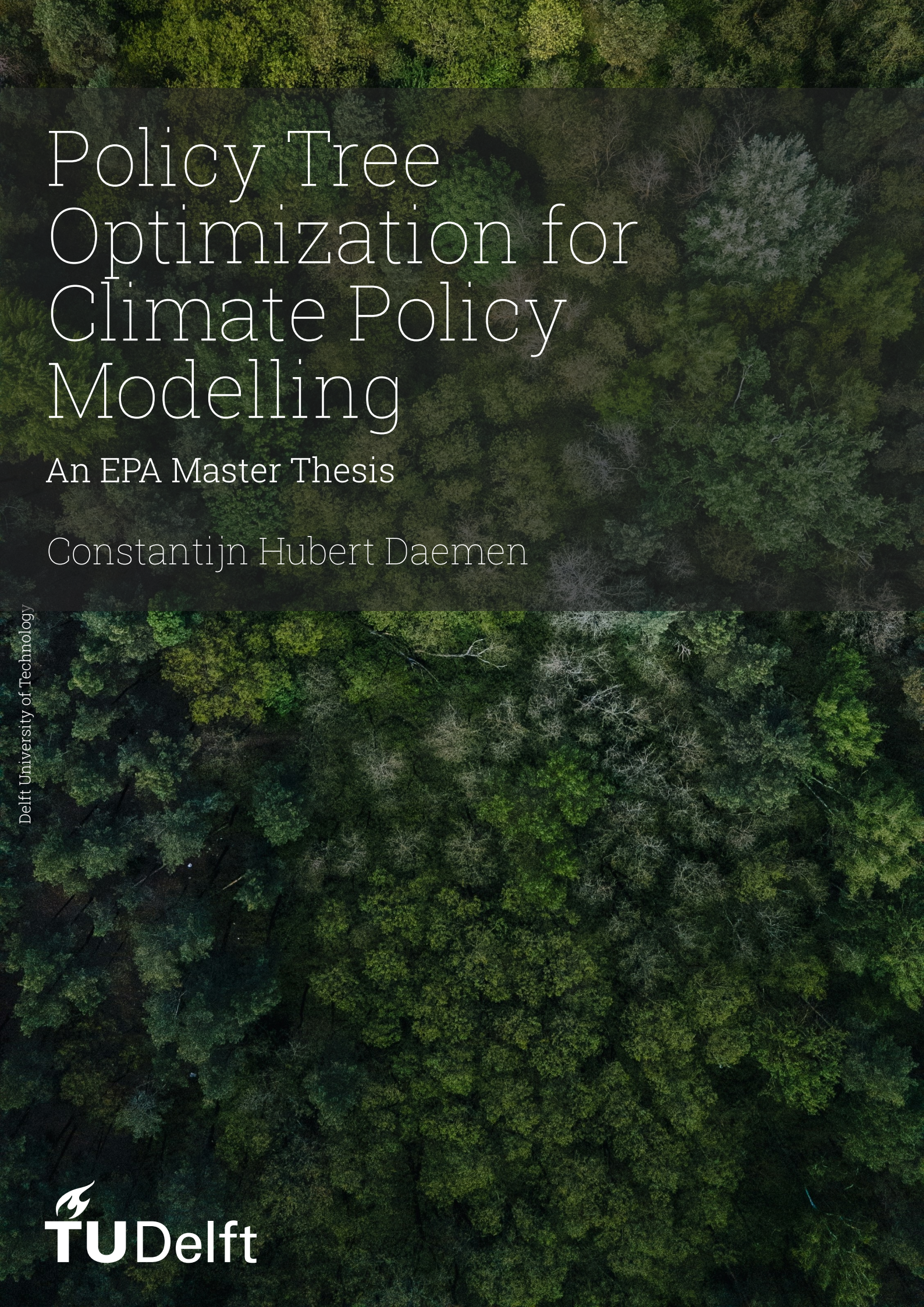

Policy Tree Optimization for Climate Policy Modelling

An EPA Master Thesis

Constantijn Hubert Daemen

Delft University of Technology

TU Delft

Policy Tree Optimization for Climate Policy Modelling

An EPA Master Thesis

by

Constantijn Hubert Daemen

Student Number 4561694
Contact stijndaemensk@gmail.com

First Supervisor: J. Salazar
Second Supervisor: M. Warnier
Daily Supervisor: D. Akoluk
Project Duration: April, 2023 - December, 2023
Faculty: Faculty of Technology, Policy and Management, Delft

The associated code is available at:
https://github.com/StijnDaemen/Thesis_Project_Trees

TU Delft

Preface

I would like to thank the supervisors of this project because they are the ones that made this thesis not only possible but also enjoyable. I feel a lot of gratitude towards the people that showed me through constructive criticism and casual conversation the fun in research and the excitement in developing something new. I owe special thanks to my direct supervisors who have helped me countless times when I was stuck in my code. It was their brief advice that would often turn days of debugging into seconds. Damla was always available whenever I had a question and she would make sure I could carry on with my project whenever I got stuck. Besides all the professional help I would also like to thank her for the pleasant conversations throughout the project. I would also like to thank Irene van Droffelaar for her advice on the development of Forest BORG even though she is not a supervisor to this project.

Many courses of the Engineering and Policy Analysis program have prepared me for this master thesis. I cannot pin point one course which was most import to the preparation of this thesis as it was the assemble of courses that introduced me to policy analysis and deep uncertainty. I should mention that the course 'Advanced Simulation' greatly improved my understanding of object oriented programming which was beneficial to the recreation of the RICE model and development of Forest BORG.

I wish to thank my parents for my education in general but also for this time in which I have been writing my thesis. Along the way of writing this thesis I had to quit my job because doing both proved to be too exhausting. Without the financial help from my father I could not have taken the time to create a working version of Forest BORG that I can feel proud of. Towards the end of the thesis I lost my home. My mother took me in and I am grateful to her in how she answered my mostly stressed mood with love. I want to thank both my parents for having someone to fall back on and for encouraging me to explore life beyond the path most taken.

Constantijn Hubert Daemen
Breda, February 2024

Executive Summary

We introduce the 'Forest BORG' framework, built on the BORG multi-objective evolutionary algorithm (MOEA), as a policy tree optimization method for socio-economical systems. Forest BORG mimics the BORG algorithm while accommodating tree-structured decision variables via specialized evolutionary operators, to allow for multi-objective and explainable decision support.

This novel framework is tested on two case studies representing socio-economic systems, the Folsom lake model and the Regional Integrated model of Climate and the Economy (RICE) Integrated Assessment Model (IAM). These case studies demonstrate the need for an algorithm that can optimize multi-objective problems under deep uncertainty. Climate change is the main driver behind the uncertainty in these models. Both case studies have multiple stakeholders, lending themselves as multi-objective decision-making problems. This research has developed the Forest BORG framework and aims to apply it to these case studies in order to find optimal policy sets, that are inherently more explainable than the ones based on real-valued decision variables used in the majority of current research.

The Folsom lake model and RICE IAM are good candidates as case studies for this research as they reflect real live problems facing uncertainty due to climate change and affect multiple stakeholders, making the case studies multi-objective problems. The conceptual models of both case studies are made operable by presenting them through the XLRM framework. RICE and the Folsom lake model are adapted to the XLRM framework of exogenous uncertainties (X), levers (L), relations (R) and metrics (M) (Lempert 2003).

This research positions itself in the gap between current MOEA capability for decision-making under deep uncertainty and the desired features that are necessary for creating explainable policies for real world multi-objective problems. Run-time dynamics, Forest BORG specific evolutionary operator dynamics and controllability are tested for each case study. The ability of Forest BORG to be a valuable asset in a decision makers toolkit is tested by analyzing the trade-offs found for each case study, and the individual policy trees that are part of the discovered pareto front. A logical extension of the current research is to design an optimized look-ahead tree (OLT) algorithm that can be coupled to Forest BORG and reapplied to both case studies. The resulting tandem framework is expected to enhance explainability and alleviate computational resources even further.

Like any other MOEA, Forest BORG employs evolutionary operators. There are seven different mutation operators, one recombination operator in the form of crossover with prune and bloat control operations and one selection operator: an adaptive tournament size function.

Each case study as well as the performance of Forest BORG is studied through five analyses. These analyses are:

1. *run-time dynamics*
2. *search operator dynamics*
3. *controllability*
4. *exposed trade-offs*
5. *highlighted policy trees*

The run time dynamics show convergence of the algorithm based on hypervolume. The search operator dynamics investigate the novel mutation operators of Forest BORG. The controllability analysis explores the influence of the hyperparameter settings on algorithm performance. The fourth analysis displays the trade-offs made visible by pareto-optimal policy alternatives through a parallel axis plot. A best performing policy is highlighted for each objective as well as a compromise policy, a policy that is closest to the average of each objective. Finally, the fifth analysis creates visuals of these selected

policies, displaying the actual policy trees. Similarities and differences between the different policies are sought after that perhaps reflect the trade-offs discovered in the fourth analysis.

The run-time dynamics show that Forest BORG performs effectively, efficiently and reliably across the case studies. The effectiveness of Forest BORG is underlined by comparing it to the original policy optimization tree (POT) algorithm for the Folsom case study. The operator dynamics were expected to shift from triple action subtree mutations to single action point mutations during the search. This shift from subtree to point mutations was not observed in the Folsom case study and not for all random trials in the RICE case study. The search operators do outperform the control of random mutation, as employed in the original POT algorithm which make them a valuable contribution to the Forest BORG algorithm. The search operators might not perform as they are expected to, but they do enhance the search. The hyperparameters of Forest BORG are case study dependent but become, unsurprisingly, more important for case studies with a limited number of discrete actions. Across both case studies, policy trees with modest maximum depths, 3 or 4, are preferred which eliminates the risk of overfitting policy trees to a certain scenario.

Forest BORG has proved itself capable of producing policies for multiple actors as it has found pareto fronts for each case study that clearly display the trade-offs that are inherent to the case study. Moreover, the structures of the selected policy trees are in accordance to their performance. Analyzing these policy trees shows what actions need to be taken and under what conditions when certain objectives are valued more than others, making the results of this analysis explainable, interpretable and valuable to decision-makers.

There are two notable observations of the results that appear to display erroneous behaviour of Forest BORG . Firstly, the trade-offs found for the RICE model are counter-intuitive. Secondly, there are small peaks and dips in the run-time analysis for both case studies. The discovered trade-offs within the RICE model are counter-intuitive as it is not welfare, but damages that poses a trade-off with the other objectives. The dips in hypervolume, observed in both case studies appear odd as an evolving pareto front should not decrease in hypervolume. The small peaks are however due to the settings of the epsilon values.

Other IAMs are also interesting to explore as they are a better representation of reality. The RICE model has been used because it is a relatively straightforward and well-established model that was suitable to test new algorithms with. The proof-of-concept, for Forest BORG, is showcased in this research. The parallel axis plots and the visualizations of the policy trees are interesting because it shows how policy trees improve explainability for climate change policies. The absolute values of the policies are however questionable given the limitations of RICE. It would therefore be interesting to employ a better RICE model or another IAM. ForestBORG was developed in an effort to accommodate many-objective problems. The policy problems of this research only employed three or four objectives which is a relatively small amount. It would therefore be interesting to perform a similar policy analysis but with more objectives.

The societal contribution of this research is the development of Forest BORG and applying it to real world case studies. Forest BORG is a new tool in a decision-makers toolkit that allows for multi-objective decision making for socio-environmental systems. To the authors knowledge there are few other algorithms that are able to handle such optimization problems. Forest BORG has improved on a competing algorithm by expanding the number of objectives that can be used as well as allowing for continuous action ranges for multiple actions. The societal implication is that the range of real world applications that can be analyzed from a decision making under deep uncertainty perspective is expanded through these improvements. Furthermore, Forest BORG is not tested on theoretical problem formulations but on real world case studies. The results from these real world case studies show confidence that the suit of case studies can be extended to other problems while maintaining algorithm performance. A logical continuation of this research is applying Forest BORG to a more detailed IAM than RICE from which policy implications are gathered that will help decision-makers develop policies against climate change.

Contents

List of Figures

List of Tables

1

Introduction

Climate change causes an increase in uncertainty in socio-economic systems. Human-induced climate change has caused a rise in global temperature, offsetting many processes and cycles in the natural world. These rapid changes in the atmosphere, ocean, cryosphere and biosphere indirectly affect the world economy and the well-being of people around the world. Some regions however are affected more than others by the change of balance in the local socio-economic and climate systems. Global surface temperatures have risen 1.1 ^oC above 1850-1900 levels in 2011-2020 mainly due to the emissions of greenhouse gasses which continue to rise (Lee et al. 2023).

In 2015, at the Conference Of the Parties 21 in Paris, the scientific and policy communities around the world established a target limit of 1.5 ^oC and a boundary limit of 2 ^oC increase in global surface temperatures above pre-industrial levels after which further warming could dangerously and inadvertibly change the natural systems on which the socio-economic systems around the world rely (Livingston and Rummukainen 2020), (Allen et al. 2018). Global cooperation to transition to a green or net-zero economic system has been challenging due to historical, economical and political tensions but countries attending the conferences of the parties have individually set goals to curb greenhouse gas emissions in the form of nationally determined contributions (NDCs). However, these NDCs as of October 2021, are insufficient to limit climate change to such an extent that warming will exceed 1.5 ^oC during the 21st century. Limiting warming below 2 ^oC becomes progressively more difficult and expensive with NDCs that do not suffice in timely eliminating greenhouse gas emissions (Lee et al. 2023). Thus, there is an urgent need for new policies that limit global warming and that the countries of the world can agree on.

A specific, localised example of systems that are changing due to climate change are water reservoirs. Local people depend on water reservoirs for their water demand. The inflow of water to these reservoirs is subject to climate conditions and therefore to deep uncertainty, due to the impact of climate change (Asadieh and Krakauer 2017). Water management problems are characterised by relating water inflow to water demand while preventing floods or periods of inadequate supply, by storing water in a reservoir. Policies are required that can incorporate the uncertainty in the system due to changes in demand and inflow, resulting from climate change. The role of operating policies in the adaptation of reservoirs to climate change is increasingly recognized, as the performance of a system is often driven by the accumulation of short-term decisions (Herman and Giuliani 2018).

Decision-makers need new and improved tools to help them navigate through this uncertain future. These tools have to formulate policies for socio-economic systems that can be presented to decision-makers. Multiple stakeholders are often involved in these socio-economic systems, be it on a global scale or on a more local scale for *e.g.* water management problems. This tool, or algorithm, must therefore be able to optimize multi-objective problems. A class of algorithms called multi-objective evolutionary algorithms (MOEA) have been developed for these kinds of problems. Some notable examples of such algorithms, ranging from classical ones to the state-of-the-art are NSGAII (Deb, Pratap, et al. 2002), ϵ-MOEA (Deb, Mohan, and Mishra 2005), MOEA/D (Zhang and H. Li 2007) and BORG (Hadka and Reed 2013). These MOEAs and search strategies have different design choices and de-

pending on the problem, one MOEA is preferred over another.

Despite a suit of MOEAs, the resulting policies that are presented to decision-makers often lack in explainability. If a decision-maker were to ask why a given set of policies is 'optimal', the answer must be that the algorithm found these to be optimal. These algorithms are well-developed and trusted but especially in socio-economic systems, a lot of value is placed on why certain decisions are made, that must go beyond the optimality argument. For most algorithms, it is difficult to explain why one policy is preferred over another, besides it performing better.

To overcome this problem, policy trees have been used as input for these MOEAs (Herman and Giuliani 2018). A policy tree relates an action to a system state. A policy based on policy trees is therefore inherently more explainable than real-valued decision variables. When asked why a certain decision is made, one can say that the system conditions were such that a specific action had to be taken, as well as the optimality argument. There are not many algorithms combining MOEAs and policy trees as decision variables which are then applied to socio-economic systems to advance the explainability of the optimal policy set.

We introduce the 'Forest BORG' framework, built on the BORG MOEA, as a policy tree optimization method for socio-economical systems. Unlike the original BORG MOEA which does not support tree-structured decision variables, Forest BORG mimics the BORG algorithm as closely as possible while accommodating tree-structured decision variables via specialized evolutionary operators.

This novel framework is tested on two case studies representing socio-economic systems, the Folsom lake model and the Regional Integrated model of Climate and the Economy (RICE) Integrated Assessment Model (IAM). These case studies demonstrate the need for an algorithm that can optimize multi-objective problems under deep uncertainty. Climate change is the main driver behind the uncertainty in these models. Both case studies have many stakeholders, lending themselves as multi-objective decision-making problems. This research has developed the Forest BORG framework and aims to apply it to these case studies in order to find optimal policy sets, that are inherently more explainable than the ones based on real-valued decision variables used in the majority of current research.

2

Literature Review

2.1. Multi-Objective Evolutionary Algorithms

An MOEA is a type of optimization algorithm capable of optimizing multiple objectives simultaneously. The main objectives of an MOEA according to (Coello 2007) are:

- Preserve nondominated points in objective space and associated solution points in decision space.
- Continue to make algorithmic progress towards the Pareto Front in objective function space.
- Maintain diversity of points on pareto front and/or of Pareto optimal solutions in decision space.
- Provide the decision maker with “enough” but limited number of Pareto points for selection resulting in decision variable values.

Note how the pareto front of point two and three are different. The Pareto Front of point two is the theoretical optimal Pareto Front, in this research indicated with capital letters. The pareto front of point three is the evolving pareto front produced by an MOEA, indicated with lowercase letters, The algorithmic evolving pareto front should get ever closer to the theoretical optimal Pareto Front.

The first MOEA is called vector evaluation genetic algorithm (VEGA) and was invented in 1985 (Schaffer 1985). VEGA was the first contribution of an evolutionary algorithm to solving the multi-objective problem. Previously mathematical programming techniques were used to solve single objective problems, but these algorithms did not transfer well to multi-objective problems as these problems could have discontinuous, nonlinear, or non-differentiable characteristics (Wang, Pei, and J. Li 2023).

Other pioneering works in the field of MOEAs are the non-dominated sorting genetic algorithm (NSGA) (Srinivas and Deb 1994), the multi-objective genetic algorithm (MOGA) (Fonseca, Fleming, et al. 1993) and the niched Pareto genetic algorithm (NPGA) (Horn, Nafpliotis, and Goldberg 1994). These first works are characterized by individual selection using the pareto front and population diversity maintenance using a fitness sharing mechanism (Wang, Pei, and J. Li 2023).

A new generation of MOEAs then developed which employed the novel elite retention mechanism feature. These MOEAs include the strength Pareto evolutionary algorithm (SPEA) (Zitzler and Thiele 1998), the Pareto archived evolution strategy (PAES) (Knowles and Corne 2000) and notably NSGA-II (Deb, Pratap, et al. 2002). Then, after 2003, a continuous development of MOEAs followed. These MOEAs, *e.g.* the e multi-objective evolutionary algorithm using decomposition (MOEA/D) (Zhang and H. Li 2007), are either developed for specific applications, usually in the field of engineering or on a more general level aim to enhance performance on problems with more than four objectives (Wang, Pei, and J. Li 2023).

While the majority of the current research in MOEA development is focused on enhancing performance on problems with large numbers of objectives, there is little research done on making the results of

an MOEA more explainable. An MOEA developed by (Herman and Giuliani 2018) incorporates tree-structured variables as decision variables to enhance the explainability of the outcomes of the algorithm, see section 2.2.1. This research builds on top of that work by introducing Forest BORG, an MOEA that uses tree-structured decision variables, able to operate multiple actions in continuous ranges and designed to be used on socio-environmental systems. The explainability of the outcome is important to decision-makers as these are not purely engineering systems, but systems in the social domain subject under deep uncertainty.

2.2. Policy Trees

2.2.1. Policy Optimization Trees

The idea of changing real-valued decision variables to policy trees is taken from (Herman and Giuliani 2018).

In the field of water management to aid adaptation decision-making under uncertainty, (Herman and Giuliani 2018) have developed a tree based algorithm that based on indicator and action variables defines a policy for the Folsom reservoir model. The goal of the algorithm is that given a state transition function of a generative simulation model, decision variables are chosen that minimize a cost function. In the case of water management, large penalties are given to flooding events or when water supply does not meet demand. Decision variables include for example what percentage of available water to hedge. The novelty of the algorithm is that previously such a problem was regarded as a real-valued optimization problem. Policy trees, instead, search for a set of hierarchical rules to select decisions at each timestep from a discrete set of actions. The optimal policy tree is a set of actions that minimizes the cost function.

A policy tree defined by (Herman and Giuliani 2018) is a binary tree consisting of indicator and action nodes. The root and closed nodes of this tree structure are indicator nodes. An indicator node is a system/ model variable and a threshold value. Starting at the root node, if the system variable exceeds the threshold, the child node on the right of the node in question is checked. The leaf nodes, or open nodes, of the tree structure are the action nodes. These action nodes equate a decision variable to a certain value. A visual representation of a policy tree, as presented in (Herman and Giuliani 2018) is given in Figure 2.1.

The policy tree is further developed in (Cohen and Herman 2021) where indicator variables, actions and policy structure are identified simultaneously during the optimization. This subsequent study identifies a set of optimal policies, as was done in (Herman and Giuliani 2018), but then analyzes what indicators and actions are most common in these optimal policies compared to non-optimal policies. Then, the indicators most frequently associated with each action and the timing of the actions are determined.

The novelty of this algorithm is the incorporation of policy trees in an MOEA. Policy trees are inherently more explainable than real-valued decision variables and are therefore more suitable when a set of Pareto optimal solutions are required for a socio-environmental system. Decision-makers do not only care what the best solutions are but also why they are the best. They have to understand the decision-making process. The algorithm has at the time of writing only been applied to water management problems. These problems face deep uncertainty due to climate change and make them similar to other socio-environmental problems, like those dealing with climate change. The shortcomings of this algorithm however, prevent it from being the best possible algorithm to be used on these problems. The first shortcoming is that only one action can be supplied to the MOEA. This action can take on different discrete values but for socio-environmental systems there are usually many levers, *i.e.* actions, inherent to the problem. The second shortcoming is the number of objectives that can be simultaneously optimized by this algorithm. Besides the multiple levers, these socio-environmental systems often have many stakeholders involved and many objectives to consider. This algorithm is expected to perform reasonably well for problems up to four or five objectives, but its performance is likely to deteriorate when applied to problems with much more objectives, *e.g.* eight objectives or more. The Forest BORG algorithm is designed to alleviate these problems.

Policy trees have two advantages over real-valued decision variables. Firstly, they are more explainable as they relate actions to system states. An optimal policy consisting of real-valued decision variables offers no further insight. Secondly, policy trees are more versatile decision variables than

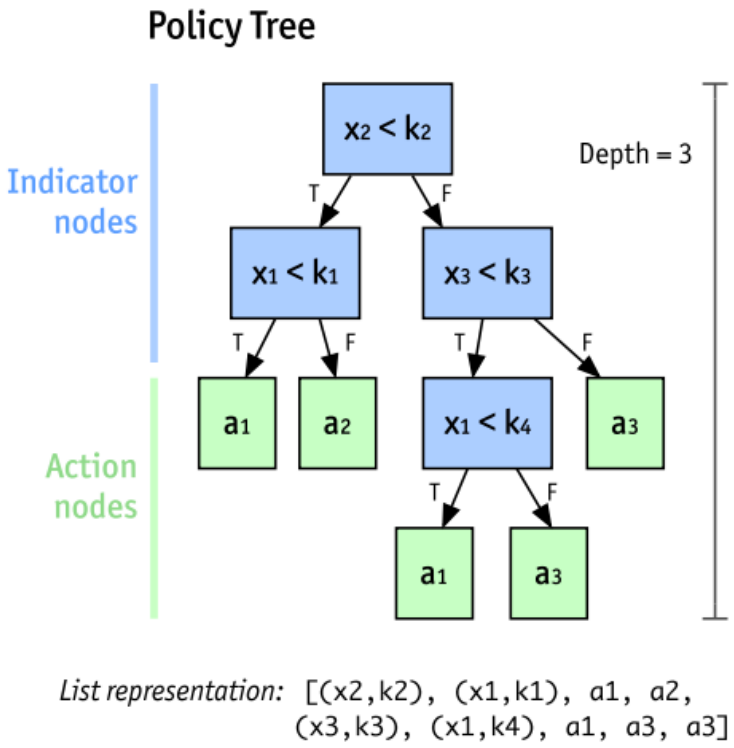

Figure 2.1: Structure of a policy tree as defined in (Herman and Giuliani 2018)

real-valued decision variables making them more adequate as decision variables for social systems with long time horizons. The implication of using tree-structured decision variables as opposed to real-valued decision variables is visualized in Figure 2.2. A simple example of a policy tree is offered as input to a hypothetical system. The policy tree dictates that when the variable 'year' in the system is smaller than 2150, the savings rate (sr) is set to 0.3. When this condition is no longer true, the variable sr is set to 0.1. The response of this hypothetical system to the policy tree is shown on the right. The green curve reflects the sr variable in the system which is the decision variable in this example, also called an action variable. The blue and dashed grey curve reflect another variable in the system that is defined as a metric. This metric can also be called objective variable, key performance indicator or solution fitness. The metric variable is sensitive to the decision variable. The blue curve, relating to the policy tree, shows a change during the simulation process. The dashed grey curve, relating to real-valued decision variables does not show a change. This hypothetical example highlights an important feature of policy trees: the actions have become dynamic, meaning they can change during the simulation. Static actions are set a the start of the simulation and cannot be changed during the simulation. The actions of real-valued decision variables are static, they are not allowed to change during the simulation. Allowing decision variables to be dynamic rather than static has important implications for long term systems, such as water management problems or integrated assessment models.

