qquad \text{(eq. 8)}$$

The time series data and the sequences of objectives are handled by the RTC-Tools 2.0. Both are imported from the comma separated files that are linked to the variables in Modelica object. The optimization process is entirely successful when the lowest goals are solved.

In RTC-Tools 2.0, a value of satisfaction tolerance between 0 and 1 could be assigned to all goals. The current goal is considered to be fully satisfied if a satisfaction variable is above 1 – satisfaction tolerance. While smaller number indicates tighter criteria, the satisfaction tolerance could be set as 1 to disable this option. In RTC-Tools 2.0, setting tight criteria might result in an unsolvable goal within the maximum number of iterations. If RTC-Tools 2.0 is unable to solve a goal, it terminates the runtime at the latest solvable goal. Furthermore, a value of constraint relaxation(c) between 0 and 1 could also be assigned to control how much the soft constraints could be violated (O_{tol}) depending on the range of value between upper and lower soft constraints(R_s). This relation is described in (eq. 9)

$$\boldsymbol{O_{tol} = R_s * c} \qquad \textbf{(eq. 9)}$$

5.3 Model schematization and Database management

As previously mentioned, the variable in Modelica object is directly connected to the hard constraints, the sequences of objectives and the time series data. Table 5.1 presents the overview of hard constraints applied which are further described in this section. The time series data injected to the variable are also discussed while the sequences of objectives are outlined in Chapter 6.

Table 5.1 The hard constraints of the optimization model

Modelica object	*Minimum*	*Maximum*
Reservoir water level	Elevation	Dam height
Reservoir relesed discharge	0	Gate capacity
Reservoir energy generation	0	Turbine capacity
River inflow discharge	0	∞
Canal inflow discharge	0	Canal capacity

The Modelica object can be easily reproduced and redefined by modifying the declarative equation and the icon visualization. The symbolization of nodes and links in Modelica and RIBASIM are shown in

Figure 5.4. Furthermore, the declarative equations for each object are shown in (eq. 10) to (eq. 14). These equations are declared to be similar to the governing equations in RIBASIM except for the surface water reservoir node. In Modelica, the conversion factor is not necessary since the variables are always defined in the standard international unit. Figure 5.5 depicts the model schematization for the Citarum cascade reservoir while the detailed object specifications are explained in a greater detail in this section.

Node/ link	*RIBASIM*	*RTC 2.0 Tools*
Inflow	✦	✦
Terminal	▬	▬
Demand	■	⊗—▬
Conjuction	●	⊗
Bifurcation	⧗	
Surface water reservoir	△	△
Channel	——	——

Figure 5.4 The symbolization of the nodes and the links in RIBASIM and RTC-Tools 2.0

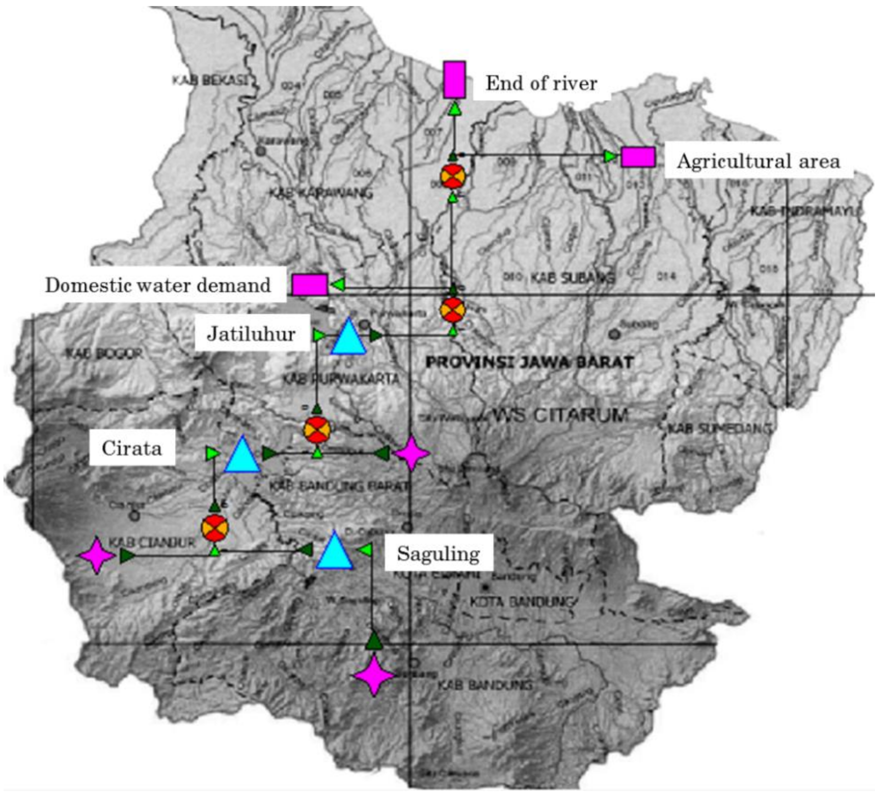


Figure 5.5 Modelica model schematization of the Citarum cascade reservoirs

Inflow

The upstream boundaries of the network are specified as the time series discharge. These time series data are mostly obtained from the data pre-processing on the rainfall-runoff simulation. In this study, the inflows time series are obtained from *Perum Jasa Tirta I,* the Indonesia state own enterprise for water resources management sector. The inflow discharge in the Saguling reservoir is obtained from the observed upstream monthly average discharge $(QOut.Q)$. However, the inflow data from other two downstream reservoirs are calculated based on the mass balance concept by reviewing the observed reservoir released discharge and water level.

$$\boldsymbol{QOut.Q = Q} \qquad (\text{eq. } 10)$$

Terminal

A terminal node represents the end of a natural channel without any restriction on the flow rate $(QIn.Q)$, for example, a river estuary. It can also symbolize the end of the canal which delivers a target discharge $(QIn.Q)$. To avoid flooding, the maximum flow to the canal is generally constrained based on the target demand.

$$\boldsymbol{Q = QIn.Q} \qquad (\text{eq. } 11)$$

Conjunction and bifurcation

This node represents the splitting of a channel into two or more parts. A physical example of this node is the main river diversion into a canal. The diversion is regulated or allocated based on the target demand. The mass balance concept as a function of the total inflows $(QInSum)$ and the total outflows $(QOutSum)$ is enforced. This node could also define a confluence where two or more water bodies meet in the river tributaries.

$$\boldsymbol{QOutSum = QInSum} \qquad (\text{eq. } 12)$$

Surface water reservoir

The surface water reservoir represents a surface water storage that controls the downstream released discharge. Hence, the reservoir operation rules are the key factor in satisfying the downstream target demands. These demands could be consist of domestic water demand, environmental flow, agriculture, hydropower release and flood control. The inviolable hard constraints or the physical constraints of the reservoir are represented by the reservoir height, gate capacities and turbines capacities. On the other hand, the reservoir operation rules such as the rule curves and the hedging rules are specified as soft constraints.

Similar to the common reservoir model, the changes in the reservoir volume $(der\,(V))$ is a function of the total upstream inflows into the reservoir $(HQUp.Q)$, the downstream released discharge $(HQDown.Q)$ and the reservoir actual evapotranspiration. This function ensures the mass balance in the surface water reservoir.

$$\boldsymbol{der\,(V) = HQUp.Q - HQDown.Q - A * evapotranspiration} \qquad (\text{eq. } 13)$$

The relation between water level and reservoir storage is explicitly defined by a look up table to represent the reservoir shape better. This relation provides better information for the overall mass balance calculation. The relation between the water level and the reservoir area also explicitly defined to calculate the reservoir evaporation. In RIBASIM, this relation is defined by a piecewise linearization that divides a nonlinear function into several linear sections. As these discontinuities often cause a problem for the optimization solver in RTC-Tools 2.0, the relation of water level and reservoir storage is defined as a function derived by B-Spline interpolation. Furthermore, a curve fit option can be specified in RTC-Tools 2.0 to ensure that the first and the second derivation of this function are always positive. Although the optimization solver in RTC-Tools 2.0 is able to handle non-linearity, a monotone function could substantially help the solver to find the optimal solution faster with slight modification to the physical representation.

The surface water reservoir covers the hydropower generation (P) in the model components. The energy generation is calculated from the relation of the released discharge via turbine gate ($Q\_turbine$) and the net head. The net head depends on the intake level, the friction losses and the tail level (H_tail)). The released discharge from the turbine gate can be utilized for the consumptive water demand. Adapting to the RIBASIM model, the efficiency plant load factor ($efficiency$) , gravity ($Modelica.Constants.g\_n$) and water density (rho) are taken into account.

$$\boldsymbol{P * 10^6 = Q\_turbine * (H - H\_tail) * Modelica.Constants.g\_n * rho * efficiency} \quad \text{(eq. 14)}$$

Channel
The connection between two or more objects represents an open channel, for example, a river or a canal. In this study, the connection ensures the mass balance between the connected objects and opens the possibility to model the losses in energy and water. This connection allows the water to flow in both upstream and downstream directions. Since the backwater event less likely presents in this case study, the minimum capacity of the open channel is set as 0 in the inviolable hard constraints.

5.4 Hydroeconomic valuations

Adapting to the RIBASIM-PS study, the economic benefit functions are formulated based on the agricultural delivered demand, the hydropower generation and the flood risk reduction. These functions formulate a single hydroeconomic objective function as a linear and convex optimization problem. The economic valuation of each function is expressed in 2010 US Dollars. A table of the summary of the hydroeconomic valuations applied in this study is presented for each section. In each of this table, the hydroeconomic valuations adapted from Van der Vat [1] are highlighted.

5.4.1 Downstream water demands

In this study, the fundamental demands that must be satisfied are not estimated economically. As shown in Table 5.2, the domestic water demand and environmental flow is set as the minimum constraints. On the other hand, the agricultural delivered demand is formulated as one of the hydroeconomic benefit functions.

Table 5.2 Summary of constraints and objective functions (water demand)

Water demand	*Constraints*	*Objective functions*	
	Quantity	*Benefit*	*Penalty*
	(m^3/s)	*(US\$/m^3)*	*(US\$ /m^3)*
Agriculture (van der Vat, 2015)	88 - 197	0.02	
- Paddy (October - May)	87 - 322	0.02	0.016
- Nuts (June - September)	8 - 190	0.043	
Drinking water	35.1	-	-
Environmental flow	1.4	-	-

5.4.1.1 Domestic water demands and environmental flow

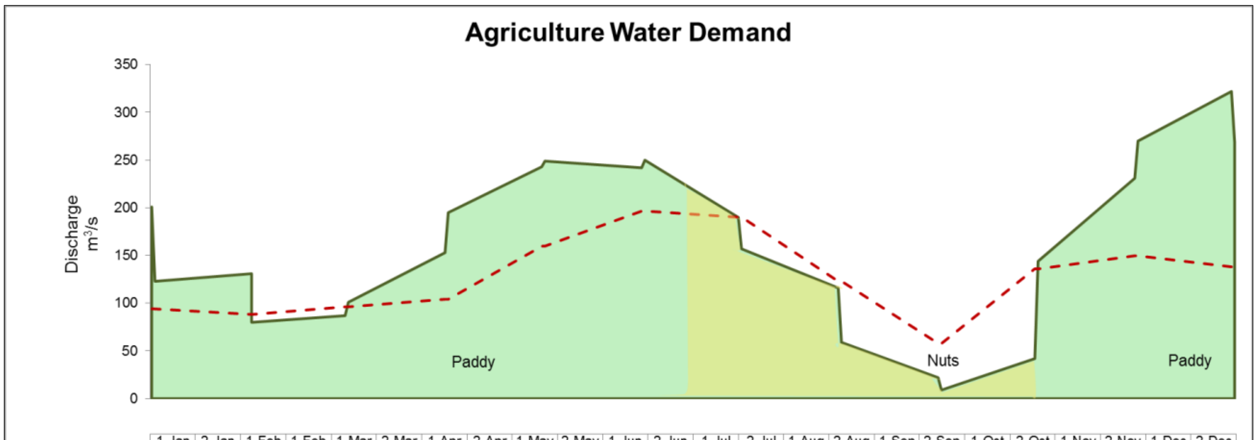
As shown in (eq. 15) and (eq. 16), the domestic water demand (Q_{dom}) is estimated based on the population and the water use per capita in the Citarum basin. The population in the basin in 2010 is around 16 million people and the average water use ($Q_{dom/cap}$) (I) is 190 l/capita/day [29]. The value of the average water use is reasonable for the city with more than 1 million inhabitants. In 2010, the estimation of the domestic water demand is 35.12 m^3/s. In 2010, the minimum environmental flow (Q_{env}) in the Citarum River is 1.4 m^3/s with a slight annual increment [30]. As the model constructed is a non-dynamic optimization model, both values are assumed to be similar for the whole period. Both domestic water and environmental flow demand are defined as priority discharge (Q_{prior}) that must be released.

$$\boldsymbol{Q_{prior} = Q_{dom} + Q_{env}} \qquad \text{(eq. 15)}$$

$$\boldsymbol{Q_{dom} = Q_{dom/cap} * I} \qquad \text{(eq. 16)}$$

5.4.1.2 Agriculture

The report published by the Indonesian MPW [29] stated that the total rice paddy field area is 348,704 Ha. The net agricultural water demand generated from the RIBASIM simulation model is presented in Figure 5.6. This figure shows that the value of the agricultural water demand shows comparatively greater quantity than the data used in the RIBASIM-PS study. The most likely cause of this difference is that the Indonesian MPW [29] performs more detail analysis on the cropping season. This difference could be helpful to capture the sensitivity of the firm rule curve to the agricultural water demand.



	1-Jan	2-Jan	1-Feb	1-Feb	1-Mar	2-Mar	1-Apr	2-Apr	1-May	2-May	1-Jun	2-Jun	1-Jul	2-Jul	1-Aug	2-Aug	1-Sep	2-Sep	1-Oct	2-Oct	1-Nov	2-Nov	1-Dec	2-Dec
🟩 Public Works	201	123	131	80	87	101	153	195	243	249	242	250	190	157	116	59	22	9	42	144	231	270	322	268
🟥 Dijkman et al	94	94	88	88	96	96	104	104	160	160	197	197	190	190	123	123	58	58	136	136	150	150	138	138

Figure 5.6 The agricultural water demand in the Citarum basin (Data source: MPW, 2012)

Paddy

As shown in (eq. 17) and (eq. 18), the irrigation benefit (B_{irri}^{i}) is defined as the economic benefit gained with the presence of reservoir operation (B_{res}^{i}). The agriculture delivered demand is valued (V_{irri}) as US$0.02/m³. The annual ($i$) benefit is calculated by adding the monthly (m) economic benefit of agricultural delivered demand (S_{irri}^{m}). This delivered demand must be less or equal to the agricultural water demand (Q_{irri}^{m}) .

$$\boldsymbol{B_{irri}^{i} = B_{res}^{i} - B_{nores}^{i}} \qquad \text{(eq. 17)}$$

$$\boldsymbol{B_{res}^{i} = \sum_{m=1}^{m=12} \min(Q_{irri}^{m}, S_{irri}^{m}) * V_{irri}} \qquad \text{(eq. 18)}$$

Van der Vat [1]

As the reduced amount of agricultural delivered demand may result in the economic loss to the farmers, the penalty function that represents the drought impact in the agricultural area is formulated. The drought penalty cost (D_{irri}^{paddy}) is obtained by defining the cost function as the average yield reduction. A report by International Rice Research Institute [48] concludes that the total drought reduces by nearly 80% of the yield ($R_{\%}^{paddy}$) over the entire season in the South and Southeast Asia. It provides an estimated value of US$0.016 /m³ as the agricultural drought penalty cost (P_{irri}^{paddy}). These relations are formulated in the (eq. 19) and (eq. 20). It is important to note that this method has not yet covered different drought damages based on the stage in the growing season.

$$\boldsymbol{P_{irri}^{paddy} = V_{irri}^{paddy} * R_{\%}^{paddy}} \qquad \text{(eq. 19)}$$

$$\boldsymbol{D_{irri}^{paddy} = \sum_{m=1}^{m=12} \min(0, Q_{irri}^{m} - S_{irri}^{m}) * P_{irri}^{paddy}} \qquad \text{(eq. 20)}$$

Secondary crops

To reduce significantly the water demand during the dry season, the cultivated crops are mostly replaced by the non-rice crops [49]. The usual secondary crop during the dry season in the Citarum basin are nuts, corn, soybeans or cassava. Retrieved from the official website of the Indonesian Ministry of Trade [50], the peanut price in September 2010 is IDR14,900/kg (US$1,030/ton). As a comparison, an overview of the rice market price in 2010-2011 provides an average rice price ($V_{paddy}^{US\$/ton}$) of US$475 /ton. Combining both values, the agricultural delivered demand in the secondary cropping season could be estimated as US$0.043 /m³. This relation is formulated in the (eq. 21). Since nuts are less prone to drought, the formulation of penalty cost is therefore considered as not necessary.

$$\boldsymbol{V_{irri}^{nuts} = \frac{V_{nuts}^{US\$/ton}}{V_{paddy}^{US\$/ton}} * V_{irri}^{paddy}} \qquad \text{(eq. 21)}$$

5.4.2 Hydropower generation

The economic benefit function of the hydropower generation depends on the water level and released discharged from the reservoir. In the RIBASIM-PS study, the constraints in the physical characteristics are directly defined in the RIBASIM model. In contrary, the physical characteristics are included as the hard constraints in RTC-Tools 2.0. The objectives and constraints related to the hydropower generation that must be specified in the RTC-Tools 2.0 are shown in Table 5.3.

Table 5.3 Summary of constraints and objective functions (hydropower generation)

Hydropower Characteristics	*Unit*	*Constraints and Objective functions*				
		Jatiluhur	*Cirata*	*Saguling*	*Peak load*	*Base load*
Benefit for generated energy (van der Vat, 2015)	US$ /MWh				66	32
Penalty on energy shortage	US$ /MWh				2x66	
Fraction of peak generation (van der Vat, 2015)	%	21	100	100		
Minimum level power generation	m+MSL	75	205	623		
Strategic level power generation	m+MSL	82	206.5	625		
Tail level hydropower plant	m+MSL	27	102	252		
Head loss	m	1	4	28		
Turbines efficiency	%	87	87	87		
Monthly firm energy demand	GWh	69.7	60	100		
Maximum turbines capacity	MW	187	700	1,008		

The economic valuation of hydropower generation is divided into peak load (V_{peak}, US$66/MWh) and base load ($V_{base}$, US$32/MWh). The operational and maintenance cost of the hydropower can be neglected since it is relatively small. The RIBASIM-PS study simplifies that the rest of the electricity produced is valued as a base load if the hydropower generation (E^m) produces higher electricity than the firm energy demand (D^m_{peak}). On the other hand, if the hydropower generates less electricity than the firm energy demand, the whole hydropower generation are valued as the peak load. As shown in the (eq. 22), the economic benefit (B^i_{power}) mainly depends on the actual hydropower generation. All of the power generated in Cirata and Saguling are valued as the peak load while Jatiluhur only produces around 21% of the peak load.

$$B^i_{power} = \sum_{m=1}^{m=12} \left[\left[\min\left(E^m, D^m_{peak}\right) * V_{peak} \right] + \left[\max\left(0, E^m - D^m_{peak}\right) * V_{base} \right] \right] \quad \text{(eq. 22)}$$

Van der Vat [1]

Some constraints on the hydropower generation aspects are described below:

Penalty on energy shortage

Cirata and Saguling reservoirs aim to fulfil the electricity demand peak on Java and Bali during 18.00 – 22.00 [39]. The drop in the firm energy demand in both reservoirs operation could bring about three hours power cut in several locations [51]. This shortage costs higher than the energy price itself because it could result in high economic losses, especially in the industrial production. In the case of this energy shortage, a high penalty value is assumed as two times higher than the peak energy value. The penalty value of US$ 132 /MW is applied to both upstream reservoirs when each of them generates lower electricity than the firm energy demand. On the other hand, most electricity generation in the Jatiluhur reservoir is valued as the base load since it mostly generates an extra energy. The penalty function is not necessary to be applied to this reservoir. The firm energy demands for each reservoir are 69.7 GWh for Jatiluhur, 60 GWh for Cirata and 100 GWh for Saguling.

Strategic minimum level
To deal better with the energy shortage, the reservoir should be operated higher than the strategic minimum water level [38]. The advised values on this water level are +82 m+MSL for Jatiluhur, +206 m+MSL for Cirata and +625 m+MSL for Saguling. These values are 2 m – 5 m higher than the minimum level of reservoir power generation.

Limiting electricity generation based on the turbines capacity
At some point, a higher potential water level could not produce more electricity since it is capped by the turbine capacity. The maximum turbines capacities are 700 MW for Saguling, 1,008 MW for Cirata and 187 MW for Jatiluhur [39].

5.4.3 Flood risk reduction

It is expected that the damage of the downstream flooding depends on the upstream released discharge. As summarised in Table 5.4, the RIBASIM-PS study applied a binary flood damage function. In this study, a linear flood damage function is formulated. The flood reduction benefit is calculated from the flood damage reduction after the application of the optimized reservoir operation rules.

Table 5.4 Summary of constraints and objective functions (flood damage)

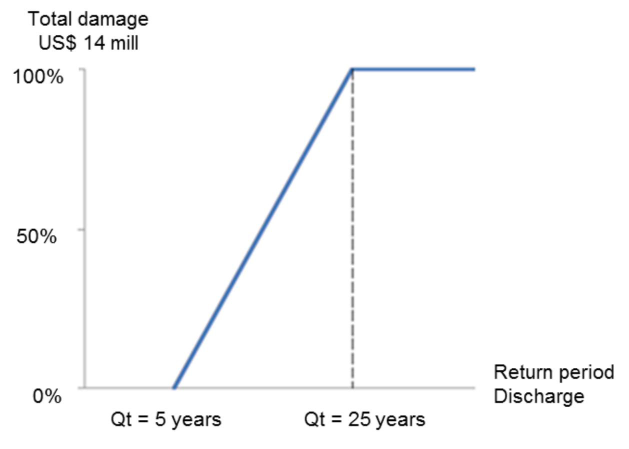
	Constraints	*Objective functions*
Flood reduction	*Discharge* (m^3/s)	*Damage* (*million US\$/m^3*)
Constant (van der Vat, 2015)	320	14
Linear Interpolation		
Q=Qmax agriculture	200	0
Q (van der Vat, 2015)	320	14
Qmax agriculture<Q	x	$\frac{x-200}{320-200} * 14$

As shown in (eq. 23), the economic benefit (B^i_{flood}) is obtained from the difference between the flood damage if the cascade reservoir is not constructed (D^i_{nores}) and the flood damage after the reservoir construction with the optimized reservoir operation rules (D^i_{res}). The flooding in the downstream area is expected if the released discharge from Jatiluhur reservoir (Q_j) passes 320 m^3/s. Although the released discharge is much higher than the threshold, the similar flood damage cost of US$ 14 million is applied. This value is based on the flood damage cost estimation for the affected household and agricultural area in the downstream cities of the Citarum basin.

$$\boldsymbol{B^i_{flood} = D^i_{nores} - \sum_{m=1}^{m=12} D^m_{res}\,(Q_j)} \qquad \text{(eq. 23)}$$

Van der Vat [1]

As depicted in Figure 5.7, a flood damage cost based on the index-based flood insurance is assumed to be linear in the Citarum Basin [52]. It is important to note that this study should estimate the social damage from the flood event apart from the insurance purpose; thus, this curve should be modified. The penalty function is presumed to be linear to the released discharge from Jatiluhur reservoir if it releases a higher discharge than the maximum monthly agricultural water demand. The starting point of the flood damage penalty function should be adjusted if the downstream water demand increases.



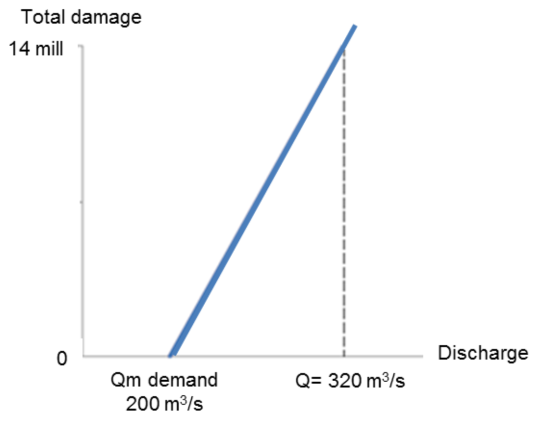


Figure 5.7 Index-based flood damage function (left) and flood social damage penalty function (right)

It should be realised that this method is a simplification since the flooded area depends on the water level of the downstream cross sections. Modelling a simplified hydrological routing could be a handy approach to analyse of the severities of the downstream flooding damage. This analysis is not carried out due to the monthly time stepping used in this study.

5.4.4 Applicability of optimized rule curves

Some additional constraints are applied to find more applicable reservoir operation rules. The optimized parameters are bound to the values based on the characteristics of physical infrastructures and the stability of the reservoirs. These constraints are presented in Table 5.5.

Table 5.5 Summary of constraints (rule curves applicability)

Rule curves applicability	*Unit*	*Constraints*		
		Jatiluhur	*Cirata*	*Saguling*
Physical characteristic				
Bottom gate level (Dead storage)	m+MSL	45	180	623
Spilling level	m+MSL	106.89	220	643
Volume at spilling level	McM	2,448	1,827	560
Live storage capacity (Turbine intake to spilling level)	McM	1,325	768	560
Minimum elevation for reservoir stability	m+MSL	87.5	-	-

The water level, area and discharge relations of the Citarum cascade reservoirs are slightly different in the various reports. These relations in Figure 5.8 are referred to the 2010 Standard Operation Procedures and its review [39].

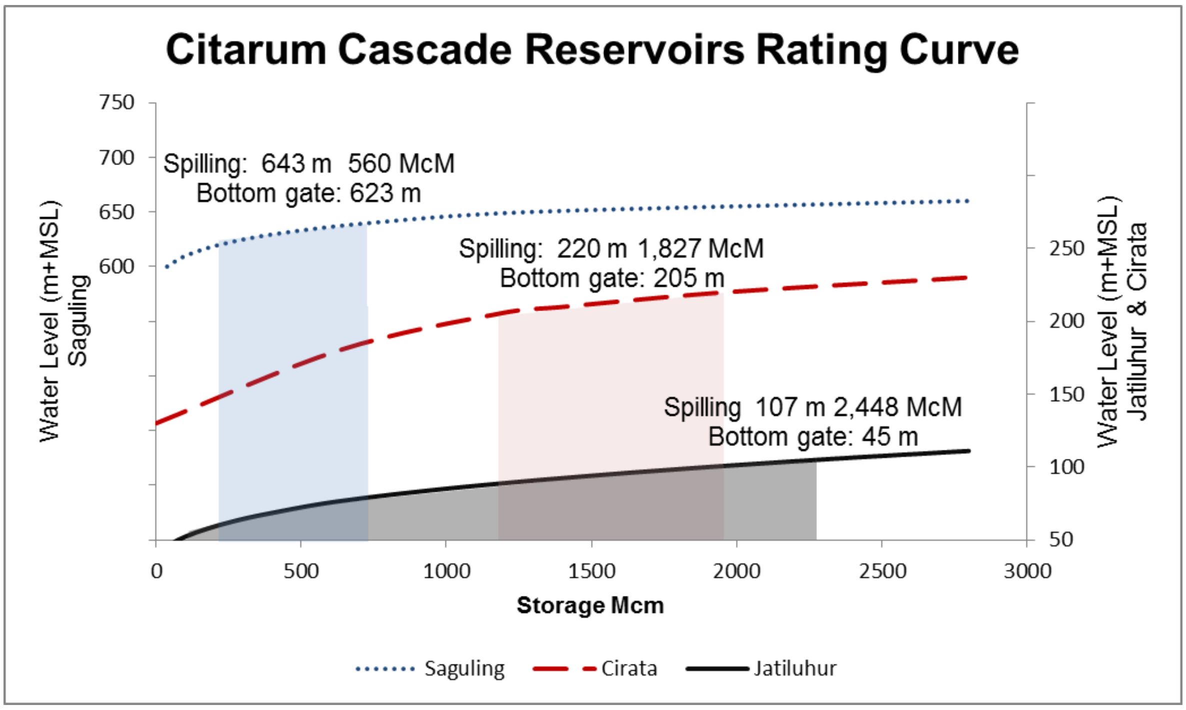


Figure 5.8 Citarum cascade reservoirs' storage and water level relation (Data source: Dijkman, *et al*, 2012)

Dijkman, *et al.* [39] found that there has been a reduction in the capacity of the reservoirs since its completion (60% for Saguling, 90% for Cirata and 75% for Jatiluhur). It could partly be explained by the high sedimentation in these reservoirs [53]. Furthermore, substantial differences on the storage and water level relation from the various reports are observed [39] [37] [38]. As this study conducts a deterministic approach, the uncertainty on the storage and water level relation is neglected.

To have a better applicability of the optimized rules, limiting the minimum water level in reservoirs is important to avoid the damages to the physical infrastructure. Both upstream reservoirs are relatively stable since they were made of rock fill dam. [25]. Since the Jatiluhur reservoir was made of a rock fill dam with an inclined clay core, the crack along the crest may occur if the water level is below a certain level. Srihadi, *et al.* [54] stated that the strategic minimum level for Jatiluhur should be higher than 87.5 m due to the dam stability. A clay core reservoir is well-known for its instability during the water level fluctuation. The rapid change in the hydraulic head often leads to the drawdown effect[6]. Since there is no specific limitation in the water elevation fluctuation, it is assumed that the Jatiluhur reservoir is always stable when the water level is higher than 87.5 m.

The Citarum cascade reservoirs are central of the freshwater fisheries in the basin. The fluctuation in water level has less influence on this activities due to the temporary floating structure used for fisheries. It is likely that the water level changes may have more influences on the fishing boats and recreational boats.

[6] Drawdown effect: A reduced stability in the upstream face of the reservoir due to the sudden drop of the water elevation [61].

5.5 Indonesian governmental policy directive

This section describes and discusses the development of the methodology based on the procedure carried out by the reservoir operators. The operators should implement the governmental policy directive no Pd T-21-2004-A regarding operation rules of the multi-purpose cascade reservoirs [55].

As presented in Table 5.6, some additional constraints applied in this section refer to the Indonesian governmental policy directive. Three alternatives of the reservoir operation rules are attained by processing the historical data that determines various hydrological years. These inputs are assessed separately with a trial and error in the NEDECO spreadsheet model resulting three different reservoir operation rules. These rules are further termed as (i) wet curve, (ii) normal curve and (iii) dry curve depending on the input data [37]. The most suitable curve for the reservoir operation rules is later decided based on the weather forecast.

Whereas the agricultural water demand must be set as a constraint in the NEDECO spreadsheet model, the RTC-Tools 2.0 provides more flexibility on defining the target demand. RTC-Tools 2.0 offers a possibility to assign agricultural target demand to one of the sequences of objectives in the goal programming while defining this target demand as a hard constraint is rather straight-forward.

Table 5.6 Summary of constraints (governmental policy directive)

Governmental directive constrain	*Unit*	*Constraints*		
		Jatiluhur	*Cirata*	*Saguling*
Equal sharing of effective volume	%	50	29	21
Minimum operational water level	m+MSL	87.5	206	623
Maximum operational water level	m+MSL	106.5	219.5	642.5

Equal sharing principle

The distributions of the total effective volume [7]in the Citarum cascade reservoir should be around 21% for Jatiluhur, 29% for Cirata and 50% for Saguling. This system is expected to avoid the involvement of the reservoir operators in the conflict of interest.

Annual deficit prevention

To avoid an annual water deficit within the Citarum cascade reservoirs, the water level at the end of the year (31 December 2000) is constrained to be greater or at least equal than the initial water level of the same year (1 January 2000). The evaluation of the derived rule curves from 2005 to 2010 reveals that the reservoir operation often violates this policy, especially during the dry year.

Operational constraints

For practical reasons, the operation rules for the cascade reservoirs are constrained by the operational water level. Retrieved from Figure 3.2, the operational water level for each reservoir must be between these elevations:

[7] Total effective volume: Reservoir live storage [55]

- Saguling : 623.0 and 642.5 m+MSL
- Cirata : 206.0 and 219.5 m+MSL
- Jatiluhur : 87.5 and 106.5 m+MSL.

5.5.1 Determination of various hydrological years

Adapting to the Indonesian governmental policy directive, a classic statistical method for inflow prediction is employed. The probability distribution *Log-Normal* and a simplified stochastic prediction ARIMA[8] are used to forecast the input. The applied inflow data is further selected based on the weather information from BMKG[9]. The definition of a hydrological year corresponds to a specific probability of exceedance shown in Table 5.7. The measured inflow data obtained from the current year observation are included in the statistical analysis for the next year inflow prediction.

Table 5.7 Hydrological year classification based on the probability of exceedance

Year Classification	*Inflow discharge probability of exceedance*
Very dry	<10%
Dry	10% - 40%
Normal	40% - 60%
Wet	60% - 90%
Very wet	>90%

As shown in the (eq. 24), the probability function of the Log-Normal distribution depends on the standard deviation (σ) and the mean (μ) of the logarithmic value of the time series data. This function is suitable for the analysis of the upstream inflow discharge since it always has a positive value. Figure 5.9 illustrates the probability distribution for Saguling historical inflow in January between 1920 and 2009.

$$f(x|\mu,\sigma) = \frac{1}{x\sigma\sqrt{2\pi}} e^{\left\{\frac{1}{}\frac{(\ln x-\mu)^2}{2\sigma^2}\right\}} \; ; x > 0 \qquad \text{(eq. 24)}$$

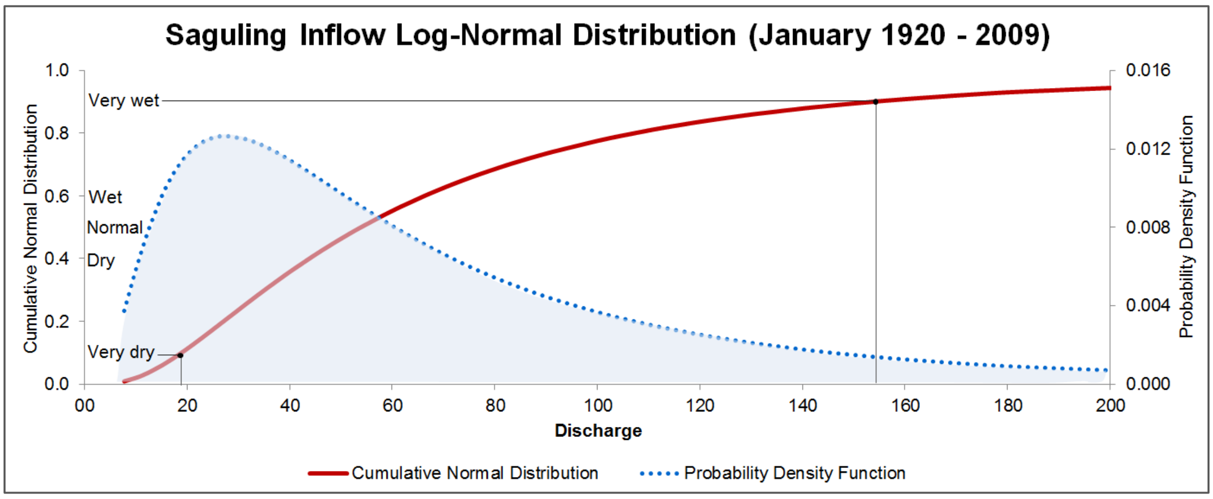


Figure 5.9 Log-Normal distribution for Saguling historical inflow (January 1920-2009)

[8] ARIMA: Autoregressive Integrated Moving Average
[9] BMKG: Indonesian Agency for Meteorology, Climatology and Geophysics

As shown in Table 5.8, the calculation based on the Indonesian government policy directive is adapted to obtain the expected monthly upstream inflow for three reservoirs. The cumulative normal distribution represents the probability of exceedance as previously discussed. A specific Log-Normal distribution function is generated for each specific month in each reservoir. It means that the analysis of the very dry inflow in January 2010 is derived from the Log-Normal distribution of January upstream inflow between 1920 and 2009. In particular, this statistical procedure is somehow problematic since the probabilities of the very dry months occur simultaneously in a year is very unlikely. As a consequence, the forecast upstream inflow as the main input in the optimization model may lead to an extreme, less practical optimization results.

