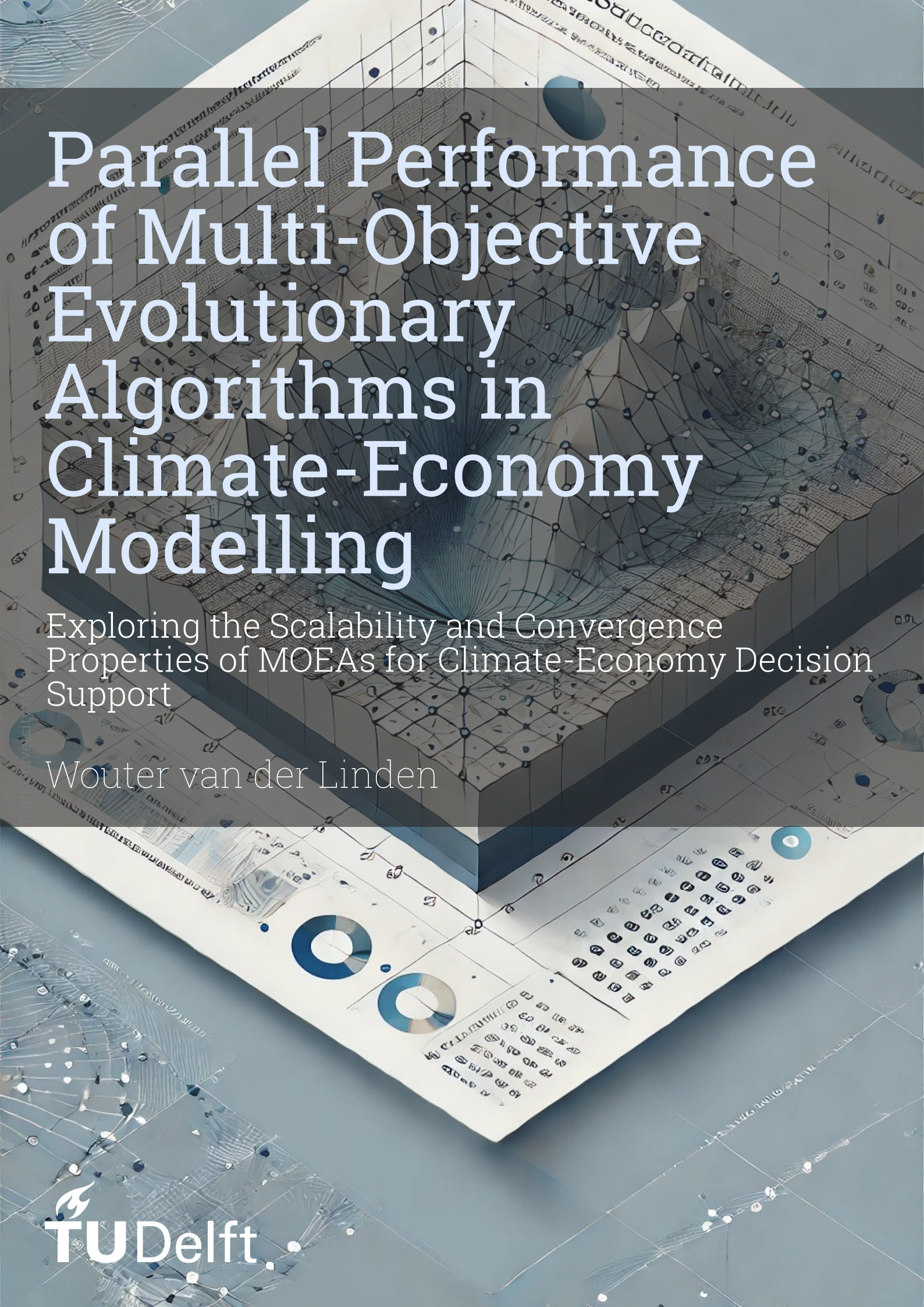

Parallel Performance of Multi-Objective Evolutionary Algorithms in Climate-Economy Modelling

Exploring the Scalability and Convergence Properties of MOEAs for Climate-Economy Decision Support

Wouter van der Linden

TU Delft

Parallel Performance of Multi-Objective Evolutionary Algorithms in Climate-Economy Modelling

Exploring the Scalability and Convergence Properties of MOEAs for Climate-Economy Decision Support

by

Wouter van der Linden

to obtain the degree of Master of Science

at the Delft University of Technology,

to be defended publicly on Monday July 7, 2025 at 15:00.

Student number:	5170451	
Project duration:	February 1, 2025 – July 7, 2025	
Thesis committee:	Prof. dr. ir. J. H. Kwakkel,	TU Delft, First Supervisor
	Prof. dr. M. E. Warnier,	TU Delft, Second Supervisor
	Ir. P. Biswas,	TU Delft, Advisor
	Dr. J. Z. Salazar,	TU Delft, Advisor

An electronic version of this thesis is available at http://repository.tudelft.nl/.

All project code is available at https://github.com/woutervdl/MOEA_convergence

TU Delft

Preface

I would like to express my deep gratitude for this thesis' entire supervisory committee. Jan Kwakkel and Martijn Warnier, thanks for providing guidance and providing the critical eyes to steer this thesis into the right direction, as well as facilitating the entire thesis and its process. Thank you Jazmin Salazar for providing essential advice whenever I needed it. Also a special thanks to Palok Biswas, who not only enabled this thesis in the first place by having built JUSTICE, but also always made time for a quick session or quick response whenever I had any questions. This thesis would definitely not have looked the way it does now without any of you.

Finally, a big thank you to my parents, who have always supported me and enabled me to fully focus on this project. A project which marks the end of my time as Engineering & Policy student at TU Delft, a time which I have thoroughly enjoyed and which has taught me so many things. While not yet sure what my future will look like, I hope to bring all that I have learned at EPA into practice and contribute to the world's greater good.

Wouter van der Linden
Delft, June 2025

Executive Summary

This thesis investigates the comparative performance of three Multi-Objective Evolutionary Algorithms (MOEAs), applied to the optimisation of complex Integrated Assessment Models (IAMs), JUSTICE specifically. The study addresses a critical gap currently present in empirical research with regards to MOEA effectiveness for real-world, highly dimensional and multi-modal problems. To address this gap, three MOEAs, ϵ-NSGA-II, Borg and Generational Borg, are evaluated.

For each MOEA, the computational efficiency, convergence dynamics and solution quality are evaluated. Their performance is not only tested on the JUSTICE IAM, but also on the DTLZ2 and DTLZ3 benchmark problems. This is done to provide extra insights and context to the JUSTICE findings, as these problems possess different characteristics. DTLZ2 is a simple problem, acting as a baseline. DTLZ3 is a highly multi-modal and thus more difficult problem. Also considering the differences observed in how MOEAs perform across these problems contextualises all results.

The findings illustrate Borg's ability to consistently outperform the other MOEAs and achieve the best performance metric values. This performance gap increases when problem complexity increases due to high-dimensionality or multi-modality. Borg's steady-state nature and auto-adaptive features, like operator selection and tournament sizing, enable it to outperform ϵ-NSGA-II and Generational Borg in practically all scenarios. However, when evaluating ϵ-NSGA-II it shows that it can offer a faster and less complex alternative. Though, important to note, when problem complexity increases, ϵ-NSGA-II's performance sharply declines, especially in heavy multi-modal problems. The main reason for this appears to be its lack of adaptive features and generational nature. This is confirmed by Generational Borg's behaviour, which performs on par with ϵ-NSGA-II on a simple problem like DTLZ2, but performs better (though still worse than Borg) on a complex multi-modal problem like DTLZ3. ϵ-NSGA-II's and Generational Borg's generational nature do shorten their runtimes on simple problems as compared to Borg, but these differences become negligible on a complex problem like JUSTICE.

The implications of the results obtained in this study are significant for real-world policy design, climate change mitigation in particular. The results have shown that MOEA choice can critically impact the quality and robustness of the eventually obtained policy solutions. While acknowledging the viability of generational MOEAs for simpler problems, or in severe cases of time-constraints, this study has highlighted the benefits of opting for steady-state MOEAs with more adaptive features when optimising a demanding problem such as JUSTICE.

Several directions for future research have been identified. Scalability gains should be more thoroughly explored, if done correctly much higher core counts than used in this thesis can also be considered. As Borg still demonstrated increasing trends after 70000 NFE for JUSTICE, it would also be interesting to further increase the computational budget. Lastly, the generalisability and validity of the findings from this study could be improved by performing similar experiments using additional MOEAs and problem configurations. Overall, this thesis tried to advance the field of multi-objective optimisation in climate-economy modelling by providing insights for researchers and policy-makers to aid in designing robust climate policies.

Contents

List of Figures

List of Tables

Nomenclature

Abbreviations

Abbreviation	Definition
DE	Differential Evolution Crossover
DPS	Direct Policy Search
EMODPS	Evolutionary Multi-Objective Direct Policy Search
HPC	High Performance Computing
IAM	Integrated Assessment Model
MPI	Message Passing Interface
MOEA	Multi-Objective Evolutionary Algorithm
MOP	Multi-Objective Problem
NFE	Number of Function Evaluations
PCX	Parent-Centric Crossover
PM	Polynomial Mutation
SBX	Simulated Binary Crossover
SPX	Simplex Crossover
UM	Uniform Mutation
UNDX	Unimodal Normal Distribution Crossover

1

Introduction

Climate change and its escalating impacts present society with one of the most critical and complex challenges to date. According to a recent IPCC report, it is very likely that the 2°C temperature increase above pre-industrial levels limit agreed upon in the 2015 Paris climate agreements will be reached in the near term, increasing even further in the higher emission scenarios (IPCC, 2023). To properly address this challenge, it is necessary to develop, test and implement effective, robust and equitable policies.

To aid the development of such policies, researchers and policymakers employ Integrated Assessment Models (IAMs). These are models that incorporate knowledge from multiple fields like climate science, economics and energy systems, combining them into a unified model. This allows for simulation of different policies, and evaluation of the long-term effects of these policies. IAMs thus enable the exploration of complex and often emergent climate-economy interactions, as well as the evaluation of the trade-offs that inevitably have to be made in policy design. While IAMs are, per definition, simplified representations of reality, the policy problem that they frame is characterised by high dimensionality and non-linearity. These characteristics reflect the uncertainty present in climate-economy developments. However, this also makes IAMs very challenging to optimise. Multiple conflicting objectives, like minimising CO_2 while also minimising economic costs for example, are present. Additionally numerous variables influence these objectives and each other. How can the best policies be found?

Traditional optimisation methods often struggle with the challenges of high dimensionality and non-linearity. Instead, the Evolutionary Multi-Objective Direct Policy Search (EMODPS) framework can be used, formulating the problem in terms of state-based control. EMODPS combines Direct Policy Search (DPS) with the use of a Multi-Objective Evolutionary Algorithm (MOEA) to find an optimal control policy. This policy is a function mapping the state of a system (e.g. global temperature) to an action (e.g. an emissions control rate). The policy is represented by a non-linear activation network, like Radial Basis Functions (RBFs) or Artificial Neural Networks (ANNs), whose parameters are the optimisable policy levers. These parameters can then be optimised with a MOEA (Salazar et al., 2022). This approach enables the exploration of high-dimensional policy spaces and the generation of a diverse set of Pareto-optimal solutions, which are policies that are optimal in at least one objective compared to all other policies, eventually providing policymakers with a spectrum of optimal solutions and trade-offs to choose from.

However, the resulting set of solutions is greatly affected by the choice of MOEA. Key differences in architecture, like a generational versus steady-state population update mechanism or the inclusion of auto-adaptive operators, create fundamental trade-offs in MOEA behaviour and capability. While a large body of research regarding the strengths and weaknesses of different MOEAs and their architectures exists already, most of this research was on the application of MOEAs to specially designed benchmark problems or other models than IAMs. Particularly, the trade-offs between MOEA convergence dynamics, solution quality and scalability applied to a demanding IAM remain underexplored. This thesis aims to address this gap by performing a quantitative comparison of three leading MOEAs, ϵ-NSGA-II, Borg and Generational Borg. Each MOEA will be used to optimise the JUSTICE

IAM, which is a new and open-source IAM specifically designed for conducting research on equitable climate-economic policy development under uncertainty (Biswas et al., 2023). Additionally, to provide more robustness to the drawn conclusions, two benchmark problems, DTLZ2 and DTLZ3, are also included in this study. By varying the number of cores, and analysing a number of performance metrics, this study aims to provide insights into the interplay of IAM problem characteristics, MOEA design and computational resources. This way researchers working with IAMs can make a better informed MOEA selection, aiding future IAM-based policy research. Therefore this thesis will answer the following central question:

How and why do ϵ-NSGA-II, Borg and Generational Borg differ regarding their computational efficiency, convergence dynamics and solution quality when optimising a demanding model such as JUSTICE using the EMODPS framework?

The remainder of this thesis is structured as follows. Chapter 2 discusses relevant literature on IAMs, EMODPS and MOEAs. Chapter 3 explicitly states the knowledge gap and formulates the main research question as well as four sub-questions. Chapter 4 describes the methodology, including problem formulations, experimental setup and performance metrics. The experimental results are presented in chapter 5. Chapter 6 discusses the implications and limitations of the obtained results. Finally, chapter 7 concludes with recommendations for practical applications as well as suggestions for future research.

2

Literature Review

To establish a solid foundation for this research, it is essential to first explore and clarify several key concepts and theoretical models. This section will discuss Integrated Assessment Models (IAMs), the Evolutionary Multi-Objective Direct Policy Search (EMODPS) framework and the intricate workings of Multi-Objective Evolutionary Algorithms (MOEAs).

2.1. Integrated Assessment Models

Firstly, it is essential to examine the role and state of modern-day IAMs. As briefly discussed in the introduction, IAMs try to model the complex and dynamic interplay between a wide range of economical and natural factors. They use theory from different disciplines like economics, climate science and energy systems to provide quantitative insights into the potential future states of the climate and economies resulting from various assumptions and policy choices. This way, IAMS allow for research to be conducted on possible future climatological and economical pathways and evaluation of policy trade-offs. A number of versions of these models have been developed over the years, consisting of completely distinct models, models derived from others and models which have been improved over the course of the past years. Examples are DICE (Nordhaus, 1993), RICE (Nordhaus and Yang, 1996), IMAGE 3.0 (Stehfest et al., 2014) and many others, each differing in scope or focus.

Despite the widespread use and continued development of such models, there also exist criticisms. These criticisms mainly stem from IAMs over- or underestimating certain effects due to erroneous assumptions and thus wrongful implementations of real-world dynamics into the IAM (Ackerman et al., 2009). Despite these criticisms being justified, this research will not take them into further consideration, as the IAM results themselves are not the main focus here, rather the contribution of MOEA dynamics to these results. Furthermore, a different IAM called JUSTICE, which is inspired by RICE and RICE50+, will be used in this research. JUSTICE is intentionally set up as a simpler and modular framework purpose-built to explore influences of underlying modelling assumptions, making it an excellent IAM for investigating MOEA scalability and dynamics (Biswas et al., 2023).

2.2. Evolutionary Multi-Objective Direct Policy Search

As mentioned in the introduction, IAMs come with a number of challenges. IAMs are inherently complex, non-linear and uncertain. Given their highly-dimensional policy space, it is not easy to optimise IAM results. The EMODPS framework comes into play here. EMODPS uses Direct Policy Search (DPS) in combination with a MOEA to directly optimise specific policy parameters in the IAM while being able to handle different and conflicting objectives (Giuliani et al., 2016). DPS defines a parameterised policy $\pi_{\boldsymbol{\theta}}(s)$ and searches for a parameter vector $\boldsymbol{\theta}$ that optimises performance. A policy function is used to map the observed system state s_t to an action u_t as $\pi_{\boldsymbol{\theta}}(s) \rightarrow u_t$. Oftentimes, Artifical Neural Networks (ANNs) or Radial Basis Functions (RBFs) are used as policy function approximators, as they allow for complex and non-linear relationships between states and optimal actions. Parameter vector $\boldsymbol{\theta}$ then represents the variables of these approximating functions (like weights, biases, layers etc.). Policy

$\pi_{\boldsymbol{\theta}}$ then is simulated, and based on the objective function values (e.g. welfare loss or temperature rise) a MOEA searches the policy parameter space for optimal parameter vectors $\boldsymbol{\theta}$, optimising the approximating policy function and ultimately the objective function values. EMODPS is not guaranteed to find the true Pareto front, but is a strong heuristic that allows for researchers to explore a wide variety of Pareto-optimal policies which balance conflicting objectives like emission reductions and economic costs.

EMODPS is applicable to a number of different fields, and has been applied as such. Salazar et al. (2016) have applied EMODPS to research on water reservoir management, Gupta (2020) applied EMODPS to energy management problems, and EMODPS is of course used in climate research and IAMs (Carlino et al., 2022). In all of these applications, EMODPS brings the benefit of allowing exploration of very high-dimensional policy spaces, under deep uncertainty, and generating a Pareto front with robust and diverse solutions.

2.3. Multi-Objective Evolutionary Algorithms

MOEAs are algorithms specifically designed to tackle problems with multiple, often conflicting, objectives. These multi-objective problems (MOPs) can be defined as;

$$\begin{aligned} \text{min} \quad & F(\mathbf{x}) = (f_1(\mathbf{x}), \ldots, f_m(\mathbf{x}))^T \\ \text{s.t.} \quad & \mathbf{x} \in \Omega \end{aligned} \tag{2.1}$$

where $\mathbf{x} \in \Omega$ is a decision vector and Ω represents the total decision space. $\mathbb{R}^m$ represents the total objective space and $F(\mathbf{x})$ contains m objective functions $f_i : \Omega \to \mathbb{R}, i = 1, \ldots, m$ (Zhou et al., 2011).

As mentioned, these MOPs often have conflicting objectives. As a result of this there is no single solution that leads to optimisation of every single objective in the objective space and trade-offs have to be made per definition. To address this issue, MOEAs look for solutions with the most optimal trade-offs. Introducing the concept of Pareto-optimality, well-known in economics, which entails that a solution belongs to the Pareto set when there is no other solution that can improve on one objective without making at least one other objective worse, thus making the solution non-dominated. Dominance and Pareto-optimality can be formally defined through the following three definitions (Zhou et al., 2011).

Definition 1 (Pareto dominance)**.** *For the MOP defined in equation* (2.1) *and with vectors* $\mathbf{u} = (u_1, \ldots, u_m)^T$ *and* $\mathbf{v} = (v_1, \ldots, v_n)^T$*, dominance can be formally defined as* $\mathbf{u} \prec \mathbf{v}$ *iff* $\forall i \in \{1, \ldots, m\}$*,* $u_i \leq v_i$ *and* $u_i \neq v_i$*. (Note that this definition holds in the case of minimisation problems which is the standard assumption in this research, in case of maximisation it holds the other way around).*

Definition 2 (Pareto set)**.** *A feasible solution* $\mathbf{x}^* \in \Omega$ *is a Pareto optimal solution iff* $\nexists \mathbf{y} \in \Omega$ *such that* $F(\mathbf{y}) \prec F(\mathbf{x}^*)$*. The entire set of Pareto optimal solutions, the Pareto set (PS), can be defined as*

$$PS = \{\mathbf{x} \in \Omega \mid \nexists \mathbf{y} \in \Omega, F(\mathbf{y}) \prec F(\mathbf{x})\} \tag{2.2}$$

Definition 3 (Pareto front)**.** *The Pareto set can be mapped to the objective space to form the Pareto front (PF), which is defined as*

$$PF = \{F(\mathbf{x}) \mid \mathbf{x} \in PS\} \tag{2.3}$$

The Pareto set and Pareto front are essential tools for decision makers, allowing them to make optimal and informed trade-offs and decisions in highly dimensional and conflicting problems. MOEAs are often used to tackle MOPs in numerous different fields, ranging from DNA sequence design (Shin et al., 2005), to neural network training (Delgado et al., 2008) to greenhouse control (Z. Zhang, 2008) and many more. To this end a lot of different types of MOEAs exist, each more suited to MOPs with specific characteristics. Their differences occur in a number of key components found in every MOEA.