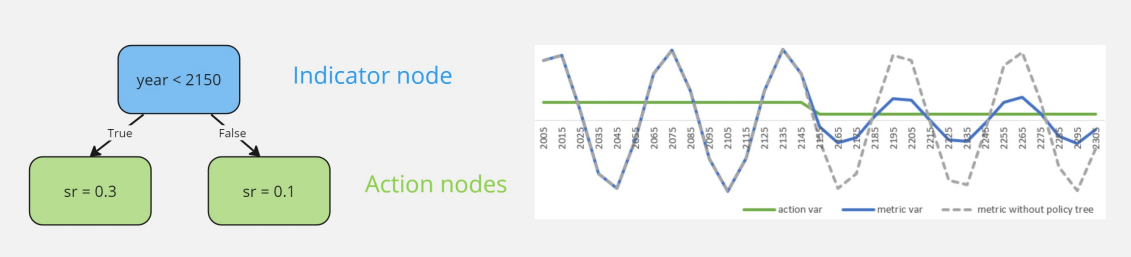

Figure 2.2: A hypothetical example of a policy tree for the RICE model. The behaviour of the savings rate (sr) is hypothetical and only serves as an example.

2.2.2. Optimized Look-Ahead Trees

Another strain of literature develops Optimized Look-ahead Trees (OLT). The OLT draws its inspiration from (Hren and Munos 2008) which developed a search tree algorithm that is able to control complex systems with relatively small trees. The root node represents a system state x and each node of depth d corresponds to a state that is reachable from x after a sequence of d transitions. The tree is developed 'optimistically, where the most promising nodes are explored first' by which depth first optimization is meant. The resulting learning tree algorithm is problem independent, meaning that the generative model, the system on which the reward function is based, has no impact on the learning tree optimization.

The OLT, however, extends this independent tree exploration to problem dependent tree exploration (Maes, Wehenkel, and Ernst 2012). The optimization relies on a path feature function which turns a path in the search tree into a vector of features describing the path. These features are chosen before the optimization and require system specific knowledge to make an apt choice. They later refine the OLT to large and continuous action spaces (Jung, Ernst, and Maes 2013) and bridge the OLT with Direct Policy Search (DPS) (Jung, Wehenkel, et al. 2014). The OLT algorithm is further explained in Appendix F.

The OLT is an interesting addition to Forest BORG. Forest BORG allows for tree-structured decision variables that improve upon real-valued decision variables by adding explainability and enhancing the flexibility of polices by turning static actions into dynamic actions. The OLT can improve upon Forest BORG in a similar manner as how Forest BORG has improved real-valued decision-making tools. The OLT expands the explainability of the results by defining the path feature function *apriori* to the optimization. The path feature function may include ethical considerations or decision-maker preferences like the aversion to large, sudden changes in decision variables. The OLT adds even more flexibility to the optimization progress by allowing the polices to be dynamic, rather than static. Forest BORG allows for dynamic actions but the policy, the set of dynamic actions, is static. The policy is defined before the optimization and is not allowed to change during the optimization, the actions are dynamic but limited to the ones defined in the policy tree, which is static. The OLT however allows for dynamic actions and dynamic policies as every action is considered at every time step in the simulation, only guided by the path feature function. This feature of an OLT can become problematic as it becomes very computationally expensive for simulation systems with many timesteps and many possible actions over wide ranges, such as IAMs or water management problems. Forest BORG is capable of alleviating this computational burden for simulation systems such as IAMs as Forest BORG uses offline resources to determine the most influential indicator and action variables and their values. This knowledge can then be used to restrict the number of action variables and their range when used in an OLT, therefore significantly reducing the online resources required by an OLT to define optimal policies.

2.3. Modeling Systems

An MOEA with policy trees as tree-structured decision variables is well suited as a decision making tool for policy makers. The most adequate models are those that allow for policy making problems characterized by uncertainty and long time horizons due to the two advantages policy trees have over real-valued decision variables. Case studies in water system management problems have been employed in (Cohen and Herman 2021). This research extends the type of problem to integrated assessment models.

IAMs are used to explore the objective space, the possible outcomes of policies. However, because IAMs unite models that could have been designed for different purposes, some IAMs will be more detailed in the description of one subdomain. For example, one IAM can describe the interactions in the climate system better than another IAM which is tailored to the precise workings of the economy or the preferences of economic agents. IAMs also differ from one another in scope. IAMs can be used as a simulation tool, predicting possible futures based on assumed underlying relationships between the variables in an IAM. An alternative use of IAMs is as optimization tools; by adjusting certain variables, questions such as 'What policies should be implemented to maximize welfare?' and 'How can a policy minimize climate damages?' can be explored. An objective function or a welfare function is introduced

and optimized so different policy pathways can be compared. IAMs are large models that lend themselves for the optimization of policies.

IAMs about the energy-climate system were first developed in the 1970s (Beek et al. 2020) and despite criticism have gained significant importance due to their central role in the global discussion about adaptation and mitigation to climate change, e.g. the IPCC or the UNEP reports (Keppo et al. 2021). Critics, however, argue that although IAMs aim to capture the key elements of real-world problems, they fail to do so and too much importance is attributed to the outcomes of IAMs. IAMs are accepted as policy making tools but face criticism for their representation of reality too.

2.4. Research Implications

This research builds on the concepts laid out in this chapter by creating a novel MOEA, specifically designed to be operated with tree-structured decision variables, policy trees, and to be applied to multi-objective problems characterized by uncertainty. The novel algorithm, Forest BORG is benchmarked on the Folsom lake model, a water management problem, and then applied to the RICE model, an IAM. The Forest BORG algorithm is presented in Section 5.1 and the two case studies are presented in Chapter 3.

3

Case Studies

3.1. Folsom Lake Model

The Folsom lake model is a simulation model of the Folsom reservoir located in California, in the US. The physical rivers and reservoir originate in the American River Basin. Three forks of the American River flow into the Folsom reservoir out of which the American river flows which connects to the Sacramento River. The relevant are is depicted in Figure 3.1. Small dams and diversion have been developed around the forks and the reservoir for generating hydroelectricity, controlling peak flows and maintaining baseflow in dry seasons (Goharian et al. 2020). The water demand fluctuates from year to year depending on climate and economic conditions (Herman and Giuliani 2018). The regional hydroclimate is characterized by a dry and a wet season. A high rate of snowmelt in the Sierra Nevada mountains during the spring season causes historic peak flows. The reservoirs are filled as much as possible during these periods of high inflow to store enough water for the summer demand, when inflow is low (Goharian et al. 2020).

Figure 3.1: An area depiction of the Folsom reservoir and the American River, taken from (Herman and Giuliani 2018)

3.2. RICE

The RICE model, a regional derivative of the global Dynamic Integrated model of Climate and the Economy (DICE) model, is a rather simple yet well-established and updated model (W. Nordhaus 2018). The DICE model is a well-known and influential. DICE was the main model in the US for the determination of the social cost of carbon, a topic of great political discussion (Council 2013). The RICE model in this research is based on the RICE2013 version (W. Nordhaus and Sztorc 2013). This version divides the world into 12 regions, see Appendix E. The RICE model has also been extended beyond its 10 to 12 regions in an effort to dis-aggregate and analyze regional differences more finely (Gazzotti 2022). Despite these badges of influence and the continued development of the DICE/RICE models, these models have been criticized too (Grubb, Wieners, and P. Yang 2021). The main argument against these models is the expression of the damage function, relating global temperature rise to (regional) economic damages, which is said to be too simplistic to reliably reflect the influence of the climate system over the economic system (Moore and Diaz 2015). The exact expression of the damage function should be treated as a large source of uncertainty in the DICE/RICE model (Burke, Hsiang, and Miguel 2015). The RICE model is flawed but simple and well-established, making it a good candidate as an IAM to test and apply Forest BORG on.

The RICE model employs four interacting submodels: an economy submodel, a carbon cycle submodel, a climate submodel and a utility submodel. The first publication of RICE featured a world divided in ten regions with each their own initial capital stock, technology and population. The growth of the first is endogenous to the model, and the latter two are exogenous. These three factors determine economic output through a Cobb-Douglas production function (W. D. Nordhaus and Z. Yang 1996). Industrial emissions, stemming from economic output, relate the economy model to the carbon cycle model. Carbon is shared between different reservoirs and has an effect on radiative forcing, which related the carbon cycle model to the climate model. The climate model determines global atmospheric temperature as a result of radiative forcing. Global atmospheric temperature is then coupled back to the economy submodel by the damages inflicted on the regional economies. These damages are estimated by a damage function. A lower, net economic output is divided into investment, used for later abatement and consumption, leading to industrial emissions in the next cycle of the model. The core logic of the RICE model is depicted in Figure 3.2, which was taken from (Reddel 2022). The RICE model is explained in depth in Appendix D.

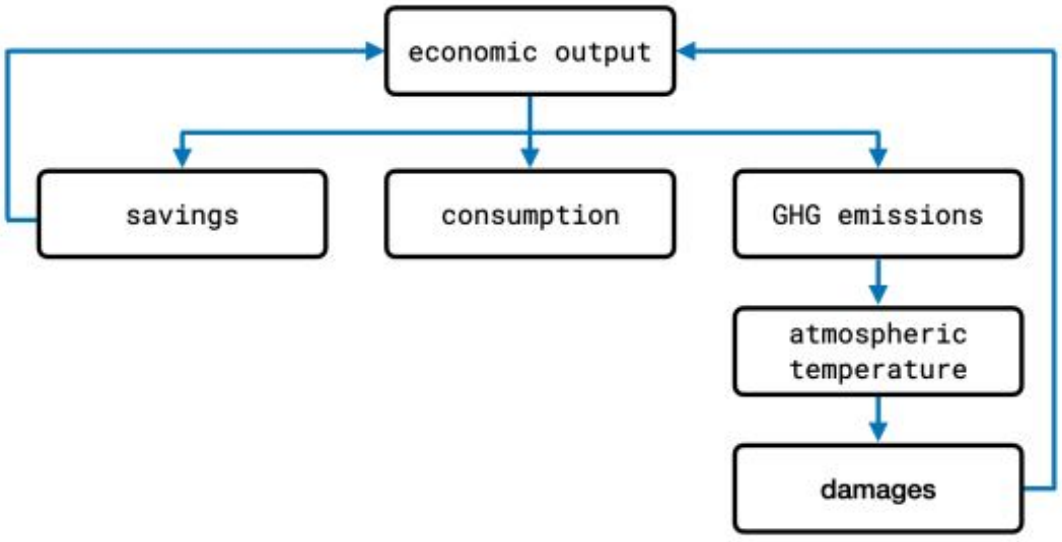

Figure 3.2: A visualization of the core logic of the RICE model, taken from (Reddel 2022)

3.3. Research Implementation

The Folsom lake model and RICE IAM are good candidates as case studies for this research as they reflect real live problems facing uncertainty due to climate change and affect multiple stakeholders,

making the case studies multi-objective problems. The conceptual models are implemented in the python programming language. The implementation of the Folsom lake model is taken from (Herman and Giuliani 2018). The RICE model was reconstructed as part of this research. The implementation of the RICE model is based on the version presented in (Reddel 2022). The structural relations within the model are identical to the predecessor version but the model was rewritten to accommodate action changes during a simulation run. The rewriting of the model was done in a more object oriented and functional programming fashion. The model implementations are operationalized by fitting them to the XLRM framework. The XLRM framework facilitates a standardized method for problems relating to decision-making under deep uncertainty. The framework also lends itself to the application of policy trees. The framework and the fitting of these case studies is presented in Section 5.3.

4

Knowledge Gap and Main Research Question

4.1. Knowledge Gap

The current methods employed in the field of climate change decision making suffer from over aggregation and lack explainability due to the neural networks present in the methods. This project aims to develop a tree based method that can be applied to assist in policy making in the climate change domain and therefore add a new tool to the DMDU (Decision making under deep uncertainty) toolkit. The tree based method provides insights in the decision making process that yields an optimal policy set. To the authors knowledge, the development of a tree based method specific to the climate change policy domain has not been done before.

From this point forward, the policy tree as described in section 2.2.1 is described as 'POT', later with a distinction based on employed MOEA and the optimized look-ahead tree as 'OLT'. Applying the POT algorithm in the climate change field has several drawbacks. It is computationally expensive, the actions are dynamic but the policies are static and the inherent explainability of the resulting policies is limited.

Finding optimal policy trees through a POT algorithm is computationally very expensive as all of the computational resources are used online, as opposed to for example Evolutionary Multi-Objective Direct Policy Search (EMODPS) where nearly all of the computational budget is deployed offline. Within a fixed depth all possible combinations of indicator variables, thresholds and actions at every possible timestep in the simulation horizon must be evaluated. The resulting policies are also static in the sense that a policy remains unchanged over the entire simulation horizon which becomes problematic when uncertain events like tipping points are included in the simulation model. Also, although the resulting policies from this approach are more explainable than policies resulting from the black box machine learning models, the explainability of a policy tree can still not be fully defended. Although it becomes clear under what circumstances actions are taken, one could not explain why those actions are taken other than that those scored the highest reward values given a reward function. Policy trees applied to the field of climate change where justice principles are incorporated are thus more akin to explainable black box models than inherently explainable models (Rudin 2019).

In addition to applying a POT algorithm in climate change simulation, the algorithm can be expanded by incorporating the idea of an OLT. First, a POT algorithm is coupled to a case study from which system specific knowledge is gained which is then used in the formulation of a path feature function which secondly allows for the coupling of an OLT algorithm to the case study. The first coupling yields the most important indicator and action variables which can then be used in the path feature function, supplemented with human preferences that do not stem from optimization results. An example is the extent of change in actions. The desire for smooth transitions over erratic transitions can be listed in the path feature function as a human preference. In the second coupling, paths that require erratic change are disregarded, thus limiting the necessary computational budget. When an explanation of the resulting

policy tree is asked for, one can point directly to the smooth transition preference, or whatever other decision-maker preference was vocalized through the path feature function.

Policies generated through OLTs are inherently dynamic as at every timestep a new action is determined instead of evaluating a system state that will translate to a predefined action as in POTs. OLTs are also more explainable than POTs as preferences can be defined in the path feature function, clearly stating what paths are more favourable than others and on which premises instead of having multiple preferences be aggregated into a single reward value. The path feature function also allows action space to condense to the influential regions which substantially decreases the required computational budget. OLTs hinge on a path feature function which improves as more system specific information is available, like which actions are important and over what ranges.

The development of a POT algorithm and applying it to the case studies is the main objective of this research. The set of policy trees can be tested on different scenarios and explain under what conditions which choices should be made. The Folsom lake model serves as a benchmarking case study and the RICE IAM as a test suite for Forest BORG.

4.2. Research Questions

To fill the knowledge gap the research question central to this research is stated as:

How can we design a tree-based multi-objective algorithm that ensures explainability in decision making for socio-environmental systems?

The algorithm must advance explainability in decision making for socio-environmental systems. The socio-environmental systems are represented through the two case studies, the Folsom lake model and the RICE IAM. The algorithm must also be benchmarked to ensure it works properly. The algorithm should be judged on efficiency, by checking convergence speed, effectiveness, by comparing the runtime dynamics to another algorithm, and on reliability, by repeating the analyses for multiple random trials. The first sub research question is then stated as:

Sub research question 1: How can algorithm performance be benchmarked?

MOEAs have multiple hyperparameters that could influence the search. This hyperparameter space should be explored to ensure optimal running conditions for each case study. Moreover, one of the hyperparameters is maximum tree depth, which should not become too large as the risk of overfitting policy trees to a specific scenario increases with maximum tree depth. The controllability of the algorithm is analyzed by the second sub research question, which is stated as:

Sub research question 2: How do the hyperparameters influence algorithm performance?

Advancing explainability in decision making for socio-environmental systems is done by analyzing the individual policy trees and the trade-offs discovered by the algorithm. The third sub research question is thus stated as:

Sub research question 3: What trade-offs are identified by the algorithm?

From the pareto front, a policy tree is selected that performs best with respect to each objective, as well as a policy tree that reflects a compromising solution across all objectives. These policy trees are analyzed for their similarities and differences and how they relate to the trade-offs that are discovered. These individual policy trees are examined in the fourth sub research question:

Sub research question 4: How do the selected policy trees advance explainability?

4.3. Research Implications

This research positions itself in the gap between current MOEA capability for decision-making under deep uncertainty and the desired features that are necessary for creating explainable policies for real world multi-objective problems. The algorithm that is developed to fill this knowledge gap, Forest BORG, is benchmarked on the Folsom lake model and tested on the RICE IAM. Run-time dynamics, Forest BORG specific evolutionary operator dynamics and controllability are tested for each case study. The ability of Forest BORG to be a valuable asset in a decision makers toolkit is tested by analyzing the trade-offs found for each case study, and the individual policy trees that are part of the discovered pareto front. A logical extension of the current research is to design an OLT algorithm that can be coupled to Forest BORG and reapplied to both case studies. The resulting tandem framework is expected to enhance explainability and alleviate computational resources even further.

5 Methods

5.1. Forest BORG Design

The Forest BORG algorithm is the product of a study of the original POT algorithm, as presented in (Herman and Giuliani 2018) and the BORG algorithm, as presented in (Hadka and Reed 2013). The original POT algorithm is one of the few algorithms that allows for tree-structured decision variables as input. It does not however permit multiple distinct actions to be used as input for these tree-structured decision variables, or for the sampling over a continuous range for these actions. Also, the MOEA design of the original POT employs a binary tournament, which performs detrimentally as the number of objectives of a case study increases.

On the other hand, the BORG algorithm is designed to deal with many objective problems and employs clever strategies for the convergence to an approximate optimal pareto front, like the keeping of an archive, the implementation of a restart procedure when search stagnates and the application of an adaptive tournament size. However, the BORG algorithm does not permit tree-structured decision variables. Forest BORG marries these two algorithms, original POT and BORG, by taking the core working of the original POT algorithm and replacing most MOEA design choices with mechanism from BORG. A novel feature of Forest BORG that is not taken from either algorithm is the custom mutation operators, specifically designed for tree-structures and case studies with more than one action.

Like any other MOEA, Forest BORG employs evolutionary operators. There are seven different mutation operators, one recombination operator in the form of crossover with prune and bloat control operations and one selection operator: an adaptive tournament size function. The following details the design of Forest BORG and the workings of these evolutionary operators. The design of the competing original POT algorithm is explained too, to facilitate comparison to Forest BORG.

Terminology

Evolutionary algorithms have some terminology associated to them that is explained here so it can be used to understand the Forest BORG algorithm. An 'individual' is an encoded solution to some problem (Coello 2007). An individual is defined by a 'genotype', which can be the set of values for each decision variable. An individual is associated with a quantifiable 'phenotype' dependent on the simulation environment, often the generative model. The generative model decodes the genotype to the phenotype. A phenotype when put in a certain environment is also called the 'fitness' of an individual, to that environment. This fitness includes or is fully composed of the objective values associated to the decision variable. A collection of individuals is called a 'population'. Evolutionary algorithms operate on a population through evolutionary operators. These are divided in mutation, recombination and selection operators. A mutation operator alters the genotype of an individual. It could *e.g.* change the value of a decision variable that an individual carries in its genotype. A recombination operator produces a new individual through the marriage of parts from other individuals, often called parents. If a genotype of an individual is defined by the setting of two values for two different decision variables than a recombination operator could generate a new individual by taking the value for one of the decision variables

from one parent and the other from another parent. A selection operator chooses individuals from a population, often based on their fitness. An evolutionary algorithm works on a population through evolutionary operators to produce individuals that are more fit to their environment with every iteration of the algorithm.

By marrying the above terminology for multi-objective problems and evolutionary algorithms, an MOEA produces a pareto front of individuals through the iterative application of evolutionary operators on a population. An individuals genotype is characterized by its position in decision space and its fitness by its position in objective space .Important to this research is the definition of a decision variable. Traditionally, a decision variable is a real-valued numerical quantity, a single point in decision space (Coello 2007). The decision variable is an individuals genotype. This research replaces the real-valued decision variable with a tree-structured data type, the policy tree.

Original POT algorithm

The original POT algorithm was designed for a water management problem. The generative model is the Folsom lake model. The results of their research Herman and Giuliani 2018 are a major inspiration for this project. By swapping the Folsom lake model for the RICE model the Original POT algorithm could possibly optimize an IAM. The original POT algorithm was however used for a single-objective problem only and the level of convergence was tested by checking after how many function evaluations the objective value stopped decreasing significantly. A challenge for this algorithm could be to move to a multi-objective problem. Firstly, as the number of objectives increases, a binary tournament will become increasingly insufficient in finding non-dominated parents. Secondly, proper convergence metrics like hypervolume are required to test algorithmic convergence on the RICE model.

The original POT algorithm is explained through its initialization and the iteration procedure. Every iteration of the algorithm produces a new generation of the population in which some parents are taken from the previous generation. The algorithm employs a population register of static size (λ) but not an archive structure. Only a binary tournament is available as a tournament selector. The initialize procedure, described in Algorithm 1, initializes the search by creating random trees until a pre-specified static population size, λ is met. The population evolves until a maximum number of function evaluations (NFE) is reached. One NFE is the evaluation of one policy tree. The evolution of one generation to the next is handled by the iterate procedure.

Algorithm 1 The initialization procedure - Original POT

1: Initialize P [Population set] with random trees until $|P| = \lambda$
2: Set $N_{eval} \leftarrow 0$ {N_{eval}: Number of function evaluations}
3: **while** $N_{eval} < N_{max}$ **do**
4: Evaluate the objectives for each member in P
5: $N_{eval} \leftarrow N_{eval} + |P|$
6: Perform iteration step
7: **end while**

The iterate procedure, described in Algorithm 2, creates new generations and updates the pareto front. A new generation, of static size, is created by selecting a static number of parents, μ, through a binary tournament. The best parent remains unchanged while the other parents are mutated, controlled by a pre-specified chance of mutation, in this research set to 0.5. This single mutation operator is repeated in Forest BORG and serves as a control to which the performance of the 6 novel operators can be tested against. The children of the new generation are created until the size of the new population equals the pre-specified population size again (λ). A child is generated either by crossover of two randomly selected parents, or by the creation of a random tree. The child generation mechanism is controlled through a pre-specified crossover probability, in this research set to 0.9.

Forest Borg

First, Forest BORG is contrasted to the BORG algorithm, then the design of Forest BORG is explained. The BORG algorithm employs different techniques for population and archive maintenance as well as

Algorithm 2 The Iterate Procedure - Original POT

1: $P_{parent} \leftarrow$ BinaryTournamentSelection(P, μ) {Select μ parents}
2: **if** pareto front PF is undefined **then**
3: $PF \leftarrow P_{parents}$ {Initialize pareto front with parents}
4: **else**
5: $P \leftarrow P \cup PF$ {Combine population with pareto front}
6: Update PF with the non-dominated solutions in P
7: **end if**
8: Except for first parent, Mutate(Parent, $P_{mutation}$) {Mutate with probability $P_{mutation}$}
9: **while** $|P_{children}| < \lambda$ **do**
10: $P_{parents} + P_{children} \leftarrow$ GenerateOffspring$(P_{parents}, P_{crossover})$ {if not crossover, generate new random tree as child solution}
11: **end while**

different genetic operators. The authors state that the critical components of the BORG algorithm are (1) population-to-archive ratio triggered restarts with adaptive population sizing; (2) ϵ-progress triggered restarts; and (3) the auto-adaptive multioperator recombination operator (Hadka and Reed 2013). The current configuration of the BORG MOEA does not support the application of tree-structured decision variables. The current configuration of the BORG MOEA allows for numeric or textual input in the form of scalars or arrays. It's performance does elicit tempting results if coupled to policy trees. The following points state the most notable comparisons between BORG and Forest BORG:

- BORG uses an ϵ-box dominance archive and Forest BORG has a similar approach for diversity and convergence management
- BORG features an adaptive population sizing operator which is reiterated in Forest BORG.
- The auto-adaptive multi-operator recombination scheme of BORG is recreated in Forest BORG although the recombination operators in BORG (SBX, DE, PCX, UNDX, SPX, UM, see (Hadka and Reed 2013)) are entirely replaced by tree-based recombination operators in Forest BORG, described in section 5.1.