This study found a marked difference in the input data between the NEDECO spreadsheet model and the data used in this study. During the dry year, the objectives tend to be easier to satisfy in the NEDECO spreadsheet model due to smaller difference between the monthly average inflow of the cascade reservoirs (RIBASIM: 77 m^3/s, NEDECO: 121 m^3/s) and the monthly average target demand (RIBASIM: 158 m^3/s, NEDECO: 137 m^3/s). For consistency reasons, the time series data used for the statistical analysis in this optimization study are extracted from the existing RIBASIM model.

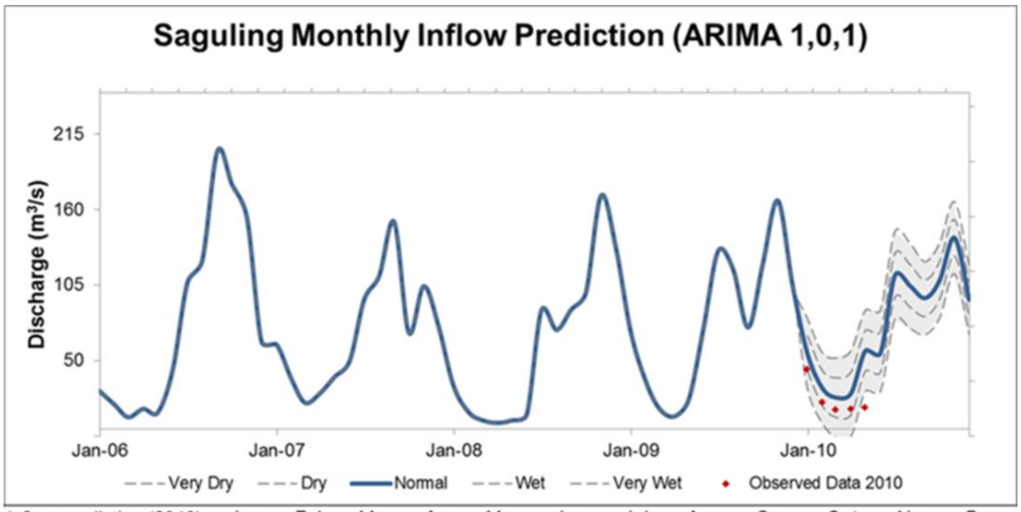
Table 5.8 Simplified stochastic inflow predictions (Log-Normal distribution)

Reservoir	*Classification*	*Monthly discharge (m^3/s)*											
		Jan	*Feb*	*Mar*	*Apr*	*May*	*Jun*	*Jul*	*Aug*	*Sep*	*Oct*	*Nov*	*Dec*
Jatiluhur	Dry	4.3	4.3	4.2	5.1	4.5	4.4	4.5	3.8	2.8	2.8	3.1	3.5
	Normal	8.9	9.3	9.8	11.6	10.6	10.1	9.9	9.0	6.7	6.4	7.2	7.5
	Wet	26.3	29.5	36.6	41.1	41.6	37.2	33.3	36.1	28.1	23.7	27.0	24.6
Cirata	Dry	34.7	36.3	40.6	47.2	45.3	40.2	35.8	41.2	37.5	35.1	29.1	32.7
	Normal	62.5	63.9	68.6	80.1	80.3	75.2	69.7	71.4	65.9	63.2	57.7	60.5
	Wet	138.7	135.6	135.6	160.2	173.2	179.1	178.9	147.7	139.5	139.8	154.4	141.0
Saguling	Dry	31.4	31.1	32.5	38.7	41.6	38.2	40.2	41.4	36.7	32.4	26.9	33.7
	Normal	53.9	55.3	58.6	70.0	74.5	71.9	74.3	72.5	64.1	58.5	52.3	56.4
	Wet	109.8	119.7	129.5	156.6	164.0	173.1	173.2	153.1	134.6	130.1	133.9	110.1

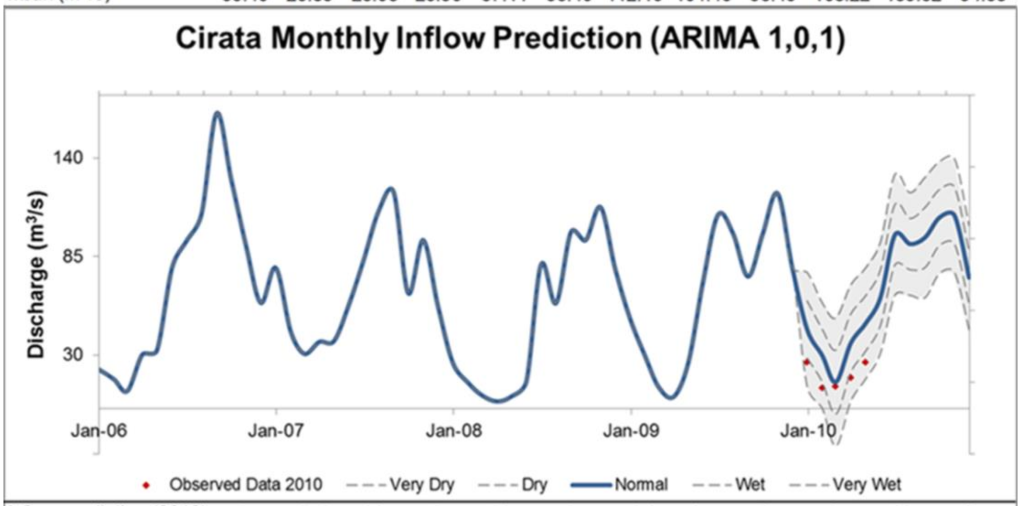
5.5.2 Simplistic stochastic forecast ARIMA

The Indonesian policy directive applies ARIMA (0,0,1) (Autoregressive Integrated Moving Average) as a stochastic approach to predict the inflow into the reservoir. Figure 5.10 present the result of the inflow time series analysis that has been done in the spreadsheet model produced in this study. The stochastic optimization analysis is not carried out since this study is limited to the deterministic model.

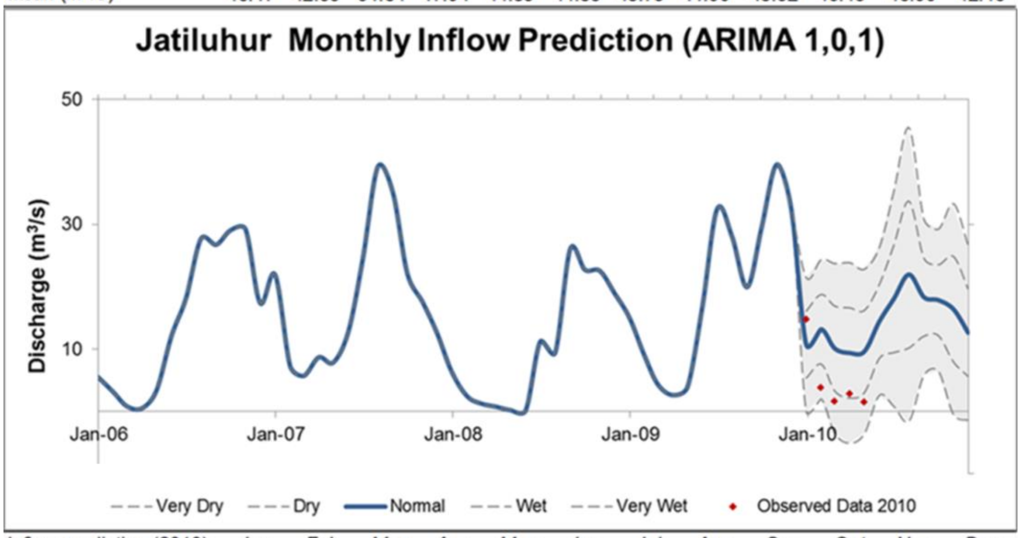
The 90 years data is applied to define the ARIMA model parameters. Moreover, the monthly data from the period 2000 and 2009 are used to forecast the upstream inflow in 2010. Although this stochastic forecast is not included in the optimization model, it provides practical information that this inflow forecast has a high level of uncertainty. The Citarum cascade reservoirs operator often addresses this issue by assessing the meteorological data to identify the best estimation of a single deterministic forecast value. An advance option is applying a stochastic optimization approach. While stochastic optimization is considered as a very expensive approach, it is a widely held view that it offers a better insight into more probable outcomes.



Inflow prediction (2010)	Jan	Feb	Mar	Apr	May	Jun	Jul	Aug	Sep	Oct	Nov	Dec
Standard deviation	36.98	37.18	41.64	45.03	42.88	42.88	46.36	45.04	38.51	39.50	38.03	36.64
Mean (m^3/s)	55.40	29.89	23.03	25.90	57.11	55.40	112.10	104.46	95.49	108.22	139.52	94.38



Inflow prediction (2010)	Jan	Feb	Mar	Apr	May	Jun	Jul	Aug	Sep	Oct	Nov	Dec
Standard deviation	44.35	29.87	14.97	36.74	47.55	62.07	97.23	92.07	95.69	106.91	107.97	73.33
Mean (m^3/s)	46.17	42.63	51.84	47.04	44.89	44.86	48.79	41.50	48.62	45.19	46.00	42.16



Inflow prediction (2010)	Jan	Feb	Mar	Apr	May	Jun	Jul	Aug	Sep	Oct	Nov	Dec
Standard deviation	10.72	13.17	10.08	9.38	9.61	14.45	18.08	21.95	18.31	17.84	16.38	12.64
Mean (m^3/s)	15.58	16.18	19.73	20.95	17.46	17.18	25.03	34.08	18.09	16.49	24.50	20.31

Figure 5.10 Stochastic inflow prediction (ARIMA (0,0,1))

6. SOFTWARE APPLICATION AND RESULT ANALYSIS

This chapter is divided into several sections; each section presents the practical application of constructing an optimization model in the RTC-Tools 2.0 and the result analysis for each of those models. Section 6.1 discusses the goal programming of the sequences of objectives derived from the existing model in RIBASIM. This section answers the first and second research questions about the practicalities of developing a similar model network as RIBASIM in the RTC-Tools 2.0. Section 6.2 focuses on the linear programming of the maximum economic benefit as a single objective function based on the RIBASIM-PS study. The sensitivity analysis of this hydroeconomic optimization model follows on the same section. The next following sections present the various optimization approaches in order to have a better applicability of the optimized reservoir operation rules than the RIBASIM-PS study. Section 6.3 explains an alternative approach called the hybrid optimization between the linear programming and the goal programming.

Section 6.4 gains insights about increasing the social benefits by modifying the hydroeconomic objective function. This reformulation is expected to be crucial for a pure holistic hydroeconomic study without any additional algorithm from the rule-based simulation model. Section 6.5 observes the changes in the reservoir operation rules if the downstream water demands are higher when the additional water demands are not quantified economically. The next Section 6.6 discusses the optimization approach adapted from the applications by reservoir operators in comparison to the water resources law. These different approaches are summarised in the Table 6.1. At last, Section 6.7 briefly summaries objectives and methodology before outlining the findings.

In this study, most data are taken from the existing RIBASIM model. The time series of upstream inflows into the reservoirs are exported between 1920 and 2009. In the same way, the monthly agricultural water demand and reservoir evaporation rate are exported; these data are assumed to be similar for each year in the simulation period as the changes in land use and the population growth is neglected. The bathymetries of the reservoirs are taken directly from the built reservoirs model in RIBASIM. The economic valuations are partly adapted from the RIBASIM-PS study whereas the penalties functions are obtain from the literature review. Since this optimization model is a non-dynamic model, discounting is taken.

The results from 1925 to 1935 are chosen as the representative period because it captures the high and low inflow into the reservoirs. In the inflow data pre-processing, the dry months (June-September) and the wet months (October-May) are separated. The typical year is chosen by comparing the average seasonal volume from the whole simulation period. The time series displays ten years results from the ninety years (1920-2009) simulation period. This chosen period comprises dry years (1925-1927), wet years (1931-1933) and typical year (1934).

Table 6.1 Software application of different optimization approaches (RTC-Tools 2.0)

Research question (s)	*Section (application and result)*	*Annual benefit for Citarum cascade reservoirs*	*Objective functions*	*Constraints*	*Hydroeconomic valuation*	*Simulation period*	*Optimization time horizon*
1 and 2	6.1	Goal Programming Sequences of hydrological objectives	Flood reduction Agriculture demand Firm energy	Infrastructure characteristics	-	90 years	90 year
3	6.2	Linear Programming A hydroeconomic objective	Maximum economic benefit	Infrastructure characteristics	van der Vat (2015)	90 years	90 years
4	6.3	Hybrid Optimization Sequences of hydrological objectives A hydroeconomic objective	Flood reduction Agriculture demand Firm energy Maximum economic benefit	Infrastructure characteristics	van der Vat (2015)	90 years	90 years
4	6.4	Linear Programming A modified hydroeconomic objective (Penalty functions)	Maximum economic benefit	Infrastructure characteristics	van der Vat (2015) & penalty functions	90 years	90 years
4	6.5	Linear Programming A modified hydroeconomic objective Additional hydrological constraints	Maximum economic benefit	Infrastructure characteristics Drinking water Environmental flow Reservoir stability	van der Vat (2015) & penalty functions	90 years	90 years
4	6.6	Goal Programming Sequences of hydrological objectives (Application by reservoir operators)	1. Drinking water 1. Environmental flow 2. Hydropower generation 2. Flood reduction 2. Agriculture demand	Infrastructure characteristics Reservoir stability	-	1 year	1 year
4	6.7	Goal Programming Sequences of hydrological objectives Sharing strategies (water resources law)	1. Hydropower generation 2. Drinking water 2. Environmental flow 3. Flood reduction 3. Agriculture demand	Infrastructure characteristics Reservoir stability	-	1 year	1 year

6.1 Goal programming of hydrological objectives (RIBASIM proxy)

This section focuses on the procedure for defining the sequences of objectives derived from the RIBASIM rule-based algorithm. This optimization is done without transforming any variable into an economic benefit function. In the RTC-Tools 2.0, the algorithm satisfies the highest priority goals first over the entire length of the time series. This procedure is markedly different to RIBASIM proxy that simulates at current and next time steps by an iterative procedure. For better comparison the optimization horizon of the RTC-Tools 2.0 model should be shortened, e.g. to two or three time steps to reduce the influence of knowledge on future inflows in the water system operation. In this case, a calendar year is considered as a practical optimization time horizon since the monthly inflow prediction is forecasted a year-ahead in Indonesia.

In this study, the goal programming approach has been done in two different time horizons, (i) whole time series optimization as a default setup in RTC-Tools 2.0 and (ii) a year optimization time horizon with the substantial help from the batch file. By integrating a batch file inside the source code of RTC-Tools 2.0, the goal programming can be run within the specified optimization time horizon. This has been done by repetitively cutting the long time series, initializing the parameters based on the previous optimization result, then optimizing the similar problem defined in the source code. The results of each optimization run are then combined into a single file.

In this study, the sequences of hydrological objectives are adapted from the algorithm in RIBASIM termed the rule curves and the hedging rules. While the rule curves are already described, Table 6.2 shows the hedging rules defined in the RIBASIM-PS study. A different percentage of the delivered target demand is determined after a specific storage zone in the reservoir is filled. This target demand entirely depends on the purpose of the reservoir. As was pointed out in the reservoir characteristics, the delivery target demands of both upstream reservoirs are the firm energy generation and the firm level fulfilment of the downstream reservoir. The Jatiluhur reservoir is the only multi-purpose reservoir that is responsible for satisfying the downstream water demands, generating energy and reducing the occurrence of the downstream flood events.

Table 6.3 presents the sequences of hydrological objectives that determine the water allocation in the network. This table is derived from the rule curves (see Section 5.1) and the hedging rules (see Table 6.2). As RIBASIM simulation routes the water from upstream to downstream, the priority on the upstream reservoir should be set higher than the downstream reservoir in RTC-Tools 2.0. In addition, the minimum capacity of the open channel has been set as 0 since RIBASIM is unable to model backwater effect. This ensures that the downstream inflow is not responsible for satisfying more upstream targets although they are set as the highest priority. Adapting to the existing RIBASIM model, the sequences of objectives below are derived based on the algorithm of online adjusted gate in RIBASIM (see Section 5.1). It is important to note that the priority setting should be different if the cascade reservoirs model disables this option. In that case, the targets of the downstream reservoir should be defined in the lower priority after all targets of the upstream reservoir are specified.

When deriving these sequences of objectives, care was taken to highlight the implicit priorities, such as the reservoir dead storage level. The upper and lower soft constraints are explicitly defined in the time series that are compiled in the comma separated file. These time series data, which may consist of a constant value for the whole simulation period, are taken directly from the existing RIBASIM model.

Table 6.2 The hedging rules of the Citarum cascade reservoirs model (RIBASIM)

Lower boundary of zone	*Saguling*		*Cirata*		*Jatiluhur*	
	% between Hfirm and Hdead	*% of target release*	*% between Hfirm and Hdead*	*% of target release*	*% between Hfirm and Hdead*	*% of target release*
Zone 1	80	90	80	90	80	100
Zone 2	60	70	60	70	65	100
Zone 3	40	50	40	50	55	90
Zone 4	20	30	20	30	45	50
Dead Level	0	10	0	10	0	0

Table 6.3 Sequences of hydrological objectives (RIBASIM proxy)

Priority	*Object*	*Objective function*	*Lower soft constraint*	*Upper soft constraint*
1	Cascade reservoirs	Water level (H)	Reservoir Hdead	Reservoir H full
2	Saguling reservoir (S)	Energy generation (P)	S: 10% FirmP	S: MaxP
3	Cirata reservoir (C)	Energy generation	C: 10% FirmP	C: MaxP
4	Cascade reservoirs	Water level	Hhedging4	Hflood
5	Saguling reservoir	Energy generation	S: 30% FirmP	S: MaxP
6	Cirata reservoir	Energy generation	C: 30% FirmP	C: MaxP
7	Jatiluhur reservoir (J)	Energy generation	J: 50% FirmP	J: MaxP
7	Agricultural terminal	Target released	50% Qagr	Qagr
8	Saguling reservoir	Water level	S: Hhedging3	S: Hflood
9	Cirata reservoir	Water level	C: Hhedging3	C: Hflood
10	Jatiluhur reservoir	Water level	J: Hhedging3	J: Hflood
11	Saguling reservoir	Energy generation	S: 50% FirmP	S: MaxP
12	Cirata reservoir	Energy generation	C: 50% FirmP	C: MaxP
13	Jatiluhur reservoir	Energy generation	J: 90% FirmP	J: MaxP
13	Agricultural terminal	Target released	90% Qagr	Qagr
14	Saguling reservoir	Water level	S: Hhedging2	S: Hflood
15	Cirata reservoir	Water level	C: Hhedging2	C: Hflood
16	Jatiluhur reservoir	Water level	J: Hhedging2	J: Hflood
17	Saguling reservoir	Energy generation	S: 70% FirmP	S: MaxP
18	Cirata reservoir	Energy generation	C: 70% FirmP	C: MaxP
19	Jatiluhur reservoir	Energy generation	J: FirmP	J: MaxP
19	Agricultural terminal	Target released	Qagr	Qagr
20	Saguling reservoir	Water level	S: Hhedging1	S: Hflood
20	Cirata reservoir	Water level	C: Hhedging1	C: Hflood
20	Jatiluhur reservoir	Water level	J: Hhedging1	J: Hflood
21	Saguling reservoir	Energy generation	S: 90% FirmP	S: MaxP
22	Cirata reservoir	Energy generation	C: 90% FirmP	C: MaxP
23	Saguling reservoir	Water level	S: Hfirm	S: Hflood
24	Cirata reservoir	Water level	C: Hfirm	C: Hflood
25	Jatiluhur reservoir	Water level	J: Hfirm	J: Hflood
26	Saguling reservoir	Energy generation	S: FirmP	S: MaxP
27	Cirata reservoir	Energy generation	C: FirmP	C: MaxP
28	Saguling reservoir	Water level	S: Htar	S: Hflood
29	Cirata reservoir	Water level	C: Htar	C: Hflood
30	Jatiluhur reservoir	Water level	J: Htar	J: Hflood

To optimize this model within 90 years as the time horizon, a default value of parameters (10^{-8}) is applied to specify the satisfaction tolerance and the constraints relaxation (see Section 5.2). These tight criteria might not be an issue if an optimization with a long time horizon is carried out since an extreme violation is likely to be distributed. Additionally, the ability of RTC-Tools 2.0 in assessing the whole time series input generally results in a long-term prevention of drought in the extreme dry year. On the other hand, these tight criteria might result in an unsolvable goal in the unseen extremely dry year due to the short optimization time horizon. To address this issue, the constraints relaxation is set as 10^{-5} and satisfaction tolerance is set as 10^{-2} for the runs conducted with a one year optimization time horizon.

When designing the model schematization, it was important to understand the physical representation of the reservoir in the case study so that an appropriate node in Modelica library is used. Figure 6.1 illustrates that the relation between reservoir water level and volume is reasonably linear. Therefore, the application of linear reservoir in the model schematization of Citarum cascade reservoirs might not considerably alter the optimization results. While the linear reservoir specifies a constant value of reservoir area, the look-up table reservoir presents a linear relation between water level and area. Although they have dissimilar physical representation, the difference between the evaporation demand in linear and look-up table reservoirs are insignificant compared to the overall mass balance in the system. Therefore, it seems that the application of linear reservoir could replace the look-up table reservoir in the model schematization of this case study.

To help the optimization solver in finding the solution faster, the monotonicity value need to be set in RTC-Tools 2.0 if the look-up table reservoir is used in the model schematization. Setting the monotonicity as 1 ensures a strict monotonicity or always increasing look-up table fitting curve. Although the improvement seems to be insignificant in this case, the small changes shown in the figure below (left) are crucial.

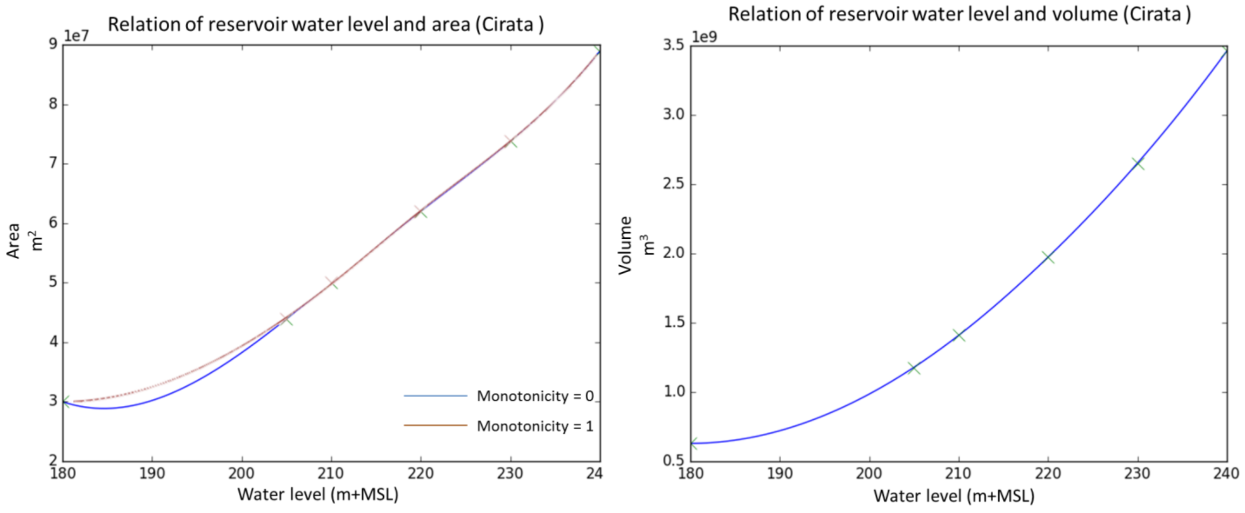


Figure 6.1 Monotonicity of look-up table fitting

Results
This section answers the first and second research question that modelling a similar network as RIBASIM in the RTC-Tools 2.0 is a possibility. It also comprises the differences between the reservoir operation rules derived from the RIBASIM simulation and the RTC-Tools 2.0 optimization. Firstly, the results of a year optimization time horizon are presented as the reproduction of the RIBASIM simulation. After, the following part of the section explains the results of the 90 year time horizon as the further step in optimizing the RIBASIM simulation.

Returning briefly to the subject of how the sequences of hydrological objectives are derived, the flood control curve is set as an upper water level soft constraint. Similarly to RIBASIM, this implies that the water level in the reservoir should not be higher than the flood curve. In order to gain a better understanding of the goal programming concept, the different sequences of objectives are applied. The supplementary results in a case when the upper flood control curves are removed are also presented in this section. This has been done for the optimization over the entire length of the time series. The reservoir operation rules of the Jatiluhur reservoir are presented since the other two reservoirs present insignificant differences in the results.

In general, the results from both models signify that the reservoir operation rules are almost similar during the wet and typical year but tend to be different during the dry year. The finding from this study suggests that the goal programming results in more frequent minor drought but significantly lower the severities of drought events. Specifically, the different optimization time horizon carried out in this study reveals that one year as the optimization time horizon results in the most comparable results with RIBASIM simulation, still, they tend to be different during the extreme dry year.

One year optimization time horizon
As shown in Figure 6.2, RTC-Tools 2.0 almost reproduces similar reservoir operation rules to RIBASIM simulation results if a shorter optimization time horizon is applied. The results tend to be more comparable during normal and wet years. A significant difference can be observed in the reservoir operation rules during the extremely dry years between 1925 and 1927. During this period, the water level drop is clearly visible, specifically in the Cirata reservoir. It seems possible that these results are due to the high satisfaction of agricultural delivered demand at the end of 1926. These low water levels at the end of 1926 are specified as initial states which greatly affect the optimization for the next year (1927). These initial states become more consequential as the upstream inflows during those years are insignificant. The reason for this different operation between both tools is not entirely clear. It is likely that the straightforward priority setting as applied in the RTC-Tools model does not precisely correspond to the behaviour of RIBASIM in a cascading reservoir situation with the agricultural demand.

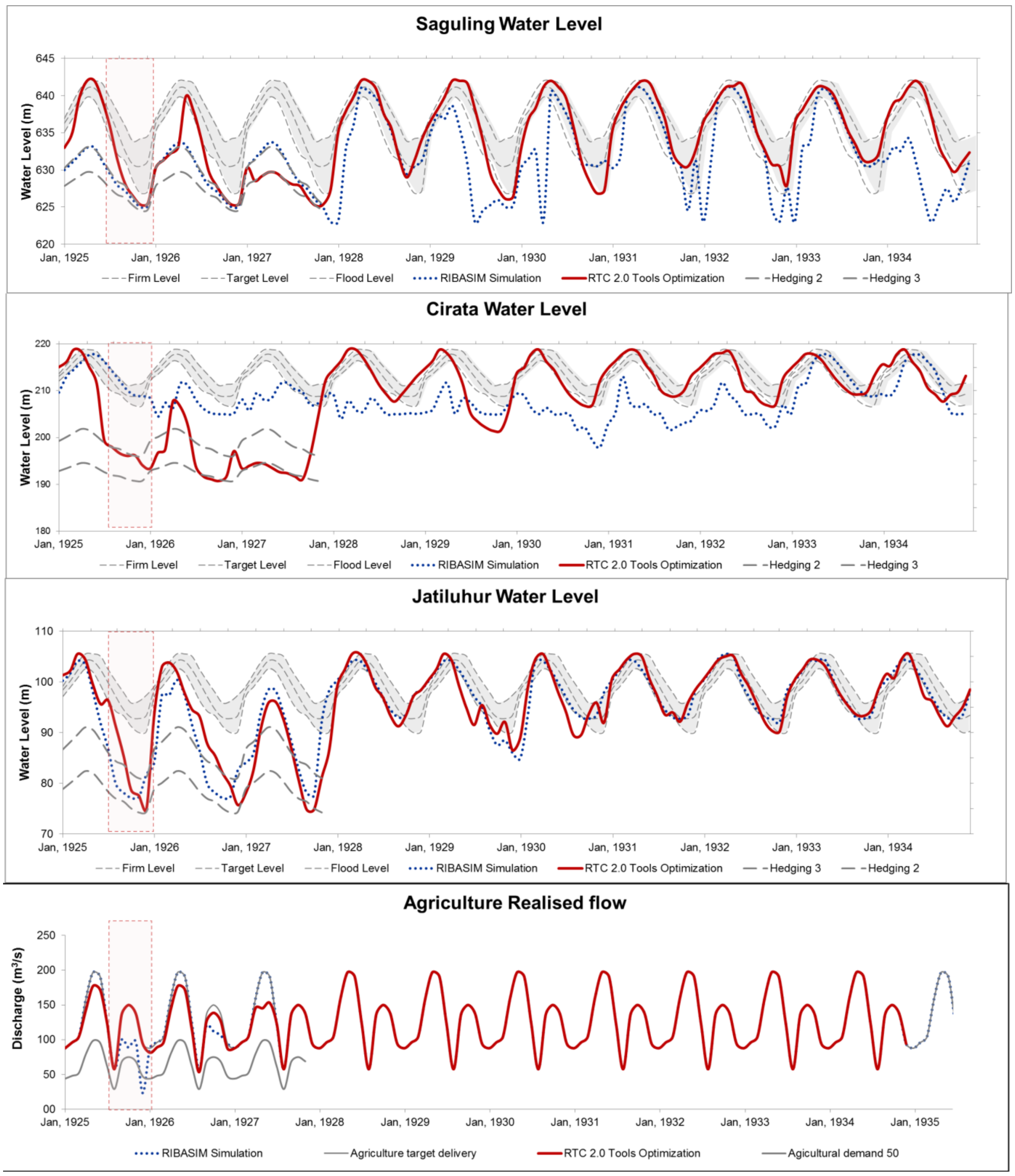


Figure 6.2 Reservoirs operation rules: Sequences of hydrological objectives (1 year horizon, RIBASIM proxy)

The present results are significant in at least one major respect; the reservoir operation rules in RTC-Tools 2.0 seem to be consistent with the priority level setting. This can be clearly seen at the end of 1926. Tracing the priority setting in Table 6.3, 90% of agricultural demand (13) should be delivered after the water level of hedging zone 3 in all reservoirs (8-10) have been fulfilled. After, the remaining water started to fill the level of hedging zone 2. It seems that this priority setting is dissimilar to RIBASIM algorithm. At the end of 1926, RIBASIM tries to fulfil the firm level of Cirata reservoir just after fulfilling 50% of agricultural water demand. Hence, it could conceivably be hypothesised that deriving roughly similar reservoir operation rules from both tools during the dry years is possible if the priority setting in RTC-Tools 2.0 is carefully adjusted based on the RIBASIM algorithm.

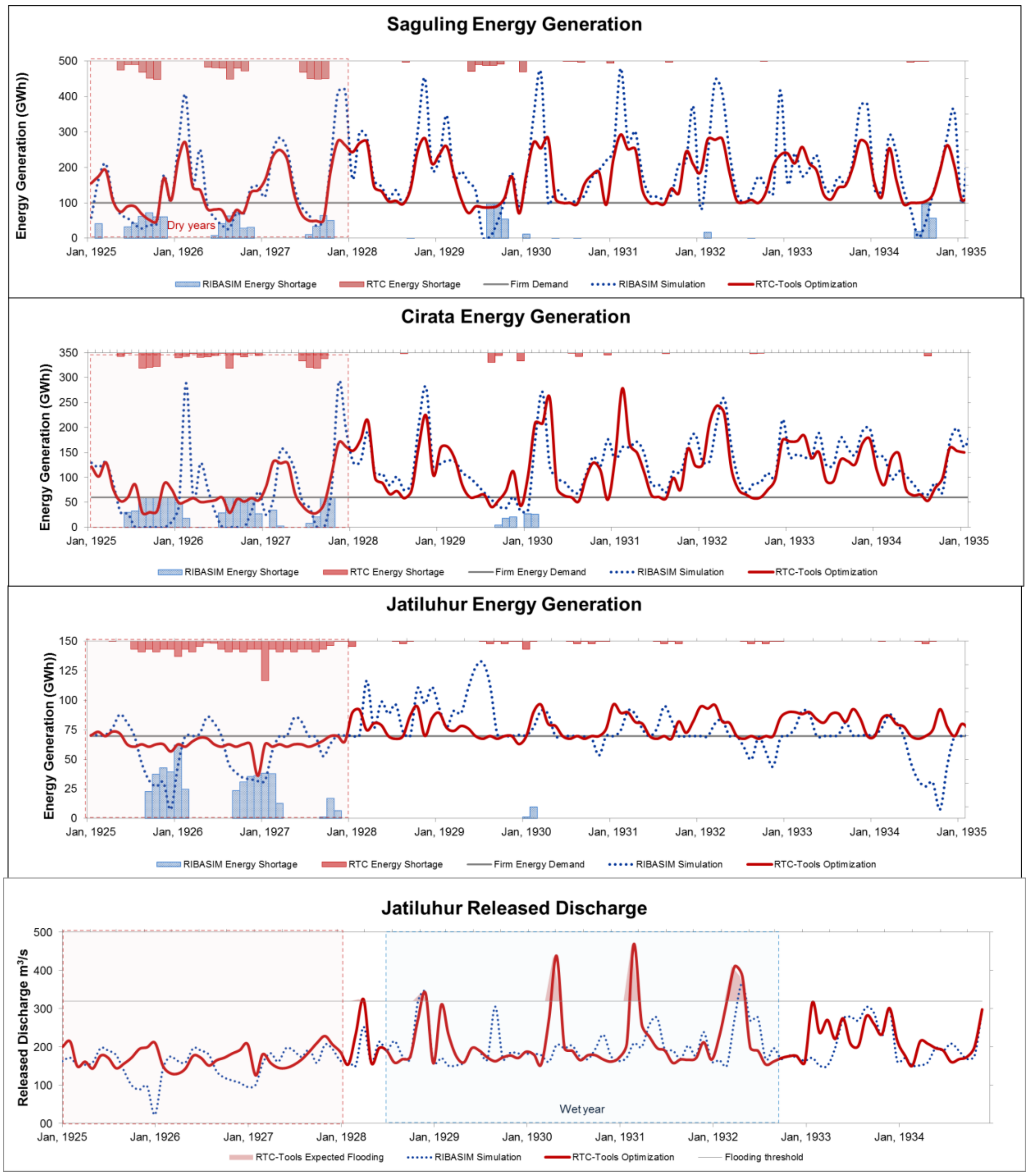


Figure 6.3 Target delivered demands: Sequences of hydrological objectives (1 year horizon, RIBASIM proxy)

The energy generation results in Figure 6.3 may help us to understand that the mass balanced concept is always satisfied in both tools. As the water level of cascade reservoirs are rather alike, the comparable energy generations imply that the released discharges from those reservoirs are roughly similar. These energy generation results also support the previous findings that RTC-Tools 2.0 tends to generate more frequent shortage but much less severity compare to the RIBASIM simulation even if one year optimization time horizon is carried out. This finding is similar to the agricultural delivered demand.