2.3.1. Key Components

Population

One of the most fundamental components of a MOEA is the concept of a population. MOEAs maintain a population of solutions at any given time. This allows for effective and simultaneous exploration of

different parts of the decision space and maintaining a set of diverse solutions capturing a big portion of the trade-off spectrum. The way this population is updated with new solutions differs per MOEA and can be classified as either generational or steady-state. This distinction is one of the main points of interest in this research. In generational MOEAs, the entire population is replaced each generation. Through the use of certain selection criteria parents from the previous population are selected and subjected to genetic manipulation, which in turn creates offspring present in the new generation. In steady-state MOEAs often only one or two individuals are replaced each iteration. Again, through selection criteria parents are selected and produce offspring through genetic manipulation. However, now this offspring is inserted into the population at the cost of removal of a less fit individual (Smith et al., 1998). This distinction between generational and steady-state population update mechanics is a core architectural trade-off. Quantifying its effect on solution quality and scalability is one of the major objectives of this thesis.

The size of the population influences the degree to which the MOEA can fully explore the decision space, where a larger population offers greater exploration but does so at the cost of slower convergence. On the other hand, a smaller population might converge faster, but lead to sub-optimal solutions. To balance exploration and exploitation some MOEAs, like Borg, use adaptive population sizing, where the population size differs throughout the optimisation based on exploration or exploitation needs (Hadka and Reed, 2013). A lot of MOEAs also make use of archiving. High-quality solutions are stored in an archive to prevent them being lost due to random genetic manipulation (M. Li, 2021).

Selection Mechanism

Selection mechanisms steer the evolution of solutions to the Pareto front through determining which solutions are selected to produce offspring and which solutions are eventually maintained. All MOEAs adhere to one of four frameworks. These are decomposition-based MOEAs, where the MOEA divides a MOP into multiple scalar subproblems which are then optimised separately (Q. Zhang and Li, 2007). Dominance-based MOEAs, which rely on Pareto dominance and ranking solutions based on their respective dominance (Branke and Deb, 2005). Indicator-based MOEAs, which rely on tracking a number of performance metrics and rank solutions based on their contribution to these metrics (Zitzler and Künzli, 2004). Lastly there exist many hybrid combinations of the first three, combining their strengths and mitigating weaknesses.

Variation Operators

Once the parents have been selected. The genetic manipulation is handled by variation operators that perform certain operations on the parental individuals. This way, new offspring, and thus greater solution diversity, is introduced into the population and a greater part of the decision space can be explored. There are two main variation operators, crossover and mutation. With crossover genetic material between parent individuals is exchanged to create new offspring, the goal being to combine good traits of parent solutions into a new solution and promote decision space exploration. Numerous different types of crossover exist, with different MOEAs opting for different types or combinations of crossover. Examples are simulated binary crossover (SBX) (Deb, Agrawal, et al., 1995), differential evolution crossover (DE) (Storn and Price, 1997), unimodal normal distribution crossover (UNDX) (Kita et al., 1999), parent-centric crossover (PCX) and simplex crossover (SPX) (Mashwani et al., 2015).

With mutation, small changes are introduced at random to certain individuals, preventing search stagnation. Just as with crossover, MOEAs use different types (or combinations) of mutation. Examples are polynomial mutation (PM) and uniform mutation (UM) (Hamdan, 2010; Rajakumar, 2013). As mentioned, certain MOEAs, are self-adaptive and can dynamically change their crossover and mutation methods based on what the optimisation needs at that point. Quantifying the effectiveness of this adaptivity compared to non-adaptive MOEAs, like ϵ-NSGA-II, in optimising complex multi-modal problems is another key focus of this thesis.

Diversity

As mentioned previously, ensuring diversity amongst the solution set is important. It makes sure all parts of the decision space are being explored, preventing premature convergence through getting stuck in local optima and providing decision-makers with a diverse and evenly distributed set of trade-offs. To ensure this diversity, MOEAs make use of implicit and explicit mechanisms (Martí et al., 2018).

Implicit mechanisms mainly consist of the aforementioned selection mechanisms and variation operators, which introduce diversity through their algorithmic operations. On the other hand, explicit mechanism explicitly enforce diversity.

The main approach to actively enforcing this diversity is through so-called niching. Multiple different approaches to niching exist, but they all share the underlying principle of encouraging the algorithm to explore and preserve multiple areas of the objective space. Examples are the crowding distance, used in NSGA-II, where the distance between solutions in the objective space is measured and solutions in sparse regions are favoured (Deb et al., 2002). Another example is ϵ-dominance, used by ϵ-NSGA-II and Borg, where the objective space is discretised into ϵ-sized grids and solutions need to differ at least one grid, ensuring a minimum resolution between solutions (Deb, Mohan, and Mishra, 2005). A different approach is using reference points, which is done by MOEA/D, where solutions are distributed in the objective space through the use of predefined directions (Q. Zhang and Li, 2007).

Another mechanism often used is a population restart. Here, based on a certain trigger, the population is reinitialised. Oftentimes MOEAs, like Borg, use a hybrid and dynamic approach to diversity management, adjusting mechanisms based on performance indicators (Hadka and Reed, 2013).

Elitism

Elitism is another essential MOEA component. Elitism ensures that high-quality solutions are kept and preserved across different generations. It retains non-dominated solutions, thus ensuring convergence to the Pareto front, through different mechanisms. Firstly, non-dominated solutions are stored in an archive. This prevents non-dominated solutions from being lost to genetic operations. Some classical MOEAs, like NSGA-II, only use this archive for storage. Other more modern MOEAs, like Borg, actively use archived solutions to generate new solutions, for example reinitialising the population with archived solutions when a population restart is triggered.

2.3.2. MOEA Parallelisation

Using EMODPS and MOEAs to optimise IAMs (or any other complex problems) often involves computationally heavy function evaluations. To gain meaningful results, many function evaluations are necessary. Parallel computing is essential in reducing the wall-clock time and increasing the efficiency of such optimisations. Literature exists on two main approaches to parallelising MOEAs, the Master-Slave- and the Island Model (Durillo et al., 2008; Limmer and Fey, 2017).

In the Master-Slave approach a central master process (CPU core) manages the main population and archive handling as well as the genetic manipulation. It passes on newly generated individuals to other worker nodes who then evaluate these individuals and calculate the objective function values. The workers pass their results back to the master which then proceeds with the next step in the algorithm. This setup is relatively straightforward. However, its effectiveness depends heavily on the time it takes for function evaluations being longer than the time it takes to communicate results. If this is not the case, the master node can become a bottleneck. Generational algorithms face the additional limitation of synchronisation overhead (Hadka et al., 2013; Zăvoianu et al., 2013).

The Island approach can be considered to be a scaled and refined version of the Master-Slave approach. Multiple master nodes each run their own instantiation of the algorithm, and have their own set of worker nodes. Effectively creating different node 'islands' (Giuliani et al., 2017). Individuals can periodically also be exchanged between different islands in case of stagnation for example. This approach further reduces communication frequency and is very fit for distributed computing (Limmer and Fey, 2017). However, given the setup of the MOEAs used in this research, the parallelisation model used will be that of the regular Master-Slave approach.

Another important distinction that needs to be made when considering MOEA parallelisability is its synchronous or asynchronous execution. This concept is very closely related to the difference between generational and steady-state MOEAs. Generational MOEAs, like ϵ-NSGA-II and Generational Borg, operate synchronously. This means that when the master node dispatches individuals to its workers, it must wait until it has received the evaluation results from each one of its workers. This is a relatively easy approach to implement, especially for embarrassingly parallel tasks where the workload can be split into many smaller independent tasks that require no communication with each other. However, a big downside is that in case the evaluation times vary, which they often do, workers that

finish earlier remain idle until all workers have finished and the master node can move on to the next step. This introduces synchronisation overhead, inefficiencies and reduced scalability, especially on High-Performance Computing (HPC) systems (Zăvoianu et al., 2013). On the other hand, steady-state MOEAs adopt an asynchronous approach. Here the master still dispatches individuals to worker nodes for evaluation, but when the worker is done evaluating and returns its results to the master node it is immediately handed a new individual for evaluation. This eliminates the bottleneck in the synchronous approach where the master has to wait for the slowest worker to finish. As a result, idle times are minimised, CPU utilisation is maximised and significantly better scalability is realised (Hadka et al., 2013). Empirically quantifying this theoretical difference between synchronous generational and asynchronous steady-state MOEAs, especially applied to a computationally heavy problem, is a key goal of this thesis. Generational Borg, which shares Borg's auto-adaptive operators and ϵ-NSGA-II's generational nature, is included in this study to be able to properly distinguish the performance gains from an asynchronous approach versus those from auto-adaptive features.

2.3.3. ϵ-NSGA-II, Borg and Generational Borg

As the focus of this study mainly lies on the differences in parallelisability between generational and steady-state MOEAs, three different MOEAs, being ϵ-NSGA-II, Borg and Generational Borg, will be examined. This section will discuss and compare each of these MOEAs properties based on the key components discussed above and illustrate their workings by providing pseudocode for each MOEA. A summary of the properties can be found in table 2.1.

ϵ-NSGA-II

ϵ-NSGA-II is a generational MOEA largely based on the older NSGA-II algorithm but using ϵ-dominance archiving instead of crowding distance as its diversity mechanism and adding adaptive population sizing (Deb et al., 2002; Kollat and Reed, 2006).

ϵ-NSGA-II starts by initialising a population, with a predefined size, of random individuals. Next, it makes use of tournament selection to select the parent individuals, using non-dominated sorting to rank them and selecting the individual with the highest rank. When the parents are selected, it defaults to use SBX for crossover and PM for mutation. It uses non-dominated sorting to rank all offspring, adding one non-dominated solution per ϵ-grid cell to the archive and replacing its predecessor in that ϵ-grid if the predecessor is dominated. This preserves diversity and prevents the archive from growing without bounds. Its adaptive population sizing and restart mechanism work as follows;

$$P_{target} = \lambda \cdot |A_{size}| \tag{2.4}$$

where P_{target} is the target population size, A_{size} is the current archive size and λ is a user-defined population-to-archive ratio. A restart can now be triggered through stagnation (a set number of iterations have passed since the last restart) or an imbalance between the population and archive sizes, defined as;

$$P_{size} - (\lambda \cdot |A_{size}|) > 0.25 \cdot (\lambda \cdot |A_{size}|) \tag{2.5}$$

where P_{size} is the current population size. In the case of a restart, it draws new parents from the archive and introduces additional randomly generated individuals to meet the target population size, introducing new diversity. New offspring is generated and the cycle repeats.

Keeping all other parameters native to MOEAs in general static makes that ϵ-NSGA-II is computationally simple and efficient. On the other hand this can also introduce challenges in more complex problems. For example, in problems with more than four objectives ϵ-boxing can become inefficient due to exponential growth in grid complexity. Additionally, with ϵ-NSGA-II being a generational MOEA, it makes use of a synchronous master-slave parallelisation approach. Therefore its scalability should theoretically be relatively limited, especially compared to other steady-state MOEAs. Nevertheless, ϵ-NSGA-II still performs very well compared to other MOEAs, and its efficient approach makes that it's still widely used today (Kim and Kim, 2021; J. Li et al., 2021; Salazar et al., 2016, 2022).

Algorithm 1 ϵ-NSGA-II

Require: Problem definition, ϵ, P_{size}, λ, nfe_{max}, Variation Operators Op_{fixed}
Ensure: Approximation of the Pareto front A

$P \leftarrow$ InitialisePopulation(P_{size})
EvaluateSolutions(P)
$nfe \leftarrow P_{size}$
$A \leftarrow$ InitialiseEpsilonArchive(ϵ)
UpdateEpsilonArchive(A, P)
while $nfe < nfe_{max}$ **do**
    $P' \leftarrow$ SelectParentsTournament(P) ▷ Based on rank
    $O \leftarrow$ ApplyVariation(P', Op_{fixed})
    EvaluateSolutions(O)
    $nfe \leftarrow nfe + |O|$
    $R \leftarrow P \cup O$
    Ranks $\leftarrow$ NonDominatedSort(R)
    $P \leftarrow$ SelectNextGeneration$(R, P_{size}, \textit{Ranks})$ ▷ Truncate based on rank
    UpdateEpsilonArchive(A, P) ▷ Add new non-dominated solutions from P
    if CheckRestartCondition(A, P, nfe) **then** ▷ Based on ϵ-progress or other criteria
        $P_{target} \leftarrow \max(P_{min}, \min(P_{max}, \lambda \cdot |A|))$ ▷ $|A|$ is archive size
        $P \leftarrow$ PerformRestart(A, P_{target}) ▷ Reinitialise using Archive + mutated/random solutions
        EvaluateSolutions$(P \setminus A)$ ▷ Evaluate only new solutions
        $nfe \leftarrow nfe + |P \setminus A|$
        $P_{size} \leftarrow P_{target}$
        UpdateEpsilonArchive(A, P)
return A

Borg

Borg is a modern steady-state MOEA, sharing a number of features with ϵ-NSGA-II while also featuring additional ones like ϵ-progress and adaptive operator selection (Hadka and Reed, 2013). Just like ϵ-NSGA-II it starts by initialising a population of random individuals and setting up an ϵ-archive with all the non-dominated solutions. It randomly selects one parent from the archive and uses tournament selection and non-dominated ranking to select the other parents. Borg features adaptive tournament sizing, setting it as a percentage of the population size. It then proceeds to apply a variation operator. Unlike with ϵ-NSGA-II however this variation operator is not static. Since different operators perform better or worse depending on the specific problem, Borg logs the success of each operator and adjusts the probability of selecting that operator accordingly. Given $K > 1$ operators, it maintains the probabilities $\{Q_1, Q_2, \ldots, Q_k\}, Q_i \in [0, 1]$ of applying each operator to produce the offspring, initialising them to $Q_i = 1/K$. Borg then periodically updates these probabilities by counting the number of solutions in the ϵ-archive produced by each operator $\{C_1, C_2, \ldots, C_k\}$ and updating Q_i by

$$Q_i = \frac{C_i + \zeta}{\sum_{j=1}^{K}(C_j + \zeta)} \tag{2.6}$$

Here $\zeta > 0$ is a constant to prevent probabilities from reaching 0, and is set to one at default. Borg usually selects from the following variation operators; SBX, DE, PCX, SPX, UNDX and UM. When using one of the first five operators Borg also applies PM. All offspring is evaluated and added to the ϵ-archive if they dominate the existing individual in an ϵ-grid cell. Borg also features a restart system. It introduces the concept of ϵ-progress, tracking search improvement by counting every time a new improved solution reaches a better ϵ-grid cell and is added to the ϵ-archive. After a set time window without any ϵ-progress a restart is triggered. Borg's restart system is very similar to that of ϵ-NSGA-II but differs in how it fills up the new population. It draws solutions from the archive and fills up remaining spots by applying UM to archived solutions.

The biggest difference between Borg and ϵ-NSGA-II is Borg's asynchronous steady-state nature. It applies a master-slave model, where one CPU node handles the population and archive management

as well as the operator selection. All evaluations are done by other worker nodes, which immediately pass the results to the master node after finishing the evaluation and are handed new offspring in return. This prevents nodes from idling, maximising CPU utilisation (Hadka et al., 2013). One of the main advantages is that this also allows for massive scaling, making Borg an excellent MOEA for highly-dimensional and complex problems and useful in a lot of current research (Giuliani et al., 2016; Marangoni et al., 2021; Salazar et al., 2024).

Algorithm 2 Borg

Require: Problem definition, ϵ, P_{size}, λ, nfe_{max}, Operator Set Ops
Ensure: Approximation of the Pareto front A

$P \leftarrow$ InitialisePopulation(P_{size})
EvaluateSolutions(P)
$nfe \leftarrow P_{size}$
$A \leftarrow$ InitialiseEpsilonArchive(ϵ)
UpdateEpsilonArchive(A, P) ▷ Add initial population to archive
$OperatorProbabilities \leftarrow$ InitialiseOperatorProbabilities(Ops)
$MutationProb \leftarrow$ InitialMutationProbability ▷ For restarts
while $nfe < nfe_{max}$ **do**
  if $|A| \leq 1$ **then**
    $P' \leftarrow$ SelectParentsTournament$(P,$ arity$)$ ▷ Select arity parents from P
  else
    $P'_{pop} \leftarrow$ SelectParentsTournament$(P,$ arity $- 1)$ ▷ Select arity-1 from P
    $P'_{arch} \leftarrow$ SelectRandom$(A, 1)$ ▷ Select 1 random from A
    $P' \leftarrow P'_{pop} \cup P'_{arch}$
  Shuffle(P')
  $op_{selected} \leftarrow$ SelectOperator$(Ops, OperatorProbabilities)$
  $O \leftarrow$ ApplyVariation$(P', \{op_{selected}\})$ ▷ Generate offspring batch
  EvaluateSolutions(O)
  $nfe \leftarrow nfe + |O|$
  for each $child \in O$ **do**
    DominatesIndices, *IsDominated* $\leftarrow$ CompareDominance$(child, P)$
    if $|$*DominatesIndices*$| > 0$ **then** ▷ Child dominates some in P
      $RemoveIndex \leftarrow$ SelectRandom(*DominatesIndices*, 1)
      RemoveFromPopulation$(P, RemoveIndex)$
      AddToPopulation$(P, child)$
    else if not *IsDominated* **then** ▷ Child is non-dominated w.r.t P
      $RemoveIndex \leftarrow$ SelectRandom$(1..|P|, 1)$
      RemoveFromPopulation$(P, RemoveIndex)$
      AddToPopulation$(P, child)$
    $Improved \leftarrow$ UpdateEpsilonArchive$(A, \{child\})$ ▷ Archive handles internal logic
    UpdateOperatorProbabilities$(OperatorProbabilities, op_{selected}, Improved)$ ▷ Based on archive improvement
  if CheckRestartCondition(A, nfe) **then** ▷ Based on ϵ-progress & potentially pop size deviation
    $P_{target} \leftarrow \max(P_{min}, \min(P_{max}, \lambda \cdot |A|))$ ▷ $|A|$ is archive size
    $MutationProb \leftarrow$ AdjustMutationProbability(*restart_history*) ▷ Adjust based on consecutive restarts
    $P \leftarrow$ PerformRestart$(A, P_{target}, MutationProb)$ ▷ Reinitialise P using Archive + mutated solutions
    EvaluateSolutions$(P \setminus A)$ ▷ Evaluate only new solutions
    $nfe \leftarrow nfe + |P \setminus A|$
    $P_{size} \leftarrow P_{target}$ ▷ Update population size target
    UpdateEpsilonArchive(A, P) ▷ Ensure restarted pop is in archive
return A

Generational Borg

Generational Borg can be considered to be a hybrid between ϵ-NSGA-II and Borg. Not a lot of literature has been written on Generational Borg, but this research adheres to the implementation found in the 'EMA_workbench' Python library (Kwakkel and contributors, 2024).

Generational Borg shares almost all of the features found in regular Borg. It makes use of the same auto-adaptive approaches for operator selection, population sizing and population restarts. The difference is that it does not make use of the steady-state approach, but uses a generational population update. This means that, intuitively, Generational Borg is less suited for parallel execution. However, comparing these MOEAs thus allows for an interesting quantification of this difference when applied to different problems. Additionally, as mentioned, including Generational Borg in this study serves the crucial purpose of enabling a better distinction between the MOEA performance contributions of auto-adaptivity versus the population update scheme.