The design of the Forest BORG algorithm is shown in Figure 5.1 and explained through Algorithms 3 to 7 as well as the evolutionary operators. The same Algorithms can be found in Appendix A but reformulated in a less mathematical form. Every iteration of the algorithm produces one offspring and therefore constitutes one increment in the number of function evaluations as shown in Algorithm 3.

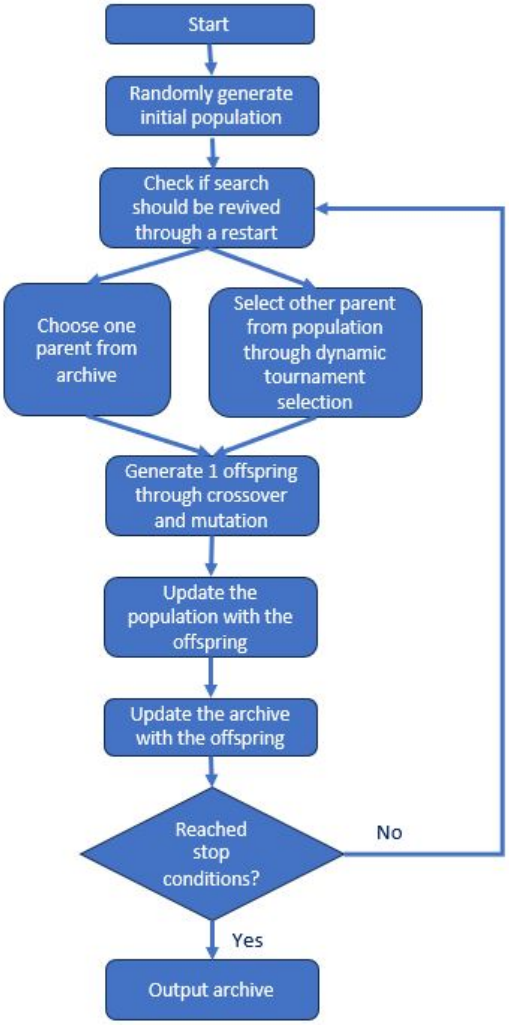

Figure 5.1: A visualization of the core logic of Forest BORG.

Algorithm 3 The initialization procedure - Forest BORG

```
Initialize P [Population set] with random trees
for all x ∈ P do
  UpdateArchive(x, A) {add tree x to archive A if eligible}
end for
while N_eval < N_max do
  Perform iteration step
  Increment N_eval by 1
end while
```

The iterate procedure, described in Algorithm 4, constitutes the main part of Forest BORG in which most of the logic behind Forest BORG is brought together. First the population-to-archive ratio and the ϵ-progress are checked to see if a restart (see Algorithm 5) is required. When the restart mechanism is not triggered, two parents are selected, one from the population through tournament selection, and one randomly from the archive. The tournament size is determined by taking a set percentage of the current population (τ), lower bounded by a binary tournament. One offspring is produced through crossover and is then allowed to mutate, for which several operators have been designed. The offspring is then added to the population, see Algorithm 7, and to the archive, see Algorithm 6.

If a restart of the search is required, Algorithm 5 is invoked. First, the population is emptied and

Algorithm 4 The Iterate Procedure - Forest BORG

Check the population-to-archive ratio γ
if γ deviates by more than 25% from target **then**
  Revive search, see Algorithm 5
else if Epsilon progress ϵ is unchanged for 10 iterations **then**
  Revive search, see Algorithm 5
end if
$parent_1 \leftarrow$ Choose from Archive
Determine tournament size k
$parent_2 \leftarrow$ TournamentSelection(P, k)
offspring $\leftarrow$ Crossover$(parent_1, parent_2)$
Mutate(offspring) with probability operators
$P \leftarrow P \cup \{\text{offspring}\}$
UpdateArchive(offspring, A) {add tree offspring to archive A if eligible}

replaced by the archive solutions. Then, the new population size is calculated by multiplying the calculated γ, different from the target population-to-archive ratio, with the population size, at this point equal to the archive size. Vacant spots are filled by mutated members of the archive until the population reaches the newly calculated population size. The mutation of archive members is controlled through the operator feedback loop. These new solution are added to the archive if applicable, see Algorithm 6.

Algorithm 5 The Restart Procedure - Forest BORG

$P \leftarrow \emptyset$ {Empty the current population P}
$P \leftarrow A$ {Refill P with solutions from archive A}
$P_{new} \leftarrow$ ComputeNewPopulationSize(a) {$P_{new} = \gamma * size(A)$}
while $|P| < P_{new}$ **do**
  $member \leftarrow$ Mutate(ArchiveMember)
  $P \leftarrow P \cup \{member\}$ {Inject mutated member into P}
  UpdateArchive($member$, A) {Update A if $member$ is eligible}
end while
Determine new tournament size based on P_{new}

The population and archive have different procedures for admitting a new member. These different procedures are described in Algorithm 6 and Algorithm 7. The defining difference is the epsilon dominance requirement for admittance into the archive.

Algorithm 6 The Update Archive Procedure - Forest BORG

for all members $a \in A$ **do**
  if Candidate $\prec_\epsilon a$ **then**
    Remove a from A
  else if $a \prec$ Candidate **then**
    Discard Candidate and exit
  end if
end for
Add Candidate to A if non-dominated by any $a \in A$

Mutation Operators

Forest BORG employs seven different mutation operators. For the binary trees that are employed in this research, two different types of mutations are possible: subtree mutations and point mutations. For a subtree mutation, an internal node is chosen at random and deemed the root node of the subtree. The value of every node in the subtree is changed at random but the type of node is retained (indicator or action node) which ensures structural integrity of mutated trees. A point mutation on the other hand

Algorithm 7 The Add to Population Procedure - Forest BORG

1: **if** Offspring $\prec$ some $p \in P$ **then**
2: Replace a randomly chosen dominated $p \in P$ with Offspring
3: **else if** Exists $p \in P$ such that $p \prec$ Offspring **then**
4: Do not add Offspring to P
5: **else**
6: Replace a randomly chosen $p \in P$ with Offspring {Offspring and every $p \in P$ are mutually non-dominated}
7: **end if**

constitutes a value change, not a type change, on only one randomly picked node. The difference between these mutations is visualized in Figure 5.2.

Given that there are multiple possible actions, for the RICE model a change in emission control rate and/or savings rate and/or initial rate of social time preference for consumption, a further division can be made between the mutation operators, namely how many actions are changed. Three action variables thus result in six different mutation operators. These six different operators offer a range of change for a policy tree. A point mutation in which only one action is changed, if an action node is selected, results in a mutated tree that should perform very similarly to the original tree. On the other side of the spectrum, a subtree mutation in which all three actions change is the most different a mutation can get from the original and should perform differently. These operators are selected based on a feedback loop in which the better performing operators have a higher probability of being used again. The probabilities are updated by counting the number of solutions in the archive that were produced through each operator, $C_1, C_2, ..., C_k$. The corresponding probability Q_i is then updated by:

$$Q_i = \frac{C_i + 1}{\sum_{j=1}^{K}(C_j + 1)} \tag{5.1}$$

This operator performance based feedback loop is inspired from BORG. The idea behind these six different operators is that at the start of the search in objective space, mutations should allow for large changes which offers a larger opportunity to search in objective space. As the pareto front however approaches the true Pareto Front, the mutations should get more distinct and shrink the search in objective space to more precise solutions.

Another possibility is a completely random mutation in which every node is susceptible to mutation, given a certain chance of mutation. This is the mutation operator employed by the original POT algorithm. Along with the other six operators, this operator is incorporated with the exception that, if a leave node is selected, every action is allowed to change, instead of just one. This seventh operator serves as control for the other six operators so their performance can be benchmarked.

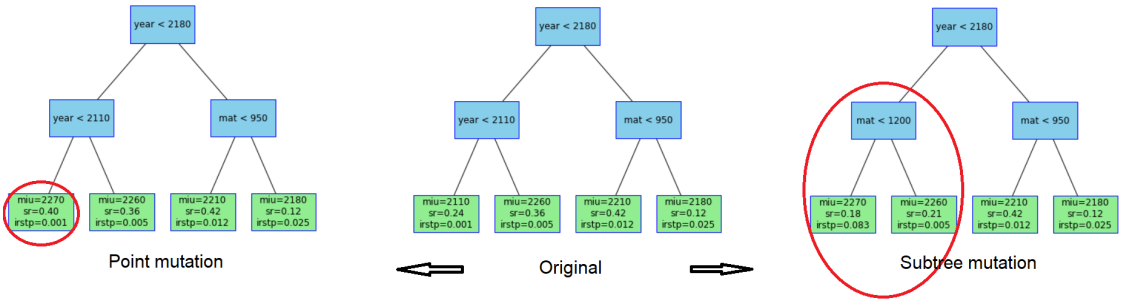

Figure 5.2: The difference of a point mutation and a subtree mutation for arbitrary policy trees. Note that if an action is mutated, one, two or all actions can be changed in value.

Recombination Operator

Forest BORG employs one recombination operator, a crossover mechanism with two after care functions, pruning and bloat control. The crossover operator takes two parent solutions and randomly

selects a node in each. All child nodes to this subtree root node are part of the subtree in each parent solution. These two subtrees are then swapped. One of these parents, with a swapped subtree is now the offspring. The offspring can contain illogical structures, *e.g.* contradicting thresholds in child nodes or duplicate actions, which are removed by a pruning function. The offspring could also have grown beyond its permissible maximum depth as a result of the crossover. A bloat control function ensures the maximum tree depth is not superseded. The maximum tree depth is determined *apriori* to the search as a hyperparameter to the algorithm. The optimal tree depth should be tested for for every new problem. The crossover and pruning operations are visualized in Figure 5.3 and Figure 5.4.

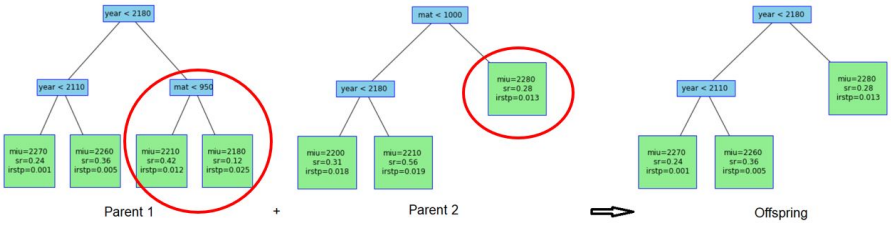

Figure 5.3: The crossover function visualized for two arbitrary parent trees.

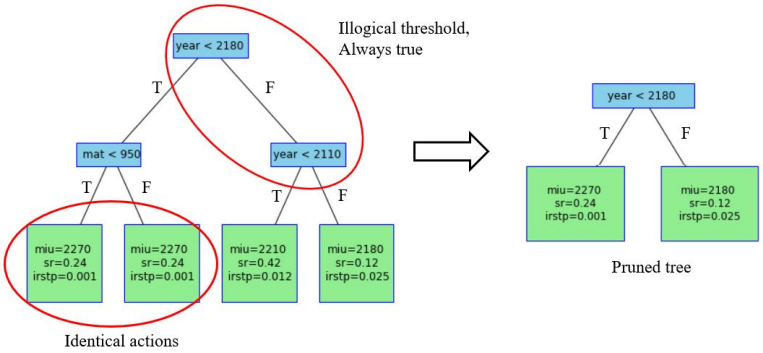

Figure 5.4: The pruning operation visualized on illogical structures in an arbitrary tree. The true (T) and false (F) conditions are added to underline the illogical structure.

Selection Operator

The selection operator in Forest BORG is an adaptive tournament function that selects individuals out of a population that will serve as parents in the next generation of the population. The logic behind this selection operator is entirely taken from the BORG algorithm (Hadka and Reed 2013).Two distinct groups evolve in Forest BORG, the population and the archive. The archive size is essentially bounded by the definition of the epsilons, the population size is dynamic and unbounded. Employing an unbounded population size makes the algorithm applicable to a wider selection of case studies as more objectives can be incorporated into the problem (Hadka and Reed 2013). Some case studies have more objectives than others. As the number of objectives in an optimization problem increases, the number of non-dominated solutions in a fixed population size will increase, which degrades the selection pressure of the search. To keep the selection pressure at an approximately fixed level, the population size has to increase. When the restart mechanism is triggered, see Algorithm 5, a new population size is calculated based on the current size of the archive. The new population size is calculated by:

$$P_{new} = \gamma * A \tag{5.2}$$

in which γ is the target population-to-archive ratio and A is the number of current archive solutions. A new tournament size is determined when the population size has changed. The new tournament size is determines through:

$$tournament\ size = max(2, \lfloor \tau(\gamma * A) \rfloor) \tag{5.3}$$

in which τ is a selection ratio that maintains tournament sizes to be a fixed percentage of population size. The lower bound of this adaptive tournament size is 2, resulting in a binary tournament which can

be enforced by setting τ to zero. The tournament size determines how many random members of a population are checked for non-dominance and are kept in the next generation as parents.

5.2. Benchmarking Metrics

Forest BORG must be able to support decision-making in socio-environmental systems and ensure explainability and interpretability for multiple actors. Forest BORG is applied to the two case studies to test whether it is capable to fill the knowledge gap. Five analyses are carried out for each case study to test the performance of Forest BORG.

To test the capability of decision-making support in socio-environmental systems, Forest BORG is applied to the Folsom lake model, a water management problem and to RICE, an IAM. Forest BORG should be effective, efficient and reliable meaning it should be able to converge to an approximate optimal Pareto Front, converge quickly and do so across multiple random trials. To test whether Forest BORG is effective, efficient and reliable, the advancing pareto front throughout a simulation is captured through hypervolume and plotted against the number of function evaluations for five different seeds. The hypervolume of a set of solutions measures the size of the portion of objective space that is dominated by those solutions collectively (While, Bradstreet, and Barone 2011). Hypervolume captures in one scalar both the closeness of the solutions to the optimal set and their spread across objective space (While, Bradstreet, and Barone 2011). Moreover, this analysis for the Folsom case shows that Forest BORG is competitive with the original POT algorithm. The hypervolume of a pareto front is determined by first normalizing each objective value and then feeding these normalized values to the hypervolume calculator created by the Walking Fish Group (While, Bradstreet, and Barone 2011). The objective values are normalized by taking all simulated pareto fronts, across seeds and when possible across algorithms, and selecting the minimum and maximum value of each objective from this enlarged set of pareto fronts. The objective value in every pareto front is then normalized according to these maximum and minimum values.

The novel mutation operators of Forest BORG are recorded and visualized in a stacked bar chart which displays the dynamics of these operators throughout a simulation. The expectation is that the more course mutation operators, subtree mutations, dominate the start of the search but give way to the more refined mutation operators, point mutations, towards the end of the simulation, as the algorithm tries to make fine improvements of the pareto front. The subtree mutation operators allow for the creation of children that are structurally more different from their parents than point mutations could allow for. Whether these large structural changes in the offspring also lead to large advancements in objective space, *i.e.* how genotype relates to phenotype, and therefore quicker convergence towards the Pareto Front remains to be seen. These search operator dynamics are also tested for their reliability by evaluating five different seeds.

The optimal hyperparameter settings of Forest BORG are visualized in a controllabilty map. The hyperparameters that are controlled for are non-exhaustive but include the most important ones, discovered throughout the development of the algorithm. The hyperparameters that are controlled for in the controllability map are maximum tree depth, the target population-to-archive ratio and the restart interval, indicating after how many function evaluations the criteria for a restart are checked. A hyperparameter like τ, the fixed percentage of the population size that determines the tournament size, or the mutation probability that is only used in the control mutation operator are not tested and kept constant throughout all simulations.

The capability of Forest BORG to produce policies for multiple actors is shown by visualizing the final pareto front through a parallel axis plot. A parallel axis plot shows a scale for each objective side by side. A policy will score on each of these scales, drawing a line between each objective axis. Trade-offs between objectives are quickly observed through a parallel axis plot. Assuming multiple actors have different preferences and therefore adhere to different objectives, Forest BORG should be able to lay bare the inherent trade-offs to the Folsom and RICE case studies. A select number of policies are highlighted from the pareto front, one for each objective that performs best for that objective compared to other policies and a compromise policy, that performs closest to the average of each objective. This

analysis is performed for one random trial, for one seed only.

To test the capability of Forest BORG to produce explainable and interpretable policies, the highlighted policy trees from the parallel axis plot are visualized. The visualization of these trees shows the actions that are taken dependent on system states, making them inherently more explainable than real-valued decision variables. The policy trees should also be interpretable, so preferably with small depths and no complex or illogical structures. Policy trees with smaller maximum depths reduce the risk of overfitting if the same policy should be applied to a different scenario and they make a policy tree easier to interpret. There should be no illogical structures in the policy trees, *e.g.* illogical thresholds for indicator variables that contradict a threshold of a parent indicator node. The pruning and bloat control operations should prevent these illogical structures but are checked in these visualizations of the policy trees. Moreover, similarities and differences in structure, indicators and actions between policies are studied that perhaps reflect the trade-offs shown in the parallel axis plots. .

5.3. XLRM Framework

The conceptual models of both case studies are made operable by presenting them through the XLRM framework. RICE and the Folsom lake model are adapted to the XLRM framework of exogenous uncertainties (X), levers (L), relations (R) and metrics (M) (Lempert 2003). The exogenous uncertainties are factors of the model over which the decision maker has no control. Different scenarios stem from different values these variables might hold. The levers are the variables over which the decision maker has full control, often upper and lower bounded, they can be set to any value before and during the simulation. The set of all levers tuned to a specific value is called a policy. In the terminology of POT the lever variables are called action variables. The relations are the internal logic of the model and can be chosen as indicator variables in a policy tree. The metrics are the model variables the decision maker finds most important and provides a value judgment of the system given a scenario or a policy. They are often referred to as objective values, or KPI's.

It is important to note that this framework is well-suited for policy trees because of the direct mapping of levers, relations and metrics in the XLRM framework to actions, indicators and objectives in a policy tree. The uncertainties of each case study are not used in this research as this research focuses on the development, rather than the application of Forest BORG. Extending the current work to robustness and vulnerability tests on pareto-optimal policy trees is easily achieved with the models and their current implementation. Designing robust policies, *i.e.* policies that perform well under a wide range of scenarios, is done by creating scenarios for each model according to the variables and their ranges.

5.3.1. Folsom Lake Model

Figure 5.5 depicts the Folsom lake model in the XLRM framework.

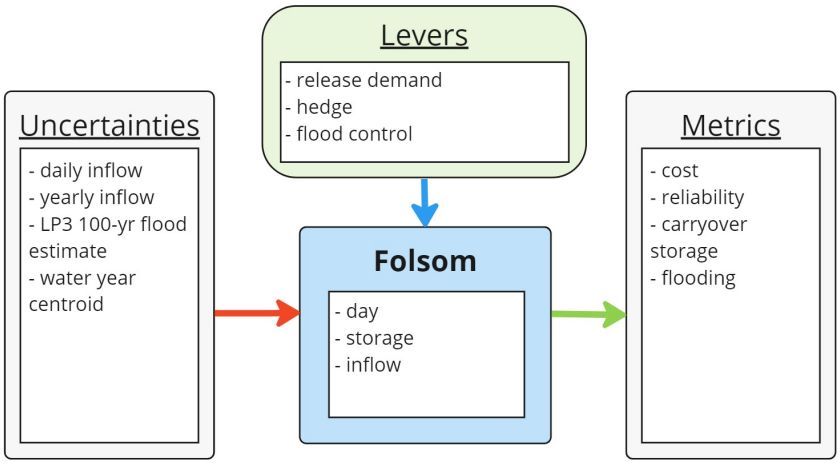

Figure 5.5: The Folsom lake model casted in the XLRM framework. The blue box represents the Folsom model, out of which three variables are taken that are used as possible indicator variables in this research.

Uncertainty

The uncertainties for the Folsom model are slightly complicated as this model was taken from literature (Herman and Giuliani 2018). The exact scenario data, built on the values and therefore ranges of uncertainty, should be requested from the authors but the uncertain variables and their descriptions are given in Table 5.1. The projected changes to volume and timing of inflow are drawn from a U.S. Bureau of Reclamation study (Reclamation 2013).

Uncertainty	Description
Inflow (30-yr moving avg)	Annual inflow to the reservoir [TAF/y]
LP3 100-yr estimate	Cummulative estimate of the 100-year flood [TAF/d]
Water year centroid	Day of water year on which 50 % of cummulative runoff is reached [d]
Inflow	Daily inflow ot the reservoir [TAF/d]

Table 5.1: Uncertainties in the Folsom model, taken from (Herman and Giuliani 2018). The units TAF/y and TAF/d stand for thousand acre-feet per year and thousand acre-feet per day. One TAF is 0.0012 km^3.

Levers

The Folsom model has one action, which can be set to different values. A distinction between the Folsom model and the RICE model is that the Folsom model has one action that can be set to different values and the RICE model has three actions that can be set to different values for each action. This distinction becomes important when the original POT algorithm is compared to Forest BORG in the different case studies. The action, or lever, in the Folsom model is the amount of water to be released. The current demand for water can be released or a percentage of it, hedging part of the water for future use. In case of large inflows of water, large amounts of water can be released through flood control. The different, discrete, values that this lever can be set to are stated in Table 5.2 and are spelled out in detail in Appendix G.

Lever	Target release rule	Description
Release Demand	$u_t = D_t$	Release current demand
Hedge p	$u_t = p * D_t$	Release p% of current demand
Flood Control	$u_t = (Q_t + S_{t+1})$	Release 20% of current inflow and next system state

Table 5.2: Levers in the Folsom model, taken from (Herman and Giuliani 2018). u_t is the amount of water released at time t and D_t is the current demand for water. See Equation 5.4 to better understand the release rule for Flood Control

Relations

The model implementation is based on a transition equation:

$$s_{t+1} = f(s_t, u_t, q_{t+1}) \tag{5.4}$$

where s, u, and q are vectors of state, decision, and disturbance variables (Herman and Giuliani 2018). A cost function is based on system state and is to be minimized by choosing the decision vectors accordingly. The disturbance variables are the inflow of water into the reservoir, subject to uncertainty. The exact model relations can be found in the associated code.

Metrics

The Folsom lake model can be run as a single objective problem, not covered in this research, or as a multi-objective problem. The cost function for the multi-objective problem is divided into four objectives. These objectives, or metrics, are stated in Table 5.3.

Metric	Description
Cost	Discounted cost of the summed annual overall cost of all actions implemented
Reliability	Weighted reliability to the Central Valley Project
Carryover Storage	Count of # years in which carryover storage falls below minimum TAF
Flooding	Represents the total volume of water exceeding downstream capacity.

Table 5.3: Levers in the Folsom model, taken from (Cohen and Herman 2021)

5.3.2. RICE

Figure 5.6 depicts the RICE model embedded in the XLRM framework.