This figure also reveals that the RTC-Tools 2.0 results in more frequent flooding events, especially during the wet years. This can be easily prevented by adding a threshold of released discharge as additional a soft constraint. This result may be explained by the fact that the reservoir gate capacity is not explicitly specified in the RTC-Tools 2.0. In RIBASIM, some part of the released discharge is allocated to the next time step if it exceeds the gate capacity. Thus, it seems possible that these extreme released discharges in RTC-Tools 2.0 could be avoided by including the reservoir gate capacity into the optimization model.

Entire length of the time series optimization time horizon

As depicted in Figure 6.4, this study finds a slight difference in the Saguling reservoir water level between the results derived from both tools (Pearson product moment correlation coefficient=0.9). This finding is similar to the other parameters such as released discharge and energy generation. It can be seen that the optimization performed in RTC-Tools 2.0 prevents the energy shortage marginally better than the RIBASIM simulation. A possible explanation for this might be that the optimization problems in the Saguling reservoir do not involve any conflicting objective. In general, therefore, it seems that an optimization approach is considered as less necessary for a single purpose reservoir.

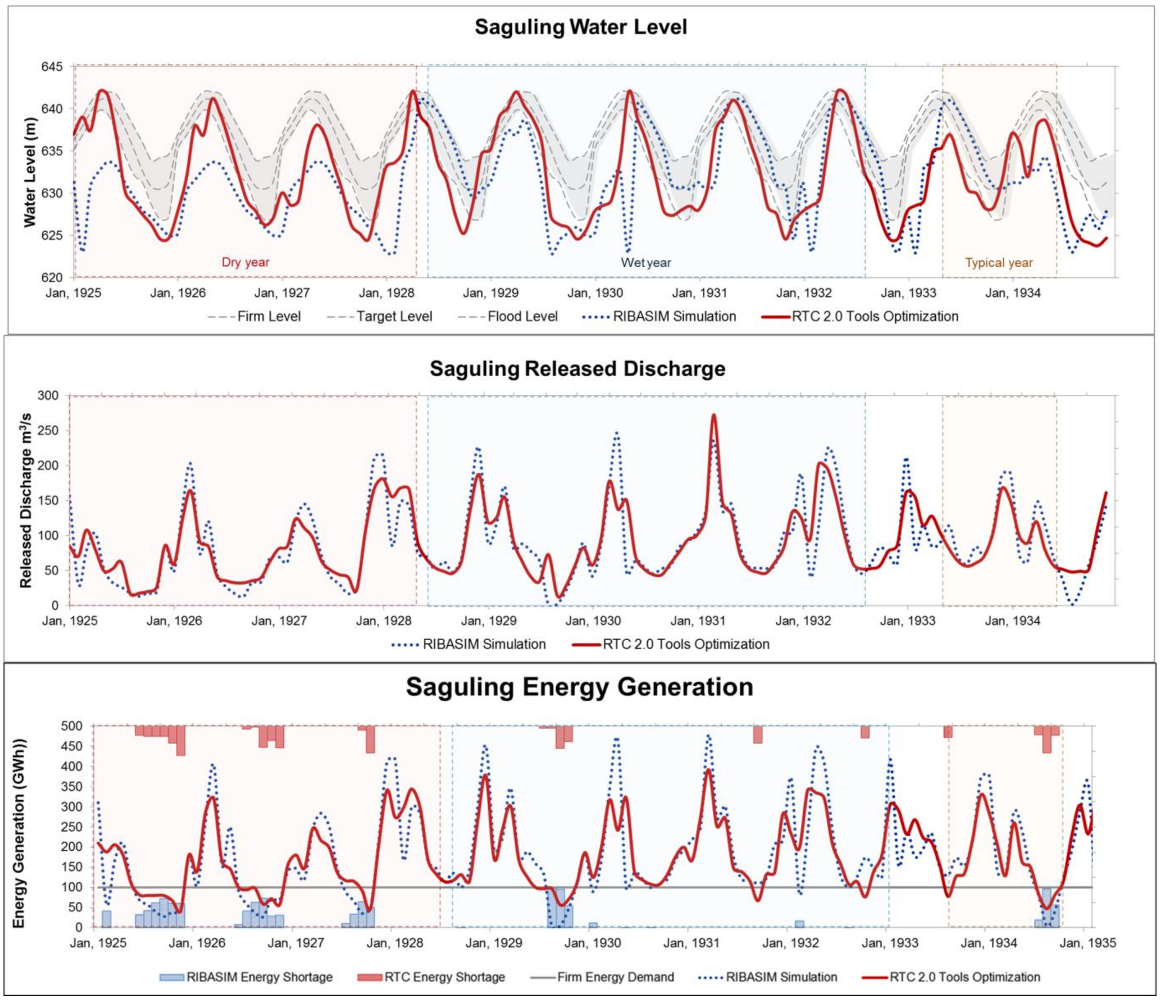


Figure 6.4 Saguling reservoir operation rules: Sequences of hydrological objectives (RIBASIM proxy)

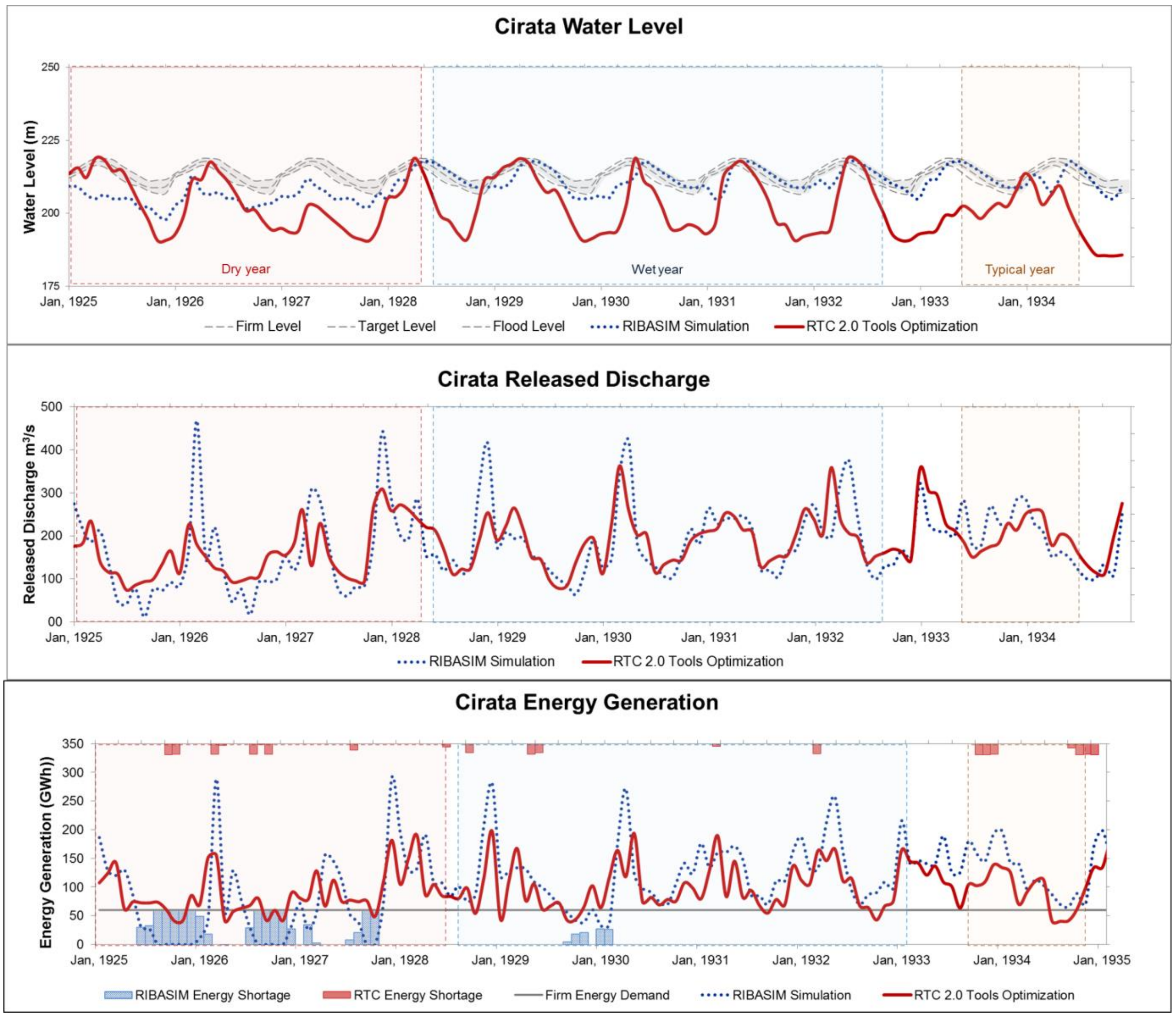


Figure 6.5 Cirata reservoir operation rules: Sequences of hydrological objectives (RIBASIM proxy)

As presented in Figure 6.5, the results from both tools show a considerably lower actual water level compared to the applied rule curves. Specifically, RTC-Tools 2.0 produces a lower, more fluctuated actual water level than the RIBASIM simulation. Unexpectedly, RTC-Tools 2.0 is able to lower the cumulative energy shortage in the RIBASIM simulation from 11% to 2%. This the energy shortage per event refers to the total of the energy shortage in each reservoir. It is assumed that an extra energy generated in a reservoir cannot cover the energy shortage in another reservoir.

These results are likely to be related to the difference between rules in both models during an extreme dry year. In the Citarum cascade reservoir simulation, RIBASIM strictly limits the target released discharge from the upstream reservoir to generate its firm energy demand and to fill the firm water level of the downstream reservoir. On the other hand, the RTC-Tools 2.0 optimization drives the water allocation based on the sequences of hydrological objectives. It leaves open the possibility of filling the downstream reservoir to its flood level from the upstream reservoir released discharge. The water level drop in the Cirata reservoir could be partly explained by its higher downstream released discharge during the dry period to keep higher water level in the Jatiluhur reservoir.

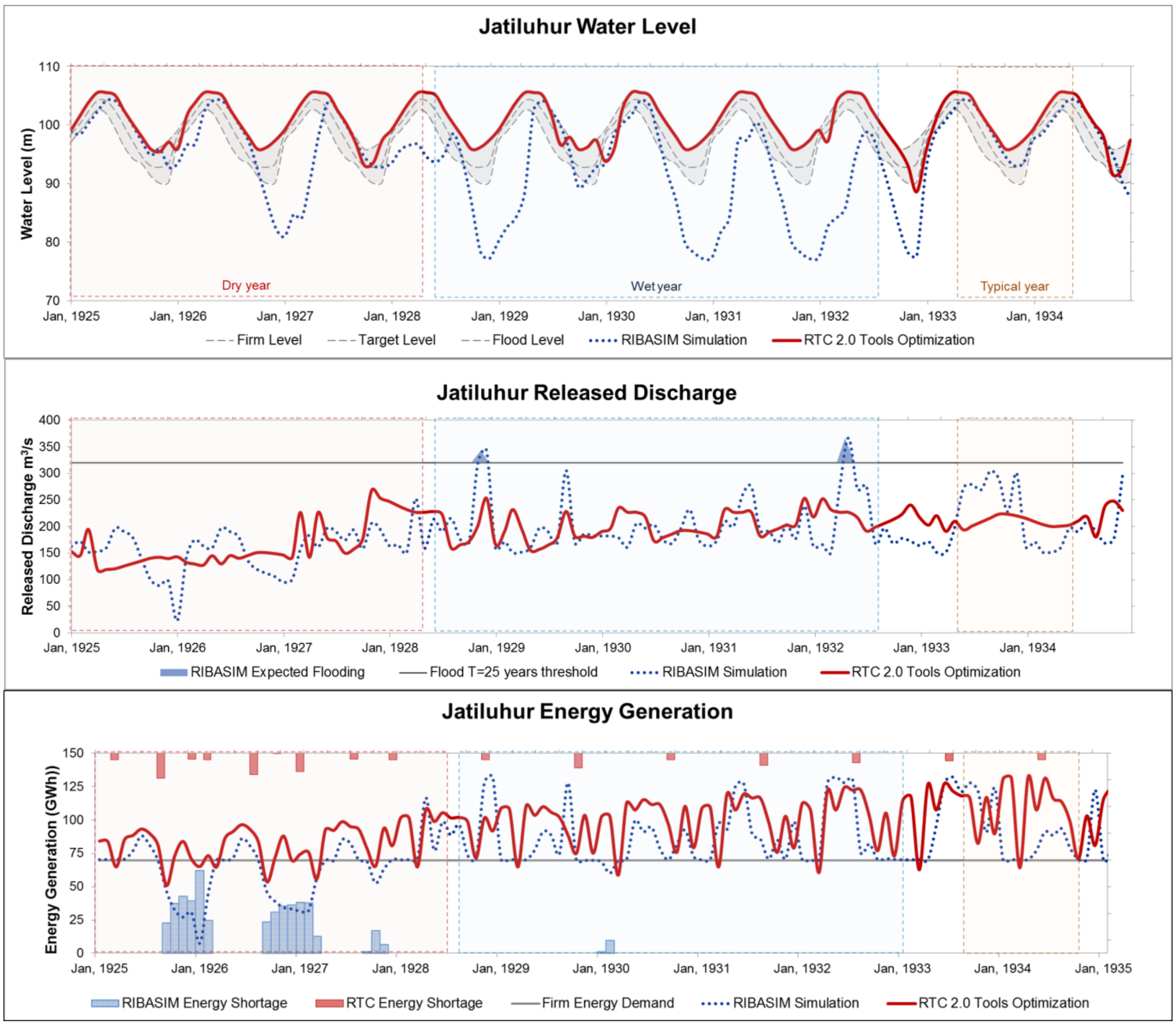


Figure 6.6 Jatiluhur reservoir operation rules: Sequences of hydrological objectives (RIBASIM proxy)

Figure 6.6 depicts the operation rules of the Jatiluhur reservoir. This figure shows that the water level is maintained to be higher in the RTC-Tools 2.0 in comparison with RIBASIM. This finding suggests that RTC-Tools 2.0 tends to store the excess water for the energy generation and the foreseen drought events. These are consistent with the findings from the reservoir released discharge results. Figure 6.6 also compares the released discharge for the Jatiluhur reservoir between the results from both tools. Whereas RIBASIM releases a great amount of water during the high upstream inflow, RTC-Tools 2.0 tends to release lesser discharge. This most likely reduces a serious downstream flooding occurrence (the peak released discharge of the Jatiluhur reservoir reaches 320 m^3/s). The results present that the downstream flooding is considerably reduced from 0.3 months/year to 0.1 months/year after the reservoir operation rules derived from the RTC-Tools 2.0 are applied.

This figure reveals that there have been several steep falls in the energy generation from the RIBASIM simulation. In contrast, the RTC-Tools 2.0 optimization is able to reduce cumulative energy shortage from 11% to 4%. As the energy generation is a function of the water level and the released discharge, the most likely cause of energy generation drop is the declined in the reservoir water level in RIBASIM simulation.

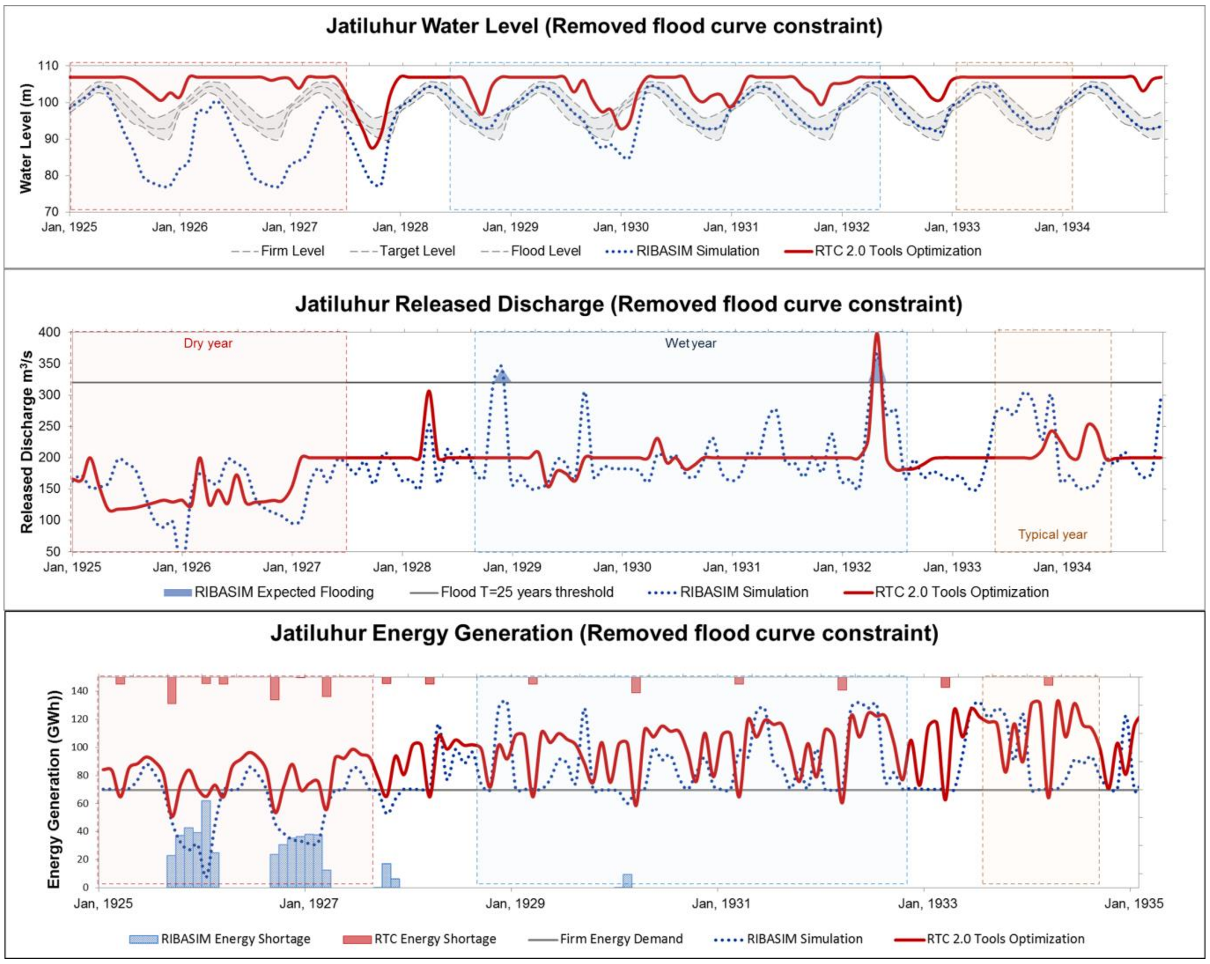


Figure 6.7 Jatiluhur reservoir operation rules: Sequences of hydrological objectives (no flood curve)

This study found some differences in a case when flood control curve as the upper soft constraint is removed from the RTC-Tools 2.0 optimization. However, both upstream reservoirs perform fairly similar. As the removal of the upper flood control curve allows the reservoir water level reaches a higher level, the results of the optimization model show a minor decrease in the energy shortage of both upstream reservoirs.

As shown in Figure 6.7, dissimilarity is observed from the water level of the Jatiluhur reservoir. This figure shows also that the water level is maintained to be even higher in the RTC-Tools 2.0 if the upper constraint is removed. Since the dam height is set as a hard constraint, the reservoir tends to keep the water level at the full reservoir level. It tends to store more water during the wet season and release relatively higher discharge during the dry season. Combining these factors, RTC-Tools 2.0 is able to lower the cumulative energy shortage in the RIBASIM simulation from 11% to 2%. It is important to highlight that the released discharge rarely passes the flooding threshold although the dam height is set as the upper constraint. A possible explanation for this is that the firm energy demand is fulfilled better with the low-risk trade-off with the flood reduction. These results suggest that further optimization of the rule curves could be carried out, but this is considered as less necessary since the multi-objective functions can be defined explicitly in the study.

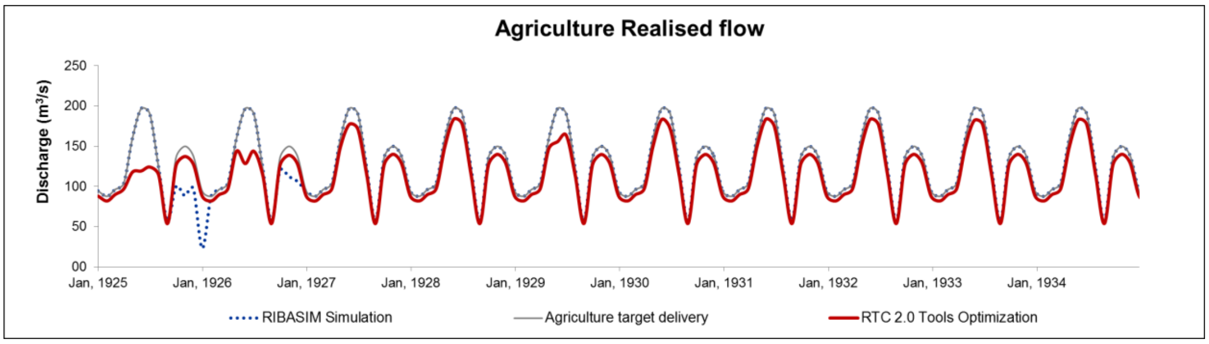


Figure 6.8 Agricultural delivered water demand: Sequences of hydrological objectives (RIBASIM proxy)

Having explained the supplementary results, this section now moves on to discuss the results of the agricultural delivered demand when the flood control curve is set as upper soft constraint. Figure 6.8 shows almost similar results of the agricultural delivered demand derived from both tools. The RTC-Tools 2.0 optimization tends to deliver relatively less water in comparison with the RIBASIM simulation (1.1 months/year, 41 m^3/s), resulting more frequent minor droughts (12 months/year) but anticipating the future extreme droughts (5 m^3/s). This may be partly the consequences of the substantial trade-off between the agricultural drought and the energy shortage in the Jatiluhur reservoir. This minor agricultural drought can be prevented by putting the agricultural water demand as a higher priority than the energy generation.

These results further support the idea of the goal programming approach functionality on a multi-purpose reservoir. Collectively, these results are in accord with the previous findings indicating that the goal programming approach is able to derive more promising reservoir operation rules in comparison to RIBASIM simulation. Furthermore, a longer optimization time horizon is likely to result in better reservoir operation rules. It could be argued that the positive results were due to the ability of the RTC-Tools 2.0 in assessing the whole time series input for each goal. This implies that this optimization method is too theoretical since the perfect knowledge of the hydrological forecast is less likely to be available for this long duration of time. These results, therefore, need to be interpreted with caution.

6.2 Linear programming of a hydroeconomic objective

This section discusses the procedure for defining a single hydroeconomic objective function in the RTC-Tools 2.0. As shown in Table 6.4, the economic valuations in this optimization model are fully adapted from the RIBASIM-PS study. The economic valuations are declared inside the Modelica script. Prior to this, care was taken to transform the variable units into the standard international units. The optimization model requires some critical transformation in the value shown in Table 6.4.

- The agricultural economic benefit of 0.02 US$/$m^3$ is transformed into 0.02/10^6 million US$/$m^3$
- The hydropower economic benefit of 66 US$/MWh is transformed to 66,000/10^6 million US$/GWh. As the standard international unit of the power is Watt, this unit must be integrated and divided by 3600.10^9 to be transformed to the value of energy generation in GWh.

- The flood penalty value of 14 million US$/month must be divided by the number of second in each month to obtain the unit of million US$/second.

It is also important to note that the standard international unit of time derivative (second) is used although the optimization model runs in a monthly time stepping. The maximum total economic benefit as a single objective function is valued in million US Dollar.

Two different approaches could be taken to solve the hydroeconomic optimization problem (i) a pure linear programming of hydroeconomic objective function and (ii) a linear programming with the additional soft constraints in the goal programming. In this study, a pure linear programming of hydroeconomic objective function is applied with the practical assistance from RTC-Tools 2.0. In this approach, the water allocation in the network is determined by the optimization of an objective function without any additional algorithm provided by a rule-based simulation model. The constraints and the initial conditions are taken directly from the existing RIBASIM model. RTC-Tools 2.0 is set to optimize the parameters for the whole simulation period.

Table 6.4 Hydroeconomic objective function (Van der Vat, 2015)

Hydroeconomic optimization (van der Vat, 2015)	*Unit*	*Jatiluhur*	*Cirata*	*Saguling*	*Others*	*Remarks*
Initial condition: Reservoir water level	m+MSL	98.8	215	624		
			Constraints			
Reservoir water level						
Minimum	m+MSL	45	180	623		Dead level
Maximum	m+MSL	106.89	220	643		Spilling level
Turbines capacity						
Maximum	MW	187	1,008	700		Physical constraints
Environmental flow	m^3/s				0	Neglegted
Drinking water demand	m^3/s				0	Neglegted
			Objective functions			
			Benefit			
Agriculture	US$/$m^3$				0.02	0.02 10^{-6} millionUS$/$m^3$
Agriculture demand	m^3/s				88 - 197	
Hydropower						
Peak	US$/MWh				66	66 10^{-3} million US$/GWh
Rest	US$/MWh				32	32 10^{-3} million US$/GWh
			Penalty			
Agriculture	US$/$m^3$				0	
Flood reduction						
Q>320 m^3/s	nillion US$/month				14	14 $(30.24.60.60)^{-1}$ million US$/s
			RIBASIM target released (not applied in RTC-Tools 2.0)			
Firm energy demand	GWh/month	69.7	60	100		
Agriculture demand	m^3/s				88 - 197	

Results

The determination of the optimal reservoir operation rules as identified by RTC-Tools 2.0 are different from the rules decision resolved from the RIBASIM-PS study. The results of the RIBASIM-PS study are the optimum annual rule curves including the flood level, the target level and the firm level of the three reservoirs. Each of the rule curves has a unique rule as specified in the RIBASIM algorithm (see Section 5.1). These rules further determine the quantity of the released discharge if a single value of the upstream inflow discharge for each time step is inputted. On the other hand, RTC-Tools 2.0 derives directly a release discharge in a time step as one of the results of the hydroeconomic optimization approach. Similarly, both tools employ the simultaneous mass balanced concept to quantify the actual water level from the released discharge. Thus, the actual water level of the reservoir derived from both tools is a suitable variable to compare.

To compare the results from both models, the historical average of the ninety years simulation period for each specific month is calculated. The time series result of agricultural delivered water demand and hydropower generation are also presented to perceive how both optimization models deal with a certain condition.

As depicted in Figure 6.9, the RTC-Tools 2.0 optimization results mostly show slightly lower actual water level, less than a meter in each reservoir. Another observed trend is that the monthly average actual water level in the RIBASIM-PS optimization mostly follows the optimized target curve. Both trends are dissimilar between August and November on the Saguling reservoir. In this period, RTC-Tools 2.0 generates nearly 10 meters higher actual water level. In RIBASIM-PS optimization, the actual water level falls under the optimized firm curve. This result shows that the water level suddenly drops at the end of time step in January. However, the water level in this period is expected to be higher since it represents the wettest month. These RIBASIM-PS results are rather confusing since the direct the relation between the average reservoir water level and the average monthly inflow is unclear.

The RTC-Tools 2.0 optimization generally calculates lower water level during the dry season compared to the RIBASIM-PS optimization. While the beginning of the dry season has a direct impact on the most upstream reservoir, a time lag is observed on the downstream reservoirs. It is recognized that the water level drops the latest in the Jatiluhur reservoir. A possible explanation for this might be that in the beginning of the dry season, the downstream reservoir still receives a high release discharge from its upstream reservoir in addition to its low local inflow.

The RIBASIM-PS study concludes that the hydroeconomic optimization considerably increases the annual economic benefit in comparison with RIBASIM simulation. In general, RTC-Tools 2.0 linear programming generates almost US$ 10 million higher maximum economic benefit in comparison with the RIBASIM-PS optimization. While the reservoir operation rules derived from RTC-Tools 2.0 successfully prevent the agricultural drought events, it performs poorly on the energy shortage reduction. Although the agricultural water demand and the firm energy generation are not explicitly defined in the RIBASIM-PS optimization, both target demands are managed by the RIBASIM algorithm. Figure 6.10 shows that this optimization approach performs very well compared to the result of the rule-based simulation in RIBASIM (see Figure 6.6). In 1927, the RIBASIM-PS optimization is able to completely eliminate the agricultural drought and to ease the severity of energy shortage.

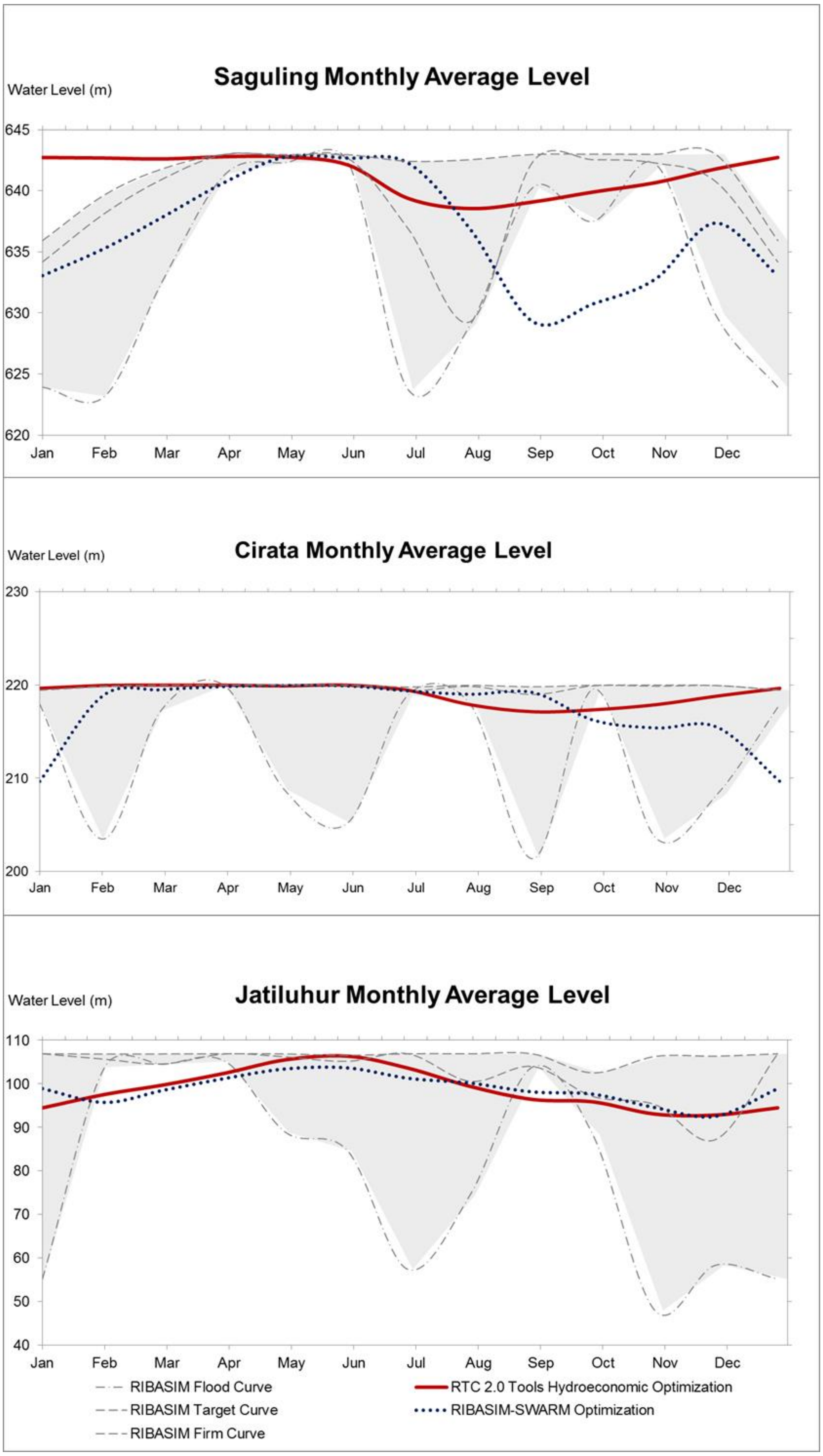


Figure 6.9 Actual water level of the reservoirs: Hydroeconomic objective

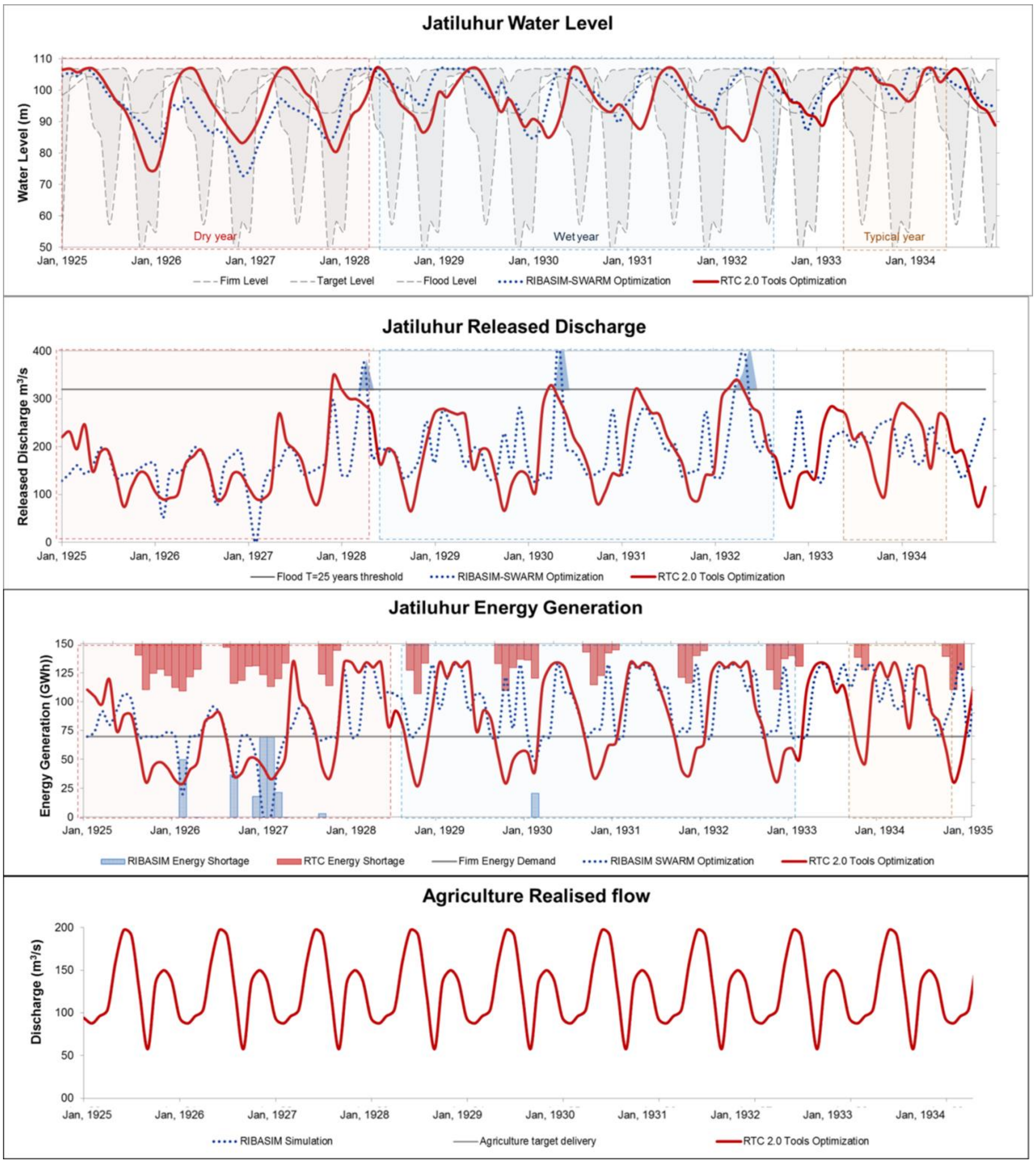


Figure 6.10 Delivery targets: Hydroeconomic objective

Figure 6.10 implies that the RIBASIM-PS optimization handles the critical dry period better, especially in the energy shortage reduction in comparison with the RTC-Tools 2.0 optimization. The RIBASIM rule-based algorithm highlights that the water demands and the firm energy demands are subjected to the target released discharge. As the firm energy demands are not explicitly defined in the RTC-Tools 2.0, the hydropower generation often drops below the firm energy demand whereas the agricultural water demand is always fulfilled. It may be the case that the economic valuation of agricultural delivered water demand is relatively dominant compared to the economic valuations of other objectives.