Algorithm 3 Generational Borg

Require: Problem definition, ϵ, P_{size}, λ, nfe_{max}, Operator Set Ops
Ensure: Approximation of the Pareto front A

$P \leftarrow$ InitialisePopulation(P_{size})
EvaluateSolutions(P)
$nfe \leftarrow P_{size}$
$A \leftarrow$ InitialiseEpsilonArchive(ϵ)
UpdateEpsilonArchive(A, P)
$OperatorProbabilities \leftarrow$ InitialiseOperatorProbabilities(Ops)
while $nfe < nfe_{max}$ **do**
    $P' \leftarrow$ SelectParentsTournament(P) ▷ Based on rank
    $op_{selected} \leftarrow$ SelectOperator$(Ops, OperatorProbabilities)$ ▷ Select operator based on probabilities
    $O \leftarrow$ ApplyVariation$(P', \{op_{selected}\})$ ▷ Use selected operator for offspring batch
    EvaluateSolutions(O)
    $nfe \leftarrow nfe + |O|$
    $R \leftarrow P \cup O$
    Ranks $\leftarrow$ NonDominatedSort(R)
    $P \leftarrow$ SelectNextGeneration$(R, P_{size}, $*Ranks*$)$ ▷ Truncate based on rank
    $Improved \leftarrow$ UpdateEpsilonArchive(A, P) ▷ Add new non-dominated solutions from P
    UpdateOperatorProbabilities$(OperatorProbabilities, op_{selected}, Improved)$ ▷ Update based on archive improvements
    if CheckRestartCondition(A, nfe) **then** ▷ Based on ϵ-progress
        $P_{target} \leftarrow \max(P_{min}, \min(P_{max}, \lambda \cdot |A|))$ ▷ $|A|$ is archive size
        $P \leftarrow$ PerformRestart(A, P_{target}) ▷ Reinitialise using Archive + mutated/random solutions
        EvaluateSolutions$(P \setminus A)$ ▷ Evaluate only new solutions
        $nfe \leftarrow nfe + |P \setminus A|$
        $P_{size} \leftarrow P_{target}$ ▷ Update population size
        UpdateEpsilonArchive(A, P)
return A

Property	ϵ-NSGA-II	Borg	Generational Borg
Population Management	Generational	Steady-State	Generational
Parallel Execution	Synchronous	Asynchronous	Synchronous
Crossover Mechanism	SBX	Auto-adaptive	Auto-adaptive
Mutation Mechanism	PM	Auto-adaptive	Auto-adaptive
Restart Mechanism	Archive + Random Individuals	Archive + Mutated Archive Individuals	Archive + Mutated Archive Individuals

Table 2.1: Key MOEA Properties

3

Research Question

The following section will use the findings from chapter 2 to synthesise a knowledge gap in the current scientific MOEA literature and formulate a research question to address this gap. Lastly, it will briefly discuss how the implications of this thesis fall within the EPA curriculum.

3.1. Knowledge Gap

The theoretical advantages and disadvantages of particular MOEA architectures, being steady-state or generational, have been readily explored (Hadka and Reed, 2013; Zăvoianu et al., 2013). Additionally, a lot of empirical research with regards to ϵ-NSGA-II and Borg performance exists already as well, often demonstrating Borg's excellent scalability and the effectiveness of applying ϵ-dominance (Hadka and Reed, 2015; Salazar et al., 2016). However, a clear research gap remains. A direct empirical comparison of ϵ-NSGA-II, Borg and Generational Borg, quantifying the trade-offs between convergence dynamics, solution quality and computational efficiency, especially in the context of optimising a complex and computationally demanding model like the JUSTICE IAM, does not yet exist.

Even though the theoretical benefits of Borg's asynchronous steady-state approach for scalability and the adaptive techniques used by both Borg and Generational Borg are known, there is no clear empirical quantification of how these translate into actual performance increases on intensive IAMs yet. How does the theoretical parallel scalability advantage of Borg compare to synchronous generational MOEAs, like ϵ-NSGA-II and Generational Borg, when considering the quality of the Pareto front approximations and the time and resources needed to reach them? Furthermore, the position of Generational Borg is interesting compared to both Borg as well as ϵ-NSGA-II. Do its additional adaptive features make it significantly outperform ϵ-NSGA-II despite facing the same generational issues with regards to parallel scalability, and how does it hold up to the steady-state Borg? Researching this on a complex and computationally heavy model like JUSTICE is crucial to gain an understanding of how different MOEA architectures perform under realistic conditions and models relevant to climate-economy modelling.

This is an important gap to address. Insights in changing MOEA dynamics due to parallelisability can yield significant benefits. It aids researchers in determining the correct choice of MOEA for specific problems. Given the differences in their efficiency, costs, suitability for distributed computing and ability to handle complexity, significant time, effort and costs can be saved by having a clear rationale to choose a particular MOEA based on the intended application.

3.2. Research Question

To address the identified knowledge gap, this thesis aims to answer the following question;

How and why do ϵ-NSGA-II, Borg and Generational Borg differ regarding their computational efficiency, convergence dynamics and solution quality when optimising a demanding model such as JUSTICE using the EMODPS framework?

To answer this question the following sub-questions have been formulated;

1. ***How do ϵ-NSGA-II, Borg and Generational Borg differ in convergence performance metrics and solution quality?***

   Answering this question is fundamental to being able to answer the main research question. It provides the baseline effectiveness of each MOEA in achieving the prime goal of finding high quality and diverse solutions for the JUSTICE model within in a certain computational and temporal budget.

2. ***Are the observed differences in JUSTICE also observed in benchmark problems like DTLZ2 and DTLZ3?***

   The answer to this question will provide additional insights in the generalisability of the JUSTICE findings. Comparing the MOEA performance on benchmark problems with that on the JUSTICE IAM can help determine if the observed differences have to do with fundamental algorithmic properties or specific interactions with the IAM and benchmark problems.

3. ***Does increasing or decreasing the number of cores used in optimisation influence the differences between the different MOEA performances?***

   This question needs to be answered to address the computational efficiency dimension of the main research question. By testing the theoretical expectations with regards to scalability discussed in chapter 2 it quantifies how effectively each MOEA can utilise an increase in computational resources, which is very helpful to know for evaluating their feasibility and cost-effectiveness especially on HPC systems.

4. ***What recommendation with regards to MOEA choice can be made based on problem characteristics and available resources?***

   Answering this question aims to synthesise the empirical findings from all previous research questions and provide an actionable advise for researchers selecting MOEAs for similar simulation and optimisation problems.

3.3. EPA Relevance

This research is highly relevant for and fits well in the domain of Engineering & Policy Analysis. The EPA philosophy advocates for a quantitative, analytical and modelling approach to understand and evaluate complex societal issues at the intersection of technology, policy and management. Using JUSTICE and the EMODPS framework this research heavily depends on advanced modelling and simulation techniques, both central to the EPA curriculum. It addresses the complex socio-technical problem of climate-economy policy optimisation, characterised by conflicting interests like economic and environmental objectives. The objective of this thesis is to research simulation tools like MOEAs to improve the quality and efficiency of decision-making support for complex policy problems characterised by deep uncertainty.

4

Method

This chapter first outlines the general design and goal of this study. It formally defines the three problems that will be optimised and discusses which MOEAs and which configurations will be used to do so. The following section outlines the exact experiments that will be performed. Next, a number of performance metrics that will aid in answering the research question are defined and discussed. Lastly, the limitations this research is subject to are discussed and recommendations to tackle these limitations in future research are made.

4.1. Research Design

The core of this research is built up out of a series of computational experiments that are run on the DelftBlue HPC. The experiments are run under a number of different controlled conditions, primarily varying the number of used CPU cores, and evaluate and compare the performance of the three selected and previously discussed MOEAs; ϵ-NSGA-II, Borg and Generational Borg. Each of the MOEAs will be run on three different problems, which can be divided into two main categories. First, the JUSTICE IAM problem will be optimised using the EMODPS framework. Second and third, the widely used benchmark problems DTLZ2 and DTLZ3 will be optimised. The latter two problems have known characteristics and analytical solutions and are used to contextualise the observed performance results from JUSTICE. DTLZ2, a simple unimodal problem, tests the baseline convergence ability of each MOEA. DTLZ3, a deceptive multi-modal problem, introduces a far greater difficulty than DTLZ2 and tests the MOEA's ability to avoid getting stuck local optima. By comparing MOEA performance across all problems, it is possible to better isolate whether the performance differences on JUSTICE stem from fundamental convergence issues or difficulties with complex and deceptive solution landscapes.

The experiments aim to quantify the trade-offs mentioned in the main research question, comparing the algorithms across three main dimensions. Firstly, the computational efficiency, focussing on parallel scalability of each of the MOEAs. Secondly, the convergence dynamics, focussing on how effectively each MOEA moves towards to the Pareto front during an optimisation run. Thirdly, the solution quality, focussing on the quality of the final Pareto front approximated by the MOEA compared to the best-known Pareto front. The performance with regards to each of these dimensions will be measured by a set of performance metrics that will be further detailed in section 4.5.

4.2. Problem Formulations

This section provides a specific definition of the MOPs used in the experiments. As mentioned, two different categories of MOPs are used. Firstly, the primary problem Justice Universality Spatial Temporal Integrated Climate Economy (JUSTICE) IAM, employing the EMODPS framework, is discussed. Next, definitions of the analytical benchmark problems DTLZ2 and DTLZ3 are provided.

4.2.1. JUSTICE

JUSTICE is a novel IAM designed as a simpler and modular model inspired by the FAIR and RICE IAMs, especially suited to act as a surrogate model for more complex IAMs for eliciting normative insights (Biswas et al., 2023). One of its key features is its modularity, which allows for researching the effects of different assumptions with regards to the economic model, damage functions and social welfare functions used in the model. However, since the purpose of this thesis lies on comparing MOEA performance, only one specific deterministic JUSTICE configuration is used as, outlined in table 4.1.

Parameter	Value
RBFs	4
Inputs per RBF	2
RBF type	Squared exponential
Welfare function	Utilitarian
Economy type	Neoclassical
Damage function	Kalkuhl
Abatement type	Enerdata
Climate reference scenario	SSP245
Climate ensemble size	40
Number of regions	57

Model structure parameters

Parameter	Value
Start year	2015
End year	2300
Data timestep	5
Timestep	1
Emission control start year	2025
Temperature year of interest	2100

Time-related parameters

Table 4.1: Parameter settings for the JUSTICE model

The goal is to optimise the following four objectives using ϵ-values based on objective scale and previous experiments in Biswas et al., 2025;

- **Welfare**, to be minimised using $\epsilon = 0.01$.
- **Years above threshold**, to be minimised using $\epsilon = 0.25$.
- **Welfare loss damage**, to be maximised using $\epsilon = 10$.
- **Welfare loss abatement**, to be maximised using $\epsilon = 10$.

The directions of optimisation for welfare, welfare loss damage and welfare loss abatement might seem counter-intuitive. However, the social welfare function used in the model returns a negative value, flipping the direction of optimisation for these objectives.

The RBF network acts as the policy function. In JUSTICE the RBF network features four basis functions and maps the system state to a set of control actions by taking two inputs, *'scaled_temperature'* and *'scaled_difference'* (change in temperature). These are used to determine the *'emissions_control_rate'* for each of the 57 regions, resulting in 57 final outputs. This mapping is defined by the RBF parameters, which are the decision variables eventually optimised by the MOEA. This can be represented as policy $\pi_{\boldsymbol{\theta}}(s)$ with a parameter vector $\boldsymbol{\theta}$ containing;

- **Center parameters**, $n_{centers} = n_{RBFs} \cdot n_{inputs} = 4 \cdot 2 = 8$ parameters. Bounded by [-1,1].
- **Radii parameters**, $n_{radii} = n_{RBFs} \cdot n_{inputs} = 4 \cdot 2 = 8$ parameters. Bounded by [0.0004,1] for numerical stability.
- **Weight parameters**, $n_{weights} = n_{RBFs} \cdot n_{regions} = 4 \cdot 57 = 228$ parameters. Bounded by [0.0004,1] for numerical stability.
- **Total parameters**, $n_{param} = 8 + 8 + 228 = 244$ parameters.

The lower bounds for the radii and weight parameters were set to prevent division by zero errors that occurred when the lower bound was set to zero. The value was determined through multiple trial and error runs. The MOEA starts by initialising a candidate $\boldsymbol{\theta}$, defining $\pi_{\boldsymbol{\theta}}(s)$. JUSTICE is simulated over the entire time horizon using this policy, resulting in final objective values for the four previously

mentioned objectives. The MOEA evaluates these values and searches for an improved θ, after which the cycle repeats.

4.2.2. DTLZ2

DTLZ2 is a benchmark problem from the DTLZ test suite, often used in MOEA research (Deb, Thiele, et al., 2005). An advantage of DTLZ2 compared to many other benchmark problems is its ability to generate a problem with an arbitrary number of objectives. Since it is an analytically solvable problem, the true Pareto front is known, allowing for a clear evaluation of the MOEA and its performance. Since the number of objectives used in JUSTICE is four, the number of objectives in the DTLZ2 problem (m) is also set to $m = 4$ with ϵ-values 0.05. This leads to the following definition;

$$\left.\begin{aligned}
&\text{min} \quad f_1(\mathbf{x}) = (1 + g(\mathbf{x}_m)) \cos\left(x_1 \frac{\pi}{2}\right) \cos\left(x_2 \frac{\pi}{2}\right) \cos\left(x_3 \frac{\pi}{2}\right), \\
&\text{min} \quad f_2(\mathbf{x}) = (1 + g(\mathbf{x}_m)) \cos\left(x_1 \frac{\pi}{2}\right) \cos\left(x_2 \frac{\pi}{2}\right) \sin\left(x_3 \frac{\pi}{2}\right), \\
&\text{min} \quad f_3(\mathbf{x}) = (1 + g(\mathbf{x}_m)) \cos\left(x_1 \frac{\pi}{2}\right) \sin\left(x_2 \frac{\pi}{2}\right), \\
&\text{min} \quad f_4(\mathbf{x}) = (1 + g(\mathbf{x}_m)) \sin\left(x_1 \frac{\pi}{2}\right), \\
&\text{with} \quad g(\mathbf{x}_m) = \sum_{i=m}^{n} \left(x_i - \frac{1}{2}\right)^2, 0 \leq x_i \leq 1 \forall i \in \{1, \ldots, n\}
\end{aligned}\right\} \tag{4.1}$$

Here $\mathbf{x} = (x_1, \cdots, x_n)$ is the decision variable vector and $\mathbf{x}_m = (x_m, \ldots, x_n)$. The number of decision variables is set as $n_{var} = m + k - 1$. Following the convention from Deb, Thiele, et al. (2005) we set $k = 10$, so $n_{var} = 13$. The Pareto front for DTLZ2 is covered by solutions where $g(\mathbf{x}_m) = 0$, so $x_i = \frac{1}{2}$ for $i = m, \ldots, n$. Here the objective values satisfy $\sum_{i=1}^{4} f_i^2(\mathbf{x}) = 1$, forming a smooth concave global Pareto front representing the first orthant of a unit hypersphere in the 4-dimensional objective space. This is one of the reasons DTLZ2 is included in this research. With its smooth concave Pareto front and unimodal distance function $g(\mathbf{x}_m)$ it is particularly suited to evaluate a MOEA's basic ability to converge towards a non-linear front and distribute solutions along this front evenly (Huband et al., 2006). By comparing these results to JUSTICE and DTLZ3, both more complex problems, insights into what part of the observed MOEA performance differences is due to basic convergence and diversity mechanics and what part is due to the ability to handle more complex problems are gained.

4.2.3. DTLZ3

DTLZ3 also is a benchmark problem from the DTLZ test suite (Deb, Thiele, et al., 2005). It is very similar to DTLZ2 and also has an analytical solution. However, it introduces a different distance function, making it a multimodal and thus more complex problem. Just like with JUSTICE and DTLZ2 we set the number of objectives to $m = 4$ with ϵ-values 0.05, leading to the following definition;

$$\left.\begin{aligned}
&\text{min} \quad f_1(\mathbf{x}) = (1 + g(\mathbf{x}_m)) \cos\left(x_1 \frac{\pi}{2}\right) \cos\left(x_2 \frac{\pi}{2}\right) \cos\left(x_3 \frac{\pi}{2}\right), \\
&\text{min} \quad f_2(\mathbf{x}) = (1 + g(\mathbf{x}_m)) \cos\left(x_1 \frac{\pi}{2}\right) \cos\left(x_2 \frac{\pi}{2}\right) \sin\left(x_3 \frac{\pi}{2}\right), \\
&\text{min} \quad f_3(\mathbf{x}) = (1 + g(\mathbf{x}_m)) \cos\left(x_1 \frac{\pi}{2}\right) \sin\left(x_2 \frac{\pi}{2}\right), \\
&\text{min} \quad f_4(\mathbf{x}) = (1 + g(\mathbf{x}_m)) \sin\left(x_1 \frac{\pi}{2}\right), \\
&\text{with} \quad g(\mathbf{x}_m) = 100 \left[|\mathbf{x}_m| + \sum_{i=m}^{n} \left(\left(x_i - \frac{1}{2}\right)^2 - \cos\left(20\pi \left(x_i - \frac{1}{2}\right)\right)\right)\right], 0 \leq x_i \leq 1 \forall i \in \{1, \ldots, n\}
\end{aligned}\right\} \tag{4.2}$$

Just like for DTLZ2 $\mathbf{x} = (x_1, \cdots, x_n)$ is the decision variable vector and $\mathbf{x}_m = (x_m, \ldots, x_n)$. The number of decision variables is set as $n_{var} = m + k - 1$ with $k = 10$, so $n_{var} = 13$. The global Pareto front for DTLZ3 is covered by solutions where $g(\mathbf{x}_m) = 0$, so $x_i = \frac{1}{2}$ for $i = m, \ldots, n$. Here the

objective values satisfy $\sum_{i=1}^{4} f_i^2(\mathbf{x}) = 1$. However, given the cosine in DTLZ3's $g(\mathbf{x}_m)$ function, $3^k - 1$ local minima are introduced (Deb, Thiele, et al., 2005). This multi-modality leads to many local Pareto fronts (while the global front is the same as for DTLZ2). Compared to the unimodal DTLZ2, DTLZ3 allows one to evaluate a MOEA's ability to handle more complex landscapes and avoid local optima. Also incorporating DTLZ3, in addition to JUSTICE and DTLZ2, allows for understanding whether MOEA performance differences are mainly due to basic convergence and diversity mechanics, ability to handle multi-modality or ability to handle even more complex landscapes found in problems like JUSTICE.

4.3. MOEA Implementation and Configurations

This section discusses the main implementation and parameter configurations for each of the three MOEAs in this research; ϵ-NSGA-II, Borg and Generational Borg. They were all implemented using Python 3.9.10 and heavily rely on the EMA_workbench (version 2.5.0) and Platypus (version 1.1.0) libraries (ProjectPlatypus, 2024; Kwakkel and contributors, 2024). All MOEA parameters were left at their default settings set by the libraries they were imported from. This choice was made given that the time scope did not allow for extensive parameter tuning, and leaving the MOEAs at default settings ensures a fair 'baseline' performance comparison reflecting a common use-case scenario where problem-specific parameter tuning does not happen. Additionally, using default parameter settings allows performance differences to be attributed more directly to fundamental architectural differences between the MOEAs, minimising the effects parameter fine-tuning could potentially introduce. The main parameters will be briefly discussed in this section. Appendix A details all the exact MOEA settings. Despite each MOEA having a unique parameter setup, a number of key parameters are shared and set to identical values. These include;

- The **stopping criterion** is set to 70000 function evaluations (NFE) for each of the three problems. Setting a shared stopping criterion ensures the results of each MOEA are based on the same computational budget and can thus be compared fairly. The number 70000 was chosen as this provides the MOEAs sufficient budget to reach meaningful results in a complex model like JUSTICE.
- The same ϵ**-values** are used for each MOEA. Based on previous experimentation with JUSTICE in Biswas et al., 2025), the JUSTICE ϵ-values are set too;

Objective	ϵ**-value**
Welfare	0.01
Years above threshold	0.25
Welfare loss damage	10
Welfare loss abatement	10

  For both the DTLZ2 and DTLZ3 problems the ϵ-values are set to 0.05 for each of the four objectives.
- The **initial population size** is set to 100 for each MOEA. This is a commonly used number that balances initial exploration with computational cost.
- The **adaptive population sizing parameters** are kept at default settings and also thus similar for each MOEA. Population-to-archive ratio $\lambda = 4$, the minimum population size $P_{min} = 10$, the maximum population size $P_{max} = 10000$ and the mutator is UM.

4.4. Experimental Setup

This section will outline the specific experiments conducted to evaluate the MOEAs performances and the computational environment in which they were conducted. Table 4.2 presents a summary of the performed experiments.

4.4.1. Computational Environment

All experiments were run on high-memory nodes of the DelftBlue HPC. The nodes contain 48 cores, provided by two 24-core Intel Xeon E5-6248R 24C 3.0GHz CPUs, with 750GB of normal memory and

a 150GB SSD. After running the experiments, the results were analysed on a local machine.