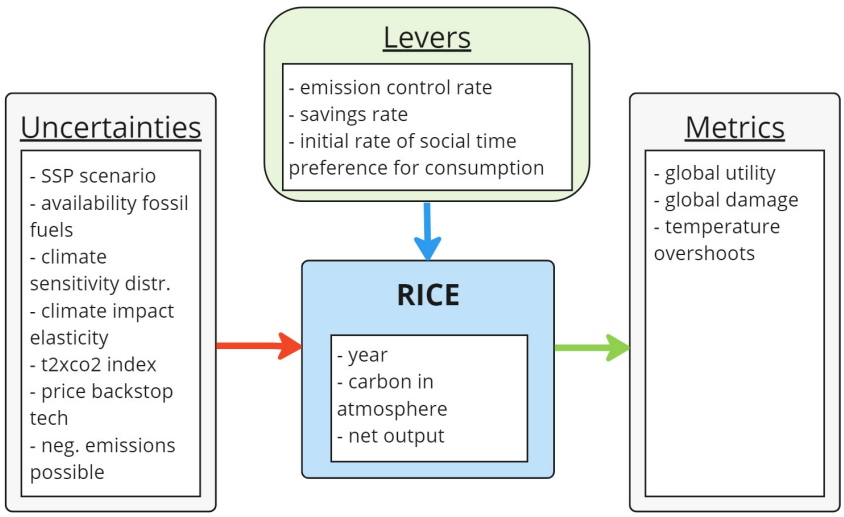

Figure 5.6: The RICE model casted in the XLRM framework. The blue box represents the RICE model, out of which three variables are taken that are used as possible indicator variables in this research.

Uncertainty

Uncertainty in the RICE model is incorporated through nine variables grouped in three distinct channels. The population and total factor productivity are grouped together in an SSP scenario and cannot be changed individually. The climate sensitivity distribution and parameter also act coherently to define the sensitivity of the climate to emissions. Seven distinct variables remain which are the uncertainties as defined in the XLRM framework. The three different channels of uncertainty are socio-economic, climate and technological backstop uncertainties. The uncertainties, their classification and the range within they operate are give in Table 5.4 and is taken from a previous version of the RICE model (Reddel 2022).

Uncertainty	Type	Range
population	socio-economic	5x SSP
total factor productivity	socio-economic	5x SSP
emission to output growth rate	socio-economic	5x SSP
availability of fossil fuel	socio-economic	[4000, 13649]
climate sensitivity distribution	climate	{normal, lognormal, Cauchy}
climate sensitivity parameter	climate	[0, 999]
elasticity of climate impact	climate	{-1, 0, 1}
price of backstop technology	backstop	[1260, 1890]
availability of negative emissions technology	backstop	yes (20%), no

Table 5.4: Uncertainties in the RICE model, adapted from (Reddel 2022)

Levers

There are three different levers in the RICE model which are given in Table 5.5 and are adapted from (Reddel 2022). The savings rate and emission control rate are set as initial values which then lead to steady-state behaviour. However, because the POT algorithm allows for the changing of levers during the simulation period these variables, which in previous RICE models (Reddel 2022) (Tjallingii 2021) were static throughout the simulation, can now be dynamic.

Lever	Range
savings rate	[0.1, 0.5]
emission control rate target	[2065, 2305]
initial rate of social time preference consumption	[0.001, 0.1]

Table 5.5: Levers in the RICE model, adapted from (Reddel 2022)

Relations

The relations of the RICE model state how information flows within and between sub-models and relates uncertainties and levers to output, the metrics. The core structure of the RICE model is shown in Appendix D in figure D.1 in which only the core variables and relations are depicted for clarity. Note that although the variables under uncertainty and levers are depicted in the figure, the metrics are not. Although the metrics are part of the RICE model, they are different for different problem formulations and often require further calculations from the RICE core model. Most stem from consumption and a utility variable has been added to depict an example of a possible metric.

Metrics

Numerous metrics can be defined based on the core working of RICE. The definition of the metrics depends on the justice principle one adheres to. Previously, metrics based on the RICE simulation model have been based on Utilitarian, Egalitarian, Prioritarian and Sufficitarian justice principles (Tjallingii 2021), (Reddel 2022). This research will focus on the Utilitarian perspective as it is commonly used as an ethical perspective and its objectives are relatively straightforward. After the proof-of-concept of the POT and OLT algorithms coupled to RICE, the objectives, or metrics, can be extended to other justice principles. The metrics employed for the Utilitarian perspective are given in table 5.6. The choice of metrics is relatively straightforward in an effort to keep the results comprehensible and comparable to other research (Marangoni et al. 2021).

Metric	Description
welfare	Welfare of discounted utility based on consumption
damages	global damages
temperature overshoots	Amount of ^oC above the threshold of 2.0 ^oC

Table 5.6: Levers in the RICE model, adapted from (Reddel 2022)

5.4. Experimental Setup

The experiments are set up per case study. Each case study as well as the performance of Forest BORG is studied through five analyses. These analyses are:

1. run-time dynamics
2. search operator dynamics
3. controllability
4. exposed trade-offs
5. highlighted policy trees

The run time dynamics show convergence of the algorithm based on hypervolume. The search operator dynamics investigate the novel mutation operators of Forest BORG. The controllability analysis explores the influence of the hyperparameter settings on algorithm performance. The fourth analysis displays the trade-offs made visible by pareto-optimal policy alternatives through a parallel axis plot. A best performing policy is highlighted for each objective as well as a compromise policy, a policy that is closest to the average of each objective. Finally, the fifth analysis creates visuals of these selected policies, displaying the actual policy trees. Similarities and differences between the different policies are sought after that perhaps reflect the trade-offs discovered in the fourth analysis.

The exact hyperparameter setting per analysis is displayed in Table 5.7. Every analysis is performed with the Forest BORG algorithm. In addition to the analyses described above, the Forest BORG algorithm is also compared to the performance of the original POT algorithm as it was originally applied to the Folsom lake model (Cohen and Herman 2021). The algorithms are compared through their run-time dynamics on the Folsom lake model with discrete actions. All analyses for the RICE model are performed with continuous actions. γ refers to the population-to-archive ratio, which is a target value. The restart mechanism is evoked, if the restart interval allows for it, when the actual population-to-archive ratio differs by more than 25% from this target value. τ is the fixed percentage of the population size that determines the tournament size. τ is kept constant across all analyses and its value is taken from the recommendations of the BORG algorithm (Hadka and Reed 2013). The population size is allowed to change in Forest BORG and therefore also the tournament size.

The two algorithms are not compared on the RICE model because of POT's limitation of only being able to handle one lever, while the RICE model has three. The actions in the RICE model can either be discrete or continuous. Continuous actions can take any value within the supplied range. Discrete actions are pre-programmed into the model and relate a handful of actions to a certain value. A table of all discrete values used in this research can be found in Appendix G for both case studies. The original POT algorithm is only equipped to deal with either multiple discrete actions, *i.e.* multiple different values for the same action, or one continuous action. The multiple discrete actions must refer to the same action, but set to different values. The RICE model has 3 separate actions and is thus not suitable to be coupled to the original POT algorithm, even if it were to be run with discrete actions. Either the original POT algorithm can be adapted or two of the three actions in RICE are randomly chosen. None of these options are explored in this research as adapting the RICE model or the original POT algorithm distracts from the other analyses.

Folsom	Analysis 1	Analysis 2	Analysis 3	Analysis 4	Analysis 5
nfe	30000	30000	20000	30000	30000
max depth	3	3	2-6	3	3
γ	4	4	3-5	4	4
restart interval	5000	5000	[500, 2000, 5000]	5000	5000
τ	0.02	0.02	0.02	0.02	0.02
# seeds	5	5	1 (45 runs)	1	1
algorithm	FB & original POT	FB	FB	FB	FB
actions	discrete	discrete	discrete	discrete	discrete
RICE	Analysis 1	Analysis 2	Analysis 3	Analysis 4	Analysis 5
nfe	30000	30000	20000	30000	30000
max depth	3	3	2-6	3	3
γ	4	4	3-5	4	4
restart interval	5000	5000	[500, 2000, 5000]	5000	5000
τ	0.02	0.02	0.02	0.02	0.02
# seeds	5	5	1 (45 runs)	1	1
algorithm	FB	FB	FB	FB	FB
actions	continuous	continuous	continuous	continuous	continuous

Table 5.7: Experimental set-up for each analysis and for each case study. Note that FB refers to the Forest BORG algorithm.

6

Results

The results of this research are presented through five analyses for each case study. First the results for the Folsom lake case study are shown and then in a similar fashion the five analyses for the RICE case study are presented. The chapter is concluded by reflecting on the performance of Forest BORG across the two case studies. The data generated to produce the figures in this chapter are the result of the simulations run according to the experimental setup, presented in Table 5.7. The data from analysis 1 is used for the hypervolume run-time dynamics figures. Analysis 2 relates to the search operator dynamics figures, analysis 3 to the controllability maps, analysis 4 to the parallel axis plots and analysis 5 to the individual policy trees visualizations.

6.1. Folsom

6.1.1. Run-time Dynamics

The run-time dynamics of Forest BORG are shown in Figure 6.1 as well as the run-time dynamics of the original POT algorithm. Each algorithm is run for five random trials. Forest BORG is both efficient and reliable as it converges quickly, around 20.000 NFE and reliably across the five seeds. There is no theoretical optimal Pareto Front known for the Folsom lake model so the effectiveness of Forest BORG is judged in relation to the competing original POT algorithm. Forest BORG is judged as effective as it outperforms the original POT algorithm across all random trials.

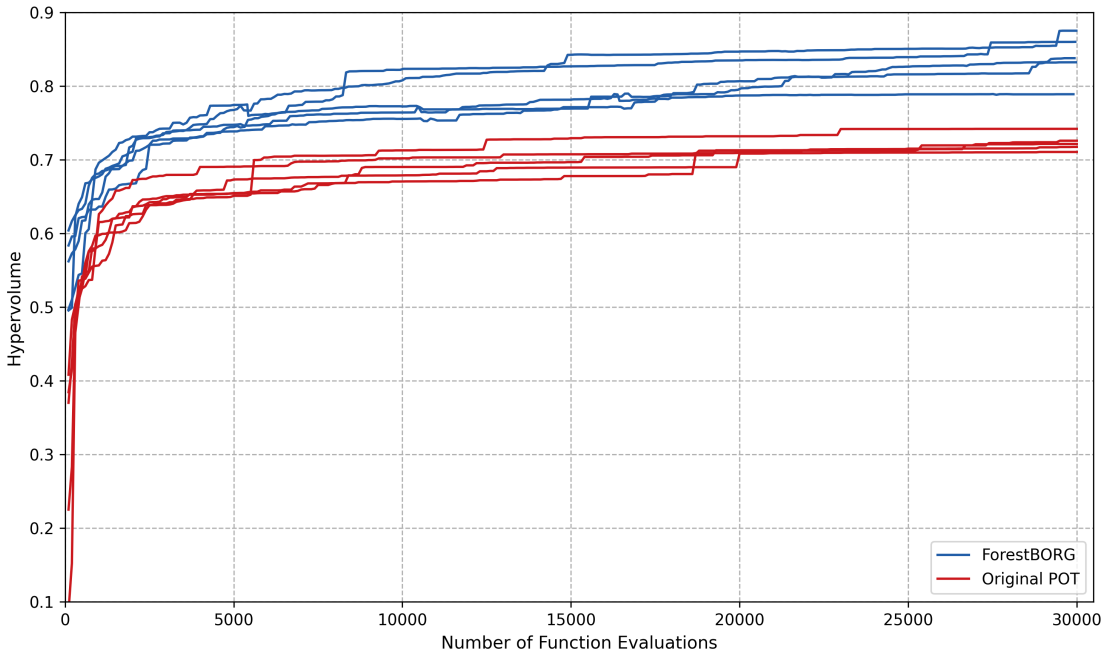

Figure 6.1: Hypervolume run-time dynamics of Forest BORG and the original POT algorithm on the Folsom lake model. Every algorithm is run with 5 seeds each.

6.1.2. Search Operator Dynamics

The search operator dynamics of Forest BORG across all random trials are presented in Figure 6.2. The operator dynamics display three features of interest. Firstly, across all random trials, the point mutation operators are dominant over the subtree mutations. Secondly, the type of point mutation is irrelevant but this is to be expected as the Folsom lake model is a one action problem, differentiating between the number of actions is irrelevant if there is only one possible action to mutate. Thirdly, the point operators outperform the control operator, the random mutation. This result shows that allowing for small point mutations is more effective than random mutations as well as a strong preference for point mutations over subtree mutations.

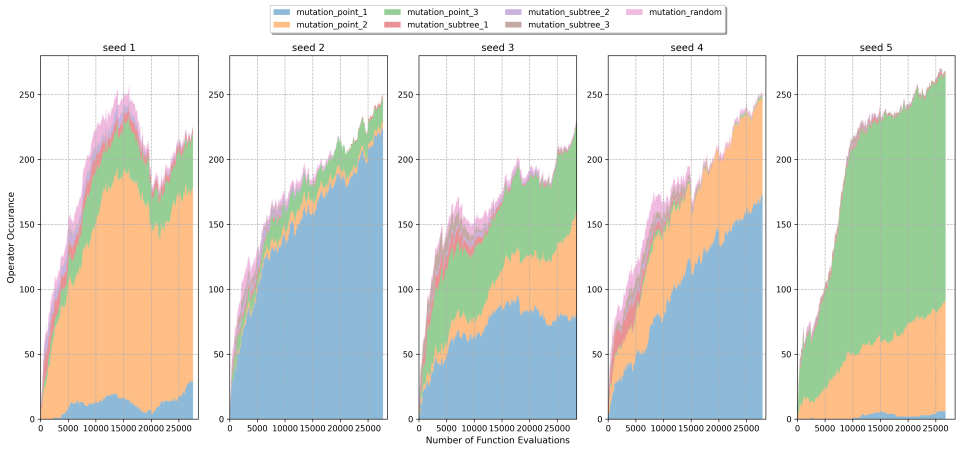

Figure 6.2: Search operator dynamics for Forest BORG on the Folsom lake model. Forest BORG is run for five different seeds.

6.1.3. Controllability map

The controllability map of Forest BORG for the Folsom lake model is shown in Figure 6.3. The map indicates that not too large maximum tree depths are preferred, which also helps to reduce overfitting and the population-to-archive ratio should not become too large. Larger target population-to-archive

ratios enlarge the margin around that target as the restart mechanism is triggered when the calculated population-to-archive ratio differs by 25%. The restart interval appears to be almost irrelevant, as long as the other hyperparameters are properly set. The hypervolume metric is used as it allows for the comparison between pareto fronts based on only one number.

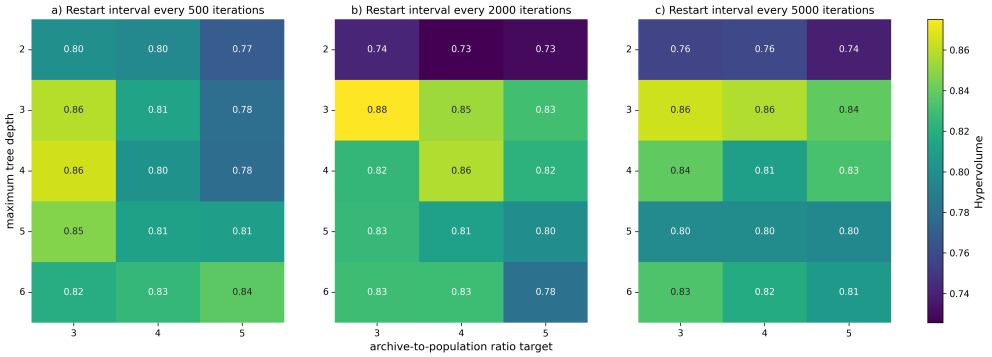

Figure 6.3: Controllability map of Forest BORG on the Folsom lake model.

6.1.4. Trade-offs

The trade-offs within the discovered pareto front are made visible in Figure 6.4. A specific policy is highlighted that reflects the best option for each separate objective and a compromise policy. Notice that the best carryover storage and best flooding policies, indicated by the purple and green line respectively, score very similarly on each objective. Another observation is that the reliability objective seems to conflict most with the other objectives. Next, the selected policy trees are analyzed for differences and similarities based on their structure.

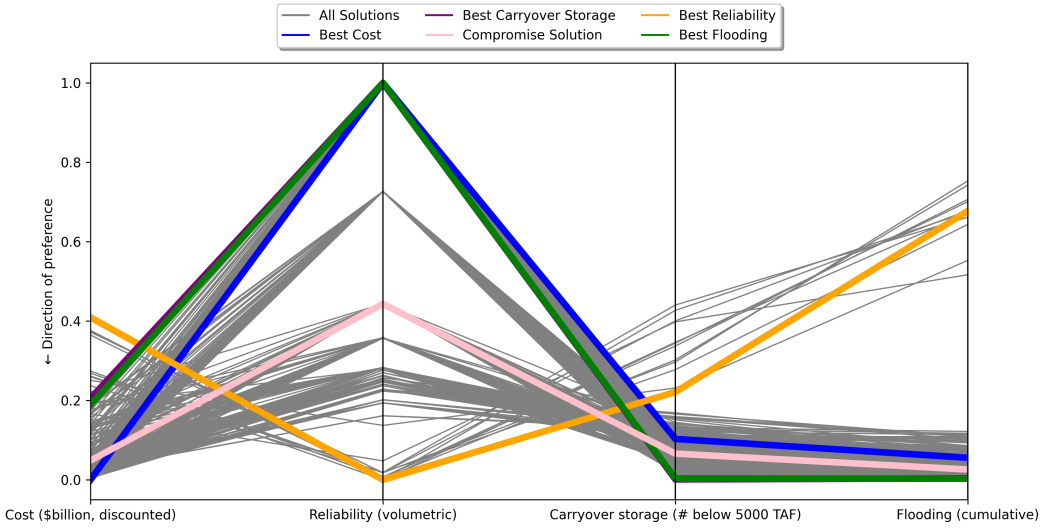

Figure 6.4: Parallel axis plot of the pareto front developed by Forest BORG for the Folsom lake model. Policies of special interest are highlighted. A parallel axis plot shows each objective in the objective space alongside each other. Here, all objectives are to be minimized so an ideal policy would be indicated by a flat line on the bottom across all objective axis. This ideal policy however does not exist. Instead, there are trade-offs, which are easily observed when two lines cross.

6.1.5. Highlighted Policy Trees

The selected policy trees are visualized in Figure 6.5. The best policy trees for carryover storage and flooding performed similarly on the four objectives, as shown in the parallel axis plot of Figure 6.4. The structures of the policy trees are similar as well. Even though the depth of the trees is not the same, nor the indicator variables, the actions are nearly identical. Only the hedging actions are featured in these

trees, of which one is shared. The release demand and flood control actions are featured in the other policy trees, as well as the other options for hedging actions. It is perhaps odd that the day indicator variable is featured in the selected policy trees but the day variable actually carries trend information as the Folsom lake model contains a strong seasonality component in the water inflow data. Every possible action is utilized in the highlighted policy trees. Some trees make more intuitive sense than others. For example, the policy tree that minimizes cost, is the only tree that incorporates the 'release demand' action, which is the most cost saving actions across all possible actions. However, the policy optimizing flooding does not incorporate the 'flood control' action. Apparently, in this scenario, 'flood control' need not be used to limit the damages and costs of flooding.

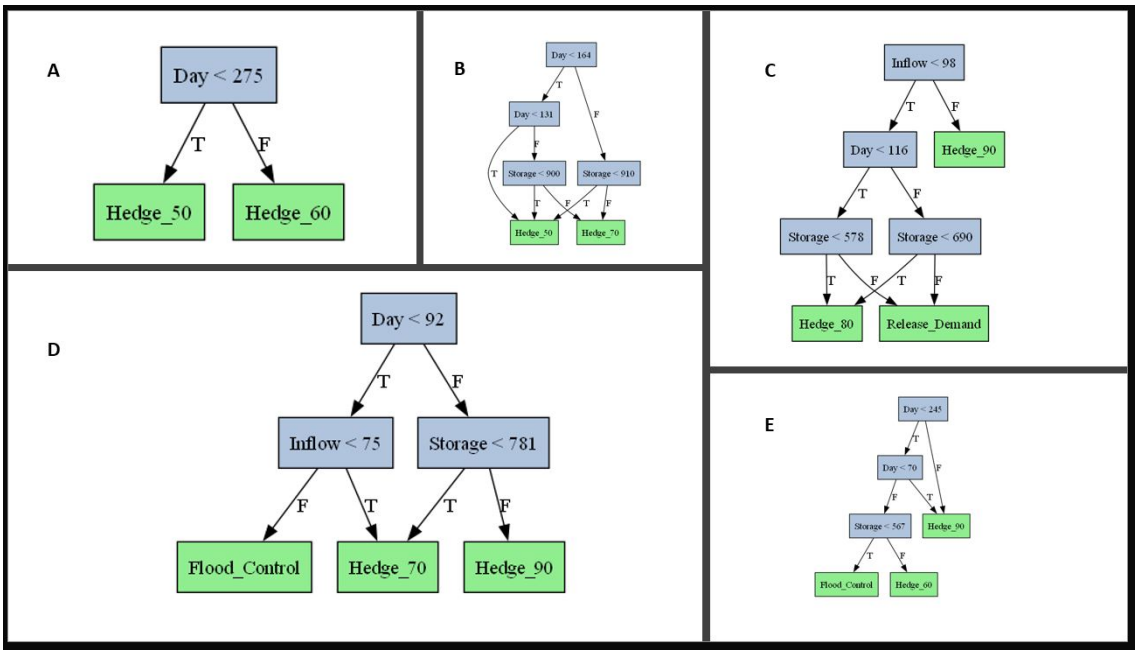

Figure 6.5: Visualization of policy trees of special interest. A) Best Carryover Storage policy tree B) Best Flooding policy tree C) Best Cost policy tree D) Compromise policy tree E) Best Reliability policy tree

6.2. RICE

6.2.1. Run-time Dynamics

The run-time dynamics of Forest BORG are shown in Figure 6.6 for five random trials. Forest BORG is both efficient and reliable as it converges quickly, around 20.000 NFE and reliably across the five seeds. This is a result that is consistent across both case studies. Because there are three actions in the RICE model, and the original POT algorithm is not equipped to handle more than one action, the effectiveness of the algorithms cannot be compared for the RICE model. The fact that there are multiple actions in the RICE case study does allow the search operators to be properly tested. The search operator dynamics of the Folsom model could only investigate the dynamics of point mutations versus subtree mutations but for the RICE model the number of actions that are mutated can be investigated, as is done in the next section.

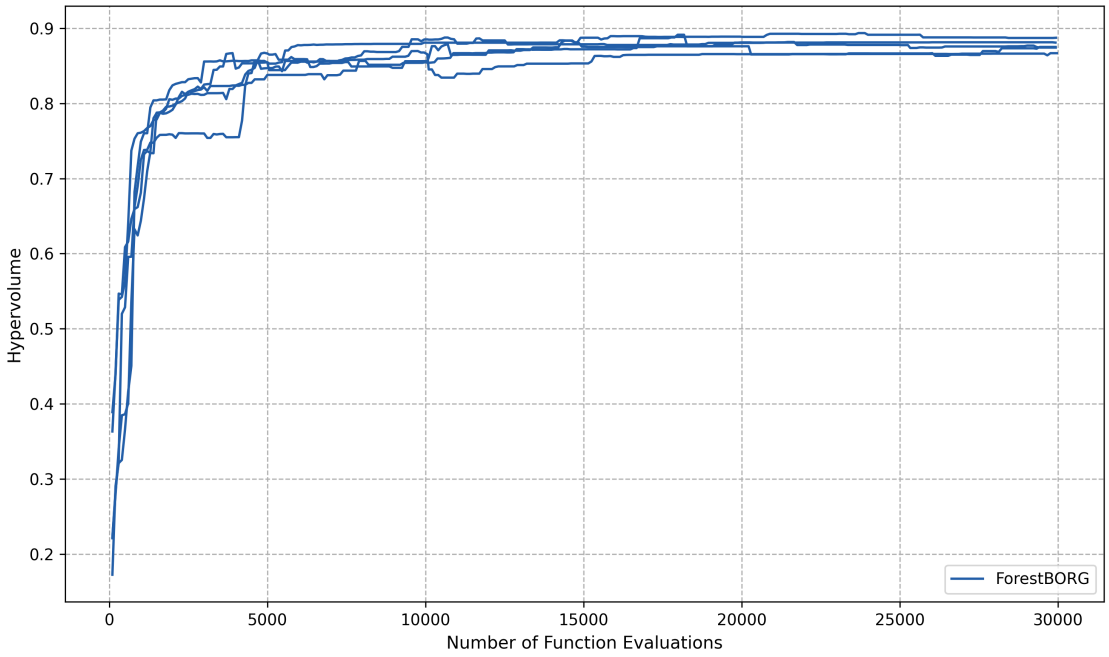

Figure 6.6: Hypervolume run-time dynamics of Forest BORG and the original POT algorithm on the RICE model. Every algorithm is run with 5 seeds each.