To reduce the frequency and the severities of this unfavourable event, the target demand should be comprised in the optimization model. This has been managed by combining the particle SWARM optimization with the rule-based simulation tools (RIBASIM). In the RTC-Tools 2.0, these target demands could be represented by (i) including the hydrological soft constraints in the goal programming or (ii) adding the penalty functions as part of the objective function.

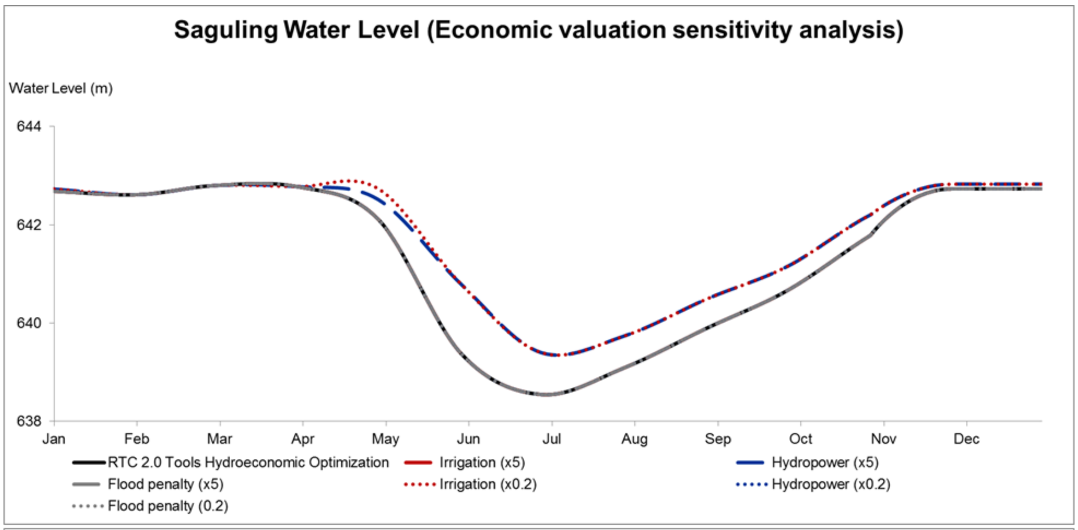
Sensitivity analysis
As pointed in the literature review, a pure hydroeconomic optimization model is expected to be sensitive to the economic valuations. A sensitivity analysis is one of the methods of addressing this issue. By running several optimization models with different values of the parameters, the changes in the water allocation in the network could be analysed if the economic valuation differs from what was previously assumed.

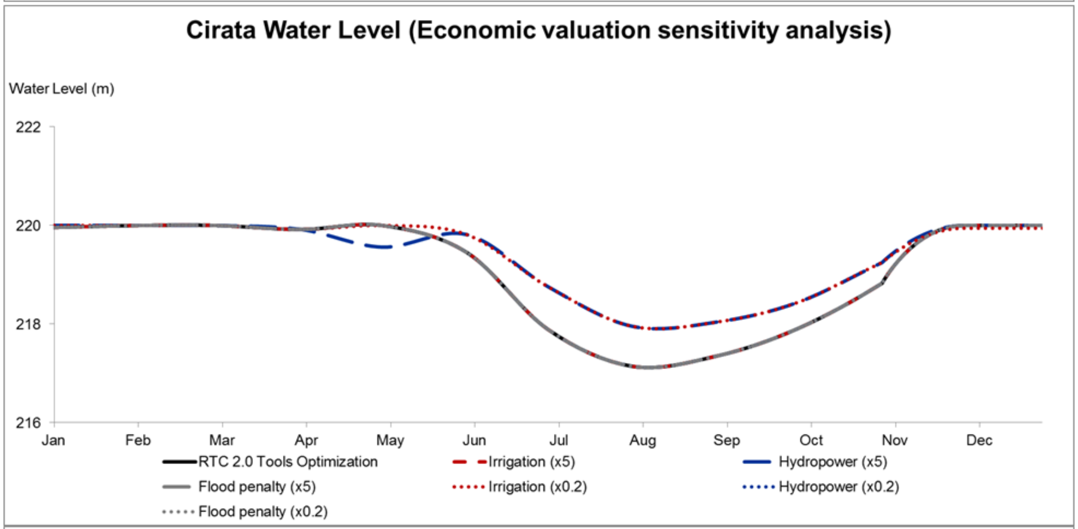
The sensitivity analysis is conducted by modifying the economic valuation appraised by Van der Vat [1]. By multiplying and dividing the economic valuation by the factor of 5, six different strategies are run. They cover the changes in the economic valuation of agricultural delivered water demand, the hydropower generation and flood damage reduction. The optimization results obtained from each strategy are later compared to the optimization results from the baseline strategy. This sensitivity analysis is exclusively conducted for a pure linear programming of hydroeconomic objective function.

As depicted in Figure 6.11, the sensitivity analysis to the economic valuation on the actual water level is almost similar for each reservoir. In general, the results from various scenarios suggest that the water allocation in the network alters along with the changes in the economic valuation. In contrast, the changes in the flood economic valuations have insignificant impact on the model; this formulation seems to be irrelevant in this hydroeconomic optimization. This might be partly explained by the ability of the RTC-Tools 2.0 to optimize with the perfect knowledge of the future events. Thus, applying the hydrological soft constraint or reformulating the economic valuation of flood damage is considered as necessary in RTC-Tools 2.0. However, this economic valuation is fundamental for the RIBASIM simulation since the released discharge from the Jatiluhur reservoir cannot be constrained explicitly.

An interesting finding is that a negligible difference is observed when the economic valuation of hydropower generation is decreased and the economic valuation of agricultural delivered water demand is increased. As both changes tend to deliver more water to the agricultural area, adjusting the reservoir operation rule is not a possibility since the agricultural water demand is always fulfilled. This result is in line with those of previous findings that the economic valuation of agricultural delivered water demand is relatively dominant.

The reduction in the economic valuation of agricultural delivered water demand and the increased in the economic valuation of hydropower generation tend to be the sensitive parameters in this model. By the factor of 5, these changes result in nearly two meters higher water levels of the three reservoirs. It seems that the water level of the Jatiluhur reservoir is the most sensitive to the changes on the economic valuations. It might partly be explained by the fact that the trade-off between the agricultural water demand and the energy generation is the most dominant in this reservoir. This figure also suggests that the economic valuation becomes more sensitive during the dry season. It signifies that the necessary trade-off between the conflicting objectives becomes more substantial in the case of water scarcity.





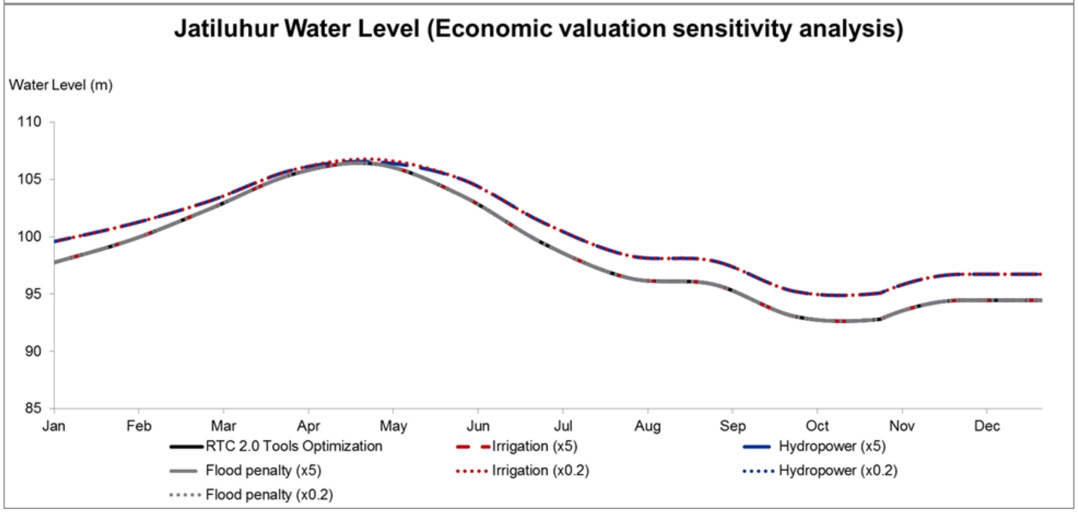
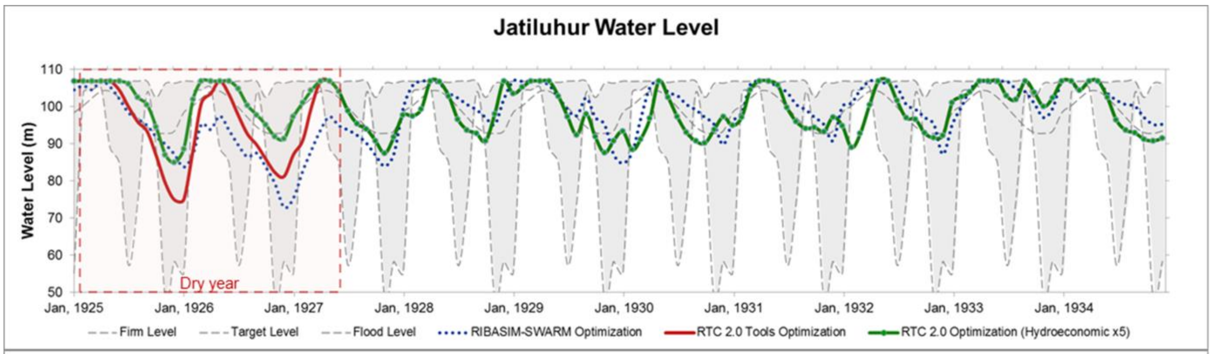


Figure 6.11 Actual water level of the reservoirs: Sensitivity analysis of hydroeconomic model



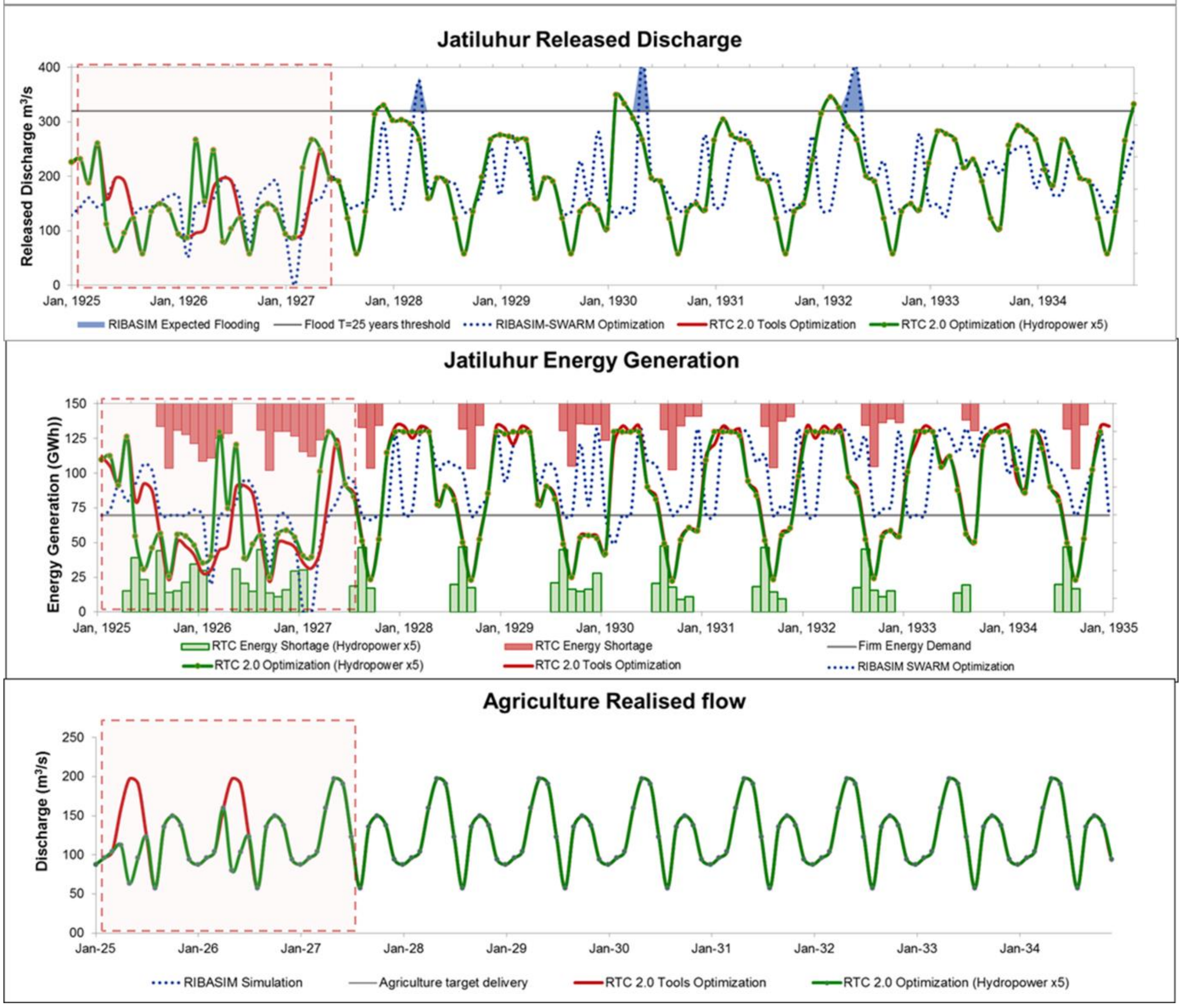


Figure 6.12 Delivery targets: Sensitivity analysis of hydroeconomic model

As presented in Figure 6.12, the agricultural delivered demand is likely to be affected by the increased economic valuation of hydropower generation, especially during the dry year. RTC-Tools 2.0 mostly keeps the reservoir water level higher in order to gain a higher benefit from the energy generation although the energy shortage is less likely to be improved. In general, a higher economic valuation of hydropower generation results in more energy generation, which further diminishes the total energy shortage but could result in the agricultural drought. This finding is similar when the economic valuation of agricultural delivered water demand is reduced. Therefore, it could conceivably be concluded that the economic valuations, in general, are the sensitive parameters on the hydroeconomic optimization model.

6.3 Hybrid optimization of hydroeconomic and hydrological objectives

As particle SWARM optimization is combined with the rule-based simulation RIBASIM, care was taken to highlight the RIBASIM algorithm that affects the results of the optimization model. An alternative approach to compromising the social benefits while maximising the economic benefits is by running the combination of a linear programming and the goal programming in the RTC-Tools 2.0. Similarly to the role of the RIBASIM algorithm on the particle SWARM optimization, these soft constraints assist the RTC-Tools 2.0 in optimizing the sequences of hydrological objectives while it searches for the highest economic benefit at the same time.

In RIBASIM, the water demand and the electricity firm demand are subjected to the target released discharges. These could be interpreted as the soft constraints in the goal programming approach. To obtain a comparable result with the RIBASIM-PS study, the domestic water demand and environmental flow are not taken into account. In addition, the maximum total economic benefit can be set as an objective function in the linear programming approach. By applying this hybrid optimization approach, preventing drought, flood event and energy shortcut can be considered as a higher priority than generating a maximum total economic benefit. As presented in Table 6.5, this hybrid optimization is formulated as a single integrated problem in the RTC-Tools 2.0.

Table 6.5 Hybrid objective functions

Hybrid optimization	*Unit*	*Jatiluhur*	*Cirata*	*Saguling*	*Others*
Sequences of hydrological objectives					
1. Goal Programming Firm energy demand (lower soft constraints)	GWh/month	69.7	60	100	
1. Goal Programming Agricultural water demand	m^3/s				88 - 197
1. Goal Programming Flooding threshold (upper soft constraint) (Maximum agricultural demand)	m^3/s	200			
Hydroeconomic objective					
2. Linear deterministic Maximum total economic benefit (van der Vat, 2015)					

Results

The consequence of applying these soft constraints is that, in general, it may generate lower maximum benefit. This optimization generates US$ 3 million lower economic benefits compared to the total economic benefit generated from the pure linear programming approach.

The addition of the goal programming considerably enhances the social benefit, especially in reducing the total cumulative energy shortage. As presented in Figure 6.13, the hybrid optimization approach remarkably improves the social benefit compared to the pure linear programming approach. Assigning the firm energy demand to the lower soft constraints reduces the total cumulative energy shortage from 11% to 2%. This optimization also slightly reduces the occurrence of downstream flood event from 0.3 months/year to 0.16 months/year. Although the agricultural drought always occurs during the whole simulation period, the value is negligible since it is relatively very small (1 m^3/s).

It is possible to hypothesise that this hybrid optimization provides an attractive alternative to the pure hydroeconomic optimization. This approach has a distinctive advantage since the soft constraints could be applied directly without any economic valuation. Applying these soft constraints is likely to ameliorate the robustness of the hydroeconomic optimization model since they are independent of the economic valuations. The concept of sequences of hydrological objective leads to more transparent water allocation in the hydroeconomic model. In addition, this hydroeconomic model becomes more flexible to the changes in priorities since the target demands are explicitly defined and ordered in the sequences of hydrological objectives.

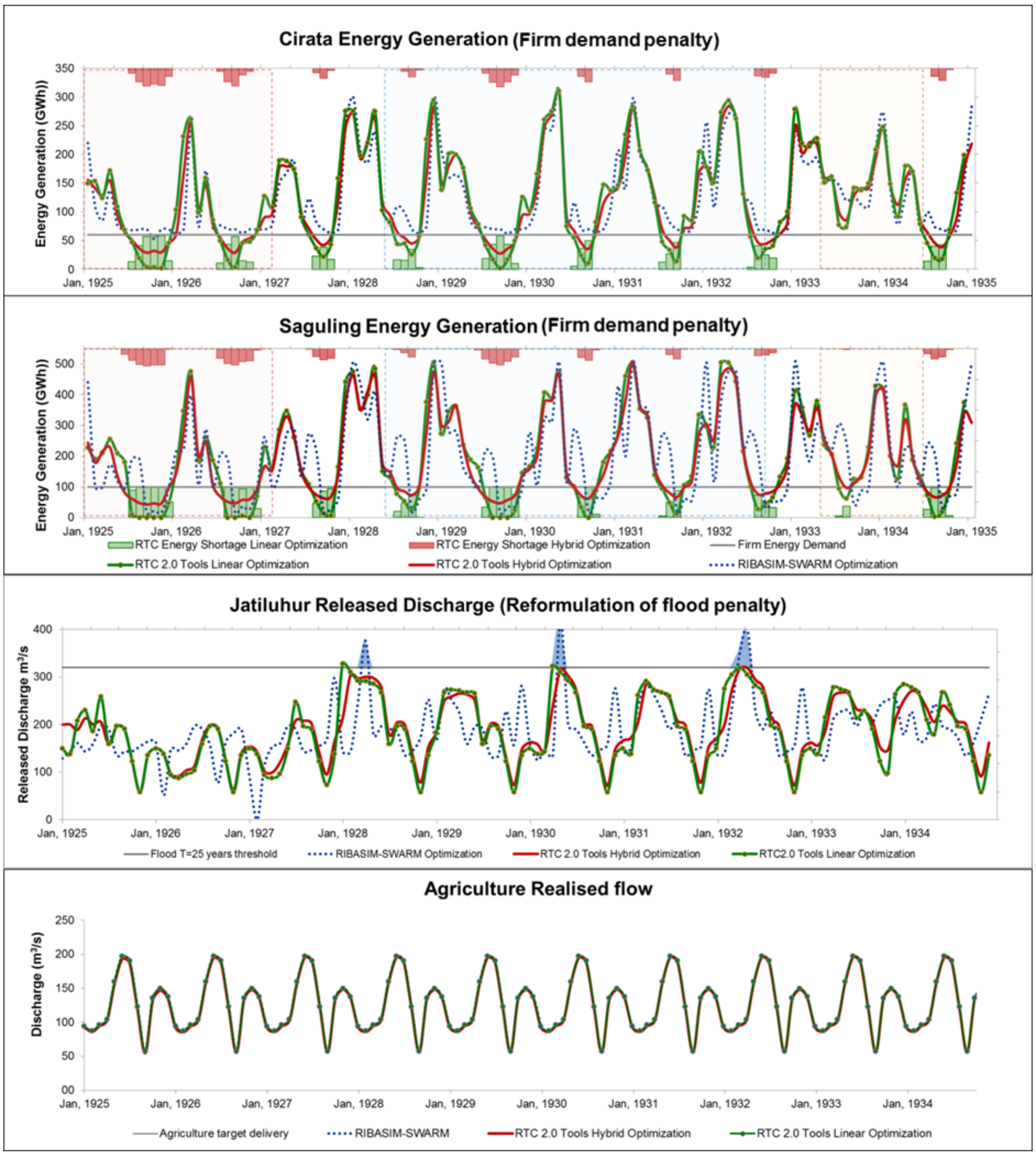


Figure 6.13 Delivery targets: Hybrid objectives

6.4 Linear programming of a modified hydroeconomic objective

The hydroeconomic objective function adapted from Van der Vat (2015) should be adjusted if a pure linear programming approach is chosen. This section discusses the optimization of a single modified hydroeconomic objective function in the RTC-Tools 2.0. A pure linear programming approach is carried out in this hydroeconomic model. Table 6.6 presents a modified hydroeconomic objective function that comprises some additional penalty functions discussed in Section 5.4. The high values of penalty functions are presumed to properly represent the importance of the social responsibilities. The expected results of this hydroeconomic optimization model can be partly related to the previously conducted sensitivity analysis.

Table 6.6 Modified hydroeconomic objective function

Hydroeconomic optimization	*Unit*	*Jatiluhur*	*Cirata*	*Saguling*	*Others*	*Remarks*
Initial condition: Reservoir water level	m+MSL	98.8	219	625		RIBASIM Model
			Constraints			
Reservoir water level						
Minimum	m+MSL	87.5	180	620		Stability
Maximum	m+MSL	106.89	220	643		Spilling level
Turbines capacity						
Maximum	MW	187	1,008	700		Physical constraints
Environmental flow	m^3/s				0	Neglegted
Drinking water demand	m^3/s				0	Neglegted
			Objective functions			
			Benefit			
Agriculture benefit						Water footprint
Paddy (November - June	US$/$m^3$				0.02	0.02 10^{-6} millionUS$/$m^3$
Nuts (July - October)	US$/$m^3$				0.043	0.043 10^{-6} millionUS$/$m^3$
Agriculture demand						MPW (2012)
Paddy (November - June	m^3/s				88 - 197	
Nuts (July - October)	m^3/s				8-190	
Hydropower						van der Vat (2015)
Peak	US$/MWh				66	66 10^{-3} million US$/GWh
Rest	US$/MWh				32	32 10^{-3} million US$/GWh
			Penalty			
Agriculture						IRRI (2006)
Paddy (November - June	US$/$m^3$				0.016	0.016 10^{-6} millionUS$/$m^3$
Nuts (July - October)	US$/$m^3$				0	
Hydropower						
Peak	US$/MWh				2x66	132 10^{-3} million US$/GWh
Flood reduction						Muin (2015)
Q<200 m^3/s					0	
Q=320 m^3/s	million US$/month				14	14 $(30.24.60.60)^{-1}$ million US$/s
Q<320 m^3/s					Linear interpolation	

Results

Compared to the result of the pure hydroeconomic optimization model, applying these penalty functions reduces the total economic benefits from US$ 378 million to US$ 370 million but substantially enhance the social benefit. Figure 6.14 shows a visible reduction on the energy shortage after applying the high penalty function (two times of peak demand, US$ 132/kWh) when the hydropower generation is lower the firm energy demand. The penalty function applied to both upstream reservoirs greatly diminishes the energy shortage even during the dry year. By contrast, this penalty function is irrelevant for the Jatiluhur reservoir since its energy generation is considered as an extra benefit.

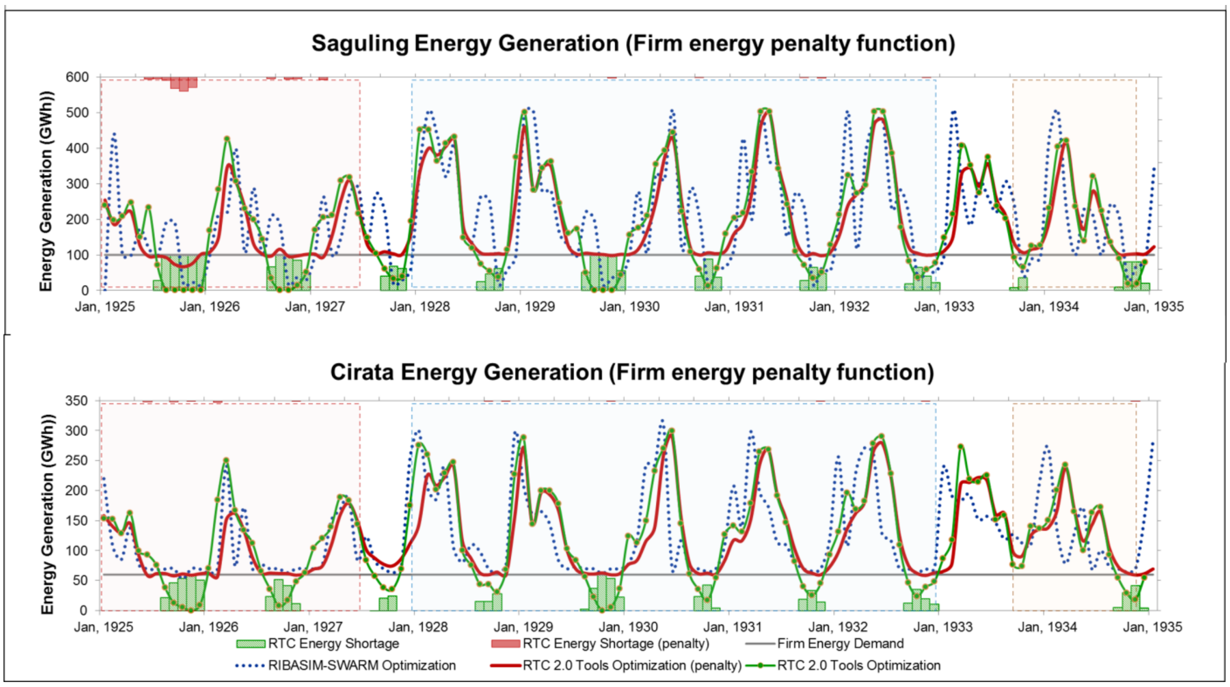


Figure 6.14 Energy generation: Modified hydroeconomic objective (shortage penalty)

Figure 6.15 shows that reformulation of the flood penalty function results in even less flooding while the economic valuation from Van der Vat [1] already significantly reduces the downstream flood event. The maximum agricultural water demand (Q=200 m^3/s) is set as the starting point of the linear flood penalty function. After this reformulation, the downstream flooding is always prevented since the released discharge never reached this threshold value.

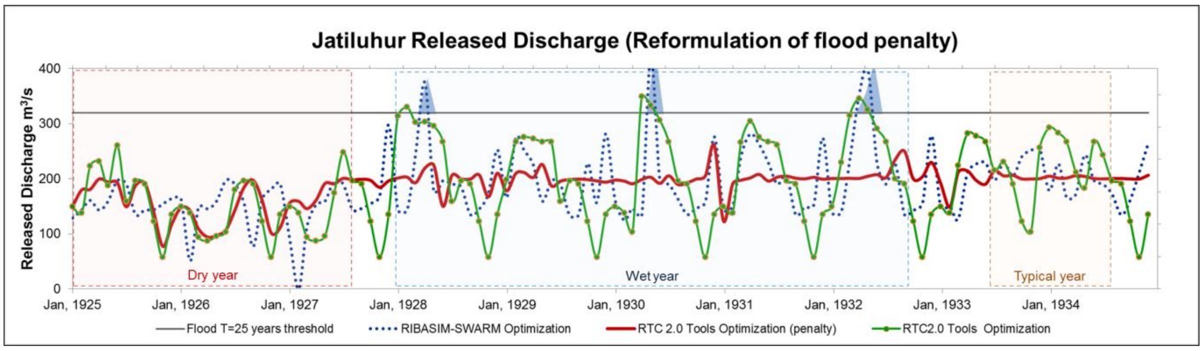


Figure 6.15 Downstream flooding: Modified hydroeconomic objective (flood penalty)

The agricultural drought penalty (US$ 0.016/m^3) and the higher economic valuations (US$ 0.02m^3 and US$ 0.043/m^3) of the agricultural delivered demand depends on seasonal cropping have less impact on the result since it is always satisfied. These findings suggest that the penalty function is considered as necessary when the economic valuation is relatively less dominant compared to the economic valuation of other conflicting objectives.

6.5 Linear programming of a modified hydroeconomic objective (Hard constraints)

For the purpose of the applicability of reservoir operation rules, the constraints are modified. The constraint for the minimum water level is set to be higher for the reservoir stability. The domestic water demand and environmental flow are defined as the main priorities whereas the previous optimization models neglect these demands. The environmental flow and the domestic water demand are specified as hard constraints since both are too difficult to be estimated economically [2]. The agricultural water demand is also modified based on the calculation that comprises the seasonal cropping analysis. As the total water demand in the network increases nearly 30%, the starting point of the flood damage penalty function for the Jatiluhur reservoir is adjusted.

Table 6.7 Modified hydroeconomic objective function (additional hard constraints)

Hydroeconomic optimization	*Unit*	*Jatiluhur*	*Cirata*	*Saguling*	*Others*	*Remarks*
Initial condition: Reservoir water level	m+MSL	98.8	219	625		RIBASIM Model
			Constraints			
Reservoir water level						
Minimum	m+MSL	87.5	180	623		Stability
Maximum	m+MSL	106.89	220	643		Spilling level
Turbines capacity						
Maximum	MW	187	1,008	700		Physical constraints
Domestic water demand	m^3/s				35.1	Hard constraint
Environmental flow	m^3/s				1.4	Hard constraint
			Objective functions			
			Benefit			
Agriculture benefit						Water footprint
Paddy (November - June	US$/$m^3$				0.02	0.02 10^{-6} millionUS$/$m^3$
Nuts (July - October)	US$/$m^3$				0.043	0.043 10^{-6} millionUS$/$m^3$
Agriculture demand						MPW (2012)
Paddy (November - June	m^3/s				87-322	
Nuts (July - October)	m^3/s				8-190	
Hydropower						van der Vat (2015)
Peak	US$/MWh				66	66 10^{-3} million US$/GWh
Rest	US$/MWh				32	32 10^{-3} million US$/GWh
			Penalty			
Agriculture						IRRI (2006)
Paddy (November - June	US$/$m^3$				0.016	0.016 10^{-6} millionUS$/$m^3$
Nuts (July - October)	US$/$m^3$				0	
Hydropower						
Peak	US$/MWh				2x66	132 10^{-3} million US$/GWh
Flood reduction						Muin (2015)
Q<236.4 m^3/s					0	
Q=320 m^3/s	million US$/month				14	14 $(30.24.60.60)^{-1}$ million US$/s
Q<320 m^3/s					Linear interpolation	

Results

By applying the domestic water demand and the environmental flow as the highest priorities, the total annual water demand increases by nearly 30%. Compared to the optimization of the modified hydroeconomic function, these demands reduce the maximum total economic benefits from US$ 373 million to US$ 370 million. While these fundamental demands are always fulfilled, the occurrences of minor agricultural drought event increase to 8 months/year (6 m^3/s). Figure 6.16 depicts that the drop in the water level of both upstream reservoirs reaches 10 m during the dry season to maintain almost similar water level in the Jatiluhur reservoir. One unanticipated finding was that this drop in water level results in nearly similar number of events and severities of total energy shortage. The decline in the total hydropower generation from the Saguling reservoir is followed by an improvement in the Jatiluhur reservoir. This result may be explained by the fact that the lower water level is compromised by the higher released discharge that supplies a higher total downstream water demand. Taken together, this finding signifies that the additional water demand has a consequential impact and should not be neglected in the optimization model although it is not valued economically.