4.4.2. Experiments

Each of the three MOEAs is used to optimise all three problems. To answer research sub-question 3, this is done for three different core counts. For the core counts the values 16, 32 and 48 were picked. Using the master-slave approach discussed in section 2.3.2 this translates to runs with 1 master and 15 workers, 1 master and 31 workers and 1 master and 47 workers. These three values were specifically chosen for a number of reasons. Given the time scope of this thesis and a single high-memory node containing 48 cores, the choice was made not to incorporate any Message Passing Interface (MPI) setup in the experiment code and use 48 cores as the maximum. Additionally, Hadka et al. (2013) have shown that for a master-slave setup higher core counts not necessarily translate to proportional increases in convergence. Depending on the time needed for a single function evaluation, Borg's efficiency peaks between 16 and 256 cores. To easily see the effects of an increasing or decreasing core count a linear increment was chosen, leading to the choice of 16, 32 and 48 cores.

All experiments were run for 70000 NFE. This number was chosen based on the available time resources, and previous JUSTICE runs made by P. Biswas using ϵ-NSGA-II showing convergence around 50000 NFE. Lastly, each experiment was run five times, using different seeds. Ideally, the number of seeds would be higher, as it not only increases the validity of the results but also leads to a more robust reference set for JUSTICE (as the 'true' Pareto front is formed by the best solutions of all runs). However, due to time and computational constraints only five seeds were run.

	JUSTICE	**DTLZ2**	**DTLZ3**
MOEAs	ϵ-NSGA-II, Borg, Generational Borg	ϵ-NSGA-II, Borg, Generational Borg	ϵ-NSGA-II, Borg, Generational Borg
Objectives	4	4	4
NFE	70000	70000	70000
Core Counts	16, 32, 48	16, 32, 48	16, 32, 48
Seeds	12345, 23403, 39349, 60930, 93489	12345, 23403, 39349, 60930, 93489	12345, 23403, 39349, 60930, 93489

Table 4.2: Experiment Summary

4.5. Performance Metrics

To quantify the trade-offs between computational efficiency, convergence dynamics and solution quality mentioned in the research questions, a number of often used performance metrics in MOEA research have been selected. All metrics are computed for independent optimisation runs. This section provides a detailed discussion of each metric. This specific group of metrics was chosen to ensure a holistic assessment of the performance. The hypervolume indicator provides a measure of convergence and diversity, while the generational distance and additive epsilon indicator both specifically quantify how close the solution approaches the reference Pareto front. The spacing metric is used to give additional insights with regards to diversity. Lastly, the epsilon progress and archive size provide insights into the MOEA's ability to effectively explore the solution space and find new solutions over time.

4.5.1. Hypervolume

First off, the hypervolume is computed. The hypervolume is a widely-used metric that quantifies the quality and diversity of the obtained convergence (i.e. the Pareto front generated by the MOEA) (Hadka and Reed, 2013; Ishibuchi et al., 2010; Yen and He, 2013). It measures the m-dimensional volume of the objective space that is weakly dominated by the obtained Pareto front (i.e. the final ϵ-archive, which we will call A) and bounded by a reference point which we will call $\mathbf{z}^{ref}$. This reference point must be set so that it is weakly dominated by all solutions in A. To determine a suitable $\mathbf{z}^{ref}$, the true Pareto front must be known or a reference Pareto front must be set. In case of the DTLZ2 and DTLZ3 problems this true Pareto front is known, simply being the analytical solution to the problem. In case of JUSTICE the true Pareto front is unknown. Here the reference set is defined by grouping all the final archives A generated by each experiment run (so combining them for all problem, MOEA and seed combinations)

and performing another non-dominated sorting. We will call the true Pareto front, or chosen reference set, Z. Next, $\mathbf{z}^{ref}$ is defined as Z's nadir point, meaning the vector composed of the worst value for each objective found across all solutions in Z. The hypervolume needs to be maximised, and can be formally defined as;

$$\begin{aligned} HV(A, \mathbf{z}^{ref}) &= \Lambda \left(\bigcup_{\mathbf{a} \in A} [\mathbf{a}, \mathbf{z}^{ref}] \right) \\ \text{where} \quad [\mathbf{a}, \mathbf{z}^{ref}] &= \{\mathbf{q} \in \mathbb{R}^m | \mathbf{a} \preceq \mathbf{q}, \mathbf{q} \preceq \mathbf{z}^{ref}\} \end{aligned} \tag{4.3}$$

Here Λ denotes the Lebesgue measure, taken from the union of all four-dimensional hyper-rectangles formed by the objective vector $\mathbf{a} = (a_1, a_2, a_3, a_4)^T$ and reference vector $\mathbf{z}^{ref} = (z_1^{ref}, z_2^{ref}, z_3^{ref}, z_4^{ref})^T$ (since we have four objectives for each problem), where $\mathbf{a}$ is a non-dominated objective vector in A and $\mathbf{z}^{ref}$ is the nadir point from Z (Guerreiro et al., 2021).

4.5.2. Generational Distance

The generational distance is a metric also widely used to assess the quality of the obtained Pareto front A (Knowles and Corne, 2002; Yen and He, 2013). The generational distance measures how far the average distance from solutions in A is compared to the true Pareto front or chosen reference set Z. It quantifies how close solutions in A have come to the optimal front Z, where a lower value for generational distance means the found solutions are closer to the optimum. The generational distance needs to be minimised and can be formally defined as;

$$GD(A, Z) = \frac{\left(\sum_{i=1}^{|A|} d_i^p \right)^{\frac{1}{p}}}{|A|} \tag{4.4}$$

Where $|A|$ denotes the number of solutions in A. We set $p = 2$, making d_i denote the Euclidean distance from solution $\mathbf{a}_i \in A$ to its nearest neighbour solution in Z.

4.5.3. Additive Epsilon Indicator

The additive epsilon indicator is another metric to quantify the quality of the obtained Pareto front A (Salazar et al., 2022). However, it takes a different approach than for example the generational distance. Instead, it identifies the minimum uniform distance ϵ with which the entire set A must shift in all objectives to weakly dominate the true or set Pareto front Z, essentially quantifying how much worse A is compared to Z in the worst case. This makes it an effective metric for capturing large gaps in trade-offs. A small additive epsilon indicator indicates less large gaps between A and Z and thus is indicative of better performance. For m objectives the additive epsilon indicator can be defined as;

$$I_{\epsilon+}(A, Z) = \max_{\mathbf{z} \in Z} \left(\min_{\mathbf{a} \in A} \left(\max_{i=1}^{m} (a_i - z_i) \right) \right) \tag{4.5}$$

4.5.4. Epsilon Progress

The epsilon progress is a measure to evaluate the improvement of solutions over time. Each time a new solution sits in a better ϵ-grid cell than the previous solution, epsilon progress occurs. This allows for an easy evaluation of the search progress and potential stagnation (Salazar et al., 2022). Essentially, here higher values are also preferred, as these indicate the search is progressing well and solutions are reaching 'better' ϵ-grid cells.

4.5.5. Archive Size

The archive size simply logs the amount non-dominated solutions at any given stage of the optimisation process. A higher value is preferred, as this indicates a lot of non-dominated solutions are found. It provides information on the diversity of the solutions, while also providing additional insights into the quality of convergence (Salazar et al., 2022).

4.5.6. Hypervolume Efficiency
The hypervolume efficiency will be computed to evaluate how effective the MOEA is in converting computational effort into improving the Pareto front. It calculates the in- or decrease in hypervolume over a unit NFE, essentially being the gradient when plotting the hypervolume over NFE.

4.5.7. Spacing
The spacing is a metric which quantifies how evenly spread out the solutions are along the Pareto front. It evaluates the uniformity of how solutions are spaced out with respect towards their nearest neighbours, with a lower value indicating a more uniform distribution of solutions. It complements other convergence metrics like hypervolume and generational distance by providing additional diversity context. This study uses the implementation found in the Platypus library, which follows as;

$$S(A) = \sqrt{\frac{1}{|A| - 1} \sum_{i=1}^{|A|} (d_i - \bar{d})^2} \tag{4.6}$$

Where $|A|$ again indicates the number of solutions in the obtained Pareto front A, d_i indicates the Euclidean distance from solution i to its nearest neighbour and $\bar{d}$ is the average of these distances (ProjectPlatypus, 2024).

4.6. Limitations
Despite efforts to minimise them, this study still is subject to a number of limitations which this section will outline in more detail. First off, a limitation is the small amount of seeds used. Due to the time constraints of this thesis and the time it takes to complete experiment runs even on a HPC such as DelftBlue, all experiments were only run for five seeds. Due to the stochastic nature of MOEAs, this does reduce the confidence in the generalisability of the results slightly. Additionally, given the way the reference set was created for JUSTICE (taking the best solutions from all performed runs), the small amount of seeds leads to a very substantial possibility that there are better solutions out there and that the performance metrics were computed relative to a sub-optimal Pareto front. Future research, with more time resources, could perform a similar study but extending it for a larger number of seeds to confirm or disprove the findings from this study.

Another area for future work to improve on is the limited computational setup used in this study. Again, due to time constraints, this study only tested the problems for three differing core counts. It is not a given that the findings from this study also generalise to other core counts, or different HPC setups in general. It would be interesting to see in future work if the results obtained here also hold for substantially higher core counts, or a higher amount of different nodes.

Similar to the limited computational setup, the findings also result from fixed problem formulations and parameter settings. Only the two DTLZ benchmark problems were tested, whereas more benchmark problems with differing characteristics exist, and both the DTLZ problems were only tested with four objectives. More importantly, JUSTICE was only run with one configuration. A different RBF setup, different welfare function, different economy type, different damage function, different abatement type, different climate reference scenario or a larger climate ensemble size all could substantially influence the model dynamics and results. Similarly, the MOEAs were only tested with their default settings. Future research should therefore test a larger amount of different model and MOEA configurations.

Lastly, whereas 70000 NFE is a substantial NFE and previous research has shown it to generally be enough for MOEAs to display a certain degree of convergence, there is no absolute certainty that the MOEAs are close to having explored the complete objective space they are capable of exploring. Future research could run the same experiments for an even higher NFE, exploring whether all MOEAs have reached full convergence.

5

Results

This chapter presents the empirical results obtained by the experiment runs. It discusses the performance metrics and their development during the optimisation process. Additionally development trends over wall-clock time will be shortly discussed. However, a significant limitation is present and will be briefly discussed here. Due to the initial experiment setup utilising a maximum of 48 cores, and a high-memory node of DelftBlue containing exactly 48 cores, no MPI was implemented in the code. Unfortunately, no exclusive node access was requested during experiment runs, and the DelftBlue Slurm Workload Manager divided experiments over one to four nodes for different runs. This introduces uncontrollable variability in exact wall-clock runtimes, which therefore are not directly comparable between runs. However, given the fact that this happened for all core counts, seeds and MOEAs, the observed trends do still offer some general insights, which is why they will presented in this section. Additionally, a number of experiments were rerun with exclusive node access to double check performance metric result validity. These results were similar for non-exclusive and exclusive node runs, strengthening the conviction that obtained performance metric results are valid. Unfortunately, due to time constraints not all experiments could be rerun using exclusive node access.

First, all JUSTICE results will be presented and briefly discussed. Next, a number of interesting results, which aid the contextualisation and interpretability of the JUSTICE results, of the DTLZ2 and DTLZ3 problems will be presented. All DTLZ2 and DTLZ3 results that are not discussed in the main text can be found in appendix C.

5.1. JUSTICE

JUSTICE, with its high-dimensional and complex solution space, serves as the primary problem for evaluating the three MOEAs. Its performance metrics will be plotted over NFE, and tables of final results averaged out over seeds will be presented. Additionally, the computational efficiency and scalability observations will be presented. More detailed results per seed can be found in appendix B.

5.1.1. Solution Quality and Convergence Dynamics

Hypervolume Figure 5.1 presents the hypervolume obtained by each MOEA for JUSTICE plotted over NFE. To reiterate, the hypervolume values were calculated using a global reference set made up out of all the non-dominated solutions contributed by each run. Therefore, these plots indicate to what degree individual runs cover the total discovered optimal Pareto front. All MOEAs show expected behaviour, increasing in hypervolume at a quick rate at the start, and slowing down after approximately 10000 NFE. ϵ-NSGA-II and Generational Borg obtain relatively similar hypervolume values on average, though ϵ-NSGA-II shows slightly bigger inter-seed differences. Borg clearly achieves the highest hypervolume as well as still displaying the highest rate of hypervolume increase even at 70000 NFE.

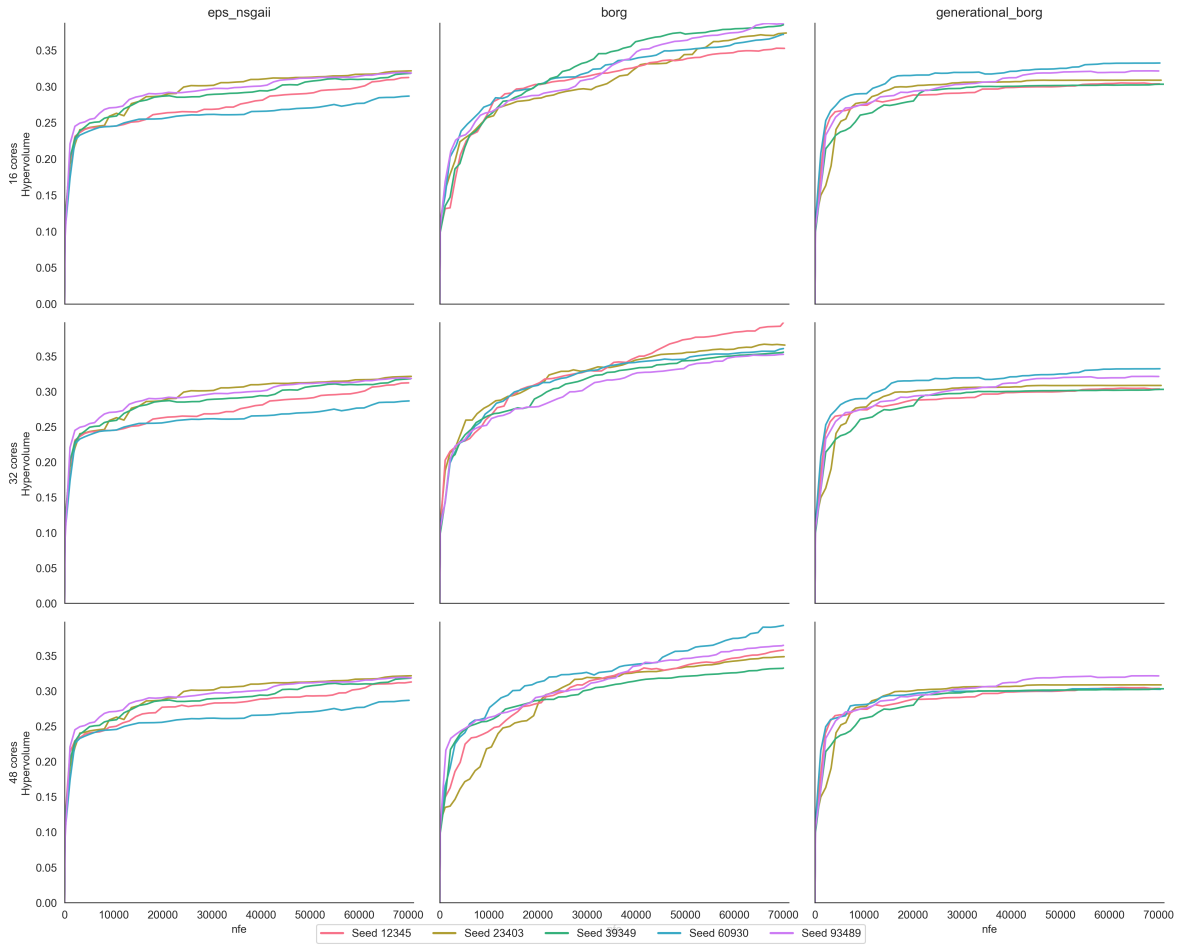

Figure 5.1: JUSTICE Hypervolume

Table 5.1 shows the hypervolume values averaged out over all five seeds, for each core count as well as the average hypervolume over all core counts for each MOEA. This gives a more detailed insight into the final hypervolume obtained by each MOEA. Looking at the final hypervolume values in table 5.1, it is clear that Borg achieves a significantly higher hypervolume than ϵ-NSGA-II and Generational Borg. As already mentioned, ϵ-NSGA-II and Generational Borg achieve very similar hypervolume values, even being identical for the 16- and 32-core runs. What also stands out is that the core count does not seem to have any substantial effect on the final hypervolume value. The fact that ϵ-NSGA-II and Generational Borg achieve very similar hypervolume values, while Borg achieves substantially higher values, indicates that the exploration of the JUSTICE solution space especially benefits from a steady-state nature. Definitely more so than the advantages that auto-adaptivity introduces.

	Hypervolume		
Core Count	**ϵ-NSGA-II**	**Borg**	**Generational Borg**
16	0.311439	0.374093	0.313802
32	0.311439	0.366613	0.313802
48	0.311510	0.359279	0.307792
Total Average	0.311463	0.366662	0.311799

Table 5.1: Average JUSTICE hypervolume

Generational Distance For the generational distance, figure 5.2 shows logical and similar behaviour for each MOEA. It starts out high and slowly converges to lower values as the search progresses. Borg

seems to consistently get lower results, and a smaller inter-seed difference. The generational distance seems to stabilise the quickest for Generational Borg, though generally being a bit worse than that of ϵ-NSGA-II (except for seed 60930). The plot suggests that Generational Borg's auto-adaptive features make it approach the true Pareto front slightly quicker than ϵ-NSGA-II, but that its lack of steady-state updates limit its final generational distance to roughly the same value as that of the generational ϵ-NSGA-II rather than that of the steady-state Borg.

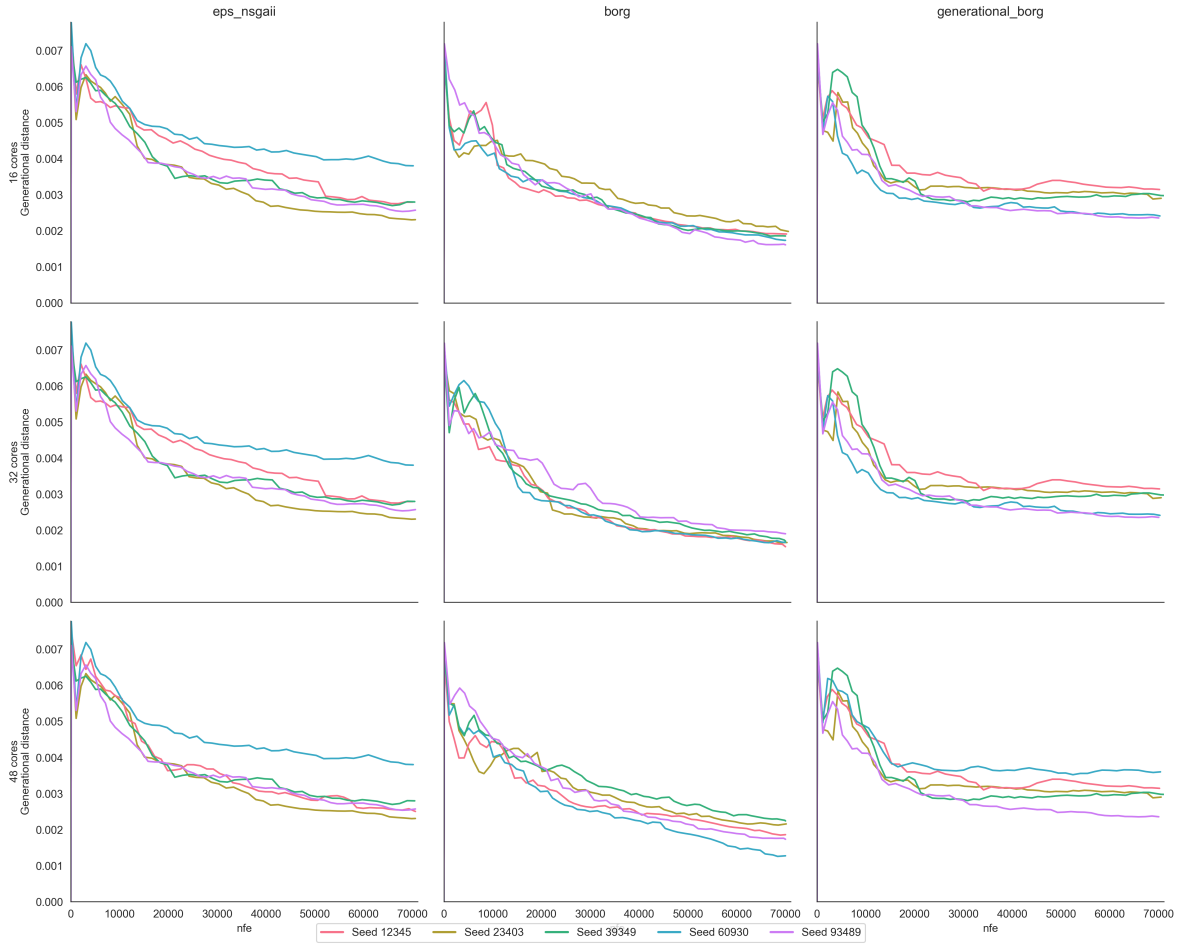

Figure 5.2: JUSTICE Generational Distances

Table 5.2 presents the final average generational distance values. Again, we see that for both ϵ-NSGA-II and Generational Borg the 16- and 32-core experiments return similar values. Borg consistently achieves the lowest and best generational distance values, while differing core counts do not seem to matter for any of the MOEAs.