6.2.2. Search Operator Dynamics

The search operator dynamics of Forest BORG on the RICE model are shown in Figure 6.7. Not all results are consistent across all random trials. It appears that at the start of the search, the subtree mutations are prominent but eventually disappear towards the end of the search, confirming the initial expectation. This behaviour is observed for all seeds except for the third one, in which the subtree mutations appear to become more prominent towards the end of the search. It should be noted however that also in this random trial, the subtree mutations in which two actions can be changed, give way to subtree mutations in which only one action can be changed, confirming the expectation that the finer mutation operators are preferred over the more course ones towards the end of the search. Of the point mutations, the ones with single actions changes are preferred over double action changes and triple action changes are almost not used, although there is some degree of variation across the random trials. Both subtree and point mutations outperform the control mutation, in which mutation is completely random. This result is consistent with the findings in the Folsom lakse case study.

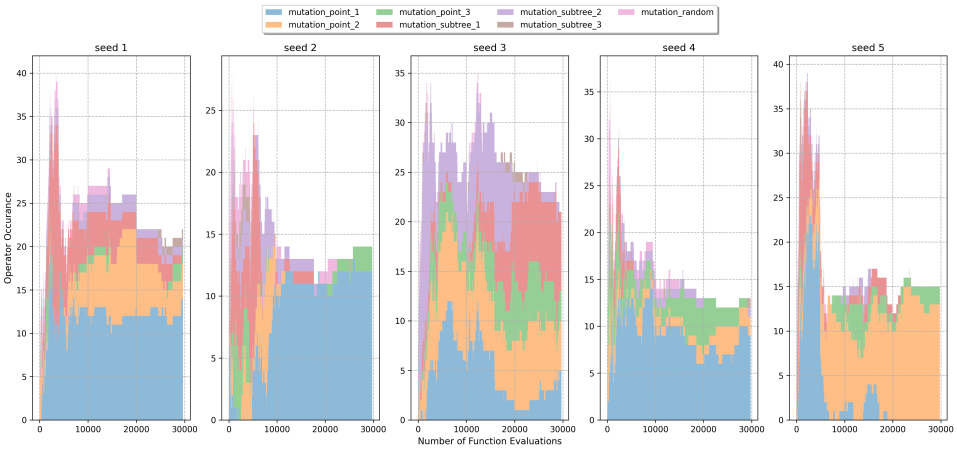

Figure 6.7: Search operator dynamics for Forest BORG on the RICE model. Forest BORG is run for five different seeds.

6.2.3. Controllability map

The controllability map of Forest BORG for the RICE model is shown in Figure 6.8. The map indicates through the high and narrow bandwidth of hypervolume values that the extent to which these hyperparameters influence performance is small, especially when it is compared to the range of performance found for the Folsom lake case study. This difference probably has to do with the fact that the Folsom model was run with a select number of discrete actions that were chosen such that all extremes over the action range were represented. The RICE model is run with continuous actions, therefore allowing pareto fronts to be much closer to one another for different hyperparameter settings. Within these small margins, there appears to be a slight preference for a medium maximum tree depth of 4 and an archive-to-population ratio of 4 and a stronger preference towards longer restart intervals, intervals of 5000 NFE.

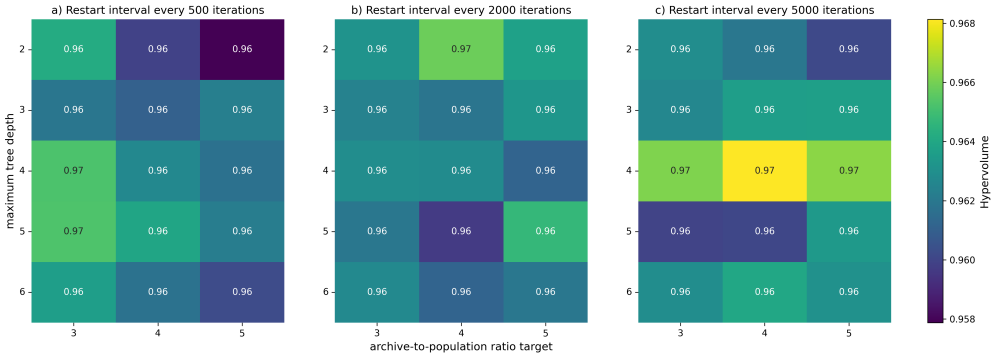

Figure 6.8: Controllability map of Forest BORG on the RICE model.

6.2.4. Trade-offs

The trade-offs within the discovered pareto front are made visible in Figure 6.4. The objective that appears to be most conflicting with the other objectives is damages. Intuitively it would be the welfare objective, this is further discussed in Chapter 7. The highlighted policies, reflecting the best possible solutions for each objective, are distinctive from one another as they score differently on almost all objectives.

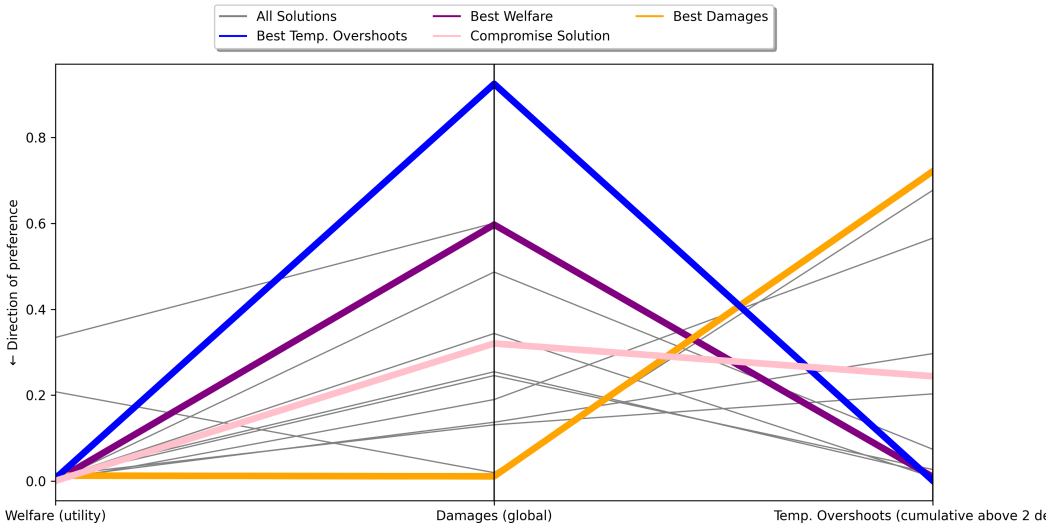

Figure 6.9: Parallel axis plot of the pareto front developed by Forest BORG for the RICE model. Policies of special interest are highlighted. A parallel axis plot shows each objective in the objective space alongside each other. Here, all objectives are to be minimized so an ideal policy would be indicated by a flat line on the bottom across all objective axis. This ideal policy however does not exist. Instead, there are trade-offs, which are easily observed when two lines cross.

6.2.5. Highlighted Policy Trees

The selected policy trees are visualized in Figure 6.10. Even though the selected policies performed distinctly different on almost each objective, as seen in the parallel axis plot of Figure 6.9, the structures across the policy trees appear to be similar. Firstly, only the carbon concentration in the atmosphere (mat) indicator, is used across all selected policy trees. This also confirms the mat variable as a suitable indicator variable. The year and net output indicators are not used. Note that the day indicator in the Folsom model is featured in the best performing policy trees but this is logically explained due to the seasonality of inflow in the Folsom model. The year variable in the RICE model can not be attributed to some system trend and these results show that Forest BORG picked up on it and did not use the indicator variable for the best policy trees. Secondly, there is not much variation in maximum tree depth across the selected policy trees, only policy trees with maximum depths of 2 or 3 are featured, while maximum depths of 4 were allowed. The actions of each policy tree are however distinctive from the other policy trees. A large range of the possible values for each action is covered across the selected policy trees.

The policy tree representing the best option to minimize temperature overshoots has a much lower threshold value for the mat indicator in the root node than the other policy trees. This is intuitively logical as the carbon concentration in the atmosphere is strongly tied to global atmospheric temperature. Similarly, relatively low saving rates are observed in the policy tree representing the best welfare option, compared to the other trees. Net output is divided through the savings rate in savings, to be used on abatement and consumption, the basis for welfare. A lower savings rate implies higher consumption. Also note how for the best welfare policy tree, the savings rate increases if there is more carbon in the atmosphere and is allowed to be lower when the system is in a state where there is relatively little carbon in the atmosphere. The policy tree representing the best option to limit damages shows an emissions control rate that is allowed to be further into the future for a system state with low concentrations of carbon in the atmosphere compared to a high concentration state when the emissions control rate should be set sooner. The system state in between, when the system is in a state of $780 < mat < 1120$ does however not follow this narrative. This could potentially be an artifact, due to the encountered counter-intuitive trade-off for the damages objective, see Chapter 7.

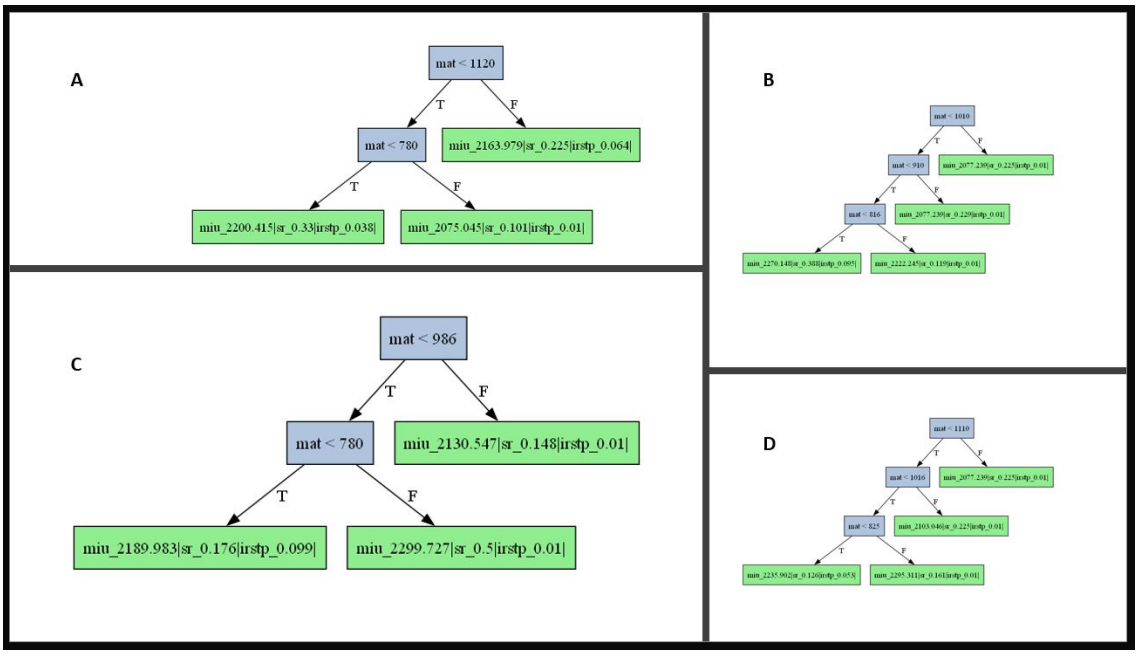

Figure 6.10: Visualization of policy trees of special interest. A) Best damages policy tree B) Best temperature overshoots policy tree C) Compromise policy tree D) Best welfare policy tree. 'mat' is the carbon concentration in the atmosphere, 'miu' the emission control rate, 'sr' the savings rate and 'irstp' the initial rate of social time preference for consumption.

6.3. Forest BORG Performance

The run-time dynamics show that Forest BORG performs effectively, efficiently and reliably across the case studies. The effectiveness of Forest BORG is underlined by comparing it to the original POT algorithm for the Folsom case study. The operator dynamics were expected to shift from triple action subtree mutations to single action point mutations during the search. This shift from subtree to point mutations was not observed in the Folsom case study and not for all random trials in the RICE case study. The search operators do outperform the control of random mutation, as employed in the original POT algorithm which make them a valuable contribution to the Forest BORG algorithm. The search operators might not perform as they are expected to, but they do enhance the search. The hyperparameters of Forest BORG are case study dependent but become, unsurprisingly, more important for case studies with a limited number of discrete actions. Across both case studies, policy trees with modest maximum depths, 3 or 4, are preferred which eliminates the risk of overfitting policy trees to a certain scenario.

Forest BORG has proved itself capable of producing policies for multiple actors as it has found pareto fronts for each case study that clearly display the trade-offs that are inherent to the case study. Moreover, the structures of the selected policy trees are in accordance to their performance, as seen in the parallel axis plots. Analyzing these policy trees shows what actions need to be taken and under what conditions when certain objectives are valued more than others, making the results of this analysis explainable and valuable to decision-makers.

7

Discussion

Forest BORG is a new tool in a decision-makers toolkit that allows for multi-objective decision making for socio-environmental systems. This research can contribute to future work that incorporates decision making under deep uncertainty. Designing robust, explainable policy trees, *i.e.* policies that perform well over a range of scenarios, is easily done for the two case studies. Forest BORG is also easily applied to other case studies. A model of a case study is coupled to Forest BORG by defining metric, indicator and action variables for the model. It could be interesting to repeat previous research of policy design for socio-environmental systems. Previously, the MOEAs employed in these studies used real-valued decision variables which Forest BORG changes to tree-structured variables. The results of these studies are thus expected to show better pareto fronts, because the actions changed from static to dynamic, and offer explainable policies.

The results of this research also require critical reflection. There are two notable observations of the results in Chapter 6 that appear to display erroneous behaviour of Forest BORG . Firstly, the trade-offs found for the RICE model, Figure 6.9, are odd, as it is not the damages objective but the welfare objective that should be in conflict with the other objectives. Secondly, there are small peaks and dips in the run-time analysis for both case studies, Figures 6.1 and 6.6. Typically, an evolving pareto front should not display any dips in hypervolume. The chapter concludes with a note on possible improvements for the RICE model and how Forest BORG should be further tested on other multi-objective problems featuring more objectives as well as a note on the OLT algorithm.

RICE trade-offs

The trade-offs of the RICE model visualized in Figure 6.9 are counter-intuitive. We ask ourselves the question if this is due to model implementation or a fault in Forest BORG. First we examine if this trade-off can be explained through model logic, then we judge model implementation.

Logically, the intuitive trade-off between the three objectives should be between welfare and damages or temperature overshoots. Damages are determined through sea level rise damages and a damage function which dictates which percentage of gross GDP is lost due to climate damages. The damage function is taken from the original DICE model (W. Nordhaus and Sztorc 2013) and is highly dependent on temperature, see Equations D.17 and D.18. Welfare is based on consumption per capita which relates to net economic output. Net economic output is split into consumption, the basis for the welfare metric, and investments, the basis for abatement and reducing climate damages in the future. The fraction of net economic output that is utilized as investment is determined by the savings rate, a lever of the model. Policies employing higher saving rates therefore intuitively should lead to lower consumption and therefore lower welfare, but also to higher investments and therefore lower damages and temperature overshoots. The metrics are aggregated over a 300 year simulation run so the counter argument would be that high saving rates would initially lead to low levels of consumption (*i.e.* welfare) but that the long term effects of such a policy minimizes climate change and therefore leads to low damage costs over the long term, leading to sustainable levels of relative high consumption and therefore welfare. If this were true, there would not be strong trade-offs between the aggregated objectives, yet these are

observed. The observed trade-off can not be logically explained.

The trade-offs of Figure 6.9 are counter-intuitive as it is not welfare, but damages that poses a trade-off with the other objectives. No logical explanation stemming from supposed model relations can be found. Instead, model implementation is at fault. The implementation of the RICE model is based on several predecessor versions which were all a continuation of its predecessor. The verification of this implementation of RICE is built on its most recent predecessor, as explained in Appendix D. An almost identical trade-off figure is found in the thesis of this predecessor version of RICE (Reddel 2022). The objectives utilized in this research are best compared to the ones in this previous work that are generated by the problem formulation described as 'utilitarian disaggregated'. This previous work showed that the pareto fronts shifted significantly when different ethical principles are employed, like egalitarian and prioritarian problem formulations, rather than the utilitarian one. As the version of RICE employed in this study is based on this predecessor version (Reddel 2022), the counter-intuitive trade-off is clearly a problem that stems from the model implementation, rather than a problem in Forest BORG. The trade-off remains counter-intuitive as other similar work has found that there is an observed trade-off between welfare and the other two objectives, damages and temperature overshoots (Marangoni et al. 2021).

Hypervolume dips

The dips in hypervolume, observed in both case studies in Figure 6.1 and Figure 6.6 appear wrong as an evolving pareto front should not decrease in hypervolume. The small peaks are however due to the settings of the epsilon values. Through testing it has been observed that from one snapshot of the pareto front to the next a few solutions have replaced many solutions because they epsilon-dominated the old solutions. If these solutions were however judged by strict dominance, many of them would not have been removed from the archive. By replacing many solutions with a few that are very close to one another in objective space, it is possible to see dips in hypervolume for a progressing pareto front. This hypothesis was tested by setting the epsilon values to arbitrarily small values, and it was observed that the dips had gone. There is however one large dip for one of the random trials in Figure 6.6 around 11000 NFE that appears to be too large and too wide to be caused by the epsilon values. The cause behind this dip is not yet discovered.

RICE2023

A new version of RICE was released during this research, after the development of this research' implementation of RICE. The new version of RICE has not been published yet but is available upon request to the author, professor Yang. This version is however currently available in the GAMS programming language and will have to be translated to python. The choice was made to not further develop the RICE model to perfectly align with its previous version on account of four reasons.

First, the small misalignment does not hinder the development of this research as the RICE model serves as a proof-of-concept for Forest BORG to be coupled to climate IAM's. The resulting policies should therefore not be taken as absolute truth but instead further research should swap the RICE model for a more complex, detailed IAM to conclude on any real-world policy. Second, the previous version was slightly out of alignment with its predecessor, straying away from the original RICE model in GAMS. Third, after the development of the RICE model but before the completion of this research a new version of RICE was made available, RICE2023, albeit in the GAMS programming language. Further effort into aligning IAM RICE with its predecessor which itself was also out of alignment could be better spent translating the new RICE2023 model to python. Fourth, the time constraint of this research imposed a choice between the further development of Forest BORG or the further development of the RICE model. Given the time available for this research the choice was made to not further develop the RICE model to perfectly align with the previous version.

Other IAMs are also interesting to explore as they are a better representation of reality. The RICE model has been used because it is a relatively straightforward and well-established model that was suitable to test new algorithms with. The proof-of-concept, for Forest BORG, is showcased in this research. The parallel axis plots and the visualizations of the policy trees are interesting because it shows how policy trees improve explainability for climate change policies. The absolute values of the policies are however questionable given the limitations of RICE. It would therefore be interesting to employ a better RICE model or another IAM.

Further Refinement of ForestBORG

ForestBORG was developed in an effort to accommodate many-objective problems. The policy problems of this research only employed three or four objectives which is a relatively small number. It would therefore be interesting to perform a similar policy analysis but with more objectives and possibly other justice principles, akin to the research performed in (Reddel 2022). Another comparison between the original POT algorithm and ForestBORG by employing a large number of objectives, eight for example, would also be interesting to observe the effect the change of tournament size has between both algorithms. Theoretically the adaptive tournament should do better than the binary tournament for an increasing number of objectives as the possibility of finding non-dominated solutions grows. Other design strategies have been tried too and are stated in Appendix C which is a record of the development process leading to the final design of Forest BORG.

Development of OLT Algorithm

A version of an OLT algorithm is described in (Jung, Ernst, and Maes 2013), (Jung, Wehenkel, et al. 2014) and (Maes, Wehenkel, and Ernst 2012). No concrete code however has been formulated that could test an OLT on RICE, the idea is at the time of writing purely theoretical. Much like the idea of applying POT to RICE, the seed that sprouted this research, the idea of applying POT-OLT to RICE could guide further research. Similarly to the development of a POT algorithm, the idea is there and (pseudo-) code is available in literature. The exact implementation and testing of the algorithm can however be challenging and time consuming, as experienced in the development of Forest BORG when the original POT algorithm was already available in literature.

8

Conclusion

This research is concluded by looking back to how the research (sub) questions are answered and by reflecting on the societal implications of this research.

8.1. Answering the Research Questions

The main research question has been answered by the creation of Forest BORG and demonstrating its performance by applying it to the case studies. The main research question was formulated as:

How can we design a tree-based multi-objective algorithm that ensures explainability in decision making for socio-environmental systems?

The following underlines how each separate sub research question has been answered.

Sub research question 1: How can algorithm performance be benchmarked?

The run-time dynamics of Forest BORG are benchmarked on the Folsom lake case study and the RICE case study. Algorithm effectiveness, efficiency and reliability were tested and Forest BORG has outperformed the competing original POT algorithm and converges quickly, after about 20.000 NFE for both case studies, and reliably across five random trials. The novel mutation operators did not entirely behave as expected. The expectation was that the search would be dominated by subtree mutations initially but then turn to point mutations towards the end of the search. The point mutation operators dominated throughout the search for the Folsom lake model and the results for the RICE model were not consistent across all random trials. The novel mutation operators did however outperform the control mutation operator. This random mutation operator randomly selects nodes in a policy tree to be mutated and is the single mutation operator in the original POT algorithm.

Sub research question 2: How do the hyperparameters influence algorithm performance?

From the controllability maps for the Folsom and RICE case studies, it can be concluded that modest maximum tree depths of 3 or 4 yield the best performing policy trees. Generally, lower population-to-archive ratios, 3 or 4, are preferred. This is probably due to the fact that it keeps the margins around the target value smaller when the population-to-archive ratio is checked to see if a restart is warranted. There does not seem to be a preference for the restart interval across the case studies. Instead, setting this hyperparameter will depend on the other hyperparameter settings. The influence of the hyperparameter settings on algorithm performance are larger for the Folsom lake case study than for the RICE case study. This is likely due to the limited number of discrete actions that are possible in the Folsom lake model while the RICE model is run with continuous action ranges.

Sub research question 3: What trade-offs are identified by the algorithm?

Forest BORG has been able to identify trade-offs for both case studies. The reliability objective in the Folsom lake model is conflicting most with the other objectives and the damages objective is identified as a trade-off across the objectives in the RICE model. This trade-off for the RICE model is built-in as the RICE model was taken from previous work that showed similar trade-offs.

Sub research question 4: How do the selected policy trees advance explainability?

The similarities and differences between the structures of the selected policy trees are analyzed. For both case studies, the selected policies that performed similarly also share structural similarities. These similarities are mainly found in the actions that are employed as well as tree depth. Forest BORG also identified that the carbon concentration in the atmosphere is a valuable system indicator for the RICE model as only this indicator was used in the selected policy trees. The net output indicator value proved less valuable as a system indicator as it did not feature in any of the selected policy trees.

8.2. Research Implications

The broader scope of this research in the context of policy analysis is to gain greater understanding in decision support methods. Decision-making under deep uncertainty on wicked problems like climate change require decision support tools that evaluate multiple perspectives. These multiple perspectives can be interpreted as different objectives from different stakeholders or as different ethical persuasions. The implication is that the support tool, Forest BORG, should be equipped to and invite to employ multi-objective problems rather than single objective problems.

The resulting policy set, a product of optimization through evolutionary algorithms should not only be optimal in the strict mathematical sense of the word but they should also be trustworthy. An optimal policy set constructed by black-box models lacks trust when it is explained to a decision-maker because it is not known why a certain action is preferred over another. Policy trees help the decision-maker explain why an optimal policy set should be implemented. Policy trees allow for decision-making based on system states. Policy trees are more explainable compared to static real-valued actions in a policy set as their output can be related to system states rather than being the mere result of an optimization.

To the authors knowledge, Forest BORG is the first algorithm that allows for continuous actions rather than discrete actions in the formulation of policy trees. These continuous actions are preferable over discrete actions as they assume no prior knowledge and thus do not invite biases into the formulation of policy trees. Policy trees allow for robust decision making under deep uncertainty when they are coupled to different scenarios. The resulting policy trees can be analysed for reoccurring or differentiating features between the optimal policy trees across scenarios. Furthermore, the ability of Forest BORG to identify non-dominated solutions informs on the extent of mitigation that is currently possible in the system. Scenarios in which the algorithm is unable to identify acceptable solutions can be regarded as the limits to mitigation. For those scenarios, changes to the system should be made, or allowing for more drastic actions in order to find acceptable solutions. This would thus identify the scenarios that go beyond mitigation and instead require adaptation.