While these fundamental water demands can be directly implemented as the hard constraints in the RTC-Tools 2.0 optimization model, defining additional water demands without quantifying their economic benefits seem to be more challenging in the RIBASIM-PS optimization. Involving these highest priority water demands in the RIBASIM schematization implies that they are included in the whole hydroeconomic optimization process while defining them as hard constraints have not been done yet.

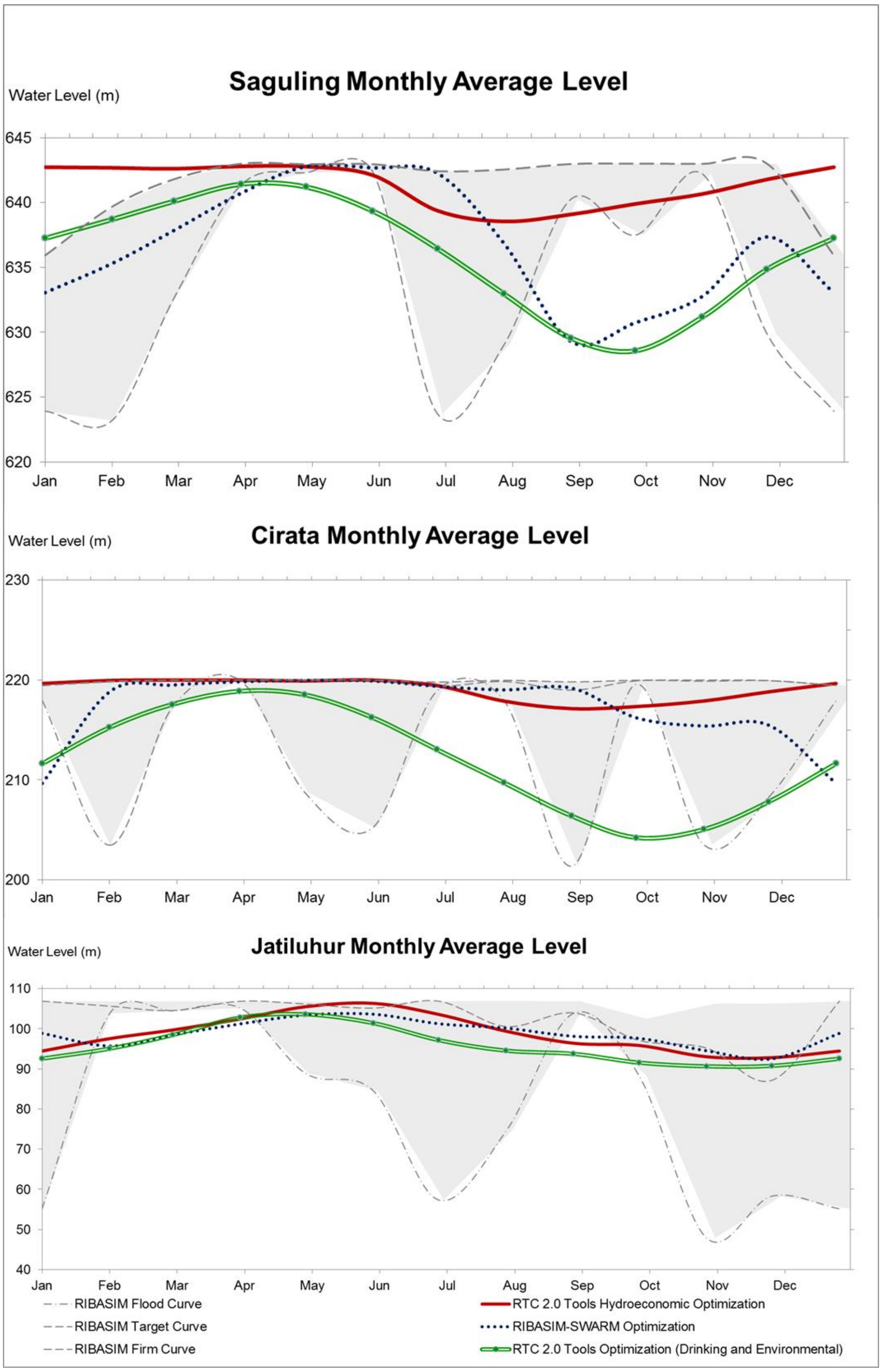


Figure 6.16 Actual water level of the reservoirs: Modified hydroeconomic objective (additional hard constraints)

6.6 Goal programming of hydrological objectives (Application)

In this study, the operational water level defined by the Indonesian government policy directive is set as the hard constraints in addition to the characteristics of physical infrastructures. The initial condition is taken from the NEDECO spreadsheet model that determines the reservoir operation rules in 2010 [37]. The constraint of annual deficit prevention is not included in this optimization model since it is often violated in the application. The principle of equal live storage sharing between the cascade reservoirs is also neglected to provide more search space for the optimization model. It is important to note the results from this optimization model are incomparable to the previous optimization models since it simulates a year period with the markedly different input data.

The sequences of hydrological objectives set in this optimization model refer to the mechanism for determining the reservoir operation rules based on the NEDECO spreadsheet model [37]. This spreadsheet model reveals that both upstream reservoirs are strictly operated to generate the maximum energy notwithstanding the high pressure in the Jatiluhur reservoir. As the local inflow discharge to the Jatiluhur reservoir is negligible, the water availability in this reservoir depends crucially on the released discharge of both upstream reservoirs. A special administrative procedure needs to be undertaken so that both upstream reservoirs release more discharge in the case of a serious downstream drought. To delineate this situation, the sequences of hydrological objectives are ordered to maximize the hydropower generation of both upstream reservoirs in the highest priority. In the lower priority, the target demands of the Jatiluhur reservoir are set. The fundamental downstream demands such as the domestic water demand and the environmental flow is set as the higher priority than the substantial trade-off in the Jatiluhur reservoir.

Table 6.8 Sequences of hydrological objectives (application by reservoir operators)

Priority	*Object*	*Objective function*	*Lower soft constraint*	*Upper soft constraint*
1	Saguling reservoir (S)	Energy generation (P)	S: MaxP	
2	Cirata reservoir (C)	Energy generation	C: MaxP	
3	Drinking terminal	Target released	Qdri	Qdri
3	Environmental flow	Target released	Qenv	-
4	Agricultural terminal	Target released	Qagr	Qagr
4	Jatiluhur reservoir (J)	Energy generation	J: FirmP	J: MaxP
4	Jatiluhur reservoir	Released Discharge	0	Flooding threshold

In order to define more applicable reservoir operation rules, it is important to define the sequences of objectives based on the Indonesian governmental policy directive. This study finds a conceptual difference between the sequences of objectives derived from the NEDECO spreadsheet model and the policy directive. As presented in Table 6.9, the domestic water demand and the environmental flow should be set as the highest priority as ordered in the Water Resources Law No. 7/2004 [56]. The upcoming priorities are set to the same level nevertheless the upstream reservoir receives inflow prior to the downstream reservoir. From this point, this strategy will be referred as the sharing strategy.

Table 6.9 Sequences of hydrological objectives (governmental policy directive)

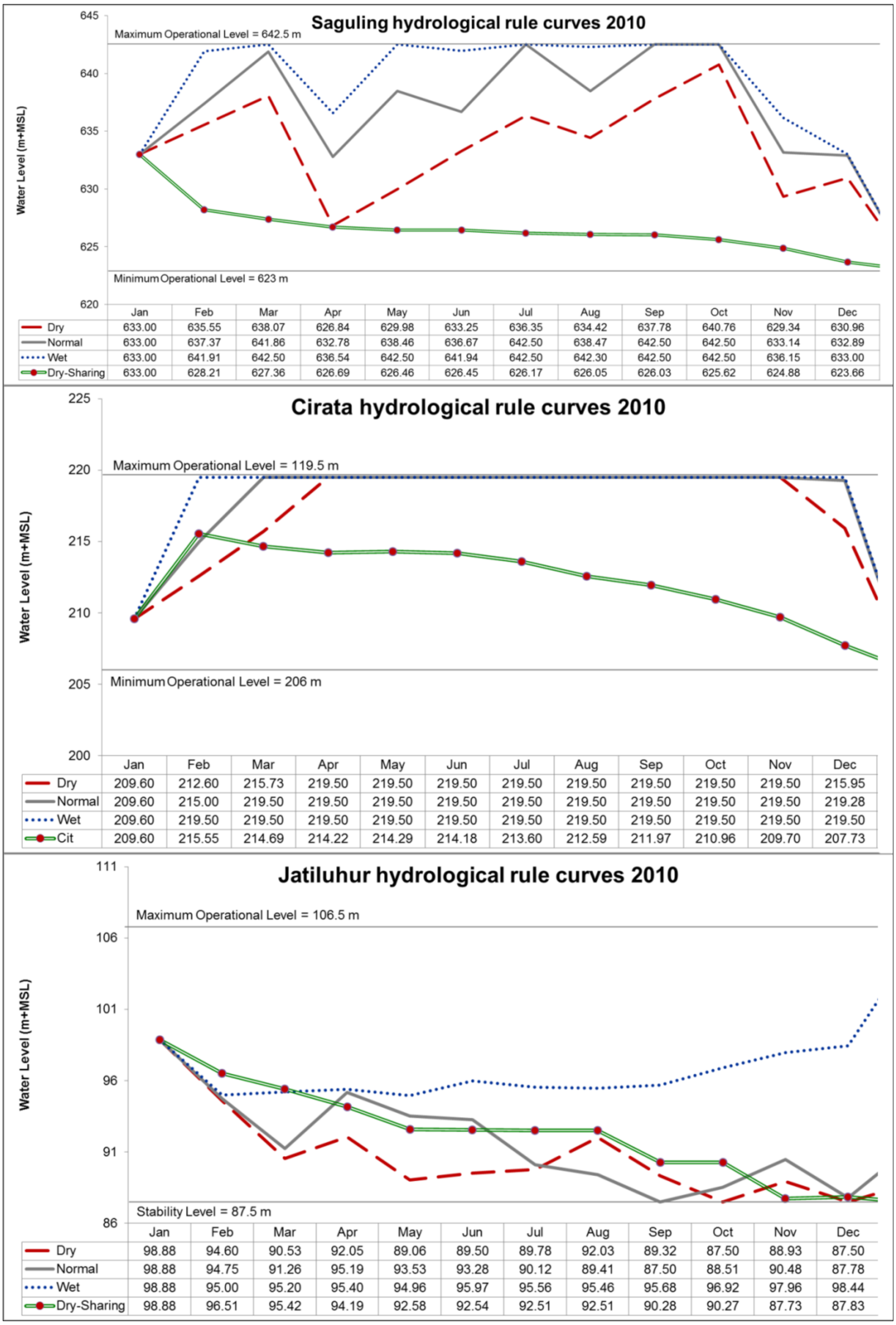
Priority	*Object*	*Objective function*	*Lower soft constraint*	*Upper soft constraint*
1	Drinking terminal	Target released	Qdri	Qdri
1	Environmental flow	Target released	Qenv	-
2	Saguling reservoir (S)	Energy generation (P)	S: FirmP	S: MaxP
2	Cirata reservoir (C)	Energy generation	C: FirmP	C: MaxP
2	Jatiluhur reservoir (J)	Energy generation	J: FirmP	J: MaxP
2	Agricultural terminal	Target released	Qagr	Qagr
2	Jatiluhur reservoir	Released Discharge	0	Flooding threshold

The deterministic models for the various hydrological years are constructed since the stochastic approach is limited in this study. This optimization approach can be illustrated briefly by three different hydrological years forecast as model input generates three distinctive actual water levels in a reservoir. This optimization of the sequences of objectives reproduces the NEDECO spreadsheet model used by the operators of the Citarum cascade reservoirs. In Indonesia, the stakeholders annually decide a single rule curve for a specific year operation based on the meteorological forecast. Thus, this rule curves concept is similar to the actual water level derived by optimization in the RTC-Tools 2.0.

Results

As presented in Figure 6.17, the results of the RTC-Tools 2.0 optimization of the current strategy have a similar trend on the reservoir operation rule curves derived by the reservoir operators. The reservoir water level is operated to be higher in the wetter year but it is always below the maximum operational water level. This result suggests that the reservoir operation rules during the wet year result in the very high reservoirs water level at the end of time step. However, this is relation is indirect in the Jatiluhur reservoir: the water level during the dry year is higher than the water level during the normal year as they have different fulfilment of agricultural delivered demand. This implies that, in the NEDECO spreadsheet model, the agricultural delivered demand should not be set as a hard constraint since it may not always be satisfied. The water scarcity analysis has not yet conducted by this spreadsheet model since the inflow data present much higher values compared to the data in this study.

It is important to note that the concept of the rule curves in the RIBASIM-PS study differs than the concept of the rule curve for a specific hydrological year in Indonesia. The RIBASIM-PS optimization results must be interpreted with caution since the trend of the actual water level mostly follows the target curve, notwithstanding the dry hydrological year is forecasted. Another important note is that, in the RIBASIM-PS algorithm, the actual rule curve applied to the reservoir operation (chosen from the firm, target and flood curve) could change in the next time step depending on the actual water level in the previous time step. This may cause confusion among the reservoir operators in Indonesia who usually determine an applied rule curve on a yearly basis. This can be clearly seen in the case of the extremely dry year is forecasted, an optimized firm curve in the RIBASIM-PS optimization cannot be directly applied as a dry year rule curve in term of the Indonesian government policy directive.



	Jan	Feb	Mar	Apr	May	Jun	Jul	Aug	Sep	Oct	Nov	Dec
Dry	633.00	635.55	638.07	626.84	629.98	633.25	636.35	634.42	637.78	640.76	629.34	630.96
Normal	633.00	637.37	641.86	632.78	638.46	636.67	642.50	638.47	642.50	642.50	633.14	632.89
Wet	633.00	641.91	642.50	636.54	642.50	641.94	642.50	642.30	642.50	642.50	636.15	633.00
Dry-Sharing	633.00	628.21	627.36	626.69	626.46	626.45	626.17	626.05	626.03	625.62	624.88	623.66

	Jan	Feb	Mar	Apr	May	Jun	Jul	Aug	Sep	Oct	Nov	Dec
Dry	209.60	212.60	215.73	219.50	219.50	219.50	219.50	219.50	219.50	219.50	219.50	215.95
Normal	209.60	215.00	219.50	219.50	219.50	219.50	219.50	219.50	219.50	219.50	219.50	219.28
Wet	209.60	219.50	219.50	219.50	219.50	219.50	219.50	219.50	219.50	219.50	219.50	219.50
Cit	209.60	215.55	214.69	214.22	214.29	214.18	213.60	212.59	211.97	210.96	209.70	207.73

	Jan	Feb	Mar	Apr	May	Jun	Jul	Aug	Sep	Oct	Nov	Dec
Dry	98.88	94.60	90.53	92.05	89.06	89.50	89.78	92.03	89.32	87.50	88.93	87.50
Normal	98.88	94.75	91.26	95.19	93.53	93.28	90.12	89.41	87.50	88.51	90.48	87.78
Wet	98.88	95.00	95.20	95.40	94.96	95.97	95.56	95.46	95.68	96.92	97.96	98.44
Dry-Sharing	98.88	96.51	95.42	94.19	92.58	92.54	92.51	92.51	90.28	90.27	87.73	87.83

Figure 6.17 Actual water level of the reservoirs: Sequences of hydrological objectives (application&policy)

This part of the section focuses on comparing the results of the optimization models if different strategies are applied. The strategy to replicating the policy directive refers to the Water Resources Law No. 7/2004. In this strategy, the fundamental water demand is put in the higher priority whereas the following priorities are set based on the purpose of each reservoir. To perceive the substantial trade-off between the conflicting objectives, the optimization results of both strategies during the dry hydrological year are presented.

The optimization results suggest that both strategies generate almost similar reservoir operation rules during the normal and wet year but these rules are significantly different during the dry year. These differences are consequential for the target demands of the Jatiluhur reservoir. In the dry year, the optimization result presents that the total economic benefit of the Citarum cascade reservoirs is increased by more than US$ 2 million along with a remarkable improvement on the economic benefit of the Jatiluhur reservoir. The benefit from the Jatiluhur reservoir is mostly gained from the higher agricultural benefit.

In general, applying this strategy results in lower water level of both upstream reservoirs so that the downstream reservoir receives an extra water to satisfy the target demands. As these target demands are set as the same priority level, the optimization model tries to satisfy these demands simultaneously notwithstanding some demands have prior access to the water. This sharing strategy seems to have a better applicability in the case of water scarcity. It is almost certain that modelling this sharing strategy, specifically to manage the substantial trade-off between the conflictive objectives, has not yet been done in the NEDECO spreadsheet model.

In the optimization results of the sequences of objectives based on the NEDECO spreadsheet model, the agricultural drought occurs frequently although the fundamental demands are always fulfilled. Adopting the sharing strategy towards the optimization model dramatically enhances the social benefits in the Jatiluhur reservoir but slightly reduces the average monthly hydropower generation from both upstream reservoirs from 180 GWh/month to 168 GWh/month. Figure 6.18 compare the agricultural delivered demand if different strategies are applied. Although the agricultural drought still occurs with the similar frequency of 5 months/year, the drought severity is reduced from 130 m^3/s to 100 m^3/s. This result shows a substantial the trade-off between the target demands of Citarum cascade reservoirs. Thus, it could be concluded that both social and economic benefits in the Citarum cascade reservoirs could be considerably improved by selecting a suitable strategy based on the policy insight.

This finding highlights the importance of adjusting the sequences of objectives to find the best strategy to address the water scarcity problem. These sequences of objectives in the goal programming could be directly associated with the stakeholders' perspective on priorities. Although the concept of priorities has been widely used in the rule-based simulation model, RTC-Tools 2.0 Tools provides a new feature of defining these priorities explicitly in the optimization model.

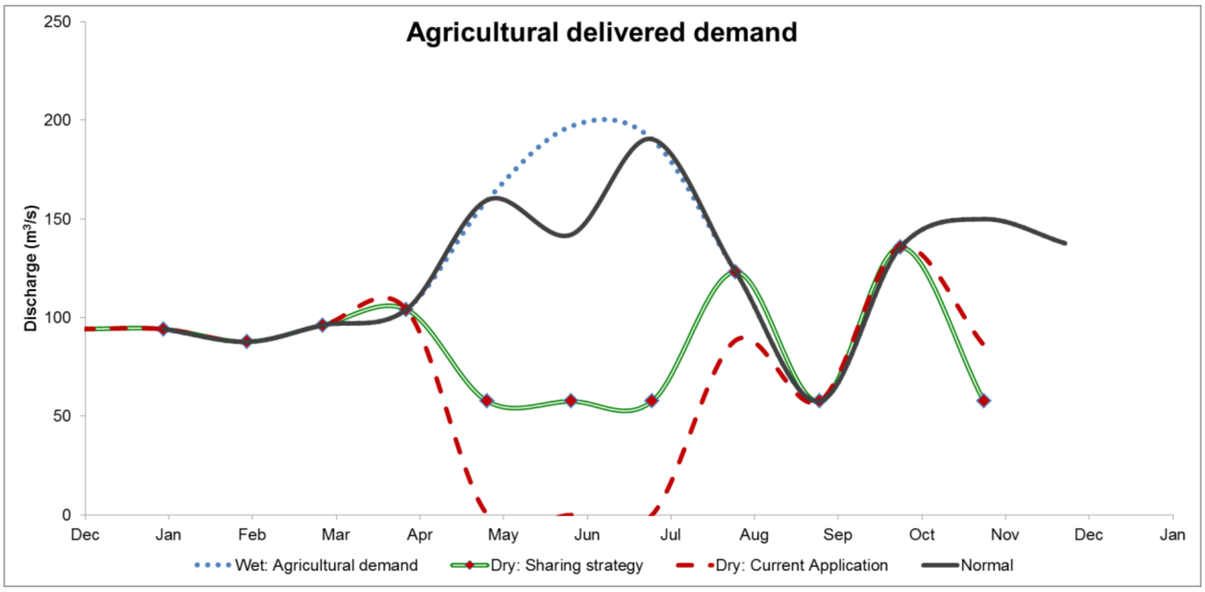


Figure 6.18 Agricultural delivered water demand: Sequences of hydrological objectives (application&policy)

6.7 Summary of findings

To avoid constructing new supply options, various methods have been developed to assess water allocation in a river basin in order to find a better operation system, for instance to derive the most promising reservoir operation. Traditionally, the optimum water allocation has been assessed by simulating various strategies in a rule-based simulation tool such as RIBASIM. As this method is time-consuming and not necessarily leads to the most promising result, recent advances in optimization techniques have facilitated the possibility to find better results.

As most reservoirs have conflicting objectives, a hydroeconomic optimization model could play an important role in solving multi-objective problems in the reservoir operation, especially in case of water scarcity. By combining the principles of economics and engineering, hydroeconomic models transform the concept of fixed demand into the economic value of water defined through water rights and priorities. Unfortunately, the management schemes and the policy insight are less likely to be easily represented by a hydroeconomic objective function. In Deltares, the necessity to explicitly implementing priority ordered by the policy on water resources allocation to a conventional hydroeconomic model has been done by combining the particle SWARM (PS) optimization with the rule-based simulation tools (RIBASIM).

To develop new alternative to the RIBASIM-PS study, a modular optimization model RTC-Tools 2.0 has been set up. This study focuses on the reservoir operation strategies to determine the most promising water allocation under similar attainment targets by constructing various hydroeconomic optimization models for a study case in the RTC–Tools 2.0. The development of the methodology for this study is based on the RIBASIM-PS study [1]. A case-study approach was adopted to provide rounded, detailed illustrations of the policy-based-management in water resources. The case study chosen is a simplified water network of the Citarum basin in West Java, Indonesia.

The goal programming approach has been chosen as the methodology for explicitly implementing the priority in the hydroeconomic optimization model in RTC-2.0 Tools. As an alternative to a conventional hydroeconomic model, this approach is likely to provide more robust, easy-to-build and communicative method to achieve a transparent water allocation based on the policy insight.

This study has been able to demonstrate the possibility to develop a similar model network as RIBASIM in the RTC-Tools 2.0. The methodology undertaken in this study has extended our knowledge of the critical step in transforming the algorithm of the simulation model into the explicit sequences of hydrological objectives for the optimization model. Furthermore, RTC-Tools 2.0 able to optimize similar hydroeconomic objective function as in RIBASIM-PS study but generate different results since RTC-Tools 2.0 is not coupled with a rule-based simulation model. While this issue has been addressed in this study, several alternatives of optimization approaches could be undertaken in RTC-Tools 2.0 to find the most promising reservoir operation rules for the case study.

This following part of the section summarises the findings related to the results of various optimization approaches carried out in the case study. These findings suggest that different optimization approaches generate distinctive results where certain results could be more suitable to the case study compared to the others. This study concludes that finding an appropriate approach and properly formulating the optimization problem are crucial steps in order to derive the most promising optimization results.

Table 6.10 presents an overview of the results from the various optimization approaches during the 90 years simulation period. The annual economic benefits presented are based on the economic valuation by Van der Vat [1]. It is important to bear in mind that the economic benefit taken directly from the modified hydroeconomic optimization models in the RTC-Tools 2.0 is not comparable since they have a distinctive economic valuation. Besides, the hydrological optimization models did not include any hydroeconomic analysis. To address this issue, a post-processing on the optimization results is carried out to obtain the total economic benefit based on the economic valuations by Van der Vat [1]. At last, the results from the optimization models with a year simulation are incomparable due to the different input data used.

As can be seen, the particle SWARM optimization approach significantly improves RIBASIM rule-based simulation model in terms of the number of the unfavourable events, their severities and the economic benefits. What is interesting about this result is that the pure linear programming from RTC-Tools 2.0 produces the highest benefit (US$ 378 million) in comparison with the other approaches but also the highest event severities. In contrast, the result of the goal programming of the sequences of hydrological objectives based on the RIBASIM presents the lowest benefit (321 US$ million) with more frequent but less severe events.

Table 6.10 Summary of benefit generated from hydroeconomic optimization models

Research question (s)	*Section (application and result)*	*Annual benefit for Citarum cascade reservoirs*	*Social priorities*		*Economic benefit (2010 value) (million US$/year)*				*Social Benefit (months/year)*				
			Drinking water	*Environ-mental flow*	*Total Benefit*	*Irrigation*	*Hydro power*	*Flood damage*	*Occurance of the agricultural drought*	*Average agricultural drought (m^3/s /month*	*Occurance of the energy shortage*	*Average energy shortage (GWh/ month)*	*Flood events*
-	-	RIBASIM Simulation without reservoir (van der Vat, 2015)	0	0	27	44	0	17.4	4.6	13	-	-	1.24
-	-	RIBASIM Simulation with reservoir (van der Vat, 2015)	0	0	347	60	289	1.7	1.1	41	5.1	60	0.12
-	-	RIBASIM-SWARM Optimization (van der Vat, 2015)	0	0	367.8	80	292	4.2	0.21	53	1.8	33.5	0.30
1 and 2	6.1	RTC 2.0 Tools Goal Programming 90 year time horizon optimization Hydrology Optimization	0	0	321.4	43	280	1.4	12	5	4.5	26	0.10
3	6.2	RTC 2.0 Tools Linear Programming Hydroeconomic Optimization	0	0	377.9	80.6	301.5	4.2	0	0	2.8	39	0.30
	6.3	RTC 2.0 Tools Hybrid Optimization Hydrology Optimization	0	0	374.6	79.5	295.2	0.14	12	1.3	2	18	0.01
4	6.4	RTC 2.0 Tools Linear Programming Modified hydroeconomic Optimization	0	0	372.6	80.6	292	0	0	0	1.6	11	0
4	6.5	RTC 2.0 Tools Linear Programming Modified hydroeconomic Optimization Additional hydrologic constraints	35.1	1.4	369.2	78.4	291	0.16	8	6.5	1.6	12.4	0.01
4	6.6	RTC 2.0 Tools Goal Programming Application by reservoir operators (Normal year)	35.1	1.4	347.6	76.6	271	0	1	54	3	10	0
4	6.7	RTC 2.0 Tools Goal Programming Sharing strategy (Normal year)	35.1	1.4	303.0	80	223	0	0	0	3.6	12	0

🟥 Poorly-satisfied 🟨 Moderately-satisfied 🟩 Well-satisfied

The result of the goal programming of the sequences of hydrological objectives based on the RIBASIM presents the lowest benefit (321 US$ million) with more frequent but less severe events. RTC-Tools 2.0 tends to generate more promising reservoir operation rules compared to the rule-based simulation model (RIBASIM), specifically when intense trade-off between reservoir conflicting objectives is substantial. The goal programming approach results in more frequent minor drought but significantly lower the severities of the drought events. Furthermore, this study employs different sequences of objectives by removing the upper flood control curves. The evidence from this analysis suggests that further optimization of the rule curves could be carried out, but this is considered as less necessary since the multi-objective functions can be applied directly in the study. This approach has a distinctive benefit as the transforming the target demands into the hydroeconomic economic function is optional nevertheless the trade-off between those demands is managed. Therefore, this approach could avoid the step in analysing the economic valuations, which is likely to be most expensive and time-consuming part of the hydroeconomic study.

The pure linear programming from RTC-Tools 2.0 provides the highest benefit (US$ 378 million) compared to the other approaches but also the highest event severities. As the firm energy demands are not explicitly defined in the RTC-Tools 2.0, the hydropower generation often drops below the firm energy demand whereas the agricultural water demand is always fulfilled. This finding reveals that the economic valuation of agricultural delivered water demand is relatively dominant compared to the economic valuations of other objectives. Additionally, this finding enhances our understanding of the importance of comprising these target demands in the optimization model to reduce the frequency and the severity of this unfavourable event. This has been managed by combining the particle SWARM optimization with the rule-based simulation tools (RIBASIM). In the RTC-Tools 2.0, these target demands have been represented by (i) including the hydrological soft constraints in the goal programming or (ii) adding the penalty functions as part of the objective function.

The sensitivity analysis conducted in this study concludes that the changes in the economic valuations that tend to deliver more water to agricultural demand have a negligible impact since the agricultural demand is always fulfilled in the current situation. On the other hand, the reduction in agricultural economic valuation and increment in the hydropower economic valuations tend to keep the water level higher to achieve higher energy generation, which further reduces the total energy shortage but results in agricultural drought.

The results from the hybrid optimization present the most promising reservoir operation rules. This optimization is able to substantially enhance the social benefit but generates a slightly lower economic benefit (375 US$ million). Assigning the firm energy demand to a soft constraint to the goal programming approach remarkably improves the social benefit compared to the pure linear programming of hydroeconomic objective function. Similarly to the role of the RIBASIM algorithm on the particle SWARM optimization, these soft constraints assist the RTC-Tools 2.0 in optimizing the sequences of hydrological objectives while the linear programming searches for the highest economic benefit at the same time. Applying these soft constraints is likely to ameliorate the robustness of the hydroeconomic optimization model since they are independent of the economic valuations.

If a pure linear programming approach is preferred, the additional penalty functions should be included in the hydroeconomic valuations by Van der Vat (2015) to represent the social responsibility. In comparison with the pure hydroeconomic optimization, applying these penalty functions reduce the total economic benefits from US$ 378 million to US$ 373 million but substantially enhance the social benefit. Applying the high penalty function on hydropower generation seems to be the most preferable approach to diminish the energy shortage. In addition, the occurrence of downstream flooding is entirely eliminated after applying more strict formulation of the penalty function on flood damage. These results seem to be consistent if the economic valuation of penalty function is high enough. On the other hand, assigning a penalty function to agricultural drought has less impact on the result since it is always satisfied. These findings suggest that the penalty function is considered as necessary when the economic valuation is relatively less dominant compared to the economic valuation of other conflicting objectives.

When this modified hydroeconomic optimization comprises the fundamental demands as the highest priorities, the maximum total economic benefit reduces from US$ 373 million to US$ 370 million. While these fundamental demands are always fulfilled, the occurrences of minor agricultural drought event increase. Taken together, this finding signifies that the additional water demand has a consequential impact and should be included in the optimization model although it is too difficult to be estimated economically.

The results in this study reveal that the reservoir operation rules derived from RTC-Tools 2.0 have a similar trend to the current application by the reservoir operators; the water level of the reservoir is operated to be higher in the wetter year but it is always between the ranges of the reservoir's operational water levels. When the different sequences of objectives are applied, the reservoir operation rules tend to be significantly different during the dry year. The new strategy based on policy directive is set as the sequences of objectives in the goal programming model. This strategy puts the reservoir purpose in the same level of priority; nevertheless the upstream reservoir receives inflow prior to the downstream reservoir. This study concludes that the social and economic benefits in the Citarum cascade reservoirs, especially in the Jatiluhur reservoir, could be improved if a suitable strategy based on policy insight is implemented.

7. CONCLUSIONS AND RECOMMENDATIONS

This concluding chapter revisits in Section 7.1 the four research questions and formulates the answers to these questions based on the analyses described in previous chapters. The summary and important findings of this study in greater details which led us to this conclusion are provided in Section 6.7. Finally, Section 7.2 recommends a number of possible future researches in this field.

7.1 Answers to research questions

Returning to the questions posed at the beginning of this study, it is now possible to state that RTC-Tools 2.0 is the appropriate modular tools to solve optimization problems in addition to its wide application in real time control situations. RTC-Tools 2.0 offers the possibility to construct various optimization models with a high level of flexibility and to understand the optimization process better since most scripting are accessible for the users. This key feature is strengthened by Modelica declarative language which enables the users to easily reproduce and reformulate the optimization variables. In general, therefore, it seems that RTC-Tools 2.0 as an open source modular tools offers a valuable solutions by solving a single or multi-objective problems with advance optimization approaches.

Research question 1: *Is RTC-Tools 2.0 able to model a similar network as RIBASIM does, using allocation rules based on demand priority and reservoir operation rules including hedging?*

This study has been able to demonstrate the possibility to develop a similar model network as RIBASIM in the RTC-Tools 2.0. A related finding, while preliminary, is that the optimization results from RTC-Tools 2.0 present more promising reservoir operation rules in comparison with the rule-based simulation results in RIBASIM for the multi-purpose reservoir. In single purpose reservoir, less significant difference between the optimization results from both tools is observed. Therefore, these findings suggest that the goal programming approach in the RTC-Tools 2.0 reveals a new potential method to derive a set of optimal reservoir operating rules, especially for the multi-purpose reservoir. Ensuring appropriate systems, the operation rules derived from RTC-Tools 2.0 are expected to provide a promising solution to the trade-offs between two or more conflicting objectives.

Research question 2: *Is it possible to formulate a set of objectives and constraints in the RTC-Tools 2.0 that will result in optimized reservoir operating rules?*

The methodology undertaken in this study has extended our knowledge of the critical step in transforming the algorithm of the simulation model into the explicit sequences of hydrological objectives for the optimization model. When identifying these, care was taken to highlight the implicit priorities, rule curves and hedging rules based on storage adapted from the algorithm of the RIBASIM simulation model.

This study has demonstrated how to define the time series of the upper and lower constraints extracted from the existing RIBASIM model as the binding value of the sequences of objectives. This study also transforms some RIBASIM model inputs into inviolable hard constraints comprised of the physical infrastructure parameters. Whereas these hard constraints tend to be easier to identify, this study finds that deriving the soft constraints adapted from the simulation model input is more challenging.