	Generational Distance		
Core Count	**ϵ-NSGA-II**	**Borg**	**Generational Borg**
16	0.002854	0.001818	0.002758
32	0.002854	0.001692	0.002758
48	0.002796	0.001852	0.002995
Total Average	0.002835	0.001787	0.002837

Table 5.2: Average JUSTICE Generational Distance

Additive Epsilon Indicator Figure 5.3 shows expected behaviour. All MOEAs quickly decrease in their additive epsilon indicator score, converging to low values. It is interesting to see that ϵ-NSGA-II does so the quickest, followed by Generational Borg, while this was reversed for the generational distance. Borg takes slightly more NFE to get to the same epsilon indicator values as the others, but does eventually converge to slightly lower values.

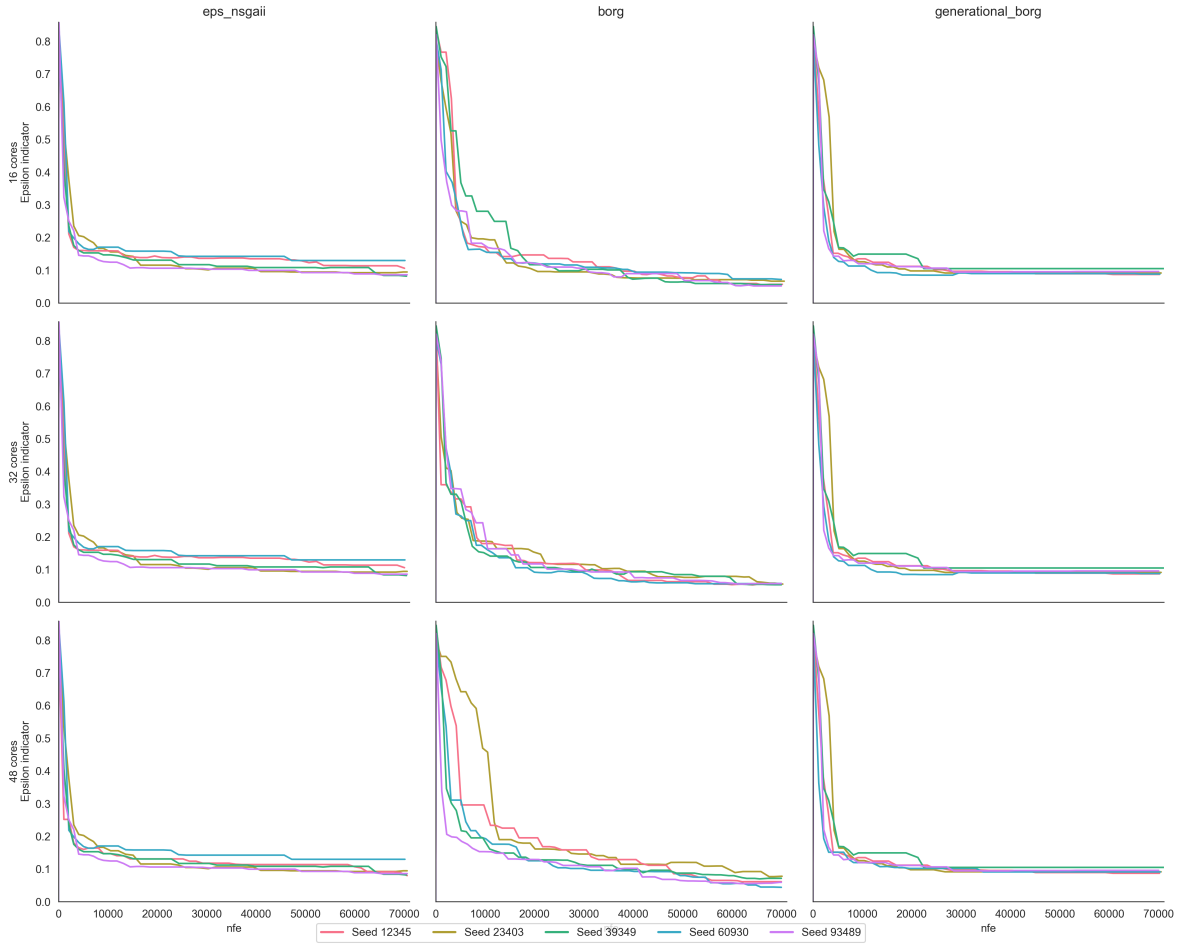

Figure 5.3: JUSTICE Epsilon Indicators

Table 5.3 confirms that Borg achieves the lowest final epsilon indicator values on average. ϵ-NSGA-II and Generational Borg show almost identical final values. Again, one can observe that core count does not significantly impact the epsilon indicator.

	Additive Epsilon Indicator		
Core Count	**ϵ-NSGA-II**	**Borg**	**Generational Borg**
16	0.099516	0.060496	0.093362
32	0.099516	0.055853	0.093362
48	0.094307	0.062682	0.093950
Total Average	0.097780	0.059677	0.093558

Table 5.3: Average JUSTICE Epsilon Indicator

Archive Size Figure 5.4 displays relatively similar behaviour for each MOEA at the start of the search. Rapidly increasing their archive size to roughly 180 after 15000 NFE. After this point Generational Borg is the first to start converging to a value of roughly 220. ϵ-NSGA-II still displays a slightly increasing

trend even after 70000 NFE. Borg again performs best, reaching the biggest archive sizes and still showing an increasing trend for almost all seeds after 70000 NFE. Different core counts do not seem to affect the archive size development.

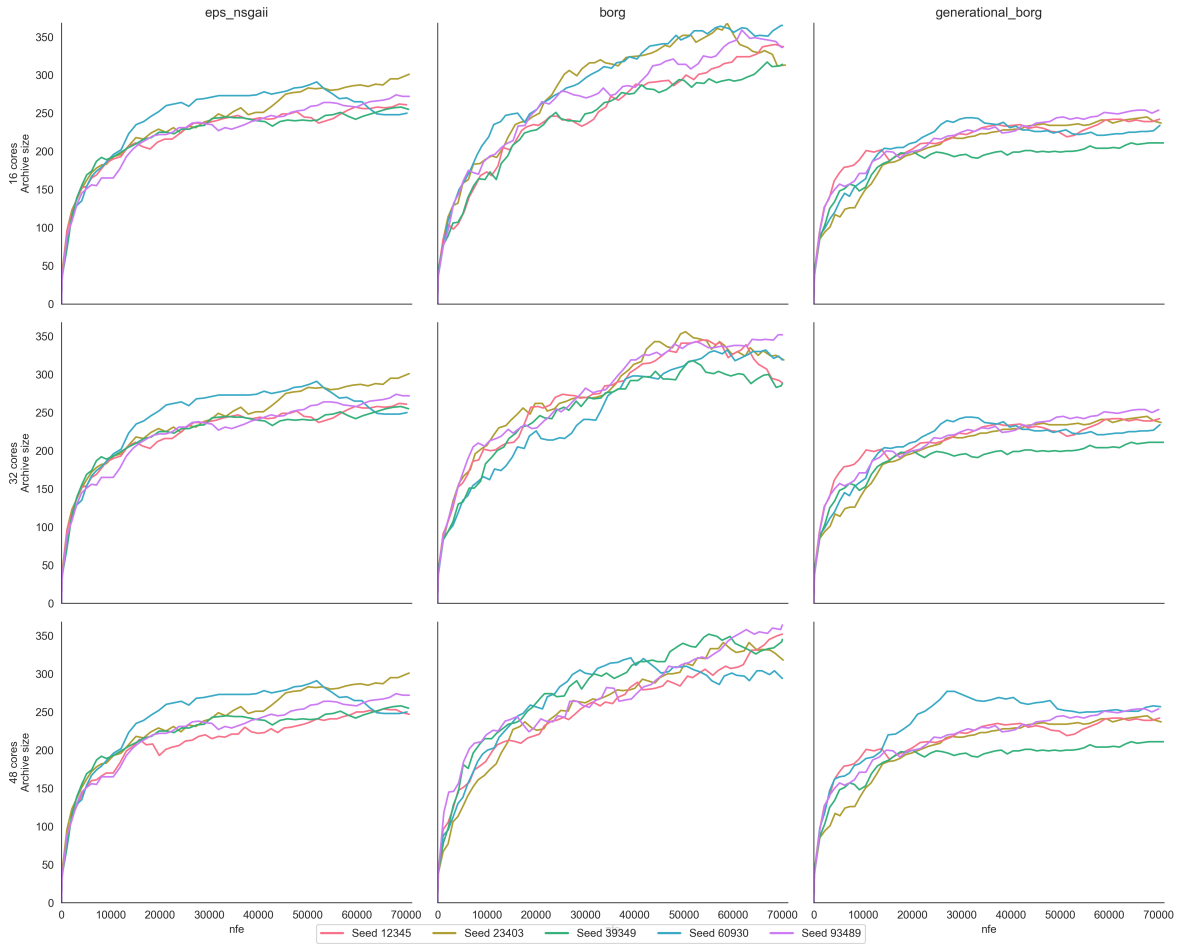

Figure 5.4: JUSTICE Archive Sizes

Table 5.4 summarises the average final archive sizes for each core count and MOEA. It is clear that regardless of core count, Borg finds the most non-dominated solutions, followed by ϵ-NSGA-II and finally Generational Borg.

	Archive Size		
Core Count	**ϵ-NSGA-II**	**Borg**	**Generational Borg**
16	267	333	235
32	267	313	235
48	265	334	240
Total Average	266	326	236

Table 5.4: Average JUSTICE Archive Size

Spacing Figure 5.5 shows the development of the spacing value for each MOEA and core count. All MOEAs follow a similar trend and reach comparable final spacing values. The most striking difference is the small inter-seed difference for ϵ-NSGA-II compared to the Borg and Generational Borg runs.

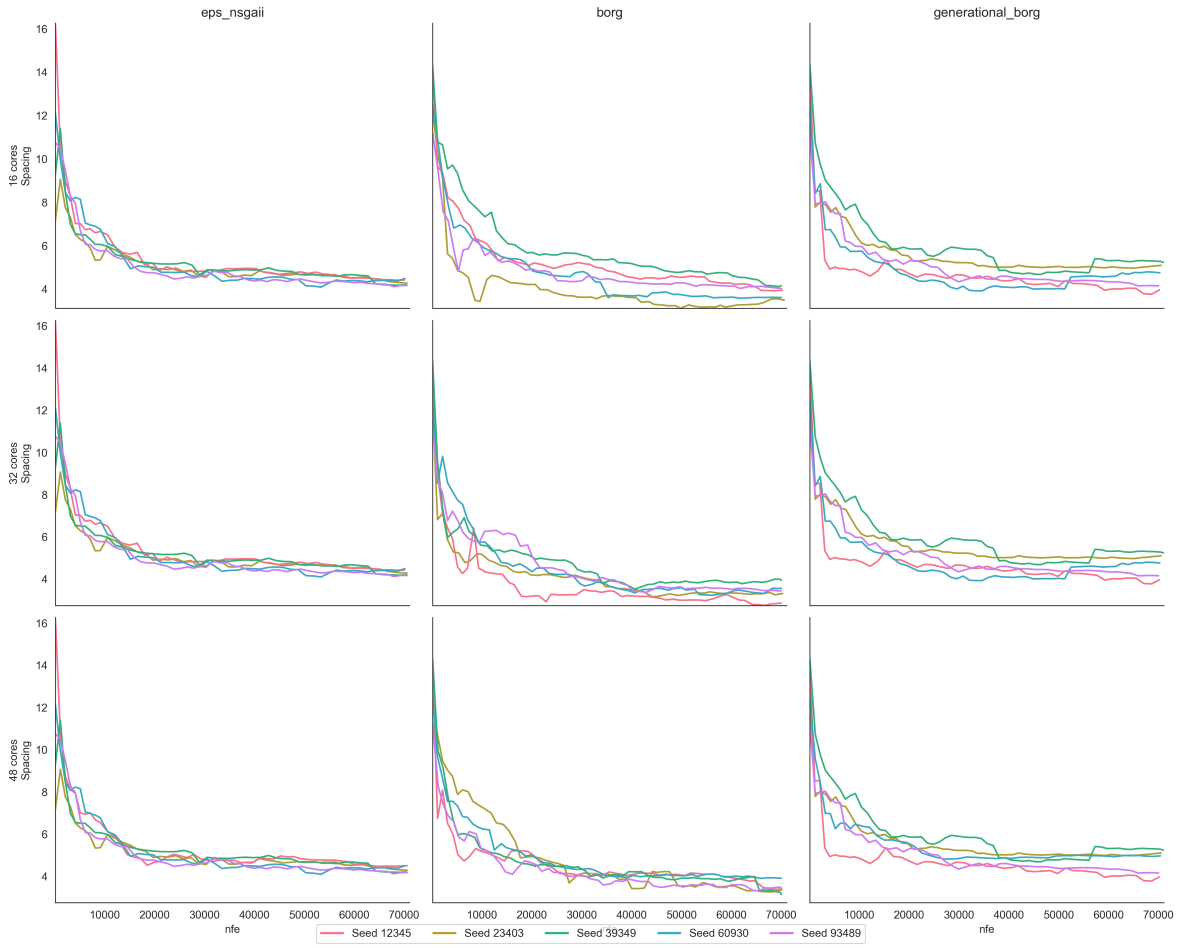

Figure 5.5: JUSTICE Spacing

Table 5.5 also shows the comparable spacing values, especially ϵ-NSGA-II and Generational Borg reach similar values. Another returning observation is that a differing core count does not have a substantial effect on the resulting spacing values.

	Spacing		
Core Count	**ϵ-NSGA-II**	**Borg**	**Generational Borg**
16	4.314934	3.853624	4.641669
32	4.314934	3.422113	4.641669
48	4.330172	3.454604	4.683329
Total Average	4.321482	3.576780	4.655566

Table 5.5: Average JUSTICE Spacing

Hypervolume Efficiency Figure 5.6 shows the hypervolume in- or decrease per function evaluation. The shown behaviour is expected, as all MOEAs rapidly increase in hypervolume at the start, but just make very small progress per function evaluation later in the search. No significant differences in behaviour between MOEAs are observed.

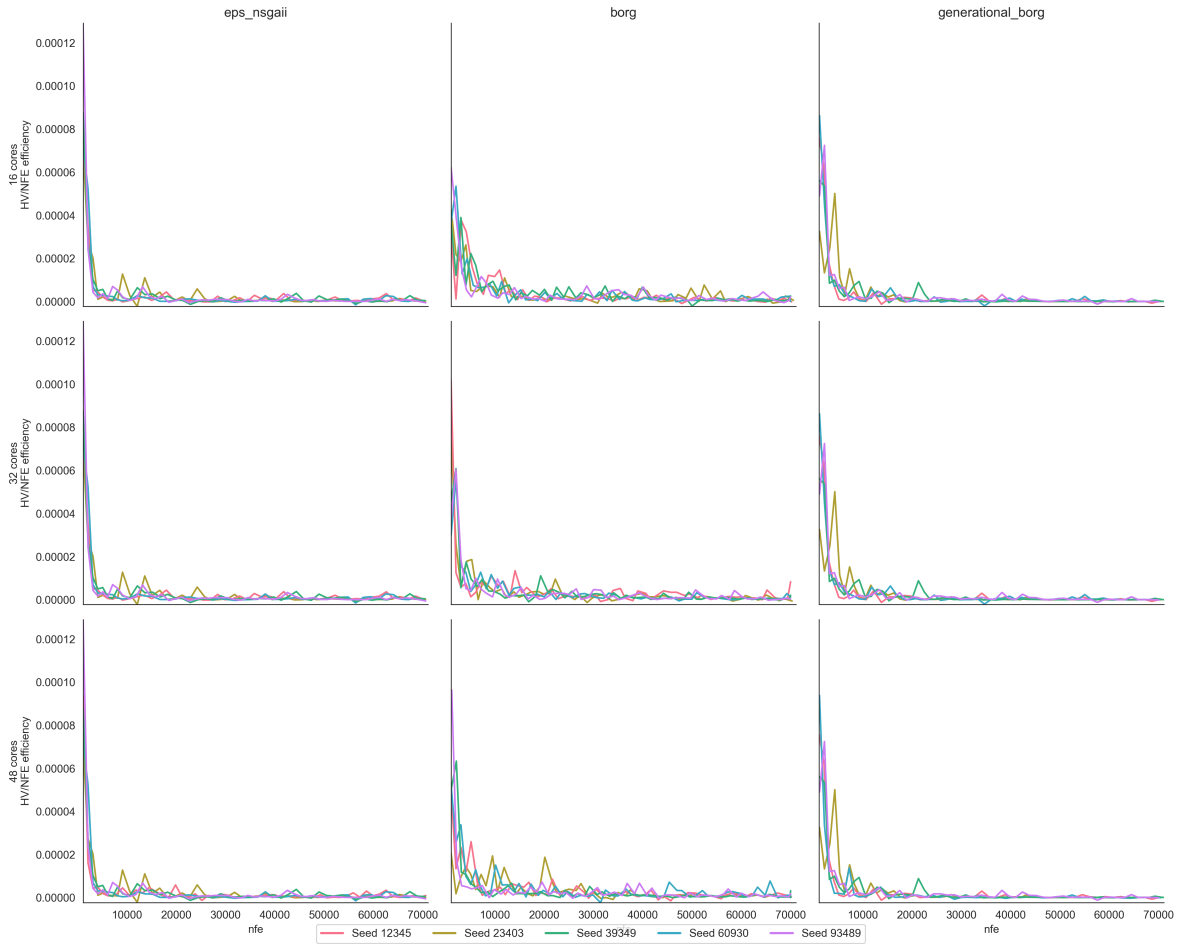

Figure 5.6: JUSTICE Hypervolume Efficiency

Epsilon Progress Figure 5.7 shows the epsilon progress for each MOEA. As expected, Borg shows the biggest epsilon progress. It shows a very linear increase, without any signs of slowing down after 70000 NFE, indicating that it keeps refining quality solutions or finding new ones (since the archive size also keeps growing after 70000 NFE). It also indicates that Borg is likely able to reach a higher hypervolume if run with a higher NFE. ϵ-NSGA-II and Generational Borg show very similar behaviour again. Both reach roughly the same epsilon progress, also still showing a slightly increasing trend even after 70000 NFE. However, given the converging archive sizes this is likely due to small refinements of existing solutions, and not the result of finding new distinct regions of the solution space.