Capturing the uncertainty in the case studies or extending the research to other actions in this Forest BORG framework is easy. Changing the actions only requires to change the string input stating the action names in the Forest BORG algorithm, as well as changing the new action name to a new variable setting in the connecting function between Forest BORG and the model of a case study. Defining different objectives is done similarly, by changing the string input of RICE and adjusting which variables are defined as metrics in the RICE model. The RICE model is built to design robust policies. Different scenario inputs can simply be fed to the RICE model, which when coupled to Forest BORG will produce policies across a range of scenarios, from which robust policy trees can be distilled.

The societal contribution of this research is the development of Forest BORG and applying it to real world case studies. Forest BORG is a new tool in a decision-makers toolkit that allows for multi-objective

decision making for socio-environmental systems. To the authors knowledge there are few other algorithms that are able to handle such optimization problems. Forest BORG has improved on a competing algorithm by expanding the number of objectives that can be used as well as allowing for continuous action ranges for multiple actions. The societal implication is that the range of real world applications that can be analyzed from a decision making under deep uncertainty perspective is expanded through these improvements. Furthermore, Forest BORG is not tested on theoretical problem formulations but on real world case studies. The results from these real world case studies show confidence that the suit of case studies can be extended to other problems while maintaining algorithm performance. A logical continuation of this research is applying Forest BORG to a more detailed IAM than RICE from which policy implications are gathered that will help decision-makers develop policies against climate change.

9

Future Work

The idea envisioned in this research is to develop a tree-based algorithm that can be integrated with a socio-environmental system to create explainable climate change policies. Two such algorithms are outlined in this research, POT and OLT, of which only POT is realized. This research has shown that POT, specifically Forest BORG, is capable to do just that. The OLT algorithm unfortunately remains an idea. However, by coupling the system knowledge that a POT can provide with the path feature functions of an OLT, a powerful framework of algorithms is created that is expected to create even more just and explainable climate policies within feasible computational times.

ForestBORG-OLT tandem framework

This research has focused on the development and application of Forest BORG to two case studies, rather than performing a policy analysis, including multiple scenarios to test policy trees for robustness, or gaining system knowledge.

Given a case study, a more complete approach to policy analysis would be to define a range of scenarios based on all uncertainties and divide the set of all possible scenarios in two, a training set and a test set. Policy trees are trained on the training set and then tested on scenarios they have not 'seen' before in the test set. This approach could point towards robust policy trees in a more formalized manner than the one currently employed in this research.

The gaining of system knowledge can also be improved by cycling through all possible indicator variables instead of only three in *e.g.* RICE. Appendix B shows a table with all possible indicator variables for RICE and proposed upper and lower bounds. This procedure could be further improved by allowing ForestBORG to learn which indicator variables work best to generate optimal pareto fronts. This research has already showed that the concentration of carbon in the atmosphere is a more informative indicator variable than the net output. Apparently more trend information within the system is captured by the concentration of carbon in the atmosphere. A system akin to the mutation operator selection could be created for using the best performing indicator variables when all of them are tested. The identification of these variables and their preferred ranges are of course of importance to defining a good basis for the path feature function in an OLT.

References

Allen, Myles et al. (2018). “IPCC SR15: Summary for policymakers”. In: *IPCC Special Report Global Warming of 1.5 ºC*. Intergovernmental Panel on Climate Change.

Asadieh, Behzad and Nir Y Krakauer (2017). “Global change in streamflow extremes under climate change over the 21st century”. In: *Hydrology and Earth System Sciences* 21.11, pp. 5863–5874.

Beek, Lisette van et al. (2020). “Anticipating futures through models: the rise of Integrated Assessment Modelling in the climate science-policy interface since 1970”. In: *Global Environmental Change* 65, p. 102191.

Burke, Marshall, Solomon M Hsiang, and Edward Miguel (2015). “Global non-linear effect of temperature on economic production”. In: *Nature* 527.7577, pp. 235–239.

Coello, Carlos A Coello (2007). *Evolutionary algorithms for solving multi-objective problems*. Springer.

Cohen, Jonathan S and Jonathan D Herman (2021). “Dynamic adaptation of water resources systems under uncertainty by learning policy structure and indicators”. In: *Water Resources Research* 57.11, e2021WR030433.

Council, Domestic Policy (2013). “Technical support document:-technical update of the social cost of carbon for regulatory impact analysis-under executive order 12866”. In: *Environmental Protection Agency*.

Deb, Kalyanmoy, Manikanth Mohan, and Shikhar Mishra (2005). “Evaluating the □-domination based multi-objective evolutionary algorithm for a quick computation of Pareto-optimal solutions”. In: *Evolutionary computation* 13.4, pp. 501–525.

Deb, Kalyanmoy, Amrit Pratap, et al. (2002). “A fast and elitist multiobjective genetic algorithm: NSGA-II”. In: *IEEE transactions on evolutionary computation* 6.2, pp. 182–197.

Fonseca, Carlos M, Peter J Fleming, et al. (1993). “Genetic algorithms for multiobjective optimization: formulationdiscussion and generalization.” In: *Icga*. Vol. 93. July. Citeseer, pp. 416–423.

Gazzotti, Paolo (2022). “RICE50+: DICE model at country and regional level”. In: *Socio-Environmental Systems Modelling* 4, pp. 18038–18038.

Goharian, Erfan et al. (2020). “Surface reservoir reoperation for managed aquifer recharge: Folsom reservoir system”. In: *Journal of Water Resources Planning and Management* 146.12, p. 04020095.

Grubb, Michael, Claudia Wieners, and Pu Yang (2021). “Modeling myths: On DICE and dynamic realism in integrated assessment models of climate change mitigation”. In: *Wiley Interdisciplinary Reviews: Climate Change* 12.3, e698.

Hadka, David and Patrick Reed (2013). “Borg: An auto-adaptive many-objective evolutionary computing framework”. In: *Evolutionary computation* 21.2, pp. 231–259.

Herman, Jonathan D and Matteo Giuliani (2018). “Policy tree optimization for threshold-based water resources management over multiple timescales”. In: *Environmental modelling & software* 99, pp. 39–51.

Horn, Jeffrey, Nicholas Nafpliotis, and David E Goldberg (1994). “A niched Pareto genetic algorithm for multiobjective optimization”. In: *Proceedings of the first IEEE conference on evolutionary computation. IEEE world congress on computational intelligence*. Ieee, pp. 82–87.

Hren, Jean-Francois and Rémi Munos (2008). “Optimistic planning of deterministic systems”. In: *Recent Advances in Reinforcement Learning: 8th European Workshop, EWRL 2008, Villeneuve d’Ascq, France, June 30-July 3, 2008, Revised and Selected Papers 8*. Springer, pp. 151–164.

Jung, Tobias, Damien Ernst, and Francis Maes (2013). “Optimized look-ahead trees: Extensions to large and continuous action spaces”. In: *2013 IEEE Symposium on Adaptive Dynamic Programming and Reinforcement Learning (ADPRL)*. IEEE, pp. 85–92.

Jung, Tobias, Louis Wehenkel, et al. (2014). “Optimized look-ahead tree policies: a bridge between look-ahead tree policies and direct policy search”. In: *International Journal of Adaptive Control and Signal Processing* 28.3-5, pp. 255–289.

Keppo, Ilkka et al. (2021). “Exploring the possibility space: taking stock of the diverse capabilities and gaps in integrated assessment models”. In: *Environmental Research Letters* 16.5, p. 053006.

Knowles, Joshua D and David W Corne (2000). “Approximating the nondominated front using the Pareto archived evolution strategy”. In: *Evolutionary computation* 8.2, pp. 149–172.
Lee, H. et al. (2023). “Synthesis Report of the IPCC Sixth Assessment Report (AR6): Summary for Policymakers”.
Lempert, Robert J (2003). “Shaping the next one hundred years: new methods for quantitative, long-term policy analysis”. In.
Lingeswaran, Shajeeshan (2019). “Redefining Integrated Assessment Models: An Exploratory Approach Towards Robust Climate-Economic Policies”. In.
Livingston, Jasmine E and Markku Rummukainen (2020). “Taking science by surprise: The knowledge politics of the IPCC Special Report on 1.5 degrees”. In: *Environmental Science & Policy* 112, pp. 10–16.
Maes, Francis, Louis Wehenkel, and Damien Ernst (2012). “Optimized look-ahead tree search policies”. In: *Recent Advances in Reinforcement Learning: 9th European Workshop, EWRL 2011, Athens, Greece, September 9-11, 2011, Revised Selected Papers 9*. Springer, pp. 189–200.
Marangoni, Giacomo et al. (2021). “Adaptive mitigation strategies hedge against extreme climate futures”. In: *Climatic Change* 166.3-4, p. 37.
Moore, Frances C and Delavane B Diaz (2015). “Temperature impacts on economic growth warrant stringent mitigation policy”. In: *Nature Climate Change* 5.2, pp. 127–131.
Nordhaus, William (2018). “Evolution of modeling of the economics of global warming: changes in the DICE model, 1992–2017”. In: *Climatic change* 148.4, pp. 623–640.
Nordhaus, William and Paul Sztorc (2013). “DICE 2013R: Introduction and user’s manual”. In: *Yale University and the National Bureau of Economic Research, USA*.
Nordhaus, William D and Zili Yang (1996). “A regional dynamic general-equilibrium model of alternative climate-change strategies”. In: *The American Economic Review*, pp. 741–765.
Reclamation, SR (2013). *Downscaled CMIP3 and CMIP5 climate and hydrology projections: Release of downscaled CMIP5 climate projections, comparison with preceding information, and summary of user needs*.
Reddel, Max (2022). “Climate Justice Behind the Veil of Aggregation: IAMs, Equity, and Pareto-Optimal Abatement Pathways”. In.
Rudin, Cynthia (2019). “Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead”. In: *Nature machine intelligence* 1.5, pp. 206–215.
Schaffer, James David (1985). *Some experiments in machine learning using vector evaluated genetic algorithms*. Tech. rep. Vanderbilt Univ., Nashville, TN (USA).
Srinivas, Nidamarthi and Kalyanmoy Deb (1994). “Muiltiobjective optimization using nondominated sorting in genetic algorithms”. In: *Evolutionary computation* 2.3, pp. 221–248.
Stocker, Thomas (2014). *Climate change 2013: the physical science basis: Working Group I contribution to the Fifth assessment report of the Intergovernmental Panel on Climate Change*. Cambridge university press.
Tjallingii, Ivar (2021). “Accounting for distributive justice in Integrated Assessment Models: Towards a more equitable climate policy agenda”. In.
Wakker, Peter P (2008). “Explaining the characteristics of the power (CRRA) utility family”. In: *Health economics* 17.12, pp. 1329–1344.
Wang, Zitong, Yan Pei, and Jianqiang Li (2023). “A Survey on Search Strategy of Evolutionary Multi-Objective Optimization Algorithms”. In: *Applied Sciences* 13.7, p. 4643.
While, Lyndon, Lucas Bradstreet, and Luigi Barone (2011). “A fast way of calculating exact hypervolumes”. In: *IEEE Transactions on Evolutionary Computation* 16.1, pp. 86–95.
Zhang, Qingfu and Hui Li (2007). “MOEA/D: A multiobjective evolutionary algorithm based on decomposition”. In: *IEEE Transactions on evolutionary computation* 11.6, pp. 712–731.
Zitzler, Eckart and Lothar Thiele (1998). “An evolutionary algorithm for multiobjective optimization: The strength pareto approach”. In: *TIK report* 43.

A

MOEA pseudo code

The pseudo-code of the Original POT and ForestBORG are reiterated here in a less formal mathematical manner.

Original POT

The Original POT algorithm is described in detail, in a practical manner as used in this research.

Algorithm 8 The initialization procedure

Create a population by generating random trees until the pre-specified population size is met
while nfe < max nfe **do**
  evaluate the objectives based on the current population
  update nfe by adding the population size
  iterate
end while

Algorithm 9 The iterate procedure

Select a parent population equal to a pre-specified size (μ) through a binary tournament of the current population
if a pareto front is not defined **then**
  Use the parents as the pareto front
else
  Add the pareto front to the population and update the pareto front by checking for non-dominated solutions in the expanded population
end if
Keep one parent unchanged but let the other mutate controlled by a mutation probability.
while population size is not equal to λ **do**
  Generate children either by cross-over of two randomly selected parents, or by the creation of a new random tree, controlled by a cross-over probability
end while

The original POT algorithm takes a number of function parameters which are stated in Table 5.7 as well as a function handler to a generative model. In order to connect RICE to the POT algorithm this function handler was built by constructing a function that takes a policy as input and the objective function value(s) as output. The policy supplied to the function is a policy tree which actions are represented in string format. The function therefore must state how each string statement in the policy maps to a change in the corresponding lever variable. E.g. if an action 'sr_04' is activated in the policy tree, the variable 'sr' in the RICE model must be set to a value of 0.4. Because the policy tree entails a static policy but dynamic actions, the threshold conditions that lead to a certain action must be checked at every time step over the simulation horizon. The output of the function however is an aggregated value of

the objective function values(s) over the simulation horizon, linking a policy tree to a single performance.

The optimization algorithm is called by the .run() function which specifies the maximum number of function evaluations, logging info and the type of evaluator which determines whether the optimization will be done with a single core or over multiple. Four important variables are defined: 'best_p', 'best_f', 'population' and 'objectives'. 'best_p' is the best solution set. Its size equals the value of parameter μ and is populated by the best performing policy trees of their generation. 'best_f' hols the corresponding scores, i.e. the objective function values, of the set of 'best_p'. 'population' is the number of policy trees in a generation, Within a generation, the 'parents' being the best performing policy trees of the previous generation and the 'children', being the mutations of the parents and new randomly generated policy trees together make up the population. Its size is equal to the value of the parameter $population\_size$. 'objectives' is the parameter that holds the objective function values of each corresponding policy tree in a population. The initial population is a collection of randomly constructed policy trees, which objectives are calculated by the executor which can be done over a single or over multiple cores. Thus, for each randomly constructed policy tree, a simulation is run in which the RICE model adopts the policy and the RICE function handler outputs the objective function values which are stored in the 'objectives' variable in the POT algorithm. After initialization of these variables, the function iterate() is called which handles the optimization from generation to generation. When every policy tree in a population of size 100 is evaluated, the number of function evaluations is incremented by 100 and new generations will be constructed until the maximum number of function evaluations is reached. The number of generations in an optimization run is thus equal to $\frac{max\_nfe}{population\_size}$.

The iterate function distinguished between single objective optimization and multi-objective optimization of which only the latter is described here. Out of a generation, the parents are chosen by a binary tournament. The binary tournament function is supplied the policy trees in a population and their corresponding performance: the objectives. Two random integers are chosen lower than the population size. The objectives of these two random samples from the population are compared to each other, if the first dominates the second, it will be returned as a parent solution and *vice versa*. If none of the solutions are dominant over one another, one of the solutions is randomly chosen as a parent. The number of binary tournaments that are held within a population is equal to μ. Note that when a solution dominates another it means that all the objectives of that solution are smaller than or equal to all the objectives of the other solution and at least one objective of the solution is smaller than the other solution, so this optimization assumes minimization.

Once the parents within a population are determined 'best_p' and 'best_f' are determined by an archive_sort function. The archive_sort function takes best_p, best_f, population and objectives as input. It appends population to best_p, and objectives to best_f, therefore momentarily extending the size of this artificial population to $\mu + population\_size$. A list as long as $\mu + population\_size$ is initialized with every value set to the boolean True. Then, a double for loop iterates over the list comparing the objective of a policy tree in the list with the one after it until the end of the artificially large population. The value in the list will change from the boolean True to the boolean False if the solution is dominated by the other one, or both solutions fall in the same box, defined by the epsilon value. The epsilon variable is by default set to 'None'. After the iteration through the artificially large population, the list with boolean values contains True, on the indices with non-dominated solutions and False on the indices of solutions that are dominated. The new best_p and best_f are the solutions in the population that correspond with the True indices.

Within the parent population, the first parent is kept unchanged but the others are mutated. The child population, in size equal to the population_size minus the number of parents, is constructed by either cross-overs from parents in the parents population, or by generating entirely new random trees. The chance of a child being a newly generated random tree is equal to $1 - P_c$, the cross-over probability. In case of a cross-over, bloating is controlled for and every time a tree is mutated, it is pruned.

ForestBORG

Algorithm 10 The initialization procedure

Generate an initial population
if a member of that initial population is non dominated **then**
  add that member to the Archive
end if
while the number of function evaluations is smaller than the maximum number of function evaluations **do**
  iterate through the borg main loop, see Algorithm 11
  Increment the number of function evaluations by one
end while

Algorithm 11 The iterate procedure

Check the population to Archive ratio
if the population to Archive ratio deviates by more than 25% **then**
  revive search, see Algorithm 12
else if epsilon progress has not changed during the previous 3 iterations **then**
  revive search, see Algorithm 12
end if
Choose one parent from the Archive
Determine tournament size
Choose another parent from the population through tournament selection
Create one offspring from the two parents through crossover
Allow the offspring to mutate through different operators
Add the offspring to the population
Update the Archive with the offspring

Algorithm 12 The revive procedure

Empty the population
fill the population with the solutions from the Archive
compute the new population size with the population to archive ratio
while the population is smaller than the newly calculated population size **do**
  inject mutated Archive members into the population
  update the Archive with the newly generated solution if applicable
end while

Algorithm 13 The update Archive procedure

check the candidate solution against every member in the Archive
if the candidate solution epsilon dominates a solution in the Archive **then**
  remove the solution in the Archive
else if a solution in the Archive dominates the candidate solution **then**
  dismiss the candidate solution and end the procedure
end if
add the candidate solution to the Archive if it dominates or is epsilon non-dominated by the solutions in the Archive

Algorithm 14 The add to population procedure

if the offspring dominates one or more population members **then**
  the offspring replaces one of these dominated members randomly
else if the offspring is dominated by at least one population member **then**
  the offspring is not added to the population
else
  the offspring is non-dominated and replaces a randomly selected member of the population
end if

B

Indicator Variables

Table B.1 shows the possible indicator variables of the RICE model and their suggested upper and lower bounds. The cuurent research employs only three, year, atmospheric carbon (mat) and global net output. To gain more system knowledge an analysis with ForestBORG could be done in which the algorithm learns to select the best performing indicator variables by cycling through them.

Indicator variable	Upper bound	Lower bound	Indicator variable	Upper bound	Lower bound
mu	1.5000	0.0002	TOTAL_SLR	2.1062	0.0609
S	0.3720	0.0893	mat	1,877.8338	393.5000
E	2.4921	0.0000	mup	3,568.6279	800.0000
damages	2.7361	0.0060	ml	15,150.2729	5,005.0000
abatement_cost	0.1280	0.0000	forcoth	0.4500	- 0.0300
abatement_fraction	0.0009	0.0000	E_worldwide_per_year	15.5604	0.0010
SLRDAMAGES	0.2983	0.0000	labour_force	716.6334	148.4213
gross_output	345.5828	6.1990	total_factor_productivity	77.6162	5.6738
net_output	342.8445	6.1929	capital_stock	1,220.7042	11.4255
I	85.0254	1.1055	sigma_ratio	0.2011	0.0068
CPC	539.6464	17.1383	Eind	2.4921	0.0000
forc	6.5443	0.8023	sigma_gr	- 0.0048	- 0.0112
temp_atm	4.4104	0.4150	damage_frac	0.0183	0.0005
temp_ocean	2.9063	0.0034	SLRTHERM	1.0264	0.0499
global_damages	46.4202	0.0436	GSICCUM	0.3537	0.0073
global_output	3,568.0760	27.6458	GISCUM	1.3844	0.0030
global_period_util_ww	4,631.5563	-198.6512	AISCUM	0.0019	-0.2194

Table B.1: Bounds for indicator variables. These bounds are constructed by taking 50% of the maximum and minimum value of a standardized simulation of the RICE model, under the Nordhaus policy and the Basic RICE scenario. These numbers correspond to the US region but the approach is equal for all regions.

C

MOEA development

This chapter documents the development of a MOEA for tree-structured decision variables. ForestBORG as presented in the main text is a functioning algorithm that outperforms the Original POT algorithm. The development of the algorithm was by no means as straight forward as the main text conveys. The journey to a functioning algorithm reads as a comedy of errors, many errors are foolish in hindsight, as most errors are. The author had adequate but limited coding skills at the start of this research but had to learn new programming concepts when the development of a new MOEA became necessary halfway through the research. Much more data and many more figures were generated than are shown here. The following figures aim to capture key points in the develpoment of the MOEA.

Original POT

The Original POT was relatively quickly coupled to the RICE model due to an excellent implementation by (Herman and Giuliani 2018). The switch from single to multi-objective optimization was also swift as it only required turning on one of the input parameters of the POT algorithm and a quick adjustment to the docking function that connects the algorithm to the model. This is the *.f()* function for the Folsom model in the original publication of the POT algorithm. There was room for improvement however as the actions a policy tree could have were either multiple but discrete or singular but continuous. The selection of parents during the evolutionary optimization is guided by a binary tournament which performance is expected to degrade as more objectives are defined.

Instead of adapting these features in the Original POT algorithm, the author decided to create a new MOEA incorporating the before-mentioned features. Creating a new MOEA was partly done out of the belief that you truly understand something if you built it from scratch and partly because some concepts present in the Original POT MOEA could be done differently and the author wanted to try those concepts out. The resulting algorithm was dubbed 'homemade MOEA' but its development halted when the idea of creating ForestBORG seemed more promising. The homemade MOEA was built on the existing code from the Original POT algorithm, especially the section of the code that creates the tree objects.

Homemade MOEA

The homemade MOEA was born out of the idea that multiple clusters of solutions could simultaneously grow towards the Pareto front which would increase diversity and convergence speed. A single cluster would have a small population, 10 to 20 members (Npop), of which 2 to 4 would be parents (Nparents). Populating a population was handled by differentiating between different cases. In case there are no parents, e.g. at the initialisation of a population, Nparents are randomly created. If there are parents but less than Nparents, create the remaining parents randomly. If there are parents but more than Nparents, Nparents are randomly chosen and the rest are taken as children. The parents are non-dominated solutions, the current Pareto front. These parents would then recombine through cross-over and mutation to produce the children until the population had grown to Npop. This population is then ready to evolve to the next generation. Every member in the population is checked for dominance with an externally kept population: the VIPs. This is a population with all the best solutions

from every cluster and is akin to the overarching Pareto front. The author realized later when working on ForestBORG that this construct is basically the 'archive' in BORG. The non dominated solutions in a generation are the parents in the next generation. At the end of a generation, all solutions from one generation, a 'family', are added to the 'graveyard'. The graveyard is a register that contains all solutions generated in the search process. It was initially useful to inspect the behaviour of the MOEA but the author realizes that its existence is unnecessary in large searches. New generations are created until a pre-specified maximum number of function evaluations is reached.

Many clusters would be deployed which would move by cooperation through the VIPs to the Pareto front. Additional features were later developed for managing the cooperation between these clusters. If for example two clusters would move to closely to each other early on in the search, one cluster would be eliminated and a random one would be created in its place. The 'position' of a cluster was calculated by taking the average position of its parents in objective space. When the diversity of the resulting Pareto fronts was rather disappointing, new search strategies were developed in which a cluster was employed to find the Pareto front for a bi-objective problem. The original many-objective problem is thus split in multiple problems which were to be solved by a cluster each. A three-objective problem would thus be split in three bi-objective problems in which two of the three objectives are taken. The solutions in each of the three Pareto fronts would be taken as the initial population for a fourth simulation that incorporates all three objectives again. This was an attempt to increase diversity. The author was aware of proper diversity management features in the scientific literature but wanted to try this splitting of objectives out as it lend itself to the structure of clusters.