Research question 3: *Are the calculations of optimal reservoir operation rules by RTC-Tools 2.0 different from the operation rules resulting from the RIBASIM-PS optimization and, if so, why?*

The study concludes that the optimal reservoir operation rules as identified by RTC-Tools 2.0 are different from the rules that resolved by RIBASIM-PS study. While the RIBASIM-PS study optimized annual rule curves, the RTC-Tools 2.0 optimization results in a time series of actual water levels. These results remain comparable since the three different rule curves from the *RIBASIM-PS* optimization result presents specific rules that later derive a time series of actual water level from the RIBASIM simulation.

The pure hydroeconomic optimization carried out in this study assists in further understanding of the role of the algorithm that comprises the fulfilment of the social benefits. From the linear programming approach, RTC-Tools 2.0 generates a slightly higher maximum benefit but more shortage events compared to the optimization result from RIBASIM-PS optimization. RIBASIM-PS optimization, which optimizes a similar objective function, is nonetheless partly influenced by the algorithm in the rule-based simulation model RIBASIM. This algorithm assists RIBASIM-PS optimization in enhancing the social benefits while it searches for the highest economic benefit.

The sensitivity analysis conducted in this study strengthens the presumption that the economic valuation is a sensitive parameter in the pure linear hydroeconomic optimization model, especially when the necessary trade-off between reservoir conflicting objectives is substantial. The changes in economic valuation could alter the operation rules in the system while determining a suitable value that comprises the management schemes and the policy insight is an expected difficulty.

Research question 4: *How can we further improve the results of the optimization approaches in order to get a better applicability of the reservoir operation rules?*

There is a number of important improvements that could be done in hydroeconomic optimization to derive more promising reservoir operation rules. This is the first study reporting the advantages of the hybrid optimization between the linear programming and additional soft constraints in the goal programming. As this approach directly applies the parameter values, setting the penalty functions is optional. This approach is thought to be a preferable option if reformulating the objective functions is difficult. This approach is likely to ameliorate the robustness of the hydroeconomic optimization model since the soft constraints are independent of the economic valuations.

The principal theoretical implication of this study is that the pure linear programming requires a suitable penalty function to represent the social benefit properly. The formulation of the penalty functions becomes more crucial for the optimization model due to the absence of the algorithm in the rule-based simulation model.

The fundamental water demand should not be neglected on the optimization model although they are not quantified economically. While the practicality of applying these demands as the hard constraints in the RTC-Tools 2.0 is rather straight-forward, separating these demands from the optimization objective function is an expected difficulty in the RIBASIM-PS optimization.

In order to derive more practical and applicable reservoir operation rules, the optimization model should incorporate the rules from the Indonesian government policy directive. The hydrological year rule curves in Indonesia, which simply refers to the expected monthly reservoir water level, are similar to the actual water level in the RTC-Tools 2.0. The results of this study indicate that RTC-Tools 2.0 is able to derive the key concept in the policy directive; the cascade reservoirs should be operated higher during the wetter year although. Still, the reservoir water level must be between the operational water level constraints.

7.2 Recommendations on future research

Considering that the great variability of nodes and links is often to be the key feature of the rule-based simulation tools such as RIBASIM, the future development of RTC-Tools 2.0 should be undertaken to expand the declarative equations in Modelica library in order to minimize the pre-processing step. This development is likely to enhance the practicalities and efficiency of constructing a spatially distributed model in the RTC-2.0 Tools. Additionally, while the current version of RTC-Tools 2.0 generates a comma separated values file, the optimization results presented in graph and chart might be a tremendous help for the users to have a brief overview of the results.

Being limited to the deterministic optimization as the scope of the study, this approach lacks of analysis in input data uncertainty. This study performs an optimization approach with the presumption of the perfect knowledge of the future events. Despite these promising results from these deterministic optimization models, questions remain. Furthermore, the simplistic statistical scholastic analysis conducted in this study suggests that the uncertainty in the inflow discharge forecast tends to be high. The stochastic optimization is expected to be an important issue for the future research considering that RTC-Tools 2.0 provides this capability.

As was pointed out earlier, the Citarum basin has a highly dynamic system facing different future changes in the increased pressure in demand and climate change. These future challenges are different spatially. Further research should be undertaken to develop this model into a dynamic model that represents the time-dependent aspects of the model behaviour. The future research that accommodates the dynamic future changes spatially is expected to provide a better insight into the multi-objective problems that will be useful for the decision-making process for the long-term master plan.

BIBLIOGRAPHY

[1] M. van der Vat, “Optimizing reservoir operation for flood storage, hydropower and irrigation using a hydro-economic model for the Citarum River, West-Java, Indonesia,” London, 2015.

[2] J. J. Harou, M. Pulido-Velazquez, D. E. Rosenberg, J. Medellín-Azuara, J. R. Lund and R. E. Howitt, “Hydro-economic models: Concepts, design, applications, and future prospects,” *Journal of Hydrology,* no. 375, p. 627–643, 2009.

[3] J. Braden, “Value of valuation: introduction,” *Journal of Water Resource Planning and Management,* vol. 126, no. 6, pp. 336-338, 2000.

[4] J. Lund and I. Ferreira, “Operating rule optimization for Missouri River reservoir system,” *Journal of Water Resources Planning and Management,* vol. 122, no. 4, p. 287–295, 1996.

[5] T. Zhu, G. F. Marques and J. R. Lund, “Hydroeconomic optimization of integrated water management and transfers under stochastic surface water supply,” *Water Resources Research,* vol. 51, p. 3568–3587, 2015.

[6] J. H. Gibbons, Use of Models for Water Resources Management, Planning, and Policy, Washington, D.C., 1982.

[7] MIT, Nonlinear Programming, Massachusetts, 2015.

[8] G. B. Dantzig, “Linear Programming and Extensions,” Princeton University Press, 1963.

[9] D. Everson and J. C. Moseley, “Simulation or Optimization Techniques for Multibasin Water Resources Planning,” *Water Resources Bulletin,* vol. 6, no. 5, pp. 725-737, 1970.

[10] M. S. Babel, A. D. Gupta and D. K. Nayak, “A Model for Optimal Allocation of Water to Competing Demands,” *Water Resources Management,* vol. 19, no. 6, p. 693–712, 2005.

[11] I. Heinz, M. Pulido-Velazquez, J. R. Lund and J. and Andreu, “Hydro-economic Modeling in River Basin Management: Implications and Applications for the European Water Framework Directive,” *Water Resources Management,* vol. 21,

no. 7, p. 1103–1125, 2007.

[12] S. Satti, B. Zaitchik and S. and Siddiqui, “The question of Sudan: a hydro-economic optimization model for the Sudanese Blue Nile.,” *Hydrology and Earth System Sciences,* vol. 19, no. 5, p. 2275–2293, 2015.

[13] X. Cai, M. W. Rosegrant and C. and Ringler, “Physical and economic efficiency of water use in the river basin: Implications for efficient water management,” *Water Resources Research,* vol. 39, no. 1, 2003.

[14] R. Bartlett, J. Baker, G. Lacombe, S. Douangsavanh and M. and Jeuland, “Analyzing Economic Tradeoffs of Water Use in the Nam Ngum River Basin , Lao PDR,” *Duke Environmental Economics Working Paper Series,* p. 37 pp, 2012.

[15] Q. Goor, C. Halleux, Y. Mohamed and A. and Tilmant, “Optimal operation of a multipurpose multireservoir system in the Eastern Nile River Basin,” *Hydrology and Earth System Sciences,* vol. 14, no. 10, p. 1895–1908, 2010.

[16] C. Rougé and A. Tilmant, “Applying SDDP to very large hydro-economic models with a simplified formulation for irrigation : the case of the Tigris-Euphrates river basin,” *Geophysical Research Abstracts,* p. 9859, 2015.

[17] A. Niazi, S. O. Prasher, J. Adamowski and T. Gleeson, “A System Dynamics Model to Conserve Arid Region Water Resources through Aquifer Storage and Recovery in Conjunction with a Dam,” *Water,* vol. 6, pp. 2300-2321, 2014.

[18] J. W. Labadie, “Optimal Operation of Multireservoir Systems: State-of-the-Art Review,” *Journal of Water Resources Planning and Management,,* vol. 130, no. 2, p. 93–111, 2004.

[19] G. A. Schultz, Ivory tower versus ghosts? - or- The Interdependency Between Systems Analysts and Real-World Decision Makers in Water Management: Closing the Gap Between Theory and Practice, Bochum: IAHS, 1989, pp. 23-32.

[20] M. McGregor and J. Dent, “An Application of Lexicographic Goal Programming to Resolve the Allocation of Water from the Rakaia River (New Zealand),” *Agricultural Systems,* vol. 41, pp. 349-367, 1993.

[21] E. A. Eschenbach, T. M. Magee and E. Zagona, “Multiobjectives Operations of Reservoir Systems via Goal Programming,” *J Water Resour Plan Management,* vol. 127, pp. 108-120, 2001.

[22] G. Leavesley, S. Markstrom, M. Brewer and R. Viger, “The modular modeling system (MMS) : The physical process modeling component of a database-centered decision support,” U.S. Geological Survey, Denver, 1995.

[23] Deltares, “Pilot case Citarum basin,” 2016. [Online]. Available: https://publicwiki.deltares.nl/display/OpenS/Pilot+case_+Citarum+basin. [Accessed 11 April 2016].

[24] CGIAR CSI, “SRTM 90m Digital Elevation Database,” 2016. [Online]. Available: http://www.cgiar-csi.org/data/srtm-90m-digital-elevation-database-v4-1. [Accessed 11 April 2016].

[25] BBWS Citarum, “Kondisi Fisik dan Spasial,” 08 August 2014. [Online]. Available: http://citarum.org/tentang-kami/sekilas-citarum/kondisi-fisik-dan-spasial.html. [Accessed 24 March 2016].

[26] BBWS Citarum, “Peta Wilayah Sungai Citarum,” 2016. [Online]. Available: http://bbwscitarum.com/wp-content/uploads/2014/12/peta-ws-citarum-keppres-12.png. [Accessed 11 April 2016].

[27] Jawa Ecoregion Profile, “Citarum Basin,” Indonesia Ministry of Environment, 2015. [Online]. Available: http://ppejawa.com/ekoregion/das-citarum/. [Accessed 10 April 2016].

[28] BPLHD, Writer, *Parakarsa pemerintah Jawa Barat dalam rangka pengembangan eco town di cekungan Bandung*. [Performance]. 2010.

[29] I. MPW, “Pola Pengelolaan Sumber Daya Air Wilayah Sungai Cidanau - Ciujung - Cidurian - Cisadane- Ciliwung - Citarum,” Kementerian Pekerjaan Umum, Jakarta, 2012.

[30] P. Rejekiningrum, “Alokasi Optimum Kebutuhan Air Untuk Pertanian Dengan Inovasi Teknologi Irigasi Berselang (Intermittent Irrigation): Studi Kasus Das Citarum, Jawa Barat,” *Prosiding Seminar Nasional Matematika, Sains, dan Teknologi.,* vol. 4, pp. 23-37, 2013.

[31] H. Shirakawa, K. Noda, P. San Miguel and K. Oki, “Integration of ecosystem services in impact assessment tools,” in *6th annual ESP conference*, Bali, 2013.

[32] Lufiandi, “Impact of Land Use Change on Water Managment in Upper Citarum River Basin,” Delft, 2011.

[33] BBWS Citarum, “Kondisi Fisik dan Spasial,” 08 August 2014. [Online]. Available: http://citarum.org/tentang-kami/sekilas-citarum/kondisi-fisik-dan-spasial.html. [Accessed 24 March 2016].

[34] Dirjen PU, “Pola Pengelolaan Sumber Daya Air Wilayah Sungai Citarum,” Direktorat Jenderal Sumber Daya Air Kementerian Pekerjaan Umum, Jakarta, 2012.

[35] FAO, “AQUASTAT global water information system,” 2015.

[36] GEO, “Current list of hydropower plants.,” 2015.

[37] Nedeco, “Jatiluhur Water Resources Management Project Preparation Study (JWRMP). Optimal Integrated Citarum Reservoir Cascade Operation. Feasibility Report. Nedeco in association with Indec, Virama Karya and Gamma,” Bandung, 1998.

[38] SPK-TPA, “Standard Operation Procedure 2010 of Cascade Reservoirs Citarum,” SPK-TPA Citarum, Bandung, 2010.

[39] J. Dijkman, W. Van der Krogt, Hendarti and J. Brinkman, “Review of the Standard Operation Procedures for the Citarum Reservoirs, Results of a rapid assessment. 6 Cis project.,” Jakarta, 2012.

[40] T. Perwitasari, “Simulation of Cascade Reservoirs Operation using RTC Tools and Distributed-Hydrological Model Citarum River Basin, West Java, Indonesia,” Delft, 2013.

[41] W. Van der Krogt, “RIBASIM Version 7.00 Technical Reference Manual,” Deltares, Delft, 2008.

[42] W. Van der Krogt and A. Boccalon, “River Basin Simulation Model RIBASIM Version 7.00, User Manual,” Deltares, Delft, 2013.

[43] K. E. Parsopoulos and M. N. Vrahatis, “Recent approaches to global optimization problems through particle swarm optimization,” *Natural Computing,* no. 1, pp. 235-306, 2002.

[44] J. M. Reddy and N. D. Kumar, “Performance evaluation of elitist-mutated multi-onjective particle swarm optimization for integrated water resources management,” *Journal of Hydroinformatics,* no. 11.1, pp. 79-83, 2009.

[45] Deltares, “RIBASIM,” 2015. [Online]. Available: https://www.deltares.nl/en/software/ribasim/. [Accessed 08 April 2016].

[46] Deltares, “RTC-Tools,” 2016. [Online]. Available: https://www.deltares.nl/en/software/rtc-tools/. [Accessed 21 March 2016].

[47] P. Fritzson, “Introduction to Object-Oriented Modeling and Simulation with Modelica Using OpenModelica,” Linköping University, Linköpin, 2012.

[48] International Rice Research Institute, “Incorporating drought tolerance as a rice breeding objective,” 2006.

[49] S. M. Miranda, “Irrigation Management for Crop Diversification in Indonesia, The

Philippines, and Sri Lanka," Colombo, 1989.

[50] Indonesian Ministry of Trade, "Tabel Harga Kebutuhan Pokok Nasional," Jakarta, 2010.

[51] PLN, "Kemarau Panjang, PLTA Tidak Bisa Beroperasi Maksimal," 29 October 2015. [Online]. Available: http://www.pln.co.id/blog/kemarau-panjang-plta-tidak-bisa-beroperasi-maksimal/. [Accessed 31 March 2016].

[52] S. F. Muin, "Pengembangan Asuransi Bencana Banjir Berbasis Indeks Untuk Sektor Pemukiman Dan Pertanian," Institut Pertanian Bogor, Bogor, 2015.

[53] H. G. S. Aji, "Evaluasi laju sedimentasi pada waduk Jatiluhur, Kabupaten Purwakarta, Jawa Barat," UT - Civil Engineering and Environment , Bogor, 2014.

[54] I. Srihadi, B. Gunady and R. Mayasari, "Operational Performance of Djuanda Dam Indonesia," Stravanger, 2015.

[55] Jasa Tirta II, Jakarta, 2004.

[56] Indonesian Ministry of Public Works, "Undang-Undang Republik Indonesia 7/2004 Tentang Sumber Daya Air," Jakarta, 2004.

[57] D. A. Iancu and N. Trichakis, "Pareto Efficiency in Robust Optimization," *Management Science,* vol. 60, no. 1, pp. 130 - 147, January 2014.

[58] A. Chapagain and A. Hoekstra, "The Green, Blue And Grey Water Footprint Of Rice From Both A Production And Consumption Perspective," UNESCO-IHE, Delft, 2010.

[59] M. Mekonnen and A. Hoekstra, "The green, blue and grey water footprint of crops and derived crop products," UNESCO-IHE Institute for Water Education, Delft, 2010.

[60] C. B. Boroughs and E. Zagona, "Daily Flow Routing With The Muskingum-Cunge Method Daily Flow Routing With The Muskingum-Cunge Method," 2002. [Online]. Available: http://webcache.googleusercontent.com/search?q=cache:lePP-Npwf-gJ:cadswes.colorado.edu/sites/default/files/PDF/RiverWare/BoroughsLV2002.pdf+&cd=1&hl=en&ct=clnk&gl=nl. [Accessed 29 March 2016].

[61] Geo-Slope International, "Rapid Drawdown with Effective Stress," Calgary, 2007.

[62] S. Sunanisari, E. Harsono and T. Tarigan, "Pengelolaan Ekosistem Dan Produktivitas DAS Citarum: Pengembangan Model Kualitas Air Waduk Saguling, Cirata, Jatiluhur," Pusat Penelitian Geoteknologi LIPI, Bandung, 2003.

[63] N. Kumar and J. Reddy, "Optimal reservoir operation for irrigation of multiple

crops using elitist-mutated particle swarm optimization," *Hydrological Sciences,* vol. 54, no. 4, pp. 686-701, August 2007.

[64] T. Juwitaningtyas, "The Most Polluted River in the World, Citarum River, Indonesia," 2016. [Online]. Available: http://www.austroindonesianartsprogram.org/blog/most-polluted-river-world-citarum-river-indonesia. [Accessed 11 April 2016].

[65] Indonesian Ministry of Trade, "Tinjauan Pasar Beras," Jakarta, 2011.

[66] A. Tilmant, D. Pinte and Q. and Goor, "Assessing marginal water values in multipurpose multireservoir systems via stochastic programming.," *Water Resources Research,* vol. 44, no. 12, 2008.

[67] A. Tilmant and W. Kinzelbach, "The cost of noncooperation in international river basins," *Water Resources Research,,* vol. 48, no. 1, 2012.

[68] Ladi, "Operating Rule Optimization for Missouri River Reservoir System," *Journal of Water Resources Planning and Management,* vol. 122, no. 5, p. 287–295, 1996.

[69] D. Whittington, X. Wu and S. C., "Water resources management in the Nile basin: the economic value of cooperation," *Water Policy,* vol. 7, no. 3, p. 227–252, 2005.

[70] C. Ringler, N. V. Huy and S. Msangi, "Water Allocation Policy Modelling for the Dong Nai River Basin: an Integrated Perspective," *Journal of the American Water Resources Association,* vol. 42, no. 6, p. 1465–1482, 2006.

[71] W. Chow, R. Brocksen, Wisniewski and Joe, "Clean Water: Factors that Influence Its Availability, Quality and Its Use," in *International Clean Water Conference*, California, 1995.

[72] A. Dinar, Bridges Over Water: Understanding Transboundary Water Conflict, Negotiation and Cooperation, World Scientific, 2007.

[73] McGraw-Hill, Linear Goal Programming and Its Solution Procedures, 2004.

[74] S. Systems, "The Dynamic Model," Sparx Systems, 2004.

[75] World Weather, "Climate Bandung," 2016. [Online]. Available: https://weather-and-climate.com/average-monthly-Rainfall-Temperature-Sunshine,bandung,Indonesia. [Accessed 2016 June 30].

APPENDIXES

APPENDIX A: GLOSSARY

ARIMA	Autoregressive Integrated Moving Average, a stochastic approach to forecast the time series based on the statistics and econometrics.
BBWS	Balai Besar Wilayah Sungai, the Indonesian central government agency for river basin organization
BMKG	Badan Meteorologi, Klimatologi dan Geofisika, the Indonesian government agency for meteorology, climatology and geophysics
BPLHD	Badan Pengelolaan Lingkungan Hidup Daerah, the regional environmental agency
Deterministic model	A model that applies a single set of historical or synthetically generated time series to obtain a single set of results.
Goal programming	The optimization model that searches the minimum total deviation from the soft constraints of sequences of objectives, also termed multi-objective optimization
Hard constraints	The inviolable constraints where the optimum solution must be inside these bounds, refer to constraints in a conventional linear programming
Hydroeconomic model	Mathematical model that transforms the concept of fixed demand into the economic value of water defined through water rights and priorities and future projections by combining the principles of economics and engineering.
Linear programming	A conventional optimization model that searches for a local minimum of a linear objective function
Modelica	An open source object-oriented programming language for simulation and optimization developed by Modelica Association
NEDECO	Netherland Engineering Consultants BV
Objective function	A function to be minimized in the optimization model, mostly consists of the economic valuations in the hydroeconomic model.

Optimization model	A mathematical model that runs to identify the local minimum of objective function limited to the constraints which represent the system
PS	Particle Swarm Optimization, is a population-based stochastic optimization approach inspired by social behaviour of bird flocking
RIBASIM	River Basin Simulation Model, a software package by that provides the sources of analysis in water allocation of a network (Deltares, 2015)
RTC-Tools	Real-Time Control Tools, an open-source toolbox for real-time control and optimization of hydraulic systems by (Deltares, 2016).
Sequences of objectives	Several specific numeric goals that are derived based on the priorities to set the series of objective functions
Simulations model	A rule-based algorithm that reproduces the system complexities in integrated water resources management, planning and policies to answer “what if” type of questions.
Soft constraints	Each goal’s lower and upper bounds that allow violations from the goal programming model
SOP	Standard Operating Procedure, a reservoir operation procedure to support maximum safety and operational efficiency depending on the hydrological data of current or forecast year.
SPK-TPA	Sekretariat Pelaksana Koordinasi - Tata Pengaturan Air, the secretariat of coordination committee on the water resources management.
Stochastic model	A model that incorporates the probabilistic character of model inputs to generate the probability results rather than a deterministic, single set of results.

APPENDIX B: MODELICA SCRIPT - MODEL SCHEMATIZATION

The nodes and links of the model schematization on Modelica graphical user interface automatically generate the code that needs to be adjusted like in the following scripting. This scripting is mostly divided into four parts.

1. Classification of variables along with their units, whether they are a parameter, an input or an output. This part of the scripting is very crucial to balance the number of equation and free variables
2. Each objects' name and annotation, some include the value of physical characteristics' parameters
3. Description of the connection between declared objects
4. Declaration of equations that connect the clarification in the first part of this scripting and the name of the Modelica objects in the second part of this scripting.

I. Goal programming on the linear reservoir model schematization

```
model citarum
  import SI = Modelica.SIunits;

/*1. DEFINE input to balance the number of equation and free variables*/

  input SI.VolumeFlowRate inflow_Saguling_Q;
  input SI.VolumeFlowRate inflow_Cirata_Q;
  input SI.VolumeFlowRate inflow_Jatiluhur_Q;
  input SI.Velocity Saguling_evaporation;
  input SI.Velocity Cirata_evaporation;
  input SI.Velocity Jatiluhur_evaporation;
  input SI.VolumeFlowRate reservoirCompact_Saguling_Qturbine;
  input SI.VolumeFlowRate reservoirCompact_Cirata_Qturbine;
  input SI.VolumeFlowRate reservoirCompact_Jatiluhur_Qturbine;
  output SI.VolumeFlowRate spill_Saguling_Q;
  output SI.VolumeFlowRate spill_Cirata_Q;
  output SI.VolumeFlowRate spill_Jatiluhur_Q;
  input SI.VolumeFlowRate node_Agriculture_QOut;
  input SI.VolumeFlowRate node_Drinking_QOut;

/*2. Each objects' name and annotation*/
/*INPUT hard constraints and physical parameters*/
  Deltares.Flow.SimpleRouting.BoundaryConditions.Inflow inflow_Saguling
  annotation(Placement(visible = true, transformation(origin = {-90, 74}, extent = {{-10, -
  10}, {10, 10}}, rotation = 0)));
  Deltares.Flow.SimpleRouting.BoundaryConditions.Inflow inflow_Cirata
  annotation(Placement(visible = true, transformation(origin = {-90, 40}, extent = {{-10, -
  10}, {10, 10}}, rotation = 0)));
  Deltares.Flow.SimpleRouting.BoundaryConditions.Inflow inflow_Jatiluhur
  annotation(Placement(visible = true, transformation(origin = {-90, 8}, extent = {{-10, -
  10}, {10, 10}}, rotation = 0)));
  Deltares.Flow.SimpleRouting.Nodes.Node node1(nout = 1, nin = 2)
  annotation(Placement(visible = true, transformation(origin = {-54, 40}, extent = {{-10, -
  10}, {10, 10}}, rotation = 0)));
  Deltares.Flow.SimpleRouting.Reservoir.Linear reservoirCompact_Saguling(H_b =
  622.6, area = 20000000, H_tail = 280, turbine_efficiency=0.87)
  annotation(Placement(visible = true, transformation(origin = {-70, 74}, extent = {{-10, -
  10}, {10, 10}}, rotation = 0)));
  Deltares.Flow.SimpleRouting.Reservoir.Linear reservoirCompact_Cirata(H_b = 180,
  area = 30000000, H_tail = 107, turbine_efficiency=0.87) annotation(Placement(visible
  = true, transformation(origin = {-34, 40}, extent = {{-10, -10}, {10, 10}}, rotation = 0)));
  Deltares.Flow.SimpleRouting.Nodes.Node node2(nout = 1, nin = 2)
  annotation(Placement(visible = true, transformation(origin = {-18, 8}, extent = {{-10, -
  10}, {10, 10}}, rotation = 0)));
  Deltares.Flow.SimpleRouting.Nodes.Node node_Drinking(nout = 2)
  annotation(Placement(visible = true, transformation(origin = {34, 8}, extent = {{-10, -
  10}, {10, 10}}, rotation = 0)));
  Deltares.Flow.SimpleRouting.Nodes.Node node_Agriculture(nout = 2)
  annotation(Placement(visible = true, transformation(origin = {50, -20}, extent = {{-10, -
  10}, {10, 10}}, rotation = 0)));
  Deltares.Flow.SimpleRouting.BoundaryConditions.Terminal terminal_Agriculture
  annotation(Placement(visible = true, transformation(origin = {82, -6}, extent = {{-10, -
  10}, {10, 10}}, rotation = 0)));
  Deltares.Flow.SimpleRouting.BoundaryConditions.Terminal terminal_River
  annotation(Placement(visible = true, transformation(origin = {82, -28}, extent = {{-10, -
  10}, {10, 10}}, rotation = 0)));
  Deltares.Flow.SimpleRouting.BoundaryConditions.Terminal terminal_Drinking
  annotation(Placement(visible = true, transformation(origin = {82, 8}, extent = {{-10, -
  10}, {10, 10}}, rotation = 0)));
  Deltares.Flow.SimpleRouting.Reservoir.Linear reservoirCompact_Jatiluhur(H_b =
  74.89, area = 53000000, H_tail = 28, turbine_efficiency=0.87)
  annotation(Placement(visible = true, transformation(origin = {2, 8}, extent = {{-10, -10},
  {10, 10}}, rotation = 0)));

/*3. DEFINE connection*/
equation
  connect(node2.QOut[1], reservoirCompact_Jatiluhur.QIn) annotation(Line(points = {{-
  10, 8}, {-6, 8}}));
  connect(reservoirCompact_Jatiluhur.QOut, node_Drinking.QIn[1])
  annotation(Line(points = {{10, 8}, {26, 8}}));
  connect(node_Drinking.QOut[1], terminal_Drinking.QIn) annotation(Line(points =
  {{42, 8}, {74, 8}}));
  connect(node_Agriculture.QOut[2], terminal_River.QIn) annotation(Line(points = {{58,
  -20}, {58, -28}, {74, -28}}));
  connect(node_Agriculture.QOut[1], terminal_Agriculture.QIn) annotation(Line(points
  = {{58, -20}, {61, -20}, {61, -6}, {74, -6}}));
  connect(node_Drinking.QOut[2], node_Agriculture.QIn[1]) annotation(Line(points =
  {{42, 8}, {42, -20}}));
  connect(reservoirCompact_Cirata.QOut, node2.QIn[2]) annotation(Line(points = {{-26,
  40}, {-26, 8}}));
  connect(inflow_Jatiluhur.QOut, node2.QIn[1]) annotation(Line(points = {{-82, 8}, {-26,
  8}}));
  connect(node1.QOut[1], reservoirCompact_Cirata.QIn) annotation(Line(points = {{-46,
  40}, {-42, 40}}));
  connect(reservoirCompact_Saguling.QOut, node1.QIn[2]) annotation(Line(points = {{-
  62, 74}, {-62, 40}}));
  connect(inflow_Saguling.QOut, reservoirCompact_Saguling.QIn)
  annotation(Line(points = {{-82, 74}, {-78, 74}}));
  connect(inflow_Cirata.QOut, node1.QIn[1]) annotation(Line(points = {{-82, 40}, {-62,
  40}}));

/*4. DECLARE additional equation, mostly to connect time series and object*/
  inflow_Saguling.Q = inflow_Saguling_Q;
  inflow_Cirata.Q = inflow_Cirata_Q;
  inflow_Jatiluhur.Q = inflow_Jatiluhur_Q;
  reservoirCompact_Saguling.evaporation_protected=Saguling_evaporation;
  reservoirCompact_Cirata.evaporation_protected=Cirata_evaporation;
  reservoirCompact_Jatiluhur.evaporation_protected=Jatiluhur_evaporation;
  node_Drinking_QOut=terminal_Drinking.Q;
  terminal_Drinking.Q = 0;
  reservoirCompact_Saguling_Qturbine = reservoirCompact_Saguling.Q_turbine;
  reservoirCompact_Cirata_Qturbine = reservoirCompact_Cirata.Q_turbine;
  reservoirCompact_Jatiluhur_Qturbine = reservoirCompact_Jatiluhur.Q_turbine;
  spill_Saguling_Q = reservoirCompact_Saguling.Q_spill;
  spill_Cirata_Q = reservoirCompact_Cirata.Q_spill;
  spill_Jatiluhur_Q = reservoirCompact_Jatiluhur.Q_spill;

annotation(Icon(coordinateSystem(extent = {{-100, -100}, {100, 100}},
preserveAspectRatio = true, initialScale = 0.1, grid = {2, 2})),
Diagram(coordinateSystem(extent = {{-100, -100}, {100, 100}}, preserveAspectRatio =
true, initialScale = 0.1, grid = {2, 2})));

end citarum;
```

II. Goal programming on look-up table reservoirs model schematization

```
model citarum
  import SI = Modelica.SIunits;
  input SI.VolumeFlowRate inflow_Saguling_Q;
  input SI.VolumeFlowRate inflow_Cirata_Q;
  input SI.VolumeFlowRate inflow_Jatiluhur_Q;
  input SI.VolumeFlowRate spill_Saguling_Q;
  input SI.VolumeFlowRate spill_Cirata_Q;
  input SI.VolumeFlowRate spill_Jatiluhur_Q;
  output SI.VolumeFlowRate reservoirCompact_Saguling_Qturbine;
  output SI.VolumeFlowRate reservoirCompact_Cirata_Qturbine;
  output SI.VolumeFlowRate reservoirCompact_Jatiluhur_Qturbine;

  input SI.Velocity Saguling_evaporation;
  input SI.Velocity Cirata_evaporation;
  input SI.Velocity Jatiluhur_evaporation;

  input SI.VolumeFlowRate node_Agriculture_QOut;
  output SI.VolumeFlowRate terminal_River_Q;
  input SI.VolumeFlowRate node_Drinking_Qout;

  input SI.Area Saguling_area(nominal = 1e8);
  input SI.Volume Saguling_volume(nominal = 1e9);
  input SI.Area Cirata_area(nominal = 1e8);
  input SI.Volume Cirata_volume(nominal = 1e9);
  input SI.Area Jatiluhur_area(nominal = 1e8);
  input SI.Volume Jatiluhur_volume(nominal = 1e10);

  Deltares.Flow.SimpleRouting.BoundaryConditions.Inflow inflow_Saguling
annotation(Placement(visible = true, transformation(origin = {-90, 74}, extent = {{-10,
-10}, {10, 10}}, rotation = 0)));
  Deltares.Flow.SimpleRouting.BoundaryConditions.Inflow inflow_Cirata
annotation(Placement(visible = true, transformation(origin = {-90, 40}, extent = {{-10,
-10}, {10, 10}}, rotation = 0)));
  Deltares.Flow.SimpleRouting.BoundaryConditions.Inflow inflow_Jatiluhur
annotation(Placement(visible = true, transformation(origin = {-90, 8}, extent = {{-10, -
10}, {10, 10}}, rotation = 0)));
  Deltares.Flow.SimpleRouting.Nodes.Node node1(nout = 1, nin = 2)
annotation(Placement(visible = true, transformation(origin = {-54, 40}, extent = {{-10,
-10}, {10, 10}}, rotation = 0)));
  Deltares.Flow.SimpleRouting.Nodes.Node node_Drinking(nin = 1, nout = 2)
annotation(Placement(visible = true, transformation(origin = {34, 8}, extent = {{-10, -
10}, {10, 10}}, rotation = 0)));
```

```
  Deltares.Flow.SimpleRouting.Nodes.Node node_Agriculture(nin = 1, nout = 2)
annotation(Placement(visible = true, transformation(origin = {50, -20}, extent = {{-10,
-10}, {10, 10}}, rotation = 0)));
  Deltares.Flow.SimpleRouting.BoundaryConditions.Terminal terminal_Agriculture
annotation(Placement(visible = true, transformation(origin = {82, -6}, extent = {{-10, -
10}, {10, 10}}, rotation = 0)));
  Deltares.Flow.SimpleRouting.BoundaryConditions.Terminal terminal_River
annotation(Placement(visible = true, transformation(origin = {82, -28}, extent = {{-10,
-10}, {10, 10}}, rotation = 0)));
  Deltares.Flow.SimpleRouting.BoundaryConditions.Terminal terminal_Drinking
annotation(Placement(visible = true, transformation(origin = {82, 8}, extent = {{-10, -
10}, {10, 10}}, rotation = 0)));
  Deltares.Flow.SimpleRouting.Reservoir.LookupTable
reservoirCompact_Cirata(H_tail = 107, turbine_efficiency = 0.87)
annotation(Placement(visible = true, transformation(origin = {-30, 40}, extent = {{-10,
-10}, {10, 10}}, rotation = 0)));
  Deltares.Flow.SimpleRouting.Reservoir.LookupTable
reservoirCompact_Jatiluhur(H_tail = 104, turbine_efficiency = 0.87)
annotation(Placement(visible = true, transformation(origin = {8, 8}, extent = {{-10, -
10}, {10, 10}}, rotation = 0)));
  Deltares.Flow.SimpleRouting.Reservoir.LookupTable
reservoirCompact_Saguling(H_tail = 280, turbine_efficiency = 0.87)
annotation(Placement(visible = true, transformation(origin = {-70, 74}, extent = {{-10,
-10}, {10, 10}}, rotation = 0)));
  Deltares.Flow.SimpleRouting.Nodes.Node node2(nout = 1, nin = 2)
annotation(Placement(visible = true, transformation(origin = {-16, 8}, extent = {{-10, -
10}, {10, 10}}, rotation = 0)));

equation
  connect(inflow_Jatiluhur.QOut, node2.QIn[1]) annotation(Line(points = {{-82, 8}, {-
24, 8}}));
  connect(reservoirCompact_Cirata.QOut, node2.QIn[2]) annotation(Line(points = {{-
22, 40}, {-24, 40}, {-24, 8}}));
  connect(node2.QOut[1], reservoirCompact_Jatiluhur.QIn) annotation(Line(points =
{{-8, 8}, {0, 8}}));
  connect(reservoirCompact_Saguling.QOut, node1.QIn[2]) annotation(Line(points =
{{-62, 74}, {-62, 40}}));
  connect(inflow_Saguling.QOut, reservoirCompact_Saguling.QIn)
annotation(Line(points = {{-82, 74}, {-78, 74}}));
  connect(node1.QOut[1], reservoirCompact_Cirata.QIn) annotation(Line(points = {{-
46, 40}, {-38, 40}, {-38, 40}, {-38, 40}}));
  connect(node_Drinking.QOut[1], terminal_Drinking.QIn) annotation(Line(points =
{{42, 8}, {74, 8}}));
  connect(node_Agriculture.QOut[2], terminal_River.QIn) annotation(Line(points =
{{58, -20}, {58, -28}, {74, -28}}));
```

```
  connect(node_Agriculture.QOut[1], terminal_Agriculture.QIn)
annotation(Line(points = {{58, -20}, {61, -20}, {61, -6}, {74, -6}}));
  connect(node_Drinking.QOut[2], node_Agriculture.QIn[1]) annotation(Line(points =
{{42, 8}, {42, -20}}));
  connect(inflow_Cirata.QOut, node1.QIn[1]) annotation(Line(points = {{-82, 40}, {-62,
40}}));

  Saguling_area = reservoirCompact_Saguling.A;
  Saguling_volume = reservoirCompact_Saguling.V;
  Cirata_area = reservoirCompact_Cirata.A;
  Cirata_volume = reservoirCompact_Cirata.V;
  Jatiluhur_area = reservoirCompact_Jatiluhur.A;
  Jatiluhur_volume = reservoirCompact_Jatiluhur.V;

  inflow_Saguling.Q = inflow_Saguling_Q;
  inflow_Cirata.Q = inflow_Cirata_Q;
  inflow_Jatiluhur.Q = inflow_Jatiluhur_Q;

  reservoirCompact_Saguling.evaporation_protected=Saguling_evaporation;
  reservoirCompact_Cirata.evaporation_protected=Cirata_evaporation;
  reservoirCompact_Jatiluhur.evaporation_protected=Jatiluhur_evaporation;

  reservoirCompact_Saguling_Qturbine = reservoirCompact_Saguling.Q_turbine;
  reservoirCompact_Cirata_Qturbine = reservoirCompact_Cirata.Q_turbine;
  reservoirCompact_Jatiluhur_Qturbine = reservoirCompact_Jatiluhur.Q_turbine;

  node_Agriculture_QOut = node_Agriculture.QOut_control[1];
  terminal_River_Q = terminal_River.Q;
  terminal_Drinking.Q = 0;
  node_Drinking_Qout = node_Drinking.QOut_control[1];

  spill_Saguling_Q = reservoirCompact_Saguling.Q_spill;
  spill_Cirata_Q = reservoirCompact_Cirata.Q_spill;
  spill_Jatiluhur_Q = reservoirCompact_Jatiluhur.Q_spill;

  annotation(Icon(coordinateSystem(extent = {{-100, -100}, {100, 100}},
preserveAspectRatio = true, initialScale = 0.1, grid = {2, 2})),
Diagram(coordinateSystem(extent = {{-100, -100}, {100, 100}}, preserveAspectRatio =
true, initialScale = 0.1, grid = {2, 2})));

end citarum;
```