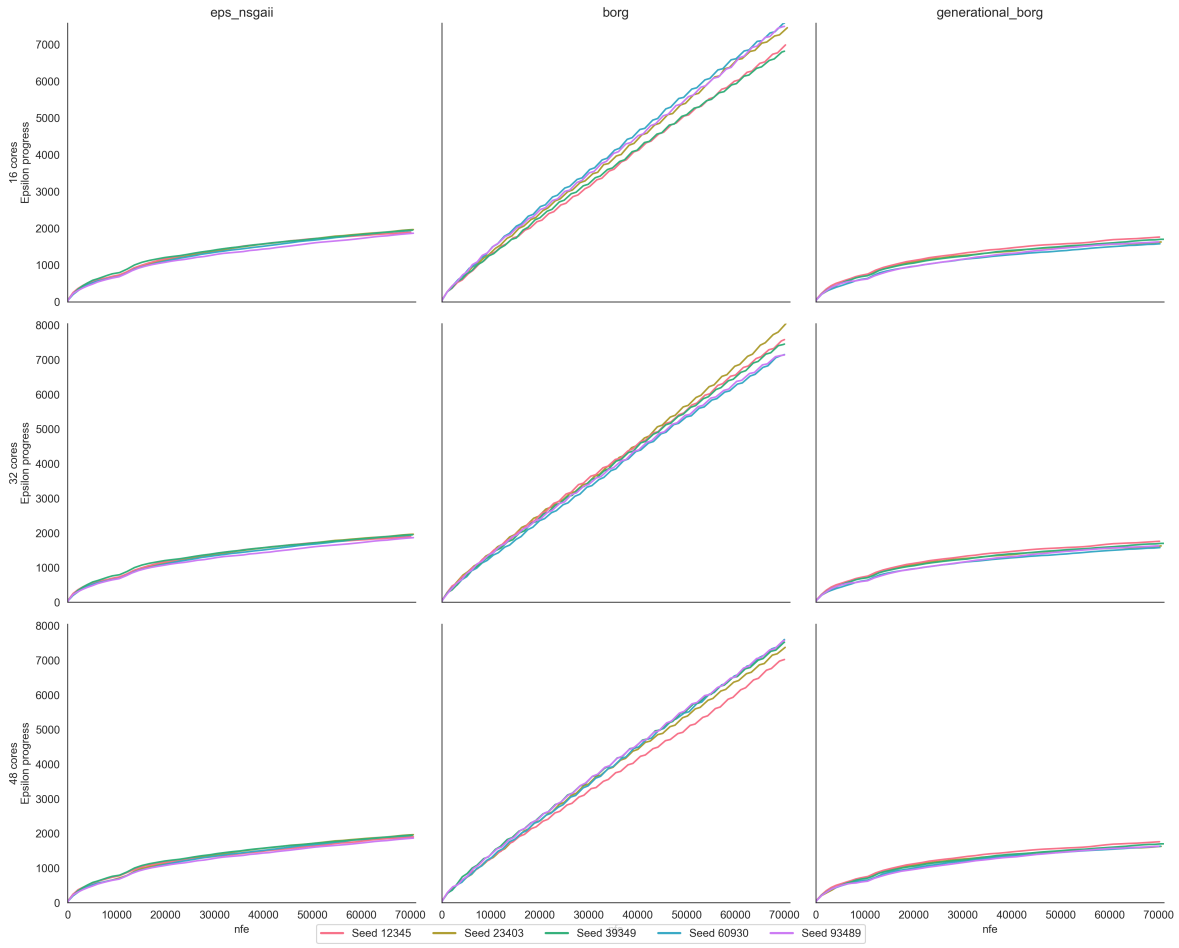

Figure 5.7: JUSTICE Epsilon Progress

Table 5.6 once more illustrates the large difference in epsilon progress made by Borg on one hand and ϵ-NSGA-II and Generational Borg on the other. It indicates that Borg is able to more consistently find solutions in better ϵ-boxes, and thus more effectively explores the solution space of JUSTICE.

	Epsilon Progress		
Core Count	**ϵ-NSGA-II**	**Borg**	**Generational Borg**
16	1920	7263	1658
32	1920	7474	1658
48	1922	7415	1668
Total Average	1920	7384	1661

Table 5.6: Average JUSTICE Epsilon Progress

5.1.2. Computational Efficiency & Scalability

Figure 5.8 presents the mean wall-clock time taken to complete the optimisation for each MOEA and core count. ϵ-NSGA-II is quickest for 32 and 48 cores, with Generational Borg only roughly matching it for 16 cores. Generational Borg is slowest for 32 and 48 cores, slower even than Borg. Normally, this would suggest that auto-adaptivity introduces more overhead, and that a steady-state nature is able to slightly compensate for this. However, the previously mentioned multi-node limitation prevents any definitive conclusions from being drawn. For all three MOEAs an increase in runtime can be observed when cores are increased. This seems unintuitive, as more cores allow for a bigger parallelisation and theoretically a faster runtime. It is very likely that the trend observed in figure 5.8 is also mainly due to

communication overhead introduced by multi-node runs.

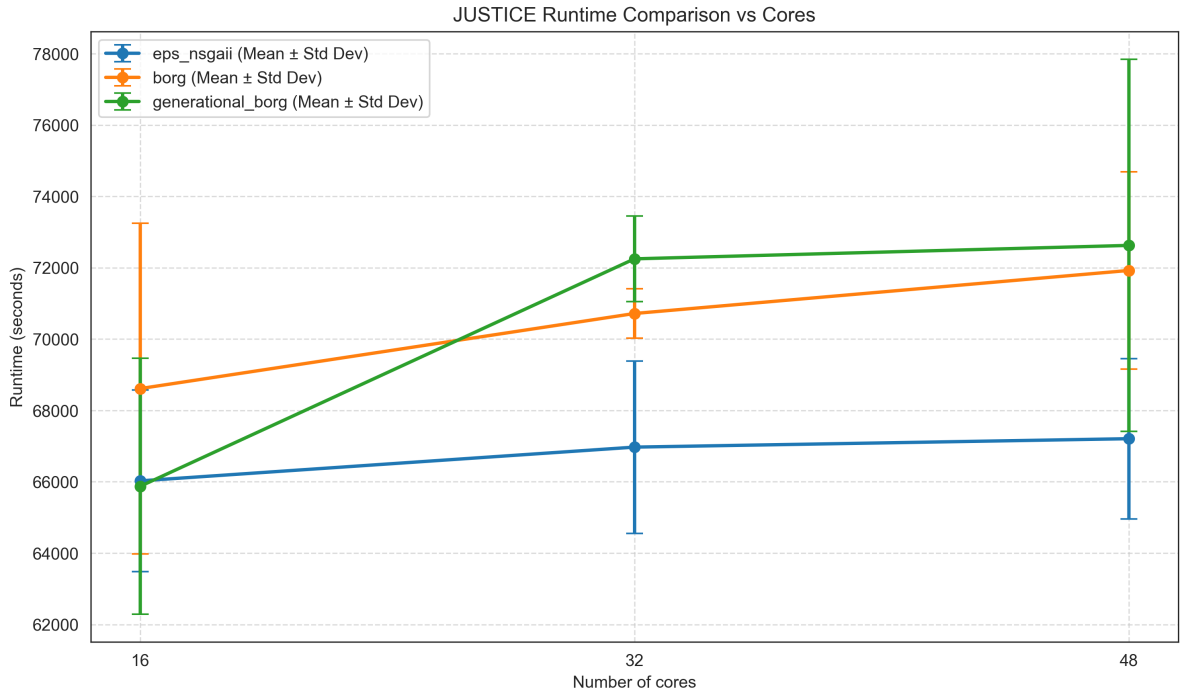

Figure 5.8: JUSTICE Runtime Comparison

Figure 5.9 displays the average obtained hypervolume plotted over wall-clock time. An important side-note here is that the hypervolume was not explicitly logged per second. Rather, the NFE was linearly scaled over the final runtime, allowing the hypervolume to also be plotted over time. However, this does make the assumption that every function evaluation takes approximately the same amount of time regardless of the stage of the search it happens in. As mentioned, no real conclusions based on runtime differences between core counts can be drawn. However, a trend which is visible is that regardless of core count both ϵ-NSGA-II and Generational Borg show a much faster initial gain in hypervolume, with Generational Borg seeing the biggest gain, than regular Borg. Suggesting that a generational nature allows for more rapid initial exploitation of promising solutions. This can likely be attributed to the initial learning time needed for Borg's adaptive features to effectively learn and tune themselves to promising solutions. This is not really observed for Generational Borg, despite also having adaptive features, because its larger batch population updates provide a quicker initial inflow of information for the adaptive features. Nevertheless, for every core count Borg quickly catches up and continues to gain hypervolume well beyond the final values ϵ-NSGA-II and Generational Borg manage to reach, indicating that a steady-state nature is a great driver of further exploration of the solution space.

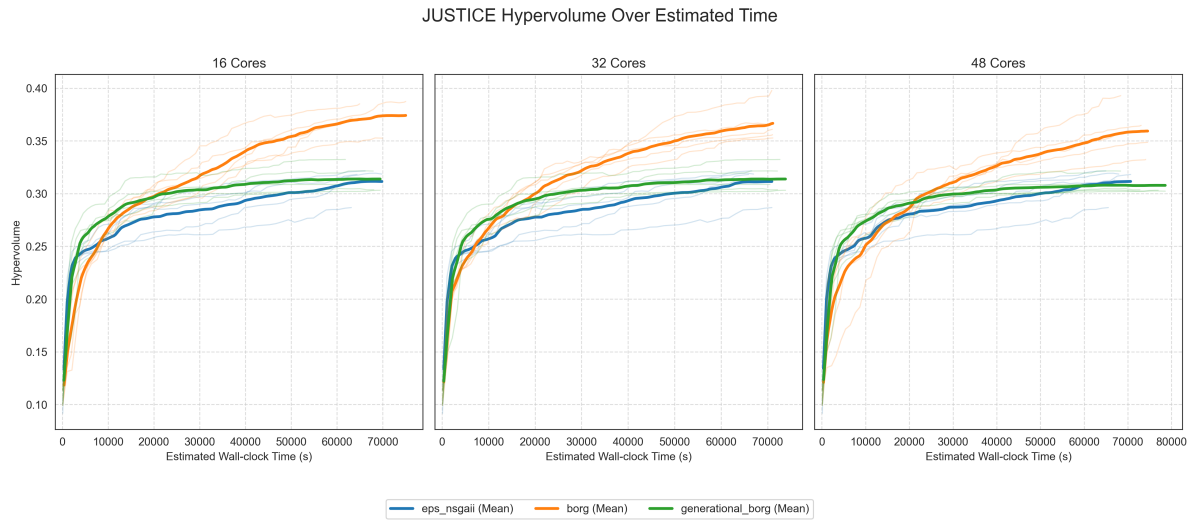

Figure 5.9: JUSTICE Hypervolume Over Wall-Clock Time

5.2. Benchmark Performance

To define a performance baseline for each MOEA it is necessary to look at optimisation problems where the true Pareto front is known. Therefore, the results of the DTLZ2 and DTLZ3 benchmark problems will now be briefly discussed. By plotting the performance metrics mentioned in section 4.5 over the NFE, insights into behaviour, strengths and weaknesses at different stages of the optimisation process can be gained for each MOEA. Additionally, for specific metrics a table with final values will be presented. Again, more detailed results and additional plots can be found in appendices B and C respectively.

5.2.1. DTLZ2

Since DTLZ2 is a simple unimodal problem with a known analytical solution, it is very well suited to quickly evaluate whether the MOEAs show any fundamental performance differences that would become apparent on any given problem they optimise. This allows for checking if the observed performance differences on JUSTICE really are due to the MOEAs ability to handle JUSTICE's specific characteristics, or due to fundamental differences.

Hypervolume Figure 5.10 shows the hypervolume for all three MOEAs for each core count for five seeds on DTLZ2.

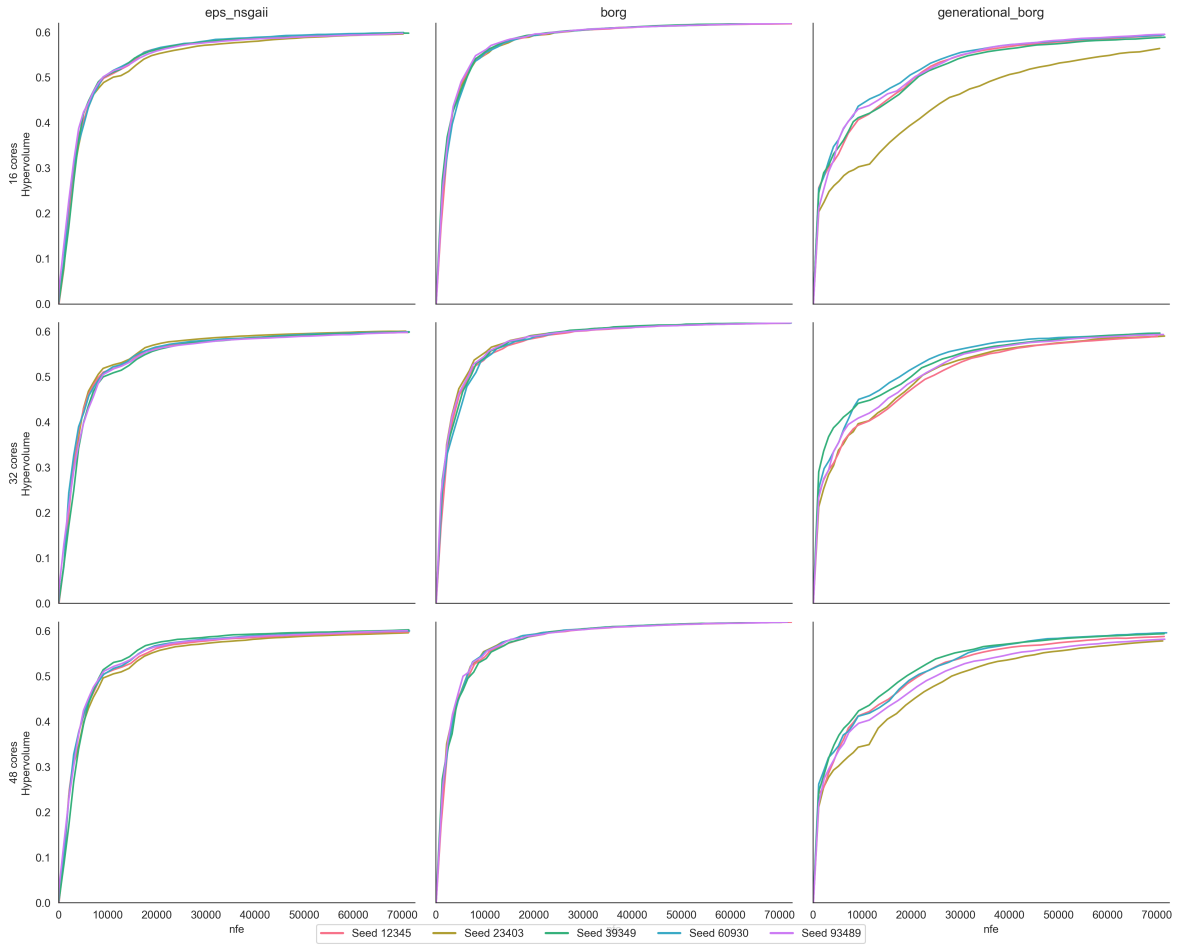

Figure 5.10: DTLZ2 Hypervolume

What stands out is that for all seeds of ϵ-NSGA-II and Borg, and for most seeds of Generational Borg, the hypervolume tends to converge to a value of around 0.6. Given the relative simple problem that is DTLZ2, this seems rather low. However, using a reference point $\mathbf{z}^{ref} = (1, 1, 1, 1)$ for the DTLZ hypervolume calculations means we calculate the volume of the unit hypercube minus the volume of the spherical volume formed by the Pareto front. For a unit m-sphere the volume of the first orthant is given by;

$$V_{orthant}(m, r) = \frac{1}{2^m} \cdot \frac{\pi^{\frac{m}{2}}}{\Gamma(\frac{m}{2} + 1)} \cdot r^m \tag{5.1}$$

And since we have $m = 4$ objectives and radius $r = 1$, the volume of the Pareto front is $\frac{1}{2^4} \cdot \frac{\pi^2}{2} = \frac{\pi^2}{32}$. As a result of this the maximum theoretically obtainable hypervolume for both DTLZ2 and DTLZ3 is;

$$HV_{max} = 1 - \frac{\pi^2}{32} \approx 0.6916 \tag{5.2}$$

This shows that the MOEAs find a Pareto front close to, but not completely covering the true Pareto front. Table 5.7 shows the obtained hypervolume values averaged out over the seeds. For each core count, Borg clearly obtains the best hypervolume, followed by ϵ-NSGA-II and then Generational Borg. As mentioned they approach the theoretical maximum hypervolume but don't quite fully reach it. Given the very slight increasing plots seen in figure 5.10 it is possible that with a higher NFE the

hypervolume would near the theoretical maximum further. However, what is the most important to note here is that the differences in hypervolume are definitely smaller than they were for JUSTICE. This confirms that compared to the other two MOEAs, Borg's characteristics were better suited for a problem like JUSTICE and responsible for achieving a higher hypervolume.

	Hypervolume		
Core Count	**ϵ-NSGA-II**	**Borg**	**Generational Borg**
16	0.597470	0.619322	0.586789
32	0.598960	0.618976	0.593103
48	0.599335	0.619188	0.587365
Total Average	0.598588	0.619162	0.589086

Table 5.7: Average DTLZ2 hypervolume

Generational Distance Figure 5.11 shows the generational distance of each MOEA, for differing core counts and different seeds. Given the obtained results for the hypervolume, the results from figure 5.10 are logical. For all core counts each MOEA rapidly improves its own solution set, getting closer to the analytically generated true Pareto front. After approximately 10000 NFE the generational distance is already approaching values near zero.

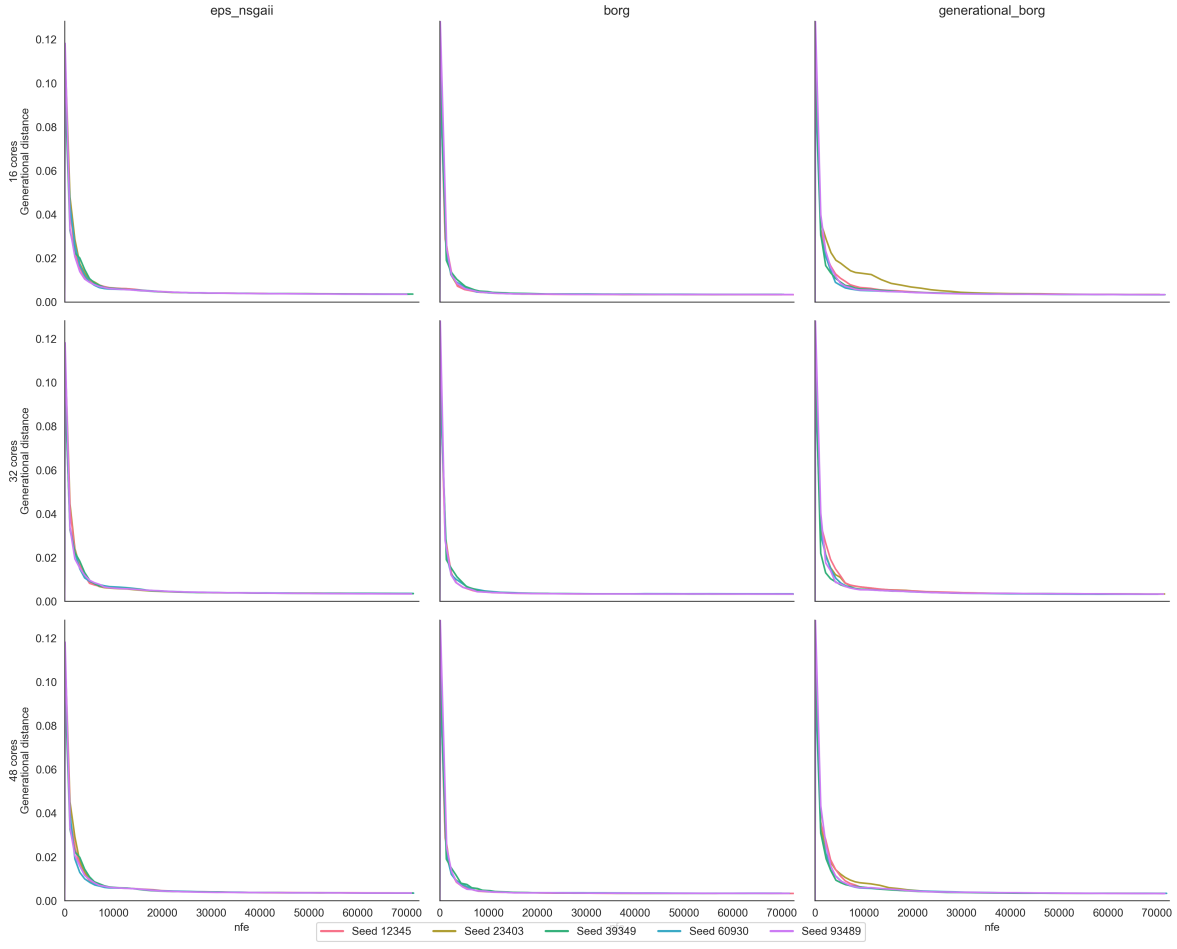

Figure 5.11: DTLZ2 Generational Distance

Table 5.8 shows the final generational distance for all MOEAs and core counts. Just like with the hypervolume values, these values do not differ very much between MOEA or core count. This

is expected as DTLZ2 is a relatively simple problem. The low values indicate that all optimisations managed to get close to the true Pareto front with their obtained solutions. On average all three MOEAs converge to similar values. In context of JUSTICE, this confirms that auto-adaptive features do not help a MOEA to better approximate the true JUSTICE Pareto front, as indicated by the similar final generational distance values on JUSTICE for ϵ-NSGA-II and Generational Borg shown in table 5.2. Instead, a steady-state nature is a bigger driver of closer Pareto front approximation, as table 5.2 does clearly show Borg reaches better final generational distance values for JUSTICE.