At this point it is worth mentioning that many tests were performed on this homemade MOEA, many different search strategies as well as various cluster features were tested. Two serious mistakes were made that could have prevented much of these tests that eventually were done in vain.
Firstly, no snapshots during the simulation were created, the only information that was gathered during the simulation was only present in convergence metrics that were shown at the end of the simulation in graphs without the underlying data being saved somewhere. The only output the algorithm provided was graphs on convergence and the ultimate Pareto front. Consciously not programming snapshots into the algorithm appears unthinkable in hindsight and points to the authors inexperience. Later, after good advice from my supervisors I added snapshots to the algorithm.
Secondly, the convergence metric was ill-constructed. Instead of using ϵ-progress or hypervolume the author decided on generational distance as the only convergence metric by which all of the beforementioned tests were judged. Because only two or 3 objective problems were considered, results were occasionally compared by 2d or 3d plots of the Pareto front. A version of the generational distance metric was constructed by the author on an erroneous notion of generational distance. Instead of tracking the distance between each solution in the Pareto front to its closest neighbour in a subsequent Pareto front, a metric was constructed in which the center of a Pareto front was defined and used as the measuring point between Pareto fronts. The center point of a Pareto front was defined as the mean of each objective for all the points on the Pareto front. In hindsight this is a terrible metric for tracking diversity, one of the problems that the MOEA was supposed to solve. Figure C.1 shows this metric on a predecessor version of ForestBORG on RICE.

All clusters did perform badly on diversity proven by the 3d plots of the Pareto fronts. Figure C.2 shows three 3d plots of Pareto fronts for the RICE model produced by the Original POT algorithm, random search and a version of the Homemade MOEA. All three simulations were allowed to run for 100.000 function evaluations. Notice how the Pareto front of the random search and the Original POT are much more diverse than the Homemade MOEA. Interestingly, the Original POT barely outperforms random search. The straight lines in the plots of Original POT and random search are due to the single action change available in the Original POT algorithm. Because there are three levers to be controlled and only one can be changed at a time in a policy tree created by Original POT, the other two were set by the Nordhaus policy (miu=2135, sr=0.248 and irstp=0.015). The resulting Pareto front therefor aligns according to the axis set out by these default values. These default values were later replaced by randomized values.

The development of the Homemade MOEA was beneficial to this research even though it eventually

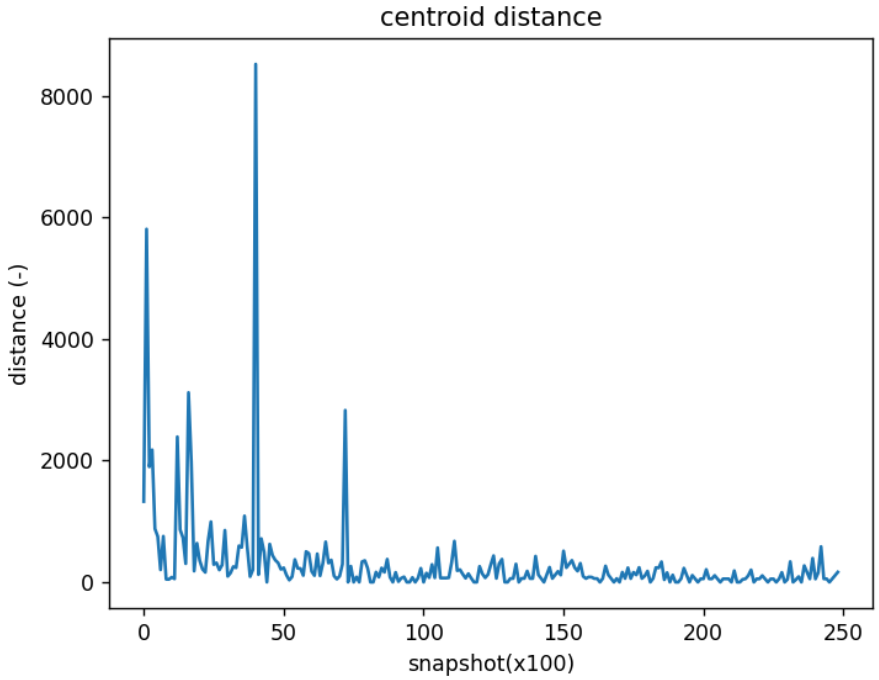

Figure C.1: The centroid distance metric on a predecessor version of ForestBORG run for 25000 function evaluations

is not used. The mistakes made were not repeated on the ForestBORG algorithm and some techniques employed to solve problems for the Homemade MOEA were transferable to ForestBORG. The main programming technique that was carried over are the random number generators. The numpy library switched to a new random number system. *np.random.seed()* is no longer the preferred way of setting a seed. Instead initialization of random number generators are encouraged. The author was not aware that random numbers produced by a random number generator are not only determined by the seed by which they are set, but also the order in which random number are produced. Concretely, this meant that whether or not to mutate a tree, controlled by chance, is determined by the previous random decision. When multiple different functions in the same class rely on the same random number generator the eventual output can differ between simulations set with the same seed. The culprit is functions that determine whether a random action is performed in another function based on a random choice. This took a long time to figure out, finding the results to be consistently different when setting the same seed was puzzling. Taking snapshots would have been helpful to speed up to the solution of this problem. Once the essence of the problem was understood, a solution was formulated by 'disentangling' the random numbers. The MOEA class itself is initialized with one master random number generator. Every function in the class that employs randomness is then endowed with its own random number generator, seeded by the master random number generator. This solution allows for reproducible results in complex randomized systems and is used in ForestBORG.

ForestBORG

As the results of the Homemade MOEA did not improve, in hindsight probably due to bad convergence metrics, the decision was made to move to a pre-existing algorithm that could handle many objective optimization. The BORG algorithm was a good candidate but it did not allow tree-structured decision variables. A search through MOEA literature did not yield any MOEAs that could, except for one or two papers whose code was unavailable. The best direction forward at the time was to recreate BORG that would work with policy trees. Professor Reed kindly shared the BORG code. As BORG is built on top of the platypus library, it was more straightforward to create ForestBORG from scratch while staying as close as possible to the original BORG.

The first results of infant ForestBORG algorithm are shown in Figure C.3. These results are created by applying the algorithms to the Folsom lake model, rather than the RICE model. The Folsom lake model is the model used in the Original POT paper. Another faulty generational distance metric, based

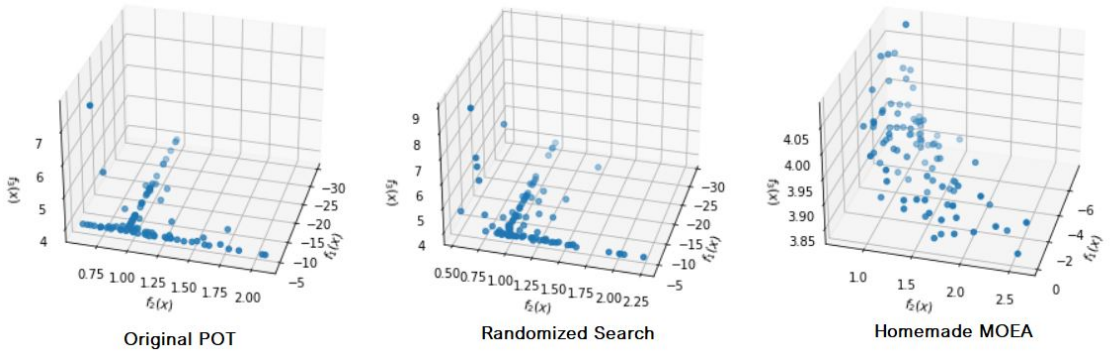

Figure C.2: Pareto front for the RICE model created by three different simulations. The first with the Original POT MOEA, the second by random search and selecting the non-dominated solutions and the third by a versinon of the Homemade MOEA. Note that the axis are not equal across the plots and that the axis themselves refer to the three objectives of the RICE model as used in this research.

on a points closest neighbour in a subsequent pareto front but wrongly implemented, is shown. It also shows the hypervolume for the Original POT algorithm and ForestBORG. These results were initially disappointing as the hypervolume for the Original POT was much higher even though both algorithms were tuned to the same epsilons. Another odd characteristic of Figure C.3 is the erratic behaviour and dips in hypervolume in ForestBORG. This behaviour was not present in the Original POT hypervolume even though both plots were made with the same definition of the hypervolume metric. The hypervolume metric was therefor not reexamined. Instead, suspicion turned to the function that allows a solution into the archive. The function, much like the rest of ForestBORG was built based on the description provided in the BORG paper (Hadka and Reed 2013), not actual code.

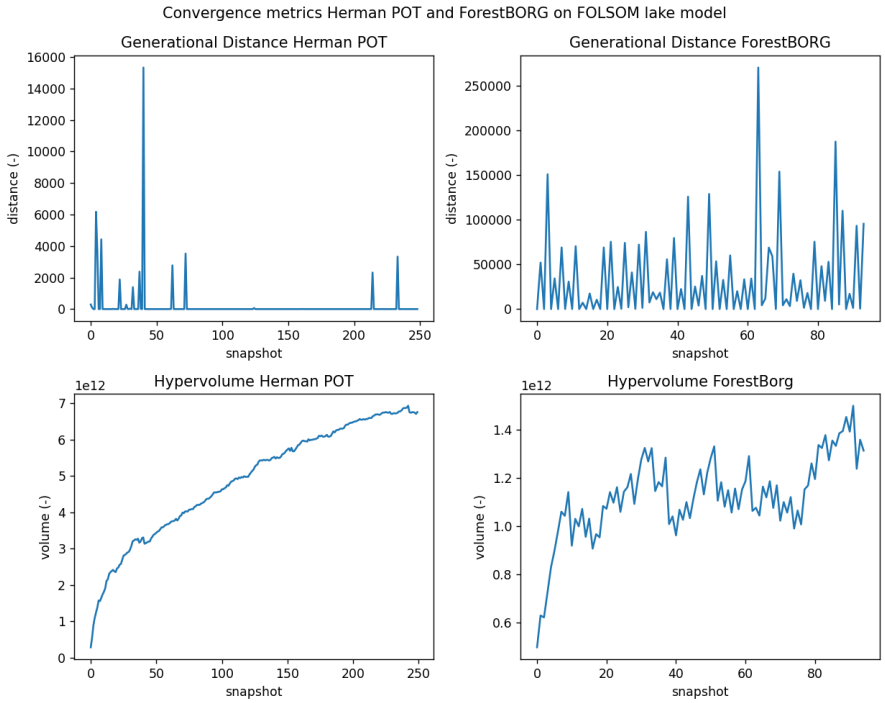

Figure C.3: First results for an infant version of ForestBORG and comparison to Original POT. Note that the metrics in these plots were faulty but this was only known in hindsight.

After much tweaking of the algorithm, testing different archive versions, disabling the restart procedure, disabling the gamma requirement in the restart procedure, disabling the epsilon progress requirement in the restart procedure and different combinations of these versions, the following Figure C.4 is created.

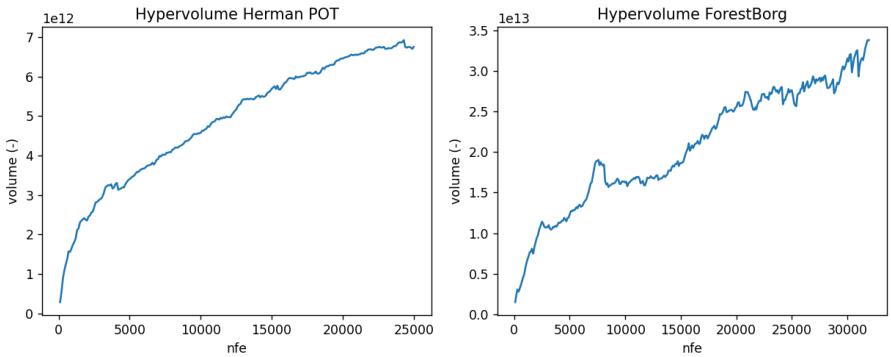

Figure C.4: Results of ForestBORG after much tweaking of the archive functions and the restart procedure.

ForestBORG seemed to be improving but there were still dips and somewhat erratic behaviour. Based on hypervolume, ForestBORG did now however appear to be outperforming the Original POT. Having established that the rules governing the archive must have been correct, the actual implementation was the sole remaining issue that could cause the erratic behaviour that seemed to mostly occur during or after restarts. The Handmade MOEA was built with list constructs in python as the author was more comfortable with lists. The first version of ForestBORG however was built with numpy arrays as they are generally recognized to be faster than python lists. The author discovered that the built-in functions handling the numpy arrays were not always programatically doing what they conceptually were supposed to be doing. This could allow for faulty population and archive updates. After correcting these oversights, the results started to look more promising, as shown in Figure C.5 which shows the epsilon progress and hypervolume development of ForestBORG.

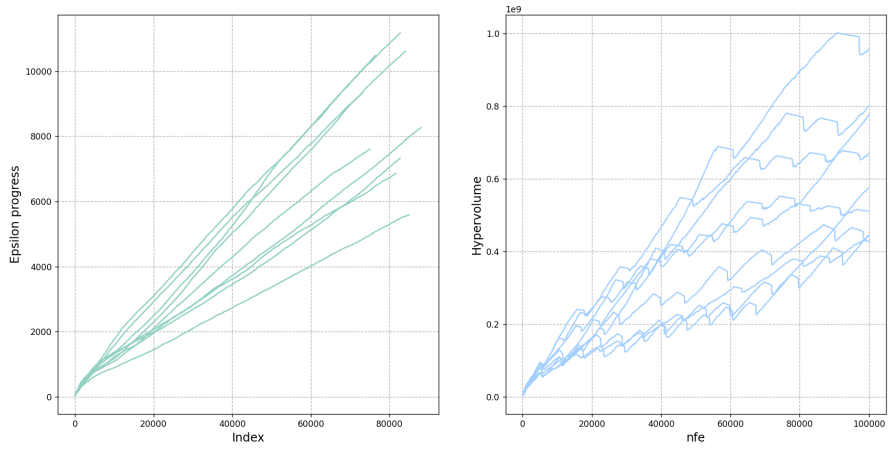

Figure C.5: Results of ForestBORG after correcting the numpy array operators governing the population and archive updates. Note that the hypervolume metric is still faulty but this was only known in hindsight.

The erratic behaviour had decreased but there were still drops when a restart was triggered. Epsilon progress showed that the algorithm did not converge after 100.000 function evaluations and running the algorithm with 9 seeds showed that the results were diverging. Not satisfied with these results, the entire archive update procedure was replaced with the implementation used in the platypus library. After minor adaptations the code was embedded in ForestBORG. The hypervolume metric was also replaced by one developed by the Walking Fish Group whose hypervolume metric can be imported in python through a library. The results were much better so a depth analysis was performed in which ForestBORG was coupled to RICE differentiating between maximum tree depths, as shown in Figure C.6.

The dips in hypervolume after a restart is triggered remain. Due to indispensable advice from the supervisor this problem was fixed by creating a deepcopy of the archive when passing it to the restart procedure. In the restart procedure the population is emptied and equated to the current archive. The population is then updated until it reaches a specific size. Besides the authors knowledge equating

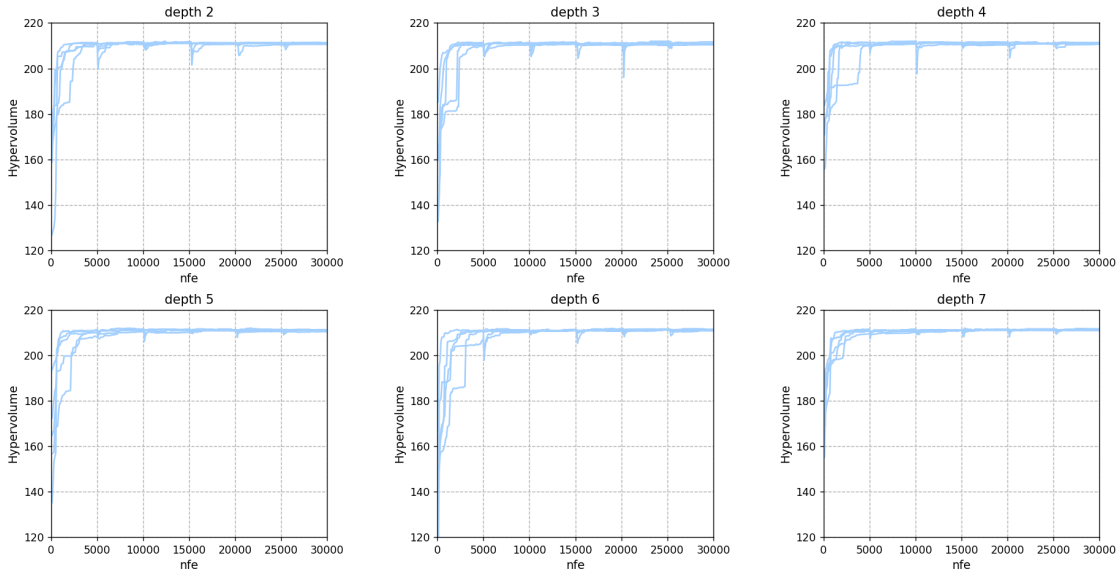

Figure C.6: Results of ForestBORG after swapping in the platypus update archive procedure and employing the hypervolume metric from the Walking Fish Group. A depth analysis in which the ForestBORG algorithm is run on the same RICE scenario but with differing maximum tree depths. Each maximum tree depth is run for five different seeds.

the population to the archive, also updated the archive when the population was updated. There are separate procedures for admittance into the population and archive but these became ineffective due to the passing of the archive to the restart procedure. In python, simply creating a deepcopy of the archive object before submitting it to the archive update procedure alleviated the problem. The results are shown in Figure C.7 and the accompanying epsilon progresses in Figure C.8.

The dips after a restart have disappeared and the algorithm appears to be converging for both the hypervolume as well as the epsilon progress metric across different depths for all seeds. These results bestowed enough confidence upon the author to compare ForestBORG to the Original POT algorithm again. The algorithms were compared on both the Folsom lake model and the RICE model. The results are shown in Figure C.9.

ForestBORG appears to be outperforming the Original POT on both the Folsom model as well as the RICE model. However, upon closer inspection of the actual tree structures in the Pareto front of ForestBORG, not all trees adhered to their set maximum depth. One of the functions that allow mutations of a tree did not correctly function. The problem was swiftly dealt with by pruning the mutated tree. The pruning function is inherited from the Original POT algorithm. The results of this final iteration of ForestBORG are reported in the main text, in chapter 6

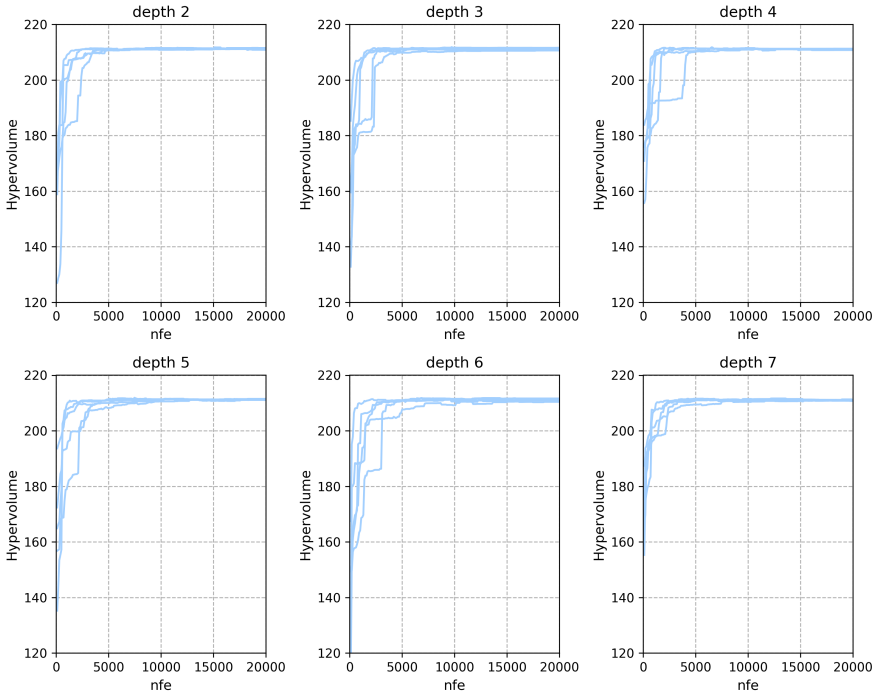

Figure C.7: Results of ForestBORG after correcting the deepcopy oversight shown as hypervolumes. A depth analysis in which the ForestBORG algorithm is run on the same RICE scenario but with differing maximum tree depths. Each maximum tree depth is run for five different seeds.

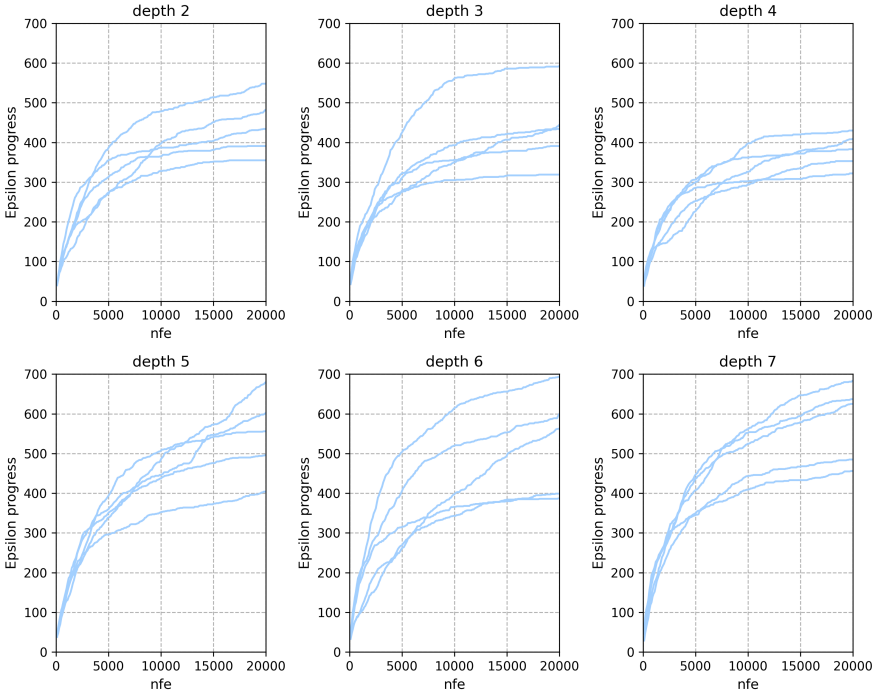

Figure C.8: Results of ForestBORG after correcting the deepcopy oversight shown as epsilon progresses. A depth analysis in which the ForestBORG algorithm is run on the same RICE scenario but with differing maximum tree depths. Each maximum tree depth is run for five different seeds.

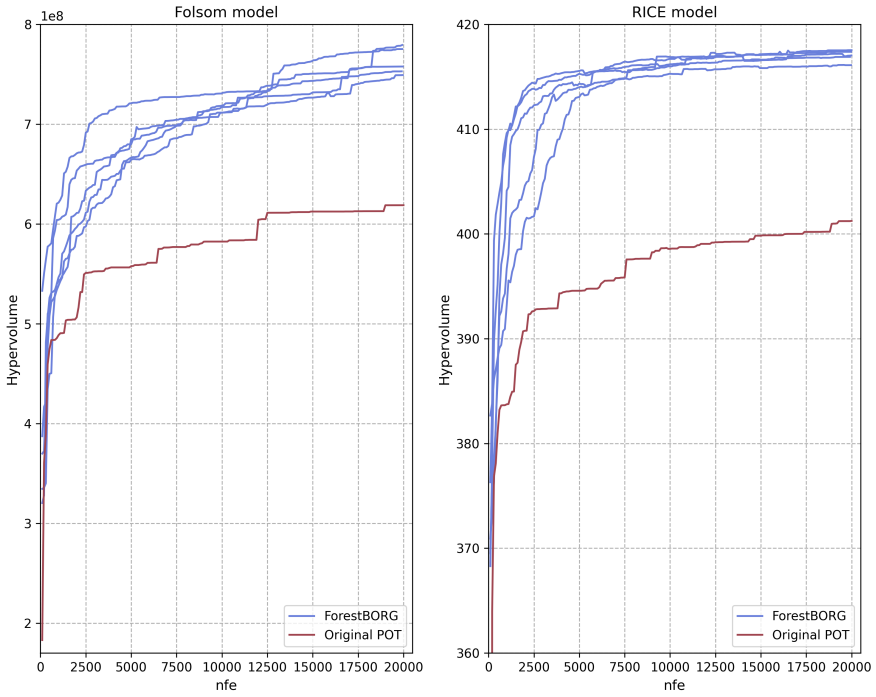

Figure C.9: The hypervolume for both the Original POT algorithm as well as ForestBORG applied to both the Folsom lake model and the RICE model. These results were generated before the pruning error was fixed in ForestBORG.

D

RICE model

The main relations of the RICE model are depicted in Figure D.1 which could be helpful when reading through these sections.