II. Hydroeconomic optimization on the linear reservoir model schematization

The total economic benefit summarises the hydroeconomic objective function which comprises of economic valuations based on the hydropower generation, agricultural delivered demand and flood damage reduction. This hydroeconomic objective function is declared in the Modelica file and it is optimized by the linear programming with the assistance of RTC-Tools 2.0. In this scripting, the economic valuations are mostly adapted from Van der Vat (2015) while the commented economic valuations, which mainly consist of penalty functions, are appraised in this study.

```
/*APPLY hydroeconomic valuations*/
/*COMMENT modified hydroeconomic valuations and additional constraints*/

model citarum
  import SI = Modelica.SIunits;

/*DEFINE input to balance the number of equation and free variables*/
/*MODELICA variables*/
  input SI.VolumeFlowRate inflow_Saguling_Q;
  input SI.VolumeFlowRate inflow_Cirata_Q;
  input SI.VolumeFlowRate inflow_Jatiluhur_Q;
  input SI.Velocity Saguling_evaporation;
  input SI.Velocity Cirata_evaporation;
  input SI.Velocity Jatiluhur_evaporation;
  input SI.VolumeFlowRate reservoirCompact_Saguling_Qturbine;
  input SI.VolumeFlowRate reservoirCompact_Cirata_Qturbine;
  input SI.VolumeFlowRate reservoirCompact_Jatiluhur_Qturbine;
  input SI.VolumeFlowRate spill_Saguling_Q;
  input SI.VolumeFlowRate spill_Cirata_Q;
  input SI.VolumeFlowRate spill_Jatiluhur_Q;
  input SI.VolumeFlowRate node_Agriculture_QOut;
  input SI.VolumeFlowRate out_Jatiluhur_Q;
  input SI.VolumeFlowRate terminal_Drinking_Qin;



/*HYDROECONOMIC VALUATION*/
/*HYDROPOWER*/
  parameter Real auxcon_Saguling (unit = "1")=0.01;
  parameter Real auxcon_Cirata (unit = "1")=0.01;
  parameter Real auxcon_Jatiluhur (unit = "1")=0;
  parameter Real c (unit = "1")=1/(1e9*3600);

  output Real output_Saguling (unit = "GWh");
  output Real output_Cirata (unit = "GWh");
  output Real output_Jatiluhur (unit = "GWh");
  parameter Real fr_peakSaguling (unit = "1")=1;
  parameter Real fr_peakCirata (unit = "1")=1;
  parameter Real fr_peakJatiluhur (unit = "1")=0.2;
  parameter Real value_Powerrest (unit = "dollar/GWh")=31.59*1000;
  parameter Real value_Powerpeak (unit = "dollar/GWh")=65.85*1000;

  output Real benefit_PowerSaguling (unit="1e6*dollar/GWh");
  output Real benefit_PowerCirata (unit="1e6*dollar/GWh");
  output Real benefit_PowerJatiluhur (unit="1e6*dollar/GWh");
  output Real benefit_Power (unit="1e6*dollar", start = 0.0, fixed = true);

/*MODIFIED HYDROPOWER
  parameter Real value_Powerpeakpenalty (unit = "dollar/GWh")=2*66*1e3;
  output Real penalty_PowerSaguling (unit="1e6*dollar/GWh");
  output Real penalty_PowerCirata (unit="1e6*dollar/GWh");
  output Real penalty_Power (unit="1e6*dollar", start = 0.0, fixed = true);
*/

/*IRRIGATION*/
  parameter Real value_Irrigationbenefit (unit="1e6*dollar/1e6*m3")=0.02;
  output Real benefit_Irrigation (unit="1e6*dollar", start = 0.0, fixed = true);

/*MODIFIED IRRIGATION
  input value_Irrigationpenalty;
  output penalty_Irrigation;
  output benefitpenalty_Irrigation;
*/

/*FLOOD DAMAGE*/
  parameter SI.VolumeFlowRate returnperiod_Q=320;
  input Real value_Floodinitial (unit="1e6*dollar/s");
  parameter Real value_Floodpenalty (unit="1e6*dollar/s")=14/(3600*24*30);
  output Real penalty_Flood (unit="1e6*dollar");
  output Real benefit_Flood (unit="1e6*dollar", start = 0.0, fixed = true);

/*MODIFIED FLOOD DAMAGE
  input value_Irrigationpenalty;
  output penalty_Irrigation;
  output benefitpenalty_Irrigation;
*/

  output Real benefit_Total (unit="1e6*dollar");
  Deltares.Flow.SimpleRouting.BoundaryConditions.Inflow inflow_Saguling
  annotation(Placement(visible = true, transformation(origin = {-90, 74}, extent = {{-10, -
  10}, {10, 10}}, rotation = 0)));
  Deltares.Flow.SimpleRouting.BoundaryConditions.Inflow inflow_Cirata
  annotation(Placement(visible = true, transformation(origin = {-90, 40}, extent = {{-10, -
  10}, {10, 10}}, rotation = 0)));
  Deltares.Flow.SimpleRouting.BoundaryConditions.Inflow inflow_Jatiluhur
  annotation(Placement(visible = true, transformation(origin = {-90, 8}, extent = {{-10, -
  10}, {10, 10}}, rotation = 0)));
  Deltares.Flow.SimpleRouting.Nodes.Node node1(nout = 1, nin = 2)
  annotation(Placement(visible = true, transformation(origin = {-54, 40}, extent = {{-10, -
  10}, {10, 10}}, rotation = 0)));
  Deltares.Flow.SimpleRouting.Reservoir.Linear reservoirCompact_Saguling(H_b =
  622.6, area = 31000000, H_tail = 280, turbine_efficiency=0.95)
  annotation(Placement(visible = true, transformation(origin = {-70, 74}, extent = {{-10, -
  10}, {10, 10}}, rotation = 0)));
  Deltares.Flow.SimpleRouting.Reservoir.Linear reservoirCompact_Cirata(H_b = 180,
  area = 33000000, H_tail = 107, turbine_efficiency=0.87) annotation(Placement(visible
  = true, transformation(origin = {-34, 40}, extent = {{-10, -10}, {10, 10}}, rotation = 0)));
  Deltares.Flow.SimpleRouting.Nodes.Node node2(nout = 1, nin = 2)
  annotation(Placement(visible = true, transformation(origin = {-18, 8}, extent = {{-10, -
  10}, {10, 10}}, rotation = 0)));
  Deltares.Flow.SimpleRouting.Nodes.Node node_Drinking(nout = 2)
  annotation(Placement(visible = true, transformation(origin = {34, 8}, extent = {{-10, -
  10}, {10, 10}}, rotation = 0)));
  Deltares.Flow.SimpleRouting.Nodes.Node node_Agriculture(nout = 2)
  annotation(Placement(visible = true, transformation(origin = {50, -20}, extent = {{-10, -
  10}, {10, 10}}, rotation = 0)));
  Deltares.Flow.SimpleRouting.BoundaryConditions.Terminal terminal_Agriculture
  annotation(Placement(visible = true, transformation(origin = {82, -6}, extent = {{-10, -
  10}, {10, 10}}, rotation = 0)));
  Deltares.Flow.SimpleRouting.BoundaryConditions.Terminal terminal_River
  annotation(Placement(visible = true, transformation(origin = {82, -28}, extent = {{-10, -
  10}, {10, 10}}, rotation = 0)));
  Deltares.Flow.SimpleRouting.BoundaryConditions.Terminal terminal_Drinking
  annotation(Placement(visible = true, transformation(origin = {82, 8}, extent = {{-10, -
  10}, {10, 10}}, rotation = 0)));
  Deltares.Flow.SimpleRouting.Reservoir.Linear reservoirCompact_Jatiluhur(H_b =
  74.89, area = 79000000, H_tail = 28, turbine_efficiency=1)
  annotation(Placement(visible = true, transformation(origin = {2, 8}, extent = {{-10, -10},
  {10, 10}}, rotation = 0)));

equation
  connect(node2.QOut[1], reservoirCompact_Jatiluhur.QIn) annotation(Line(points = {{-
  10, 8}, {-6, 8}}));
  connect(reservoirCompact_Jatiluhur.QOut, node_Drinking.QIn[1])
  annotation(Line(points = {{10, 8}, {26, 8}}));
  connect(node_Drinking.QOut[1], terminal_Drinking.QIn) annotation(Line(points =
  {{42, 8}, {74, 8}}));
  connect(node_Agriculture.QOut[2], terminal_River.QIn) annotation(Line(points = {{58,
  -20}, {58, -28}, {74, -28}}));
  connect(node_Agriculture.QOut[1], terminal_Agriculture.QIn) annotation(Line(points
  = {{58, -20}, {61, -20}, {61, -6}, {74, -6}}));
  connect(node_Drinking.QOut[2], node_Agriculture.QIn[1]) annotation(Line(points =
  {{42, 8}, {42, -20}}));
  connect(reservoirCompact_Cirata.QOut, node2.QIn[2]) annotation(Line(points = {{-26,
  40}, {-26, 8}}));
  connect(inflow_Jatiluhur.QOut, node2.QIn[1]) annotation(Line(points = {{-82, 8}, {-26,
  8}}));
  connect(node1.QOut[1], reservoirCompact_Cirata.QIn) annotation(Line(points = {{-46,
  40}, {-42, 40}}));
  connect(reservoirCompact_Saguling.QOut, node1.QIn[2]) annotation(Line(points = {{-
  62, 74}, {-62, 40}}));
  connect(inflow_Saguling.QOut, reservoirCompact_Saguling.QIn)
  annotation(Line(points = {{-82, 74}, {-78, 74}}));
  connect(inflow_Cirata.QOut, node1.QIn[1]) annotation(Line(points = {{-82, 40}, {-62,
  40}}));

/*DECLARE additional equation, mostly to connect time series and object*/
  inflow_Saguling.Q = inflow_Saguling_Q;
  inflow_Cirata.Q = inflow_Cirata_Q;
  inflow_Jatiluhur.Q = inflow_Jatiluhur_Q;
  reservoirCompact_Saguling.evaporation=Saguling_evaporation;
  reservoirCompact_Cirata.evaporation=Cirata_evaporation;
  reservoirCompact_Jatiluhur.evaporation=Jatiluhur_evaporation;
  terminal_Drinking.Q = 0;
  reservoirCompact_Saguling_Qturbine = reservoirCompact_Saguling.Q_turbine;
  reservoirCompact_Cirata_Qturbine = reservoirCompact_Cirata.Q_turbine;
  reservoirCompact_Jatiluhur_Qturbine = reservoirCompact_Jatiluhur.Q_turbine;
  spill_Saguling_Q = reservoirCompact_Saguling.Q_spill;
  spill_Cirata_Q = reservoirCompact_Cirata.Q_spill;
  spill_Jatiluhur_Q = reservoirCompact_Jatiluhur.Q_spill;
  out_Jatiluhur_Q=spill_Jatiluhur_Q+reservoirCompact_Jatiluhur_Qturbine;

/*ADD constraints for domestic/drinking  water demand
   Environmental flow is added to bounds in RTC-Tools 2.0
terminal_Drinking.Q = 35;
*/
```

```
/*
  --------------------------------------------------------
  BENEFIT AND PENALTY FUNCTIONS
  --------------------------------------------------------
*/

/*---------------------------HYDROPOWER GENERATION---------------------------*/
output_Saguling=c*(1-auxcon_Saguling)*reservoirCompact_Saguling.P;
output_Cirata=c*(1-auxcon_Cirata)*reservoirCompact_Cirata.P;
output_Jatiluhur=c*(1-auxcon_Jatiluhur)*reservoirCompact_Jatiluhur.P;


benefit_PowerSaguling=(output_Saguling*((fr_peakSaguling*value_Powerpeak)+((1-
fr_peakSaguling)*value_Powerrest)))/1e6;
benefit_PowerCirata=(output_Cirata*((fr_peakCirata*value_Powerpeak)+((1-
fr_peakCirata)*value_Powerrest)))/1e6;
benefit_PowerJatiluhur=(output_Jatiluhur*((fr_peakJatiluhur*value_Powerpeak)+((1-
fr_peakJatiluhur)*value_Powerrest)))/1e6;

der(benefit_Power)=(benefit_PowerSaguling+benefit_PowerCirata+benefit_PowerJatilu
hur);

/*-------------------------------------IRRIGATION------------------------------------*/
der(benefit_Irrigation)=terminal_Agriculture.Q*value_Irrigationbenefit/1e6;

/*-----------------------------MODIFIFED IRRIGATION----------------------------*/
/*
benefit_Irrigation+=terminal_Agriculture.Q*value_Irrigationbenefit;
penalty_Irrigation+=(Qagr-terminal_Agriculture.Q)*value_Irrigationpenalty;
benefitpenalty_Irrigation+=benefit_Irrigation-penalty_Irrigation
*/

/*-----------------------------------FLOOD REDUCTION-----------------------------------*/
if out_Jatiluhur_Q>returnperiod_Q then
   penalty_Flood=value_Floodpenalty;
else
  penalty_Flood=0;
end if;

der(benefit_Flood)=(value_Floodinitial/(3600*24*30))-penalty_Flood;

/*---------------------------MODIFIED FLOOD REDUCTION---------------------------*/
/*
returnperiod_Q5=66;
returnperiod_Q25=334;
```

```
value_Floodpenalty=14000000;

if reservoirCompact_Jatiluhur_Qout>=returnperiod_Q25 then
   penalty_Flood=value_Floodpenalty;
elseif reservoirCompact_Jatiluhur_Qout>= returnperiod_Q5 &
linear_Jatiluhur.HQ.Q<returnperiod_Q25:
  penalty_Flood=(linear_Jatiluhur.HQ.Q-
returnperiod_Q5)*value_Floodpenalty/(returnperiod_Q25-returnperiod_Q5)
else:
  penalty_Flood=0
benefit_Flood+=value_Floodinitial-penalty_Flood;
*/

/*---------------------------TOTAL BENEFIT---------------------------*/
benefit_Total=benefit_Power+benefit_Irrigation+benefit_Flood;

annotation(Icon(coordinateSystem(extent = {{-100, -100}, {100, 100}},
preserveAspectRatio = true, initialScale = 0.1, grid = {2, 2})),
Diagram(coordinateSystem(extent = {{-100, -100}, {100, 100}}, preserveAspectRatio =
true, initialScale = 0.1, grid = {2, 2})));

end citarum
```

APPENDIX C: PYTHON SCRIPT – OPTIMIZATION

The following python scripting in this appendix might need to be slightly adjusted depending on the new development of RTC-Tools 2.0.

I. Goal programming of the sequences of hydrological objectives

The Python script below has been run in Docker that incorporate distributed applications integrated by RTC-2.0 Tools under 32-bits Python 2.7.3. This hydroeconomic optimization (~40 goals) took 20 minutes simulation time Intel(R) Core(TM) i5 2.5GHz. This python scripting is linked to the model schematization of the linear reservoirs.

```
"""
OTPIMIZATION of hydrological problems on the linear cascade reservoirs

Author:  Tiaravanni Hermawan
Date  :  July 28, 2016
"""

#IMPORT modules
from rtctools.optimization.collocated_integrated_optimization_problem import
CollocatedIntegratedOptimizationProblem
from rtctools.optimization.goal_programming_mixin import GoalProgrammingMixin,
Goal
from rtctools.optimization.modelica_mixin import ModelicaMixin
from rtctools.optimization.timeseries import Timeseries
from rtctools.optimization.csv_mixin import CSVMixin
from rtctools.util import run_optimization_problem
from casadi import MX
import logging
import numpy as np
import sys
import os
import csv
from time import sleep

logger = logging.getLogger("rtctools")
```

```
#PRIORITIES are based on RIBASIM, including rule curves and hedging rules
#ONLINE adjusted gate: Most upstream reservoir has the highest priority for a specific
rule curves
#IF NOT ONLINE: Most upstream reservoir has the highest priority

#TARGET before lowest hedging
#A class for a specific goal, either range of soft constraints or minimize a function
class Saguling_P10(Goal):
    def __init__(self, time, _min, _max):
        self.time = time
        self._min = _min
        self._max = _max

#OBJETIVE function on a Modelica variable
    def function(self, o):
        return o.state_at('reservoirCompact_Saguling.P', self.time)

    @property
    def min(self):
        return self._min

    @property
    def max(self):
        return self._max

    @property
    def function_range(self):
        return (0,1e11)

    # HIGHER priority =lower number
    @property
    def priority(self):
        return 1

class Cirata_P10(Goal):
    def __init__(self, time, _min, _max):
        self.time = time
        self._min = _min
        self._max = _max

    def function(self, o):
        return o.state_at('reservoirCompact_Cirata.P', self.time)

    @property
    def min(self):
        return self._min

    @property
    def max(self):
        return self._max

    @property
    def function_range(self):
        return (0,1e11)

    @property
    def priority(self):
        return 2

#Water level of hedging 4

class Saguling_H4(Goal):
    def __init__(self, time, _min, _max):
        self.time = time
        self._min = _min
        self._max = _max

    def function(self, o):
        return o.state_at('reservoirCompact_Saguling.H', self.time)

    @property
    def min(self):
        return self._min

    @property
    def max(self):
        return self._max

    @property
    def function_range(self):
        return (0,1e3)

    @property
    def priority(self):
        return 3

class Cirata_H4(Goal):
    def __init__(self, time, _min, _max):
        self.time = time
        self._min = _min
        self._max = _max

    def function(self, o):
        return o.state_at('reservoirCompact_Cirata.H', self.time)

    @property
    def min(self):
        return self._min

    @property
    def max(self):
        return self._max

    @property
    def function_range(self):
        return (0,1e3)

    @property
    def priority(self):
        return 4

class Jatiluhur_H4(Goal):
    def __init__(self, time, _min, _max):
        self.time = time
        self._min = _min
        self._max = _max

    def function(self, o):
        return o.state_at('reservoirCompact_Jatiluhur.H', self.time)

    @property
    def min(self):
        return self._min

    @property
    def max(self):
        return self._max

    @property
    def function_range(self):
        return (0,1e3)

    @property
    def priority(self):
        return 5
```

```
#TARGET

class Saguling_P30(Goal):
    def __init__(self, time, _min, _max):
        self.time = time
        self._min = _min
        self._max = _max

    def function(self, o):
        return o.state_at('reservoirCompact_Saguling.P', self.time)

    @property
    def min(self):
        return self._min

    @property
    def max(self):
        return self._max

    @property
    def function_range(self):
        return (0,1e11)

    @property
    def priority(self):
        return 6

class Cirata_P30(Goal):
    def __init__(self, time, _min, _max):
        self.time = time
        self._min = _min
        self._max = _max

    def function(self, o):
        return o.state_at('reservoirCompact_Cirata.P', self.time)

    @property
    def min(self):
        return self._min

    @property
    def max(self):
        return self._max
    @property
    def function_range(self):
        return (0,1e11)

    @property
    def priority(self):
        return 7

class Jatiluhur_P50(Goal):
    def __init__(self, time, _min, _max):
        self.time = time
        self._min = _min
        self._max = _max

    def function(self, o):
        return o.state_at('reservoirCompact_Jatiluhur.P', self.time)

    @property
    def min(self):
        return self._min

    @property
    def max(self):
        return self._max

    @property
    def function_range(self):
        return (0,1e11)

    @property
    def priority(self):
        return 8

class Agriculture_Q50(Goal):
    def __init__(self, time, _min, _max):
        self.time = time
        self._min = _min
        self._max = _max

    def function(self, o):
        return o.state_at('terminal_Agriculture.Q', self.time)

    @property
    def min(self):
        return self._min
    @property
    def max(self):
        return self._max

    @property
    def function_range(self):
        return (0,1e3)

    @property
    def priority(self):
        return 8

#LEVEL Hedging 3

class Saguling_H3(Goal):
    def __init__(self, time, _min, _max):
        self.time = time
        self._min = _min
        self._max = _max

    def function(self, o):
        return o.state_at('reservoirCompact_Saguling.H', self.time)

    @property
    def min(self):
        return self._min

    @property
    def max(self):
        return self._max

    @property
    def function_range(self):
        return (0,1e3)

    @property
    def priority(self):
        return 9

class Cirata_H3(Goal):
    def __init__(self, time, _min, _max):
        self.time = time
        self._min = _min
        self._max = _max
    def function(self, o):
        return o.state_at('reservoirCompact_Cirata.H', self.time)

    @property
    def min(self):
        return self._min

    @property
    def max(self):
        return self._max

    @property
    def function_range(self):
        return (0,1e3)

    @property
    def priority(self):
        return 10

class Jatiluhur_H3(Goal):
    def __init__(self, time, _min, _max):
        self.time = time
        self._min = _min
        self._max = _max

    def function(self, o):
        return o.state_at('reservoirCompact_Jatiluhur.H', self.time)

    @property
    def min(self):
        return self._min

    @property
    def max(self):
        return self._max

    @property
    def function_range(self):
        return (0,1e3)

    @property
    def priority(self):
        return 11
```

```
#TARGET

class Saguling_P50(Goal):
    def __init__(self, time, _min, _max):
        self.time = time
        self._min = _min
        self._max = _max

    def function(self, o):
        return o.state_at('reservoirCompact_Saguling.P', self.time)

    @property
    def min(self):
        return self._min

    @property
    def max(self):
        return self._max

    @property
    def function_range(self):
        return (0,1e11)

    @property
    def priority(self):
        return 5

class Cirata_P50(Goal):
    def __init__(self, time, _min, _max):
        self.time = time
        self._min = _min
        self._max = _max

    def function(self, o):
        return o.state_at('reservoirCompact_Cirata.P', self.time)

    @property
    def min(self):
        return self._min

    @property
    def max(self):
        return self._max
    @property
    def function_range(self):
        return (0,1e11)

    @property
    def priority(self):
        return 12

class Jatiluhur_P90(Goal):
    def __init__(self, time, _min, _max):
        self.time = time
        self._min = _min
        self._max = _max

    def function(self, o):
        return o.state_at('reservoirCompact_Jatiluhur.P', self.time)

    @property
    def min(self):
        return self._min

    @property
    def max(self):
        return self._max

    @property
    def function_range(self):
        return (0,1e11)

    # Because we want to satisfy our water level target first, this has a
    # higher priority (=lower number).
    @property
    def priority(self):
        return 13

class Agriculture_Q90(Goal):
    def __init__(self, time, _min, _max):
        self.time = time
        self._min = _min
        self._max = _max

    def function(self, o):
        return o.state_at('terminal_Agriculture.Q', self.time)

    @property
    def min(self):
        return self._min

    @property
    def max(self):
        return self._max

    @property
    def function_range(self):
        return (0,1e3)

    @property
    def priority(self):
        return 14

#LEVEL Hedging 2

class Saguling_H2(Goal):
    def __init__(self, time, _min, _max):
        self.time = time
        self._min = _min
        self._max = _max

    def function(self, o):
        return o.state_at('reservoirCompact_Saguling.H', self.time)

    @property
    def min(self):
        return self._min

    @property
    def max(self):
        return self._max

    @property
    def function_range(self):
        return (0,1e3)

    @property
    def priority(self):
        return 15



class Cirata_H2(Goal):
    def __init__(self, time, _min, _max):
        self.time = time
        self._min = _min
        self._max = _max

    def function(self, o):
        return o.state_at('reservoirCompact_Cirata.H', self.time)

    @property
    def min(self):
        return self._min

    @property
    def max(self):
        return self._max

    @property
    def function_range(self):
        return (0,1e3)

    @property
    def priority(self):
        return 16

class Jatiluhur_H2(Goal):
    def __init__(self, time, _min, _max):
        self.time = time
        self._min = _min
        self._max = _max

    def function(self, o):
        return o.state_at('reservoirCompact_Jatiluhur.H', self.time)

    @property
    def min(self):
        return self._min

    @property
    def max(self):
        return self._max

    @property
    def function_range(self):
        return (0,1e3)
    @property
    def priority(self):
        return 17

#TARGET

class Saguling_P70(Goal):
    def __init__(self, time, _min, _max):
        self.time = time
        self._min = _min
        self._max = _max

    def function(self, o):
        return o.state_at('reservoirCompact_Saguling.P', self.time)

    @property
    def min(self):
        return self._min

    @property
    def max(self):
        return self._max

    @property
    def function_range(self):
        return (0,1e11)

    @property
    def priority(self):
        return 18

class Cirata_P70(Goal):
    def __init__(self, time, _min, _max):
        self.time = time
        self._min = _min
        self._max = _max

    def function(self, o):
        return o.state_at('reservoirCompact_Cirata.P', self.time)

    @property
    def min(self):
        return self._min

    @property
    def max(self):
        return self._max

    @property
    def function_range(self):
        return (0,1e11)

    @property
    def priority(self):
        return 19

class Jatiluhur_PFirm(Goal):
    def __init__(self, time, _min, _max):
        self.time = time
        self._min = _min
        self._max = _max

    def function(self, o):
        return o.state_at('reservoirCompact_Jatiluhur.P', self.time)

    @property
    def min(self):
        return self._min

    @property
    def max(self):
        return self._max

    @property
    def function_range(self):
        return (0,1e11)

    @property
    def priority(self):
        return 20

class Agriculture_QOut(Goal):
    def __init__(self, time, _min, _max):
        self.time = time
        self._min = _min
        self._max = _max

    def function(self, o):
        return o.state_at('node_Agriculture.QOut_control[1]', self.time)
    @property
    def min(self):
        return self._min

    @property
    def max(self):
        return self._max

    @property
    def function_range(self):
        return (0,1e3)

    @property
    def priority(self):
        return 21

#LEVEL Hedging 1

class Saguling_H1(Goal):
    def __init__(self, time, _min, _max):
        self.time = time
        self._min = _min
        self._max = _max

    def function(self, o):
        return o.state_at('reservoirCompact_Saguling.H', self.time)

    @property
    def min(self):
        return self._min

    @property
    def max(self):
        return self._max

    @property
    def function_range(self):
        return (0,1e3)

    @property
    def priority(self):
        return 22


class Cirata_H1(Goal):
    def __init__(self, time, _min, _max):
        self.time = time
        self._min = _min
        self._max = _max

    def function(self, o):
        return o.state_at('reservoirCompact_Cirata.H', self.time)

    @property
    def min(self):
        return self._min

    @property
    def max(self):
        return self._max

    @property
    def function_range(self):
        return (0,1e3)

    @property
    def priority(self):
        return 23

class Jatiluhur_H1(Goal):
    def __init__(self, time, _min, _max):
        self.time = time
        self._min = _min
        self._max = _max

    def function(self, o):
        return o.state_at('reservoirCompact_Jatiluhur.H', self.time)

    @property
    def min(self):
        return self._min

    @property
    def max(self):
        return self._max

    @property
    def function_range(self):
        return (0,1e3)
    @property
    def priority(self):
        return 24

#TARGET

class Saguling_P90(Goal):
    def __init__(self, time, _min, _max):
        self.time = time
        self._min = _min
        self._max = _max

    def function(self, o):
        return o.state_at('reservoirCompact_Saguling.P', self.time)

    @property
    def min(self):
        return self._min

    @property
    def max(self):
        return self._max

    @property
    def function_range(self):
        return (0,1e11)

    @property
    def priority(self):
        return 25

class Cirata_P90(Goal):
    def __init__(self, time, _min, _max):
        self.time = time
        self._min = _min
        self._max = _max

    def function(self, o):
        return o.state_at('reservoirCompact_Cirata.P', self.time)

    @property
    def min(self):
        return self._min

    @property
    def max(self):
        return self._max

    @property
    def function_range(self):
        return (0,1e11)

    @property
    def priority(self):
        return 26

#LEVEL Firm

class Saguling_HFirm(Goal):
    def __init__(self, time, _min, _max):
        self.time = time
        self._min = _min
        self._max = _max

    def function(self, o):
        return o.state_at('reservoirCompact_Saguling.H', self.time)

    @property
    def min(self):
        return self._min

    @property
    def max(self):
        return self._max

    @property
    def function_range(self):
        return (0,1e3)

    @property
    def priority(self):
        return 27

class Cirata_HFirm(Goal):
    def __init__(self, time, _min, _max):
        self.time = time
        self._min = _min
        self._max = _max

    def function(self, o):
        return o.state_at('reservoirCompact_Cirata.H', self.time)

    @property
    def min(self):
        return self._min

    @property
    def max(self):
        return self._max

    @property
    def function_range(self):
        return (0,1e3)

    @property
    def priority(self):
        return 28

class Jatiluhur_HFirm(Goal):
    def __init__(self, time, _min, _max):
        self.time = time
        self._min = _min
        self._max = _max

    def function(self, o):
        return o.state_at('reservoirCompact_Jatiluhur.H', self.time)

    @property
    def min(self):
        return self._min

    @property
    def max(self):
        return self._max

    @property
    def function_range(self):
        return (0,1e3)

    @property
    def priority(self):
        return 29

#TARGET Firm
```

```
class Saguling_PFirm(Goal):
    def __init__(self, time, _min, _max):
        self.time = time
        self._min = _min
        self._max = _max

    def function(self, o):
        return o.state_at('reservoirCompact_Saguling.P', self.time)

    @property
    def min(self):
        return self._min

    @property
    def max(self):
        return self._max

    @property
    def function_range(self):
        return (0,1e11)

    @property
    def priority(self):
        return 30

class Cirata_PFirm(Goal):
    def __init__(self, time, _min, _max):
        self.time = time
        self._min = _min
        self._max = _max

    def function(self, o):
        return o.state_at('reservoirCompact_Cirata.P', self.time)

    @property
    def min(self):
        return self._min

    @property
    def max(self):
        return self._max

    @property
    def function_range(self):
        return (0,1e11)
    @property
    def priority(self):
        return 31

#LEVEL Target

class Saguling_HTarget(Goal):
    def __init__(self, time, _min, _max):
        self.time = time
        self._min = _min
        self._max = _max

    def function(self, o):
        return o.state_at('reservoirCompact_Saguling.H', self.time)

    @property
    def min(self):
        return self._min

    @property
    def max(self):
        return self._max

    @property
    def function_range(self):
        return (0,1e3)

    @property
    def priority(self):
        return 32

class Cirata_HTarget(Goal):
    def __init__(self, time, _min, _max):
        self.time = time
        self._min = _min
        self._max = _max

    def function(self, o):
        return o.state_at('reservoirCompact_Cirata.H', self.time)

    @property
    def min(self):
        return self._min
    @property
    def max(self):
        return self._max

    @property
    def function_range(self):
        return (0,1e3)

    @property
    def priority(self):
        return 33

class Jatiluhur_HTarget(Goal):
    def __init__(self, time, _min, _max):
        self.time = time
        self._min = _min
        self._max = _max

    def function(self, o):
        return o.state_at('reservoirCompact_Jatiluhur.H', self.time)

    @property
    def min(self):
        return self._min

    @property
    def max(self):
        return self._max

    @property
    def function_range(self):
        return (0,1e3)

    @property
    def priority(self):
        return 34

#OPTIMIZATION problem
class CitarumGP(GoalProgrammingMixin, CSVMixin, ModelicaMixin,
CollocatedIntegratedOptimizationProblem):
    def __init__(self, model_folder, input_folder, output_folder):

        # Call constructors
        GoalProgrammingMixin.__init__(self)
        CSVMixin.__init__(self,
                 input_folder=input_folder,
                 output_folder=output_folder,
                 equidistant=False)
    ModelicaMixin.__init__(self,
                    model_name='citarum',
                    model_folder=model_folder,
                    #HEADER of .csv time series
                    constant_inputs=['inflow_Saguling_Q',
                                'inflow_Cirata_Q',
                                'inflow_Jatiluhur_Q',
                                'node_Agriculture_Q50',
                                'node_Agriculture_Q90',
                                'node_Agriculture_QOut',
                                'Saguling_evaporation',
                                'Cirata_evaporation',
                                'Jatiluhur_evaporation',
                                'Saguling_H4',
                                'Saguling_H3',
                                'Saguling_H2',
                                'Saguling_H1',
                                'Saguling_HFirm',
                                'Saguling_HTarget',
                                'SagulingHFlood',
                                'Cirata_H4',
                                'Cirata_H3',
                                'Cirata_H2',
                                'Cirata_H1',
                                'Cirata_HFirm',
                                'Cirata_HTarget',
                                'CirataHFlood',
                                'Jatiluhur_H4',
                                'Jatiluhur_H3',
                                'Jatiluhur_H2',
                                'Jatiluhur_H1',
                                'Jatiluhur_HFirm',
                                'Jatiluhur_HTarget',
                                'JatiluhurHFlood'])
    CollocatedIntegratedOptimizationProblem.__init__(self)

    # We keep track of our intermediate results, so that we can print some
    # information about the progress of goals at the end of our run.
    self.intermediate_results = []


    # Store settings
```

```
        #self.output_folder = output_folder

    def bounds(self):
        # HARD CONSTRAINTS
        return {'reservoirCompact_Saguling.P': (0.0, 700000000),
                'reservoirCompact_Cirata.P': (0.0, 1008000000),
                'reservoirCompact_Jatiluhur.P': (0.0, 180000000),
                'reservoirCompact_Jatiluhur.H': (74.89,106.89),
                'reservoirCompact_Cirata.H': (180, 220),
                'reservoirCompact_Saguling.H': (622.6, 643),
                'reservoirCompact_Saguling.Q_spill': (0.0, None),
                'reservoirCompact_Cirata.Q_spill': (0.0, None),
                'reservoirCompact_Jatiluhur.Q_spill': (0.0, None),
                'reservoirCompact_Saguling.Q_turbine': (0.0, None),
                'reservoirCompact_Cirata.Q_turbine': (0.0, None),
                'reservoirCompact_Jatiluhur.Q_turbine': (0.0, None),
                'node_Agriculture.QOut_control[1]': (0,
self.timeseries('node_Agriculture_QOut')),
                'terminal_River.Q':(0.0, None),
                'reservoirCompact_Saguling.V':(0.0, None),
                'reservoirCompact_Cirata.V':(0.0, None),
                'reservoirCompact_Jatiluhur.V':(0.0, None)}

    @property
    def goals(self):
        g = []
        # Use a for loop to add goals for every time step
        for t in self.times():
            g.append(Saguling_P10(t, 13440860,700000000))
            g.append(Saguling_P30(t, 40322581,700000000))
            g.append(Saguling_P50(t, 67204301,700000000))
            g.append(Saguling_P70(t, 94086022,700000000))
            g.append(Saguling_P90(t, 120967742,700000000))
            g.append(Saguling_PFirm(t, 134402602,700000000))

            g.append(Cirata_P10(t, 8064516,1008000000))
            g.append(Cirata_P30(t, 24193548,1008000000))
            g.append(Cirata_P50(t, 40322581,1008000000))
            g.append(Cirata_P70(t, 56451613,1008000000))
            g.append(Cirata_P90(t, 72580645,1008000000))
            g.append(Cirata_PFirm(t, 80645161,1008000000))

            g.append(Jatiluhur_P50(t, 46841398,187000000))
            g.append(Jatiluhur_P90(t, 84314516,187000000))
            g.append(Jatiluhur_PFirm(t, 93682796,187000000))
        g.append(Saguling_H4(t, self.interpolate(t, self.timeseries('Saguling_H4').times,
self.timeseries('Saguling_H4').values), self.interpolate(t,
self.timeseries('SagulingHFlood').times, self.timeseries('SagulingHFlood').values)))
        g.append(Saguling_H3(t, self.interpolate(t, self.timeseries('Saguling_H3').times,
self.timeseries('Saguling_H3').values), self.interpolate(t,
self.timeseries('SagulingHFlood').times, self.timeseries('SagulingHFlood').values)))
        g.append(Saguling_H3(t, self.interpolate(t, self.timeseries('Saguling_H2').times,
self.timeseries('Saguling_H2').values), self.interpolate(t,
self.timeseries('SagulingHFlood').times, self.timeseries('SagulingHFlood').values)))
        g.append(Saguling_H1(t, self.interpolate(t, self.timeseries('Saguling_H1').times,
self.timeseries('Saguling_H1').values), self.interpolate(t,
self.timeseries('SagulingHFlood').times, self.timeseries('SagulingHFlood').values)))
        g.append(Saguling_HFirm(t, self.interpolate(t,
self.timeseries('Saguling_HFirm').times, self.timeseries('Saguling_HFirm').values),
self.interpolate(t, self.timeseries('SagulingHFlood').times,
self.timeseries('SagulingHFlood').values)))
        g.append(Saguling_HTarget(t, self.interpolate(t,
self.timeseries('Saguling_HTarget').times, self.timeseries('Saguling_HTarget').values),
self.interpolate(t, self.timeseries('SagulingHFlood').times,
self.timeseries('SagulingHFlood').values)))

        g.append(Cirata_H4(t, self.interpolate(t, self.timeseries('Cirata_H4').times,
self.timeseries('Cirata_H4').values), self.interpolate(t,
self.timeseries('CirataHFlood').times, self.timeseries('CirataHFlood').values)))
        g.append(Cirata_H3(t, self.interpolate(t, self.timeseries('Cirata_H3').times,
self.timeseries('Cirata_H3').values), self.interpolate(t,
self.timeseries('CirataHFlood').times, self.timeseries('CirataHFlood').values)))
        g.append(Cirata_H3(t, self.interpolate(t, self.timeseries('Cirata_H2').times,
self.timeseries('Cirata_H2').values), self.interpolate(t,
self.timeseries('CirataHFlood').times, self.timeseries('CirataHFlood').values)))
        g.append(Cirata_H1(t, self.interpolate(t, self.timeseries('Cirata_H1').times,
self.timeseries('Cirata_H1').values), self.interpolate(t,
self.timeseries('CirataHFlood').times, self.timeseries('CirataHFlood').values)))
        g.append(Cirata_HFirm(t, self.interpolate(t,
self.timeseries('Cirata_HFirm').times, self.timeseries('Cirata_HFirm').values),
self.interpolate(t, self.timeseries('CirataHFlood').times,
self.timeseries('CirataHFlood').values)))
        g.append(Cirata_HTarget(t, self.interpolate(t,
self.timeseries('Cirata_HTarget').times, self.timeseries('Cirata_HTarget').values),
self.interpolate(t, self.timeseries('CirataHFlood').times,
self.timeseries('CirataHFlood').values)))
```

```
        g.append(Jatiluhur_H4(t, self.interpolate(t, self.timeseries('Jatiluhur_H4').times,
self.timeseries('Jatiluhur_H4').values), self.interpolate(t,
self.timeseries('JatiluhurHFlood').times, self.timeseries('JatiluhurHFlood').values)))
        g.append(Jatiluhur_H3(t, self.interpolate(t, self.timeseries('Jatiluhur_H3').times,
self.timeseries('Jatiluhur_H3').values), self.interpolate(t,
self.timeseries('JatiluhurHFlood').times, self.timeseries('JatiluhurHFlood').values)))
        g.append(Jatiluhur_H3(t, self.interpolate(t, self.timeseries('Jatiluhur_H2').times,
self.timeseries('Jatiluhur_H2').values), self.interpolate(t,
self.timeseries('JatiluhurHFlood').times, self.timeseries('JatiluhurHFlood').values)))
        g.append(Jatiluhur_H1(t, self.interpolate(t, self.timeseries('Jatiluhur_H1').times,
self.timeseries('Jatiluhur_H1').values), self.interpolate(t,
self.timeseries('JatiluhurHFlood').times, self.timeseries('JatiluhurHFlood').values)))
        g.append(Jatiluhur_HFirm(t, self.interpolate(t,
self.timeseries('Jatiluhur_HFirm').times, self.timeseries('Jatiluhur_HFirm').values),
self.interpolate(t, self.timeseries('JatiluhurHFlood').times,
self.timeseries('JatiluhurHFlood').values)))
        g.append(Jatiluhur_HTarget(t, self.interpolate(t,
self.timeseries('Jatiluhur_HTarget').times, self.timeseries('Jatiluhur_HTarget').values),
self.interpolate(t, self.timeseries('JatiluhurHFlood').times,
self.timeseries('JatiluhurHFlood').values)))

        g.append(Agriculture_Q50(t, self.interpolate(t,
self.timeseries('node_Agriculture_Q50').times,
self.timeseries('node_Agriculture_Q50').values), self.interpolate(t,
self.timeseries('node_Agriculture_QOut').times,
self.timeseries('node_Agriculture_QOut').values)))
        g.append(Agriculture_Q90(t, self.interpolate(t,
self.timeseries('node_Agriculture_Q90').times,
self.timeseries('node_Agriculture_Q90').values), self.interpolate(t,
self.timeseries('node_Agriculture_QOut').times,
self.timeseries('node_Agriculture_QOut').values)))
        g.append(Agriculture_QOut(t, self.interpolate(t,
self.timeseries('node_Agriculture_QOut').times,
self.timeseries('node_Agriculture_QOut').values), self.interpolate(t,
self.timeseries('node_Agriculture_QOut').times,
self.timeseries('node_Agriculture_QOut').values)))

      return g

# Run
run_optimization_problem(CitarumGP, base_folder='..')
```