	Generational Distance		
Core Count	**ϵ-NSGA-II**	**Borg**	**Generational Borg**
16	0.003498	0.003354	0.003318
32	0.003492	0.003331	0.003284
48	0.003478	0.003341	0.003288
Total Average	0.003489	0.003342	0.003297

Table 5.8: Average DTLZ2 Generational Distance

Computational Efficiency & Scalability

Figure 5.12 displays the average hypervolume over wall-clock time. Despite the often mentioned multi-node limitation, the plot does show a number of interesting facts. Regardless of core count, ϵ-NSGA-II is the fastest to complete, reaching the second best hypervolume every time. Generational Borg is the second fastest, but also converges to the smallest hypervolume every time. Lastly, Borg consistently reaches the highest hypervolume, while also taking the longest every time. What is most interesting to see is that the differences in runtime between the generational MOEAs and the steady-state Borg are far more outspoken for this simple problem than for JUSTICE. Furthermore, it is interesting to see that ϵ-NSGA-II initially sees a far more rapid increase in hypervolume than the other MOEAs. This suggests two things. Auto-adaptive operators introduce a certain computational overhead on simple problems, as seen by the later hypervolume increases of Borg and Generational Borg, as their added complexity is not necessary to solve simple problems. Additionally, simple problems do not necessarily require a steady-state update scheme, as the fine-grained search pressure introduced by a steady-state update scheme does not translate into substantially better hypervolume values in these cases. However, it does lead to substantially longer runtimes. This clearly changes for more complex problems, as seen for JUSTICE in figure 5.9.

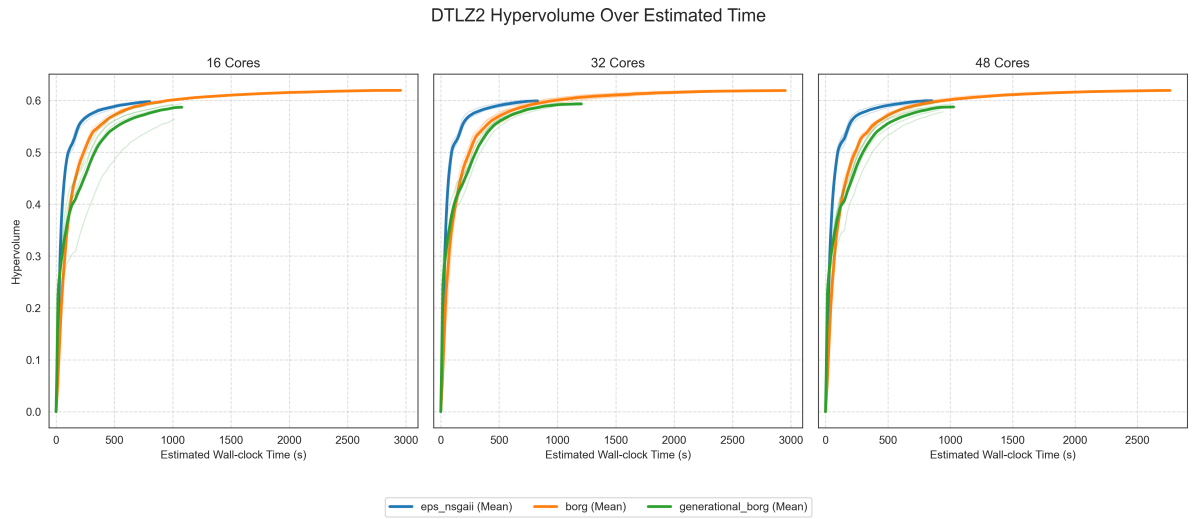

Figure 5.12: DTLZ2 Hypervolume Over Wall-Clock Time

5.2.2. DTLZ3

Just like DTLZ2, DTLZ3 is a benchmark problem with an analytical solution. However, it differentiates itself through its deceptiveness, introduced by its multi-modality. Therefore it enables a better understanding of whether an observed performance difference is introduced due to pure multi-modality, or other factors such as the high-dimensionality and thus large solution space as seen in JUSTICE. This leads to better insights regarding the effects auto-adaptivity and a generational versus steady-state nature have on a MOEA's ability to tackle a problem with specific characteristics.

Solution Quality and Convergence Dynamics

Hypervolume Figure 5.13 displays the obtained hypervolume for all MOEAs for each seed and core count. As previously explained for DTLZ2, the theoretical maximum for the hypervolume of DTLZ3 in this research also is $HV_{max} \approx 0.6916$. From the plot it immediately becomes clear that the many local optima make DTLZ3 a more deceptive problem than DTLZ2. ϵ-NSGA-II does not obtain a positive hypervolume for any run. This is behaviour that has been observed before, with the NSGA-II and NSGA-III MOEAs also not being able to obtain any hypervolume on a three-objective DTLZ3 minimisation problem (Ishibuchi et al., 2016). Generational Borg obtains a good hypervolume for four runs, spread over three seeds and three core counts, total. Only Borg seems to more consistently approach the global Pareto front, but still has runs with zero or very low hypervolume. This shows that all MOEAs have trouble navigating DTLZ3's deceptive true Pareto front. However, as exemplified by seed 23403 in the 48-core Generational Borg run (which only started increasing after 60000 NFE), there is a possibility that the MOEAs simply need a higher NFE to find good solutions for DTLZ3. The observed behaviour strongly points towards the lack of auto-adaptive features limiting a MOEA's ability to escape local optima and handle multi-modal problems, as indicated by the differences between ϵ-NSGA-II and Generational Borg (and Borg of course). Additionally, the difference between Generational Borg and Borg also suggests that having a generational instead of steady-state nature inhibits a MOEA's search pressure and, thus, ability to explore diverse regions of the solution space.

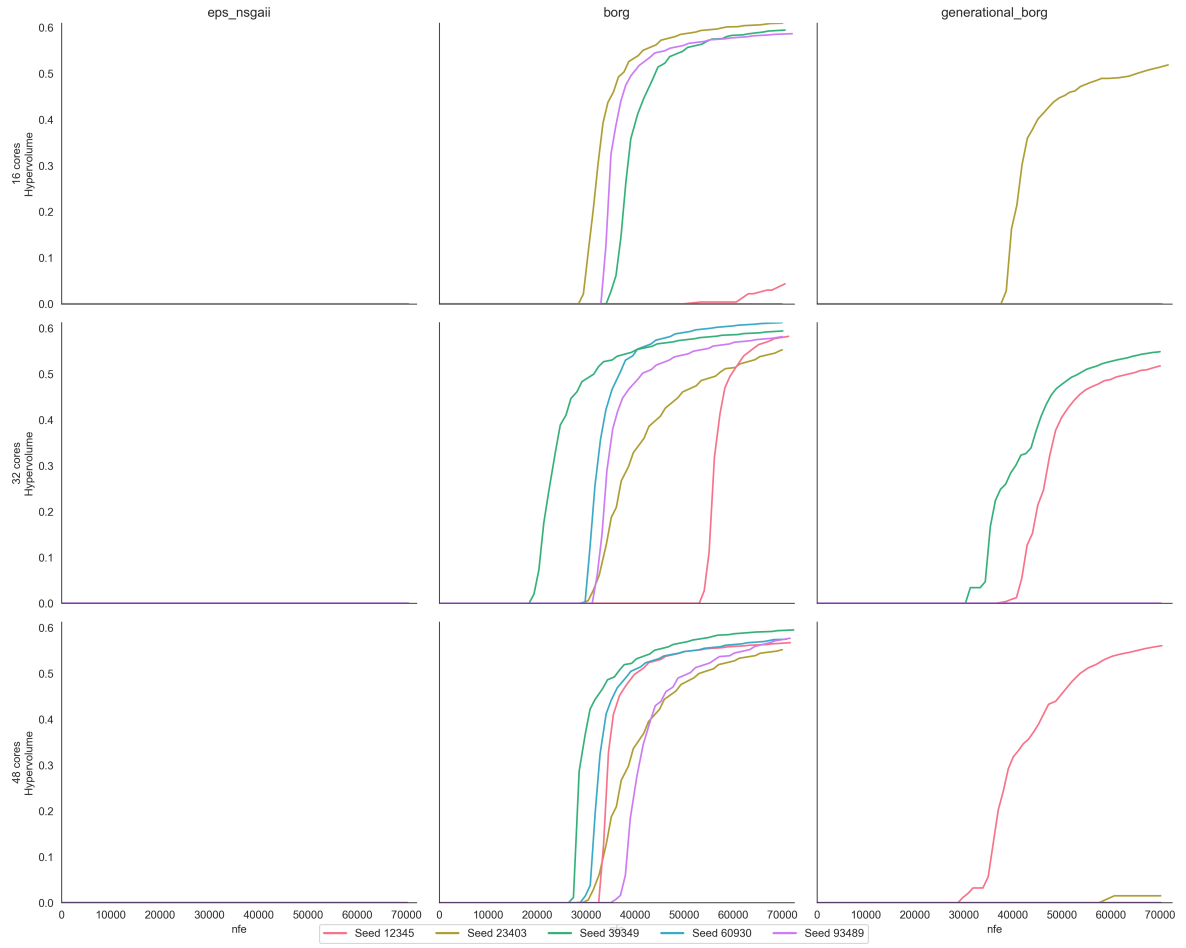

Figure 5.13: DTLZ3 Hypervolume

Interpreting the values in table 5.9 requires extra attention, as these are values averaged over seeds. The plot showed that most of the times a run would either approach the true front well, or not at all. What it does show is that Borg consistently had more runs successfully navigating the DTLZ3 landscape, with Generational Borg sometimes managing to do so as well, and ϵ-NSGA-II not managing at all. The results observed here do also say something about the JUSTICE problem. Given the hypervolume values observed for JUSTICE as seen in table 5.1, where every MOEA obtained a (very) roughly similar positive hypervolume for each seed, it is likely that JUSTICE is substantially less deceptive than DTLZ3. The MOEAs were less likely to become stuck in local optima and not gain any hypervolume at all. However, the relatively low JUSTICE hypervolume values indicate that instead it has a very big solution space. This is likely introduced through its high dimensionality, and the MOEAs struggle to explore a large part of this area within the given computational budget.

	Hypervolume		
Core Count	**ϵ-NSGA-II**	**Borg**	**Generational Borg**
16	0	0.366820	0.103760
32	0	0.584704	0.213371
48	0	0.573190	0.115125
Total Average	0	0.508238	0.144085

Table 5.9: Average DTLZ3 Hypervolume

Generational Distance Figure 5.14 shows the DTLZ3 generational distance for each MOEA and core count. An important note is that while having a hypervolume of zero, a solution set can still improve its generational distance. The generational distance merely measures the average distance to the true Pareto front, whereas hypervolume indicates if the found solutions dominate $\mathbf{z}^{ref}$. Therefore ϵ-NSGA-II still displays a decreasing trend in its generational distance. However, as expected, Borg shows the steepest and fastest decline in generational distance. Generally, the plots look typical for the generational distance, with a steep decrease early on in the search, which then slowly converges.

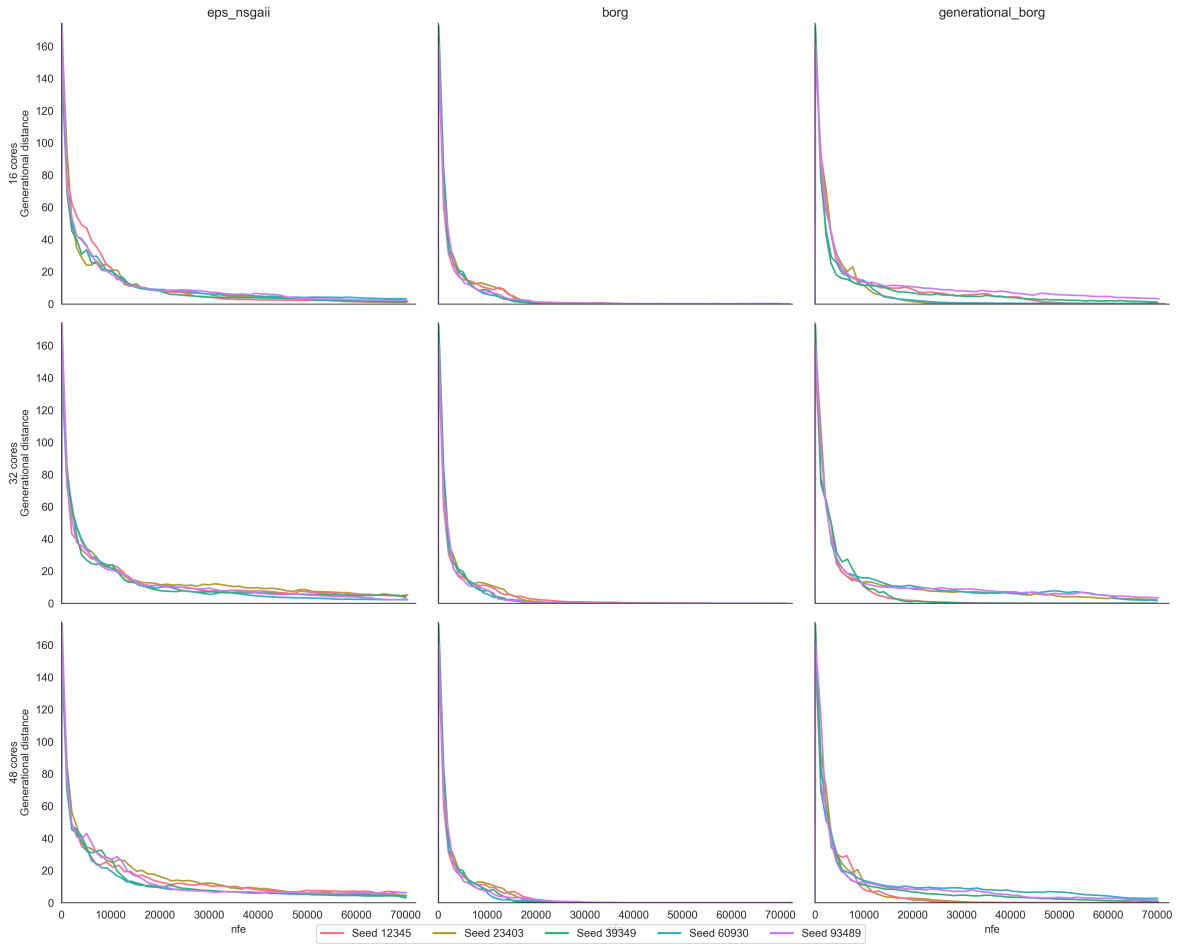

Figure 5.14: DTLZ3 Generational Distances

Table 5.10 again confirms that ϵ-NSGA-II does find solutions closer and closer to the true Pareto front, but not close enough to dominate $\mathbf{z}^{ref}$. Generational Borg, and especially Borg, obtain lower values for generational distance. This is to be expected, as they actually displayed runs where their solutions managed to dominate $\mathbf{z}^{ref}$. Again, the values in table 5.10 all are worse than the generational distance values for JUSTICE shown in table 5.2. With ϵ-NSGA-II obviously showing the biggest differences. This confirms that all three MOEAs had more difficulty approaching the deceptive true Pareto front of DTLZ3 than that of the high-dimensional JUSTICE problem. However, given that Borg outperformed the other MOEAs for both problems, auto-adaptive features and a steady-state nature clearly aid the exploration of complex solution spaces. This seems to especially be the case for auto-adaptive features, given that Generational Borg did obtain a positive hypervolume for some runs on DTLZ3.

Archive Size Figure 5.15 shows the development of the archive size over NFE for each MOEA. Compared to the DTLZ2 archive size plots shown in figure C.3, it immediately stands out that ϵ-NSGA-II has a very low number of solutions, and that it does not show any increasing trend. This does not guarantee that ϵ-NSGA-II would not have found more solutions if given more NFE, but it does indicate

	Generational Distance		
Core Count	**ϵ-NSGA-II**	**Borg**	**Generational Borg**
16	1.764981	0.046274	0.970969
32	3.572047	0.003385	1.394063
48	4.810238	0.003236	0.962975
Total Average	3.382422	0.017632	1.109336

Table 5.10: Average DTLZ3 Generational Distance

how much it is struggling with finding any solutions for the DTLZ3 problem. Comparing the plots for Borg and Generational Borg, the plots coincide relatively well with the hypervolume plots, with runs that obtain higher hypervolume values also ending the run with a larger archive size. What is an interesting observation is that for some runs, the archive size seems to substantially decrease again after reaching a peak.

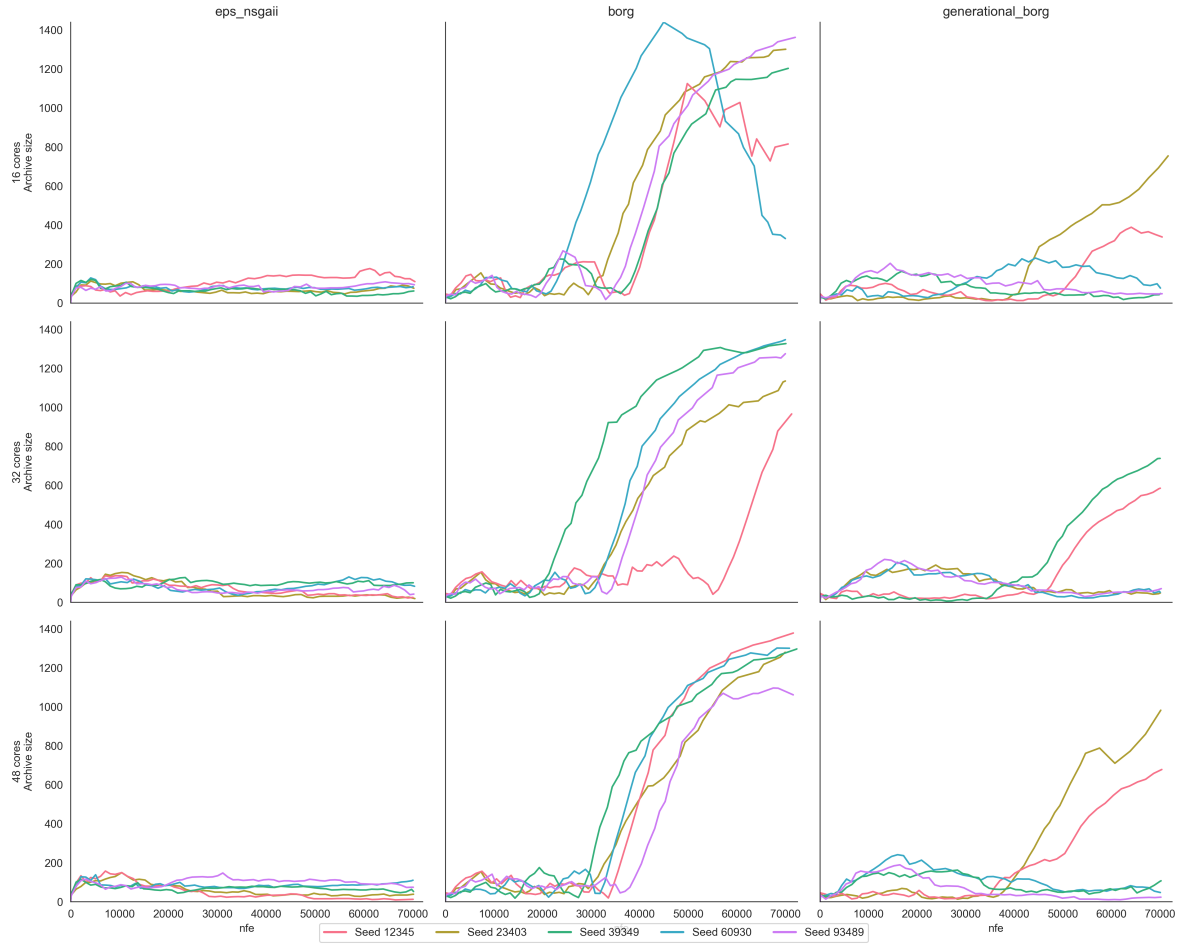

Figure 5.15: DTLZ3 Archive Sizes

Table 5.11 summarises the average final archive sizes per core count. It shows the very small amount of solutions found by ϵ-NSGA-II, especially compared to the archive size of Borg. Both Borg and Generational Borg show a slight increase in final average archive size with a core count increase. Again, when compared to the final archive sizes of JUSTICE seen in table 5.4, the differences between the MOEAs are far greater for DTLZ3. This indicates how much more especially ϵ-NSGA-II, with its generational nature and lack of auto-adaptivity, struggles with deceptive problems.