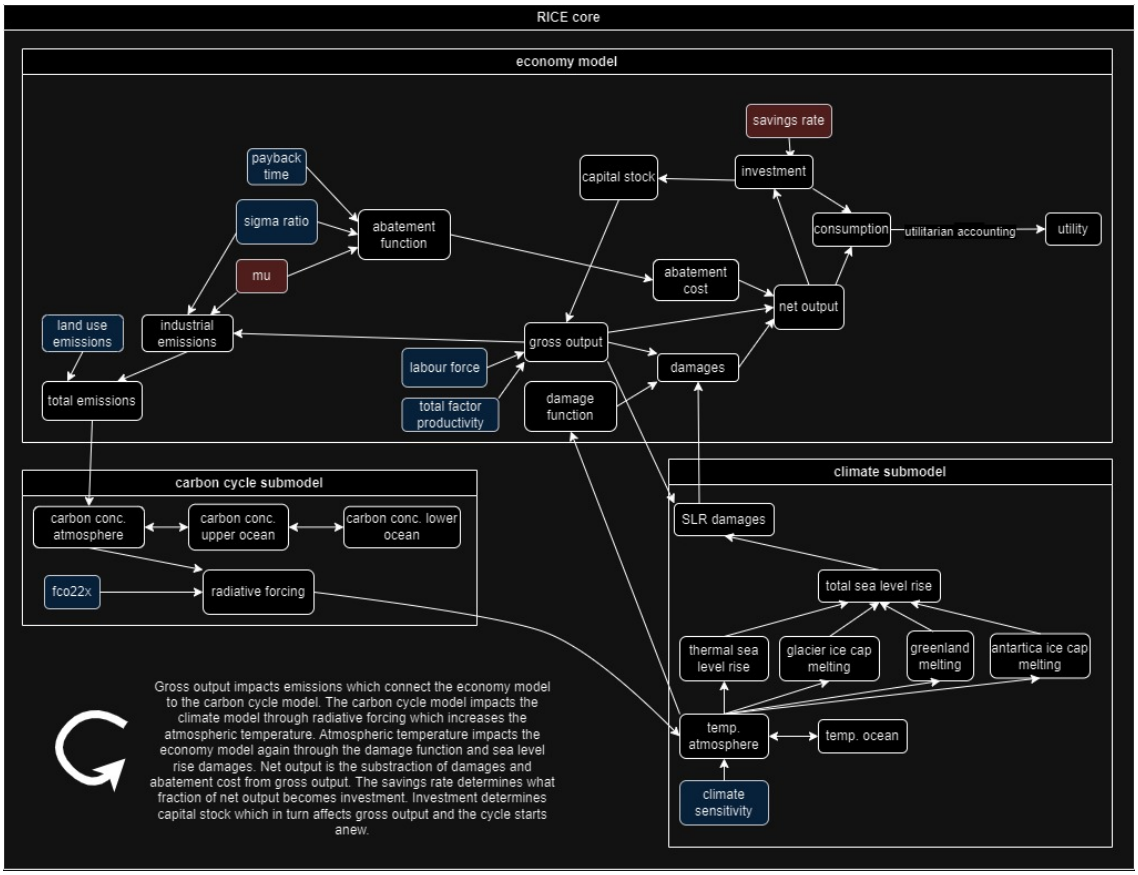

Figure D.1: Core structure of the RICE model. Various metrics stemming from different justice principles are not visualized. The blue containers depict variables under uncertainty and the red ones depict lever variables.

D.0.1. Model Equations

The following details the relationships within and the interactions between the submodels in RICE. Although most of these relationships are also described in the DICE2013R manual (W. Nordhaus and Sztorc 2013), they are reiterated here for the sake of completion and brevity.

Gross Economy Model

The version of RICE employed in this study divides the world into 12 regions, see Appendix **??** for more detailed information about the regions. Each region is endowed with an initial total factor productivity, which is a measure of a regions technological development, a population, here assumed equal to the labor force, and an initial capital stock. The first two variables are exogenous to the model while the latter is endogenous. Total factor productivity and labor force are exogenously determined through their respective growth and are calculated in the model through:

$$A_{t,r} = A_{t-1,r} * 2.71828^{A_{growth,t,r}*\Delta t} \tag{D.1}$$

$$L_{t,r} = L_{t-1,r} * 2.71828^{L_{growth,t,r}*\Delta t} \tag{D.2}$$

in which $A_{growth,t,r}$ and $L_{growth,t,r}$ are the growth rates of total factor productivity and labor force respectively. The total factor productivity time series is produced through Hicks-neutral technological change and the labor force through a similar logistic-type equation (W. Nordhaus and Sztorc 2013). The capital stock dynamics is characterized by an exponential depreciation rate and is determined by:

$$K_{t+1,r} = I_{t,r} + (1-\delta) * K_{t,r} \tag{D.3}$$

in which $K_{t,r}$ is the capital stock of region r at time t, I is the investment at that region at that time, and δ is the depreciation rate of capital, here equal to 0.1. Investment however depends on net gross output, so the initial value of a regions capital stock cannot be calculated and is taken from the original RICE model as initial input values. A region's gross output is calculated through:

$$Y_{gross,t,r} = A_{t,r} * L_{t,r}^{1-\gamma} * K_{t,r}^{\gamma} \tag{D.4}$$

in which A is total factor productivity, L the labor force, K the regions capital stock and γ the output elasticity of capital, here equal to 0.3. A region's gross output has an effect on global GHG emissions. Global GHG emissions are the sum of industrial emissions, which are related to a region's gross output and emissions from land use. These are determined through:

$$E_{land,t,r} = (1 - r_{E_{land}}) * E_{land,t-1,r} \tag{D.5}$$

in which $r_{E_{land}}$ is the decline rate of emissions from land use. It is determined exogenously and assumed to decline due to efficiency improvements. Emissions from land use changes are estimated based on the fifth assessment report of the IPCC and the emissions are estimated to be around 3 $GtCO_2$ per year (Stocker 2014). While emissions from land use are not controlled in the model, emissions through industry are, by means of the emissions control rate. This policy parameter reflects the year in which emissions reach net zero globally.

$$E_{ind,t,r} = \sigma_{t,r} * (1-\mu) * Y_{gross,t,r} \tag{D.6}$$

in which μ is the emissions control rate, and σ is the output to emissions ratio, relating a level of gross output to a level of emissions, and is determined through:

$$\sigma_{t,r} = \sigma_{t-1,r} * 2.71828^{\sigma_{growth}*\Delta t} \tag{D.7}$$

in which σ_{growth} is the rate of change of the output to emissions ratio which is assumed to decrease due to technological efficiencies. Total emissions are then defined as:

$$E_{t,r} = E_{ind,t,r} + E_{land,t,r} \tag{D.8}$$

Total emissions per region is the entity that relates the gross economic model to the carbon cycle model.

Carbon Cycle Model

Total emissions from each region are added into one worldwide carbon emissions variable and is taken up into the atmosphere. There are three carbon reservoirs in the model: these are the atmosphere, the deep oceans, and the upper oceans and biosphere, which is situated in between the other two reservoirs. Carbon enters the atmosphere and flows through the upper oceans and biosphere reservoir. From the biosphere it flows to the deep oceans which acts as a carbon sink. The flows of carbon are

bi-directional and occur at different speeds. The equations that model the quantity of carbon between the reservoirs are:

$$M_{AT,t} = E_t + \phi_{11} M_{AT,t-1} + \phi_{21} M_{UP,t-1} \tag{D.9}$$

$$M_{UP,t} = \phi_{12} M_{AT,t-1} + \phi_{22} M_{UP,t-1} + \phi_{32} M_{LO,t-1} \tag{D.10}$$

$$M_{LO,t} = \phi_{23} M_{UP,t-1} + \phi_{33} M_{LO,t-1} \tag{D.11}$$

in which M_{AT} is the carbon in the atmospheric reservoir, M_{UP} the upper oceans and biosphere reservoir, M_{LO} the deep oceans reservoir and ϕ the rate of flow variables between the reservoirs. The flow variables are exogenously determined and compared to the Fifth Assessment Report of the IPCC. The flow variables are determined such that the deep oceans act as a carbon sink and that 35% of a pulse of 100GtC into the atmosphere will remain there for 100 years. The accumulation of carbon in the three reservoirs impacts the climate system through the radiative forcing equation which calculates the impact on the radiation balance of the globe. The relationship between accumulation of GHG and radiative forcing ultimately relates the GHG in the reservoirs from the emissions due to gross output to the rise in global temperatures and is given by:

$$F_t = \eta * (log_2[\frac{M_{AT,t}}{M_{AT,1750}}]) + F_{EX,t} \tag{D.12}$$

in which F_t is the human induced change in total radiative forcings of GHGs since 1750 at timestep t, η is a constant that describes the forcing due to CO_2 and is set here to 3.8 Wm^{-2}. $F_{EX,t}$ is forcing different from CO_2, like aerosols, ozone and albedo changes. Exogenous forcing $F_{EX,t}$ is relatively small compared to forcing due to CO_2 (F_t). Radiative forcing relates the carbon cycle model to the climate model.

Climate Model

An increase in radiative forcing warms the atmospheric layer. Warming of the atmosphere warms the upper ocean layer which ultimately warms the deep ocean layer. The relations between radiative forcing, atmospheric mean surface temperature and deep ocean temperature are given by:

$$T_{atm,t} = T_{atm,t-1} + \zeta_1(F_t - \zeta_2 T_{atm,t-1} - \zeta_3(T_{atm,t-1} - T_{ocean,t-1})) \tag{D.13}$$

$$T_{ocean,t} = T_{ocean,t-1} + \zeta_4(T_{atm,t-1} - T_{ocean,t-1}) \tag{D.14}$$

in which F_t is radiative forcing and ζ are diffusion parameters between the different layers. Important to note is that with the current weights the climate sensitivity of a doubling of equilibrium CO_2 results in 2.9 ^0C mean surface temperature warming.

An increase in atmospheric temperature impacts global sea level rise, which through damages impacts the world economy. Atmospheric temperature impacts sea level rise through four channels: thermal expansion of sea water, melting of glacier ice caps, melting of ice in Greenland, and melting of the Antarctica ice cap. Damages through sea level rise are given through:

$$SLR_{dam,t,r} = 100 * SLR_{mult} * (SLR_{total,t-1} * SLR_{dam1} + SLR_{total}^2 * SLR_{dam2}) * (\frac{Y_{gross,t-1,r}}{Y_{gross,t_0,r}})^{\frac{1}{SLR_{elas}}} \tag{D.15}$$

in which SLR_{mult} is set at 2, SLR_{elas} is set at 4, SLR_{dam} is a region specific damage impact, SLR_{total} is the total sea level rise as a sum of the four channels, and $\frac{Y_{gross,t-1,r}}{Y_{gross,t_0,r}}$ is the fraction of current region gross output over initial region gross output.

The atmospheric temperature and sea level rise damages relate the climate model to the net economy model.

Net Economy Model

The net economy model is a continuation of the gross economy model with the acceptance of the impact of climate change on economic output. Net economic output is a function of gross economic output and damages as a result of climate change and the cost of abatement to climate change:

$$Y_{net,t,r} = Y_{gross,t,r} - \Lambda_{t,r} - \Omega_{t,r} \tag{D.16}$$

in which $\Omega_{t,r}$ is the cost of climate damages and $\Lambda_{t,r}$ is the cost of abatement. The cost of climate damages is calculated through:

$$\Omega_{t,r} = Y_{gross,t,r} * damagefraction_{t,r} + SLR_{damages,t,r} \tag{D.17}$$

in which $SLR_{damages,t,r}$ is the cost of damages relating to sea level rise. The damage fraction is determined through:

$$damagefraction_{t,r} = \lambda_1 T_{atm,t} + \lambda_2 T_{atm,t}^{\lambda_3} \tag{D.18}$$

in which λ are the damage parameters which together with the functional form of the damage function are highly controversial (W. Nordhaus and Sztorc 2013). Important to note is that the damage function and parameters are calibrated for a damage in the range of 0 to 3 ^0C. The abatement cost is determined through:

$$\Lambda_{t,r} = Y_{gross,t,r} * abatementfraction_{t,r} \tag{D.19}$$

in which the abatement fraction is given by:

$$abatementfraction_{t,r} = \beta_{1,t,r} \mu_{t,r}^{\beta_2} \tag{D.20}$$

in which μ is the emissions control rate, β_2 a constant set there at 2.8, and β_1 a variable that is determined by:

$$\beta_{1,t,r} = paybacktime_{t,r} * \frac{\sigma_{t,r}}{\beta_2} \tag{D.21}$$

in which payback time is a function that assumes the existence of a backstop technology that will be developed in the future and that is able to replace all fossil fuel based technologies. The cost of this backstop technology is initially high but reduces over the years. The backstop technology is incorporated in the payback time function by setting the backstop price for a given year equal to the marginal cost of abatement at a control rate of 100%. Here, the year of the backstop technology is set at 2250. $\sigma_{t,r}$ is the output to emissions ratio.
With the net economic output known, investment and consumption are calculated through:

$$I_{t,r} = S_{t,r} * Y_{net,t,r} \tag{D.22}$$

in which $I_{t,r}$ is investment and $S_{t,r}$ is the savings rate, which is a lever of the model.

$$C_{t,r} = Y_{net,t,r} - I_{t,r} \tag{D.23}$$

in which $C_{t,r}$ is total consumption in a region. Consumption per capita is calculated by dividing the consumption of a region by that region's labor force.
Note that with investment known, a new iteration of the RICE model can start. Investment impacts capital stock, which impacts gross output. Gross output impacts the carbon cycle through emissions which on its turns impacts the climate model through radiative forcing. The increase in atmospheric temperatures impacts the net economic module through climate damages and abatement costs, which impacts investment. The two economic levers in the current model are μ, the emissions control rate which impacts the abatement costs, and S the savings rate which impacts investment. The earlier μ is set, the lower overall abatement costs will be and the higher S is set, the lower the deterioration of the climate. The logic behind these relations is that as societies set aside part of their output for investment instead of consuming it, they are able to invest in their economies in the long term therefor securing consumption in the future. In the RICE model, climate change is incorporated into the economic model through climate damages and abatement costs thus extending the logic of using investment to limit the detrimental effects of climate change to regional economies.

Welfare

A variable that does not influence the model but does inform on the state of the model is the welfare of discounted utility based on consumption. It is a basis point for further ethical principles and is given by:

$$W_{c,t,r} = U(C)_{t,r} * R_t * L_{t,r} \tag{D.24}$$

in which $U(C)_{t,r}$ is the utility of consumption of region r at time t, R_t is the social discount factor for consumption at time t and $L_{t,r}$ is the population of region r at time t as used in the core of the RICE

model.
The utility of consumption is defined by the Constant Relative Risk Aversion function (Wakker 2008) and is given by:

$$U(C)_{t,r} = \frac{C_{t,r}^{1-\eta}}{1-\eta} \tag{D.25}$$

in which $C_{t,r}$ is the consumption per capita and η the elasticity of marginal utility of consumption. In the current RICE model this value is set to 1.5. The consumption per capita is taken from the core of the RICE model.
The social discount factor for consumption is defined by:

$$R_t = \frac{1}{(1+irstp)^{\delta t * t}} \tag{D.26}$$

in which $irstp$ is the initial rate of time preference for consumption which is a lever of the model.

D.0.2. Model Verification

The implementation of the RICE model in the python programming language is based on a number of previous implementations. The original DICE optimization model by Nordhaus (W. Nordhaus and Sztorc 2013) was translated to a stochastic simulation model in the python language by (Lingeswaran 2019). The (global) DICE python model was then adapted to the (regional) RICE model in python (Tjallingii 2021). This RICE python model was then further adapted and restructured in an object oriented fashion (Reddel 2022). This research uses a rewritten version of the RICE model based on the most recent implementation (Reddel 2022). The model had to be rewritten because the previous implementation did not allow for changes in action variables mid-simulation. The POT algorithm is centered around the notion of sequential decision making as opposed to 'one-shot' simulation. The previous implementation allowed for one-shot simulation in which the initial values for all variables are set and run a deterministic course throughout the simulation. Sequential decision making sets initial values for all variables as well but allows to change them throughout the simulation. The POT algorithm requires actions to be changed depending on system states and therefor warranted a complete revision of the python RICE model. For ease of reference, the version of RICE used in this research is dubbed 'IAM RICE' and the version of RICE to which it is compared for verification is named 'pyRICE 2022'.
A number of important variables in the model are recorded for both versions and compared to one another. Model conditions are identical and both have been run with the Nordhaus policy (miu=2135, sr=0.248 and the irstp=0.015). The difference between the versions is analysed by subtracting the pyRICE 2022 values from the IAM RICE values and normalizing the difference by dividing it by the pyRICE 2022 values. The analysis is summarized in Figure D.2 which shows that the differences for a number of important variables is not zero but sufficiently small.

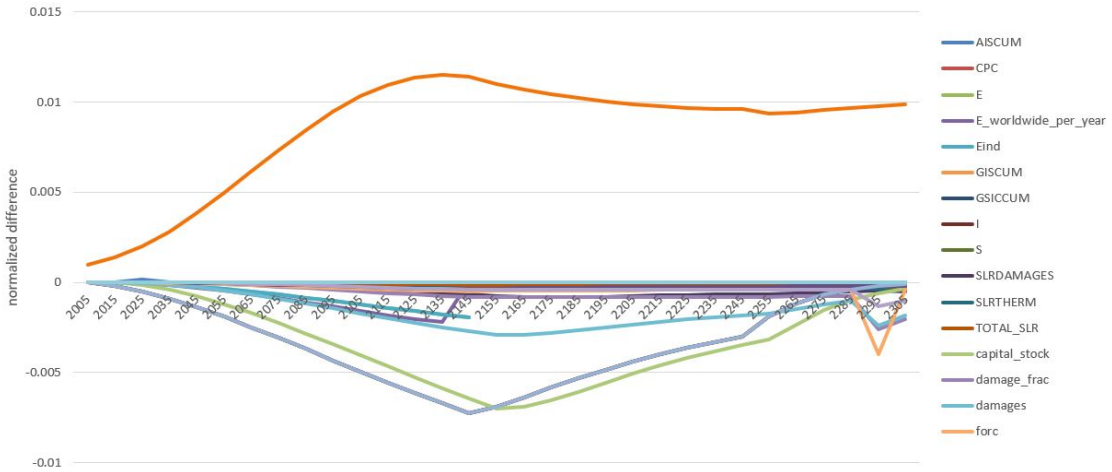

Figure D.2: Normalized difference between IAM RICE and pyRICE 2022 for a selection of model variables. The normalized difference is calculated by dividing the difference between new and the old model with the old model (new-old)/old

E

RICE regions

Across different versions of RICE it can take unnecessarily long to find the exact countries that belong to a region. For the development of IAM RICE, an attempt was made to gather all necessary input from online sources so that they could be automatically updated. The information on the countries that make up each region is however not detailed enough to gather the input data from a source other than the Nordhaus papers (W. Nordhaus and Sztorc 2013). Table E.1 states the regions in this implementation of RICE as detailed as they are available.

Region	Description
US	United States of America
OECD-Europe	European countries that are members of the OECD
Japan	Japan
Russia	Russia
Non-Russia Eurasia	Countries that are in Eurasia excluding Russia
China	China
India	India
Middle East	Middle-Eastern countries
Africa	Countries on the African continent
Latin America	Countries of Latin America
OHI	Other higher-income countries
Other non-OECD-Asia	Countries in Asia that are not members of the OECD

Table E.1: Regions represented in the IAM RICE model.

F

Optimized Look-Ahead Tree algorithm

The structure of an OLT is such that each node represents a system state and the depth of the tree equals the number of timesteps or system evaluations. At some initial system state multiple actions is possible which transform the system to a new system state each associated with a reward value. For each potential new system state there are again a number of actions possible to take the system to potential new system states. The tree with possible system states and associated rewards thus branches out until the end of the simulation horizon. The path from the root node to the leaf node with the highest cumulative reward is the optimal path of actions, *i.e.* the optimal policy. The structure of such a tree is shown in F.1.

This type of tree structure can grow very rapidly for many possible actions and long simulation horizons up to a point where it is computationally no longer trivial to find the optimal path. Search strategies like breadth first or depth first strategies exist to find the optimal path with a feasible computational budget. An OLT, however, employs a path feature function to traverse the many possibilities in a look-ahead tree. Given a path in the look-ahead search tree, a path-feature function outputs a vector of features describing that path which can be global or local properties. An example given in (Maes, Wehenkel, and Ernst 2012) formulates a path-feature function based on the global properties of depth and reward. These variables essentially cover the trade-off between breadth first or depth first search and the trade-off between random or greedy search. More elaborate path-feature functions can be constructed when more detailed system specific information is available. Given a computational budget, the look-ahead search tree can now be incrementally expanded with the aid of the path-feature function.

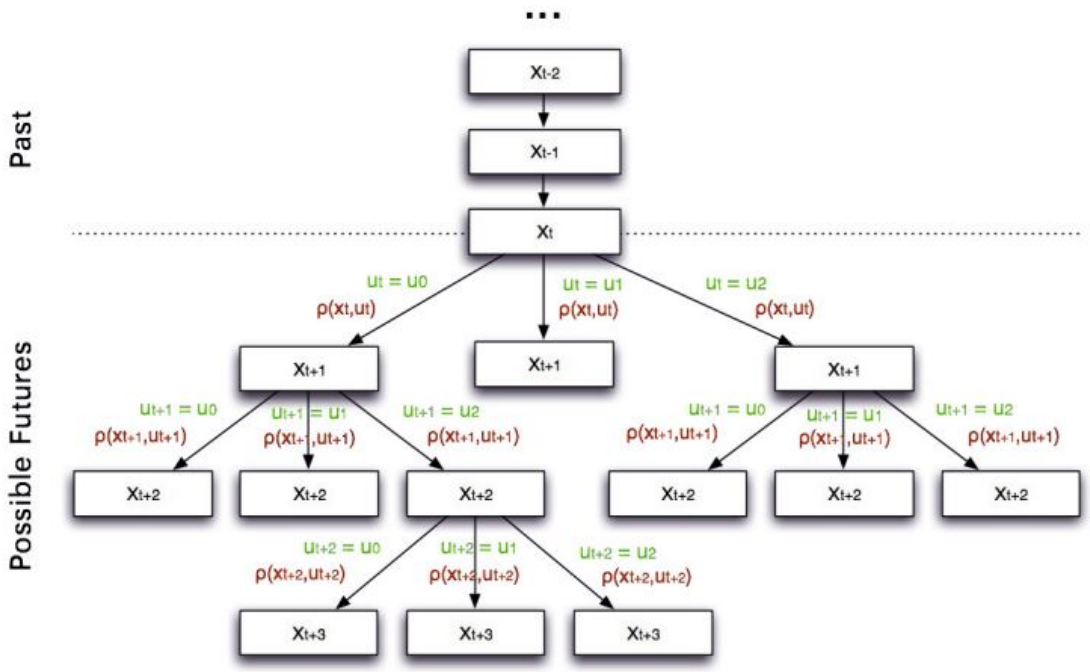

Figure F.1: Structure of a look-ahead tree for which 3 different actions are possible at each future timestep. The nodes represent a system state x, each associated with an action u and a reward ρ at a certain point in time (Maes, Wehenkel, and Ernst 2012).

G

Discrete Actions Specification

Model	Action name	Action in model
Folsom	Release_Demand	target[t] = D[t]
	Hedge_90	target[t] = 0.9 * D[t]
	Hedge_80	target[t] = 0.8 * D[t]
	Hedge_70	target[t] = 0.7 * D[t]
	Hedge_60	target[t] = 0.6 * D[t]
	Hedge_50	target[t] = 0.5 * D[t]
	Flood_Control	target[t] = max(0.2 * (Q[t] + S[t - 1] - 0.0), 0.0)
RICE	miu_2065	mu_target = 2065
	miu_2100	mu_target = 2100
	miu_2180	mu_target = 2180
	miu_2250	mu_target = 2250
	miu_2305	mu_target = 2305
	sr_01	sr = 0.1
	sr_02	sr = 0.2
	sr_03	sr = 0.3
	sr_04	sr = 0.4
	sr_05	sr = 0.5
	irstp_0005	irstp = 0.005
	irstp_0015	irstp = 0.015
	irstp_0025	irstp = 0.025

Table G.1: Specification of discrete actions that are recognized by the Folsom lake model and the RICE model.