II. Conventional linear programming of a hydroeconomic objective

The Python script below has been run in Docker that incorporate distributed applications integrated by RTC-2.0 Tools under 32-bits Python 2.7.3. This hydroeconomic optimization took 3 minutes simulation time (~150 iterations) in Intel(R) Core(TM) i5 2.5GHz. The simulation time is likely to strongly depend on the formulation of the objective function. This python scripting is linked to the model schematization of the linear reservoirs.

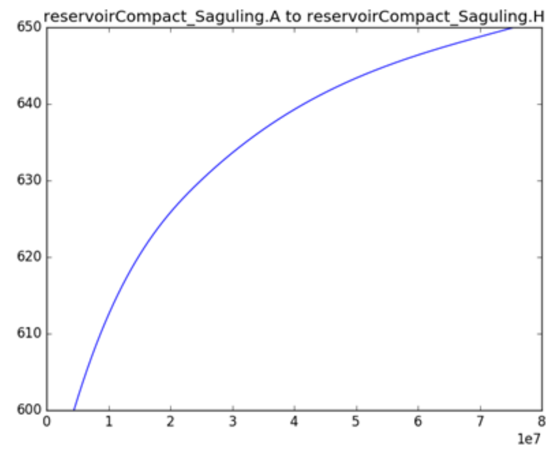
```
#IMPORT modules
from rtctools.optimization.collocated_integrated_optimization_problem import
CollocatedIntegratedOptimizationProblem
from rtctools.optimization.modelica_mixin import ModelicaMixin
from rtctools.optimization.timeseries import Timeseries
from rtctools.optimization.csv_mixin import CSVMixin
from rtctools.util import run_optimization_problem
from casadi import MX
import logging
import numpy as np
import sys
import os
logger = logging.getLogger("rtctools")

#DEFINE the optimization problem
class TestProblem(CSVMixin, ModelicaMixin,
CollocatedIntegratedOptimizationProblem):
    def __init__(self, model_folder, input_folder, output_folder):
        # CALL constructors
        CSVMixin.__init__(self,
                         input_folder=input_folder,
                         output_folder=output_folder,
                         equidistant=False)
        ModelicaMixin.__init__(self,
                              model_name='citarum',
                              model_folder=model_folder,
                              constant_inputs=['inflow_Saguling_Q',
                                               'inflow_Cirata_Q',
                                               'inflow_Jatiluhur_Q',
                                              'node_Agriculture_QOut',
                                              'Saguling_evaporation',
                                              'Cirata_evaporation',
                                              'Jatiluhur_evaporation',
                                              'value_Floodinitial'])
        CollocatedIntegratedOptimizationProblem.__init__(self)
        # STORE settings
        self.output_folder = output_folder

    #DEFINE a single hydroeconomic objective function
    def objective(self,ensemble_member):
        # MAXIMIZE generation
        return -self.state_at('benefit_Total', self.times()[-1])

    def bounds(self):
        # HARD CONSTRAINT
        return {'reservoirCompact_Saguling.P': (0.0, 700000000),
                'reservoirCompact_Cirata.P': (0.0, 1008000000),
                'reservoirCompact_Jatiluhur.P': (0.0, 180000000),
                'reservoirCompact_Jatiluhur.H': (74.89,106.89),
                'reservoirCompact_Cirata.H': (180, 220),
                'reservoirCompact_Saguling.H': (622.6, 643),
                'reservoirCompact_Saguling.Q_spill': (0.0, None),
                'reservoirCompact_Cirata.Q_spill': (0.0, None),
                'reservoirCompact_Jatiluhur.Q_spill': (0.0, None),
                'reservoirCompact_Saguling.Q_turbine': (0.0, None),
                'reservoirCompact_Cirata.Q_turbine': (0.0, None),
                'reservoirCompact_Jatiluhur.Q_turbine': (0.0, None),
                 'node_Agriculture.QOut_control[1]': (0.0,
                                                  self.timeseries('node_Agriculture_QOut')),
                'terminal_River.Q':(0.0, None),
                #Environmental flow 'terminal_River.Q':(1.4, None),
                'reservoirCompact_Saguling.V':(0.0, None),
                'reservoirCompact_Cirata.V':(0.0, None),
                'reservoirCompact_Jatiluhur.V':(0.0, None)}

    def constraints(self,ensemble_member):
        return [ ]

# RUN optimization problem
run_optimization_problem(TestProblem, log_level=logging.INFO)
```

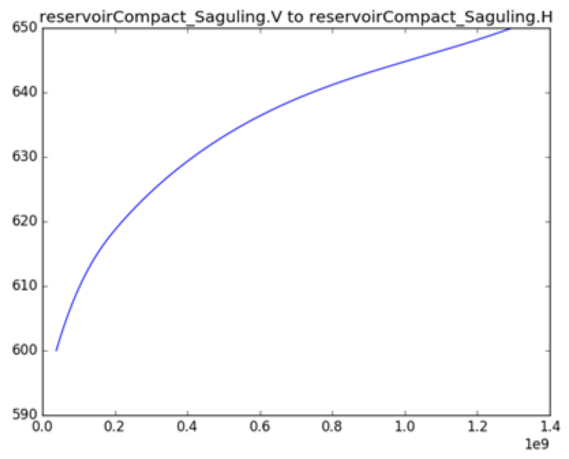
III. Conventional linear programming of a hydroeconomic objective (lookup table)

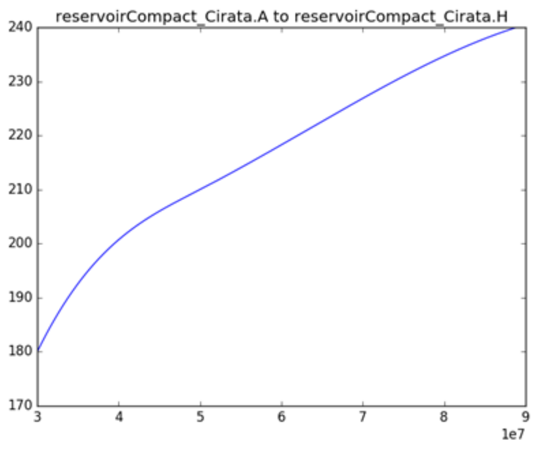
The Python script below has been run in Docker that incorporates distributed applications integrated by RTC-2.0 Tools under 32-bits Python 2.7.3. This hydroeconomic optimization took 30 minutes simulation time in Intel(R) Core(TM) i5 2.5GHz. The simulation time is likely to strongly depend on the formulation of the objective function. This python scripting is linked to the model schematization of the look-up table reservoirs. The curves fitting of the look-up table of reservoirs are presented below.

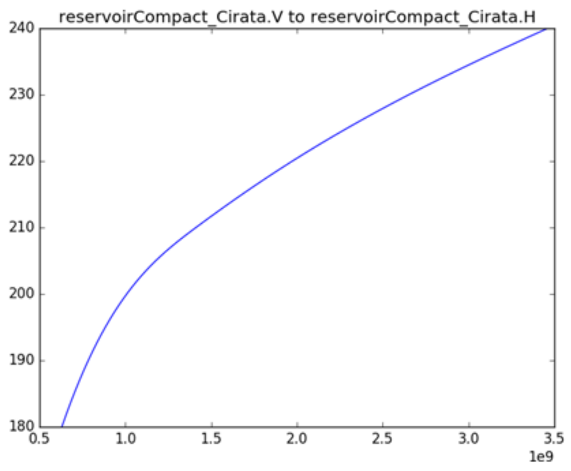
```
"""
Linear programming/linear optimization of a single hydroeconomic objective funtion

Author:  Tiaravanni Hermawan
Date  :  11 August, 2016
"""


from rtctools.optimization.collocated_integrated_optimization_problem import
CollocatedIntegratedOptimizationProblem
from rtctools.optimization.goal_programming_mixin import GoalProgrammingMixin,
Goal
from rtctools.optimization.modelica_mixin import ModelicaMixin
from rtctools.optimization.csv_lookup_table_mixin import CSVLookupTableMixin
from rtctools.optimization.timeseries import Timeseries
from rtctools.optimization.csv_mixin import CSVMixin
from rtctools.util import run_optimization_problem
from casadi import MX
import logging
import numpy as np
import sys
import os
import csv
from time import sleep

import numpy as np
from abc import ABCMeta, abstractmethod
from sets import Set
import scipy.interpolate
import itertools
import logging
import glob
import datetime

import matplotlib
matplotlib.use('Agg')
```

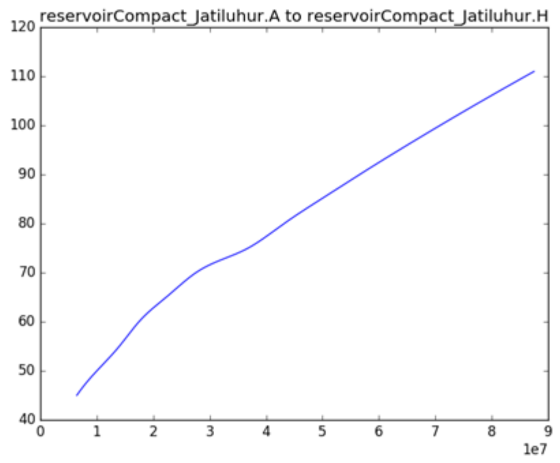
```
import pylab

logger = logging.getLogger("rtctools")

class MaximizeBenefit(Goal):
    def function(self, optimization_problem):
        return -optimization_problem.integral('benefit_total')

    @property
    def function_range(self):
        return (1e2, 1e5)

    @property
    def priority(self):
        return 1

class Lookup(CSVLookupTableMixin,GoalProgrammingMixin, CSVMixin,
ModelicaMixin, CollocatedIntegratedOptimizationProblem):
    def __init__(self, model_folder, input_folder, output_folder):
        #lookup_tables = [splitext(f)[0] for f in listdir(join(input_folder, 'lookup_tables'))]
        # Call constructors
        CSVLookupTableMixin.__init__(self, input_folder=input_folder)
        GoalProgrammingMixin.__init__(self)
        CSVMixin.__init__(self,
                          input_folder=input_folder,
                          output_folder=output_folder,
                          equidistant=False)
        ModelicaMixin.__init__(self,
                               model_name='citarum',
                               model_folder=model_folder,
                               constant_inputs=['inflow_Saguling_Q',
                                                'inflow_Cirata_Q',
                                                'inflow_Jatiluhur_Q',
                                                'node_Agriculture_QOut',
                                                'Saguling_evaporation',
                                                'Cirata_evaporation',
                                                'Jatiluhur_evaporation',
                                lookup_tables=['Saguling_area',
                                         'Saguling_volume'
                                         'Cirata_area'
                                         'Cirata_volume'
                                         'Jatiluhur_area'
                                         'Jatiluhur_volume'])
        CollocatedIntegratedOptimizationProblem.__init__(self)
        for filename in glob.glob(os.path.join(self, input_folder, "lookup_tables/*.csv")):
            logger.debug("Reading lookup tables from {}".format(filename))

            csvinput = np.genfromtxt(filename, delimiter=",", dtype=None, names=True,
deletechars="")

            input_name = csvinput.dtype.names[0]
            input_values = csvinput[input_name]
            for output_name in csvinput.dtype.names[1:]:
                logger.debug("Reading lookup table from {} to {}".format(input_name,
output_name))
               #Plot lookup table
                tck = scipy.interpolate.splrep(input_values, csvinput[output_name], k=3, s=0)
                t_ = np.linspace(input_values[0], input_values[-1], 1000)
                o = scipy.interpolate.splev(t_, tck)
                pylab.clf()
                pylab.plot(t_, o)
                pylab.title(input_name + ' to ' + output_name)
                pylab.savefig(os.path.join(self, output_folder, input_name.replace(':','_') + '_' +
output_name.replace(':','_') + '.png'))

                logger.debug("Done computing B-Spline coefficients")

        # We keep track of our intermediate results, so that we can print some
        # information about the progress of goals at the end of our run.
        self.intermediate_results = []

        # Store settings
        #self.output_folder = output_folder

    @property
    def bounds(self):
        # Bounds
        return {'reservoirCompact_Saguling.P': (0.0, 700000000),
                'reservoirCompact_Cirata.P': (0.0, 1008000000),
                'reservoirCompact_Jatiluhur.P': (0.0, 180000000),
                'reservoirCompact_Jatiluhur.H': (74.89,106.89),
                'reservoirCompact_Cirata.H': (180, 220),
                'reservoirCompact_Saguling.H': (622.6, 643),
                'reservoirCompact_Saguling.Q_spill': (0.0, None),
                'reservoirCompact_Cirata.Q_spill': (0.0, None),
                'reservoirCompact_Jatiluhur.Q_spill': (0.0, None),
                'reservoirCompact_Saguling.Q_turbine': (0.0, None),
                'reservoirCompact_Cirata.Q_turbine': (0.0, None),
                'reservoirCompact_Jatiluhur.Q_turbine': (0.0, None),
             'node_Agriculture.QOut_control[1]': (0,
self.timeseries('node_Agriculture_QOut')),
             'terminal_River.Q':(0.0, None),
             'reservoirCompact_Saguling.V':(0.0, None),
             'reservoirCompact_Cirata.V':(0.0, None),
             'reservoirCompact_Jatiluhur.V':(0.0, None)}
   @property
   def goals(self):
      g = []
      g.append(MaximizeBenefit())
      return g

# Run
run_optimization_problem(Lookup, base_folder='..')
```

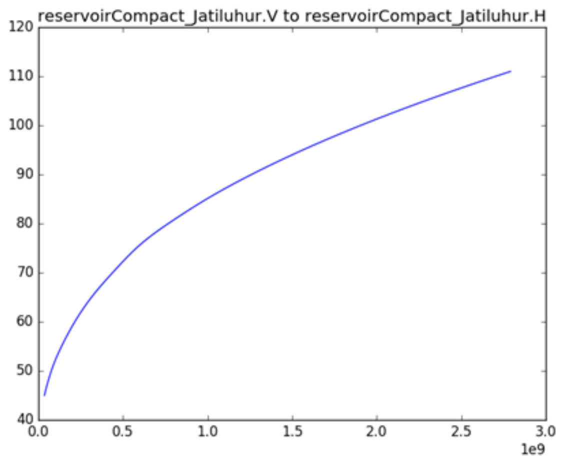

reservoirCompact_Saguling.A to reservoirCompact_Saguling.H
650
640
630
620
610
600
0
1
2
3
4
5
6
7
8
1e7


reservoirCompact_Saguling.V to reservoirCompact_Saguling.H
650
640
630
620
610
600
590
0.0
0.2
0.4
0.6
0.8
1.0
1.2
1.4
1e9


reservoirCompact_Cirata.A to reservoirCompact_Cirata.H
240
230
220
210
200
190
180
170
3
4
5
6
7
8
9
1e7


reservoirCompact_Cirata.V to reservoirCompact_Cirata.H
240
230
220
210
200
190
180
0.5
1.0
1.5
2.0
2.5
3.0
3.5
1e9


reservoirCompact_Jatiluhur.A to reservoirCompact_Jatiluhur.H
120
110
100
90
80
70
60
50
40
0
1
2
3
4
5
6
7
8
9
1e7


reservoirCompact_Jatiluhur.V to reservoirCompact_Jatiluhur.H
120
110
100
90
80
70
60
50
40
0.0
0.5
1.0
1.5
2.0
2.5
3.0
1e9

IV. Hybrid optimization between goal programming and conventional linear programming

The Python script below has been run in Docker that incorporates distributed applications integrated by RTC-2.0 Tools under 32-bits Python 2.7.3. This hydroeconomic optimization (~4 sequences of objectives and a single hydroeconomic objective) took 5 minutes simulation time Intel(R) Core(TM) i5 2.5GHz.

```
"""
Hybrid optimization on the linear cascade reservoirs
Author:  Tiaravanni Hermawan
Date  :  August 10, 2016
"""

from rtctools.optimization.collocated_integrated_optimization_problem import
CollocatedIntegratedOptimizationProblem
from rtctools.optimization.goal_programming_mixin import GoalProgrammingMixin,
Goal
from rtctools.optimization.modelica_mixin import ModelicaMixin
from rtctools.optimization.timeseries import Timeseries
from rtctools.optimization.csv_mixin import CSVMixin
from rtctools.util import run_optimization_problem
from casadi import MX
import logging
import numpy as np
import sys
import os

logger = logging.getLogger("rtctools")

#LATEST PRIORITY
class MaxTotalBenefit(Goal):
    # If we do not specify any minimum or maximum value in this class, the
    # goal programming mixin will try to minimize the following function.
    def function(self, optimization_problem):
        # Maximize generation
        return optimization_problem.state_at('benefit_Total',
optimization_problem.times()[-1])

    # Every goal needs a rough (over)estimate of the range of the function
    # defined above.
    # decent estimate.
    @property
    def function_range(self):
        return (0, 1e6)
    # The lower the number returned by this function, the higher the priority.
    @property
    def priority(self):
        return 100

#HIGHEST PRIORITY: FIRM DEMAND
#Put the firm target of the cascade linear reservoirs in the similar level

class Saguling_PFirm(Goal):
    def __init__(self, time, _min, _max):
        self.time = time
        self._min = _min
        self._max = _max

    def function(self, o):
        return o.state_at('reservoirCompact_Saguling.P', self.time)

    @property
    def min(self):
        return self._min

    @property
    def max(self):
        return self._max

    @property
    def function_range(self):
        return (0,1e12)

    # Lowest number is the highest priority
    @property
    def priority(self):
        return 1

class Cirata_PFirm(Goal):
    def __init__(self, time, _min, _max):
        self.time = time
        self._min = _min
        self._max = _max

    def function(self, o):
        return o.state_at('reservoirCompact_Cirata.P', self.time)

    @property
    def min(self):
        return self._min

    @property
    def max(self):
        return self._max

    @property
    def function_range(self):
        return (0,1e12)

    @property
    def priority(self):
        return 1

class Jatiluhur_PFirm(Goal):
    def __init__(self, time, _min, _max):
        self.time = time
        self._min = _min
        self._max = _max

    def function(self, o):
        return o.state_at('reservoirCompact_Jatiluhur.P', self.time)

    @property
    def min(self):
        return self._min

    @property
    def max(self):
        return self._max

    @property
    def function_range(self):
        return (0,1e12)

    @property
    def priority(self):
        return 1

class Agriculture_QOut(Goal):
    def __init__(self, time, _min, _max):
        self.time = time
        self._min = _min
        self._max = _max
    def function(self, o):
        return o.state_at('terminal_Agriculture.Q', self.time)

    @property
    def min(self):
        return self._min

    @property
    def max(self):
        return self._max

    @property
    def function_range(self):
        return (0,1e3)

    @property
    def priority(self):
        return 1

class Jatiluhur_QOut(Goal):
    def __init__(self, time, _min, _max):
        self.time = time
        self._min = _min
        self._max = _max

    def function(self, o):
        return o.state_at('out_Jatiluhur_Q', self.time)

    @property
    def min(self):
        return self._min

    @property
    def max(self):
        return self._max

    @property
    def function_range(self):
        return (0,1e3)

    @property
    def priority(self):
        return 1
```

```
#OPTIMIZATION PROBLEM
class Hybrid(GoalProgrammingMixin, CSVMixin, ModelicaMixin,
CollocatedIntegratedOptimizationProblem):
    def __init__(self, model_folder, input_folder, output_folder):
        # Call constructors
        GoalProgrammingMixin.__init__(self)
        CSVMixin.__init__(self,
                          input_folder=input_folder,
                          output_folder=output_folder,
                          equidistant=False)
        ModelicaMixin.__init__(self,
                               model_name='citarum',
                               model_folder=model_folder,
                               constant_inputs=['inflow_Saguling_Q',
                                                'inflow_Cirata_Q',
                                                'inflow_Jatiluhur_Q',
                                                'node_Agriculture_QOut',
                                                'Saguling_evaporation',
                                                'Cirata_evaporation',
                                                'Jatiluhur_evaporation',
                                                'value_Floodinitial'])
        CollocatedIntegratedOptimizationProblem.__init__(self)

        # We keep track of our intermediate results, so that we can print some
        # information about the progress of goals at the end of our run.
        self.intermediate_results = []

    def bounds(self):
        #HARD CONSTRAINTS
        return {'reservoirCompact_Saguling.P': (0.0, 700000000),
                'reservoirCompact_Cirata.P': (0.0, 1008000000),
                'reservoirCompact_Jatiluhur.P': (0.0, 180000000),
                'reservoirCompact_Jatiluhur.H': (74.89,106.89),
                'reservoirCompact_Cirata.H': (180, 220),
                'reservoirCompact_Saguling.H': (622.6, 643),
                'reservoirCompact_Saguling.Q_spill': (0.0, None),
                'reservoirCompact_Cirata.Q_spill': (0.0, None),
                'reservoirCompact_Jatiluhur.Q_spill': (0.0, None),
                'reservoirCompact_Saguling.Q_turbine': (0.0, None),
                'reservoirCompact_Cirata.Q_turbine': (0.0, None),
                'reservoirCompact_Jatiluhur.Q_turbine': (0.0, None),
                'node_Agriculture.QOut_control[1]': (0.0,
                                                     self.timeseries('node_Agriculture_QOut')),
                'terminal_River.Q':(1.4, None),
    @property
    def goals(self):
        g = []
        #MAX Total benefit at the latest goal
        g.append(MaxTotalBenefit())

        #SATISFY firm demand as soft constraints at the higher goal
        # Use a for loop to add goals for every time step
        for t in self.times():
           g.append(Saguling_PFirm(t, 148814016.428571,700000000))
           g.append(Cirata_PFirm(t, 89285713.9642857,1008000000))
           g.append(Jatiluhur_PFirm(t, 103720238.428571,187000000))
           g.append(Agriculture_QOut(t, 0, self.interpolate(t,
                       self.timeseries('node_Agriculture_QOut').times,
                       self.timeseries('node_Agriculture_QOut').values)))
           g.append(Jatiluhur_QOut(t, 57.6,200))

        return g

    def configure_solver(self, options):
        # Guideline: O(goal tolerance) = |QSTGoal.function_range| * tol
        options['tol']=1e-6
        options['expand']=True

# Run
run_optimization_problem(Hybrid, log_level=logging.DEBUG)
```

APPENDIX D: PYTHON SCRIPT – BATCH FILE

The batch file below is developed to automatically run RTC-Tools 2.0 several times with different input data but similar objective functions.

```
#DEFINE folder location
directory = os.path.join(sys.path[0], '../longinput/')
directory2 = os.path.join(sys.path[0], '../input/')
directory3 = os.path.join(sys.path[0], '../output/')

#SPECIFY the length of cutting and simulation period
cutting=12
ts=1200
run=int(ts/cutting)
numrun=0

# EXTRACT output file of the cut time series to a single extract.csv file (function)
def extract():
    for o in range(cutting):
          f_ext.write(line_out[(o+2)])
          f_out.close()

for numrun in range(run):
    print numrun
    #CUT time series input
    for subdir, dirs, files in os.walk(directory):
        for file in files:
            o=0
            f=open(directory+file,'rb')
            with open(directory+file,'rb') as openfile:
                listsdata = []
                for linedata in openfile:
                    linesdata = linedata.split(',')
                    linesdata[-1] = linesdata[-1].replace('\n', '').replace('\r', '')
                    listsdata.append(linesdata)
            i=0
            f_cut = open(directory2+file,'wb')
            #lines = f.readlines()
            a=csv.writer(f_cut, delimiter=",", dtype=None, names=True, deletechars="")
            a.writerow(listsdata[i])
            #f_cut.write(listsdata[i])
            for i in range(cutting+2):
                #print lines[(cutting*(numrun))+(i+1)]
                a.writerow(listsdata[(cutting*(numrun))+(i+1)])
                f.close()

    #EXECUTE run.py in RTC-Tools 2.0
    os.system("python run.py")

    with open(directory3+'timeseries_export.csv','r') as openfile:
        lines = []
        for line in openfile:
            lists = line.split(',')
            lines.append(lists)

        column_jat = lines[0].index('reservoirCompact_Jatiluhur.H')
        value_jat = float(lines[-1][column_jat])
        column_cir = lines[0].index('reservoirCompact_Cirata.H')
        value_cir = float(lines[-1][column_cir])
        column_sag = lines[0].index('reservoirCompact_Saguling.H')
        value_sag = float(lines[-1][column_sag])

    head=['reservoirCompact_Saguling.H', 'reservoirCompact_Cirata.H',
                'reservoirCompact_Jatiluhur.H']
    b= [value_sag,value_cir,value_jat]

    f_init = open(directory2+'initial_state.csv','w')
    f_init.write('reservoirCompact_Saguling.H, reservoirCompact_Cirata.H,
                        reservoirCompact_Jatiluhur.H\n')
    a=csv.writer(f_init)
    a.writerow(b)
    f_init.close()

    #EXTRACT output file
    f_out=open(directory3+'timeseries_export.csv','r')
    f_ext=open(directory3+'extract.csv','a')
    line_out = f_out.readlines()

    if numrun==0:
        f_ext.write(line_out[0])
        extract()
    else:
        extract()
```