Core Count	*Archive Size* ϵ-NSGA-II	Borg	Generational Borg
16	84	1002	252
32	53	1210	298
48	57	1263	367
Total Average	64	1158	305

Table 5.11: Average DTLZ3 Archive Size

Epsilon Progress Lastly, we consider the epsilon progress shown in figure 5.16 and table 5.12. Borg still shows an increasing trend after 70000 NFE, making it more likely that the seeds that were unable to obtain any hypervolume after 70000 NFE might still improve if given more function evaluations. For both ϵ-NSGA-II and Generational Borg this trend is far less outspoken (with exceptions for some Generational Borg seeds). However, even for ϵ-NSGA-II the epsilon progress keeps increasing ever so slightly, indicating that it still is finding new and improved solutions. Though, given the high values for the generational distance and epsilon indicator, as well as a very low value for the archive size, it is more likely that it is simply refining solutions for a local optima rather than discovering new high-quality solutions.

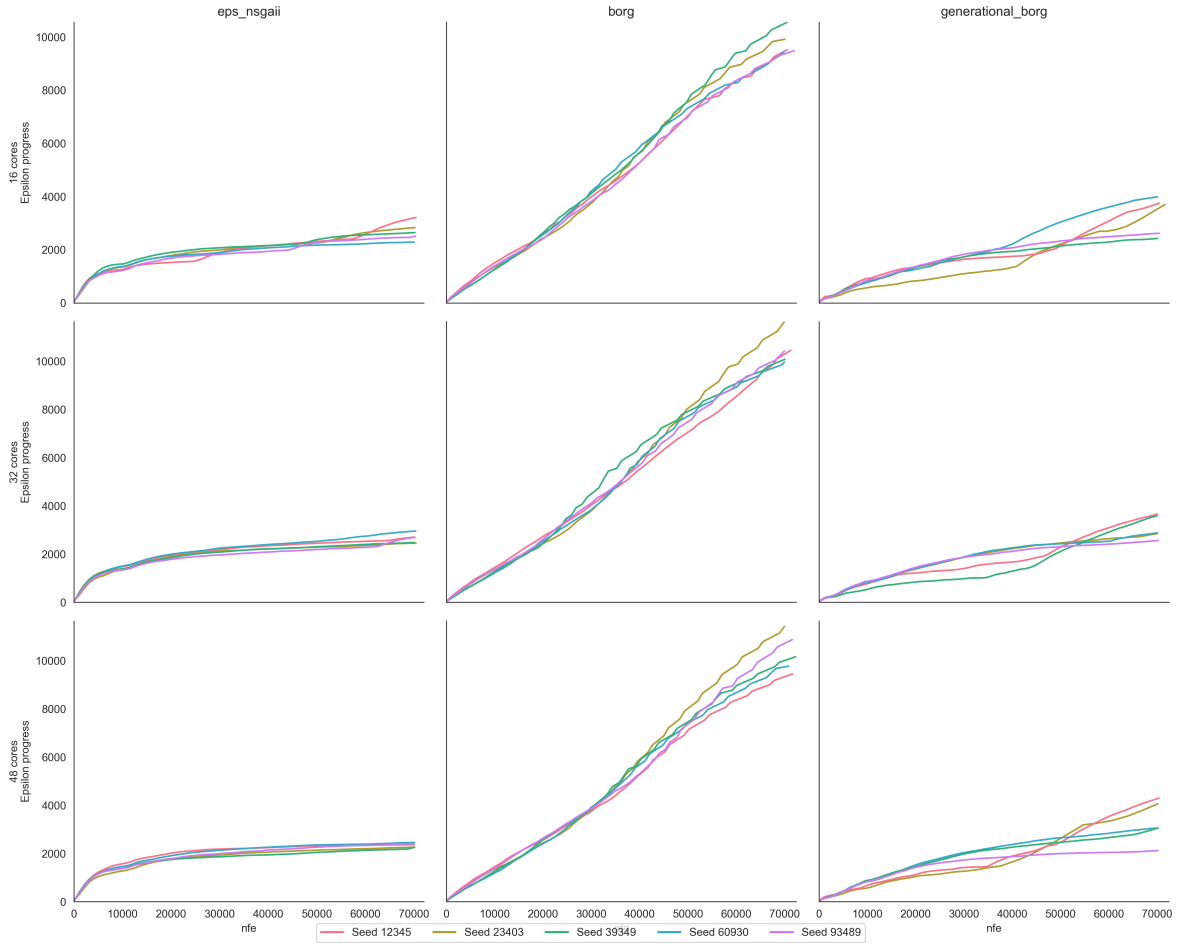

Figure 5.16: DTLZ3 Epsilon Progress

Table 5.12 gives the final values for the epsilon progress. Confirming that Borg finds substantially more improvements, followed by Generational Borg and ϵ-NSGA-II closely trailing behind that. Each MOEA reaches a higher epsilon progress on DTLZ3 than they do on JUSTICE, as can be seen when

comparing tables 5.12 and 5.6. This could again confirm the suspicion that JUSTICE simply is a larger but less deceptive problem than DTLZ3. When the MOEAs get stuck in a local optima, they can keep on refining solutions within that local optimum, thus increasing epsilon progress, without getting closer to the true global Pareto front.

	Epsilon Progress		
Core Count	**ϵ-NSGA-II**	**Borg**	**Generational Borg**
16	2696	9786	3297
32	2656	10512	3109
48	2347	10331	3317
Total Average	2566	10209	3241

Table 5.12: Average DTLZ3 Epsilon Progress

Computational Efficiency & Scalability

Figure 5.17 presents the mean wall-clock time each MOEA needed to complete 70000 NFE on DTLZ3. Just like with DTLZ2, Borg takes substantially longer to complete its 70000 NFE than both ϵ-NSGA-II and Generational Borg. It is interesting that all MOEAs take less time for DTLZ3 than they did for DTLZ2. An explanation could be the MOEAs struggling with finding the true Pareto front, and getting stuck in local optima, speeding up the search. Just like with DTLZ2, Borg demonstrates the greatest runtime decrease over an increasing core count (though still very minimal), which would make sense as it is the only true steady-state MOEA of the three. However, again, the multi-node limitation likely exerted a substantial influence on these results, therefore rendering the conclusions drawn from them uncertain.

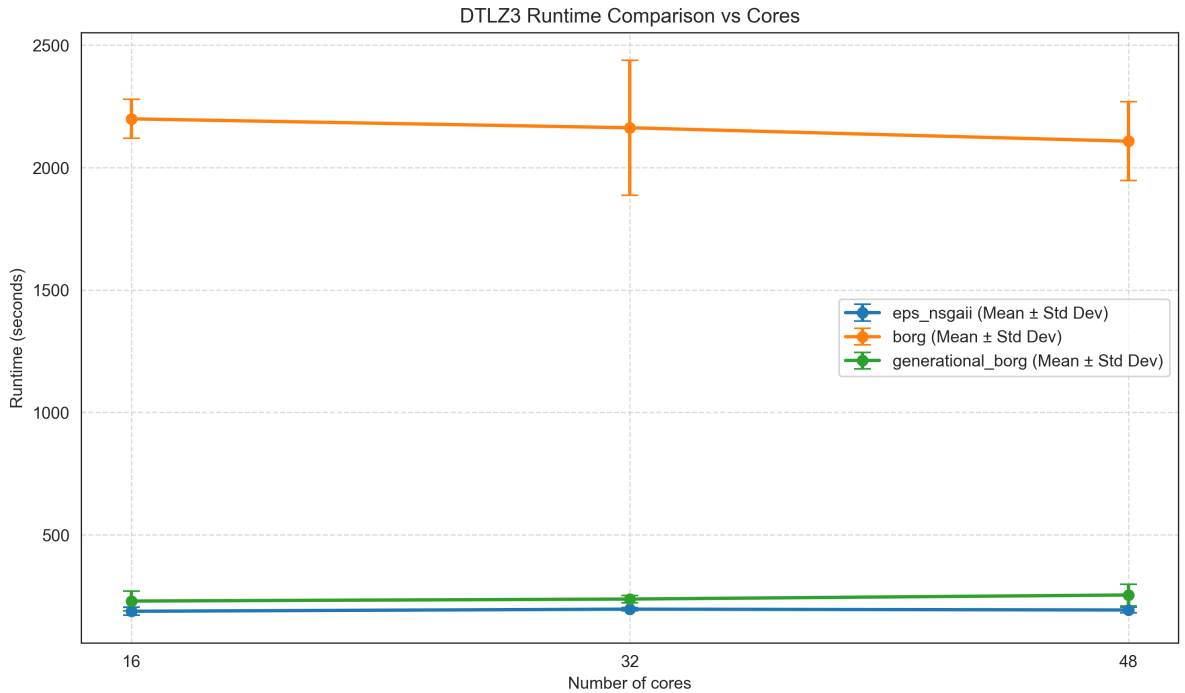

Figure 5.17: DTLZ3 Runtime Comparison

Figure 5.18 shows the average hypervolume development over the estimated wall-clock time in bold, with the shaded lines representing individual runs. The plots reflect the behaviour seen in the previously discussed performance metrics. ϵ-NSGA-II did not obtain any hypervolume during its search. For some runs, Generational Borg was successful in obtaining some hypervolume, and did so relatively quick. Borg took the longest but was able to find a decent hypervolume in most cases.

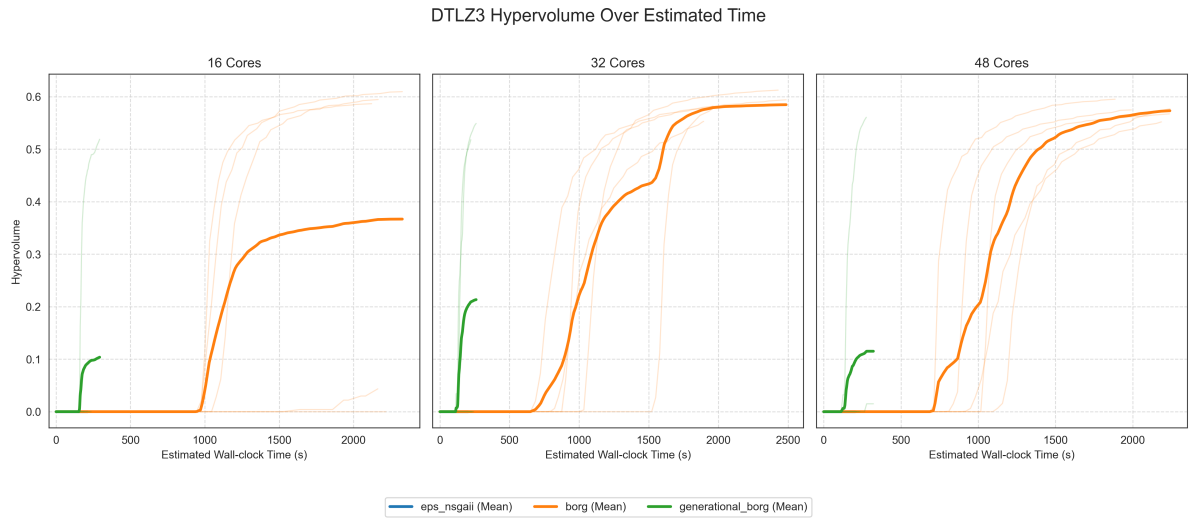

Figure 5.18: DTLZ3 Hypervolume Over Wall-Clock Time

6

Discussion

This chapter provides a detailed discussion of the observed results from chapter 5, their implications and reasoning behind their observation. It generalises the results and proposes a rough visual framework to aid future MOEA selection. Additionally, it restates a number of key limitations. Lastly, it explicitly states a number of contributions made by this research, discusses the implications of the findings and proposes directions for future research.

6.1. MOEA Performance Synthesis

This section will provide a detailed synthesis of the results presented in chapter 5 and aims to give an explanation for them. Instead of interpreting the results for each problem in isolation, they can be generalised into a conceptual framework to aid future MOEA selection. This framework maps the specific problem characteristics *deceptiveness* (multi-modality, as seen primarily in DTLZ3) and *solution space size* (due to high-dimensionality, as seen in JUSTICE) to fundamental MOEA architectural properties. Figure 6.1 gives a very rough visual representation on how to base the MOEA architecture choice on problem characteristics. This section will analyse the observed performance on each problem to provide evidence supporting this framework and aim to justify why the MOEAs performed on each problem the way they did. Finally, a few notable observations will be discussed in more detail.

6.1.1. Solution Quality & Convergence Dynamics

We start by discussing the performance of each MOEA with regards to final solution quality and convergence dynamics on DTLZ2, DTLZ3 and JUSTICE.

DTLZ2

DTLZ2 is the simplest of all three problems. Its smooth and unimodal true Pareto front is relatively easy to approach in the optimisation process. This is clearly reflected in the obtained results, with all three MOEAs performing well. Nevertheless, Borg did achieve the best hypervolume, approaching the theoretical maximum hypervolume for DTLZ2. Additionally, it achieved the best values for the epsilon indicator, archive size, spacing and epsilon progress. ϵ-NSGA-II and Generational Borg performed very similarly, being slightly worse than Borg, but still converging to good solutions for DTLZ2. These marginal differences confirm that for simpler problems without a substantial degree of multi-modality or high-dimensionality, like DTLZ2, advanced MOEA characteristics like auto-adaptive mechanisms, epsilon progress tracking and steady-state versus generational natures do not necessarily make a MOEA outperform other MOEAs lacking those features by a great amount.

DTLZ3

DTLZ3 differs from DTLZ2 in that it is a significantly more deceptive model, its multi-modal nature introduces a lot of local optima which MOEAs can easily get stuck in. This increase in problem difficulty was clearly observed in the obtained results, with performance differences between the MOEAs becoming much more pronounced. Borg now clearly outperforms both ϵ-NSGA-II and Generational Borg across

practically all performance metrics. Most striking is the difference in hypervolume values. Borg was the only MOEA that was able to consistently obtain a positive hypervolume value (with the exception of one single run). This indicates a strong ability to avoid getting stuck in local optima and successfully explore more deceptive solution spaces. In contrast, ϵ-NSGA-II fails to obtain any hypervolume at all. Despite its generational distance and epsilon progress plots in figures 5.14 and C.1 showing that it does get closer to the true Pareto front during the optimisation process, its solutions fail to dominate $\mathbf{z}^{ref}$ and thus do not generate any hypervolume.

This extreme difference in performance can thus be attributed to Borg's defining features, namely its adaptive tournament sizing, auto-adaptive operator selection and epsilon progress-based restart mechanism. As stated by Hadka and Reed (2013) these features greatly enhance performance on multi-modal problems. Additionally, its steady-state nature ensures a more constant in- and outflow of solutions in the ϵ-archive. This results in a more fine-grained and continuous selection pressure. Together with its continuous elitism, and more frequent feedback for adaptive features, this helps Borg to preserve solution diversity and prevent getting stuck in local optima.

This steady-state nature effect can be clearly seen when considering Generational Borg's performance on DTLZ3. Generational Borg shares almost all features of Borg, except for that it uses a generational population update mechanism. Its performance on DTLZ3 falls in between that of ϵ-NSGA-II and Borg. Despite its auto-adaptive mechanisms and epsilon progress tracking, its epsilon archive is updated less frequently and in greater batches than that of Borg. As a result of this, high-quality solutions are introduced into the population less quickly, delaying useful feedback to the adaptation mechanisms and reducing the responsiveness of the search process. This becomes clearly visible when considering the archive sizes shown in table 5.11. Borg finds approximately four times as many solutions as Generational Borg. However, on its turn Generational Borg finds roughly six times as many solutions as ϵ-NSGA-II. This clearly illustrates the effectiveness of using adaptive features and epsilon progress tracking as well as the additional effect a steady-state population update approach has on successfully dealing with deceptive multi-modal problems.

JUSTICE

The observed results for JUSTICE provide more interesting insights, also with regards to JUSTICE its own characteristics. For JUSTICE, all three MOEAs manage to obtain a positive hypervolume, though Borg does outperform the other two again. However, despite all MOEAs obtaining positive hypervolume values relatively close to each other, none of them are particularly high. Combined with the fact that ϵ-NSGA-II does consistently obtain a hypervolume for JUSTICE (which was not the case for DTLZ3) this indicates that it is very likely that JUSTICE is a less deceptive problem than DTLZ3. Instead, its very high dimensionality (JUSTICE has 244 optimisable parameters versus only 13 for DTLZ2 and DTLZ3) possibly makes for an extremely large solution space which all MOEAs are having trouble with fully exploring.

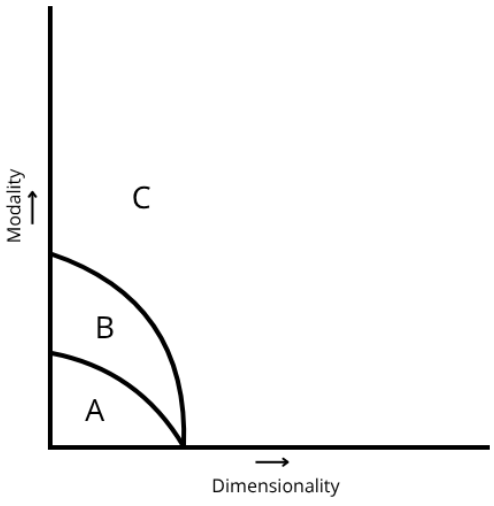

Figure 6.1: MOEA selection based on problem characteristics
A: Generational & Non-Adaptive (e.g. ϵ-NSGA-II)
B: Generational & Auto-Adaptive (e.g. Generational Borg)
C: Steady-State & Auto-Adaptive (e.g. Borg)

An important observation here though is that Borg is the only MOEA successfully continuing optimisation after 70000 NFE. For Borg, figures 5.1, 5.4 and 5.7 all show continued upward trends for hypervolume, archive size and epsilon progress respectively, suggesting that Borg has not converged yet, and is able to better approach the true Pareto front if given a larger computational budget. On the other hand, ϵ-NSGA-II and Generational Borg both show much slower and more stagnant improvement after 70000 NFE, suggesting they are likely not able to further explore the solution space by much.

A final observation particular to JUSTICE, is the difference in convergence behaviour during the early stage of the optimisation process. Despite Borg consistently achieving the best final

hypervolume values, it starts off converging more slowly than the other two MOEAs. Both ϵ-NSGA-II and Generational Borg initially show faster hypervolume gains, only later being overtaken by Borg. This behaviour is likely due to the difference in generational and steady-state nature. The generational nature of ϵ-NSGA-II and Generational Borg allow for quick population updates in greater batches, making for a higher early stage search pressure. Opposed to this is Borg's more gradual update strategy, resulting in slower initial hypervolume gains but more steady improvement later in the search. This suggests a key trade-off between the rate of initial convergence and the quality of the final converged solution set. The key takeaway here though is that for problems defined by high-dimensionality, such as JUSTICE, the continuous search pressure of a steady-state update mechanism is the most effective feature for ensuring maximum exploration of the solution space.

To give the reader a final illustration of how MOEA selection and MOEA characteristics concretely affect objective values, figure 6.2 presents all the non-dominated solutions found for JUSTICE per MOEA.

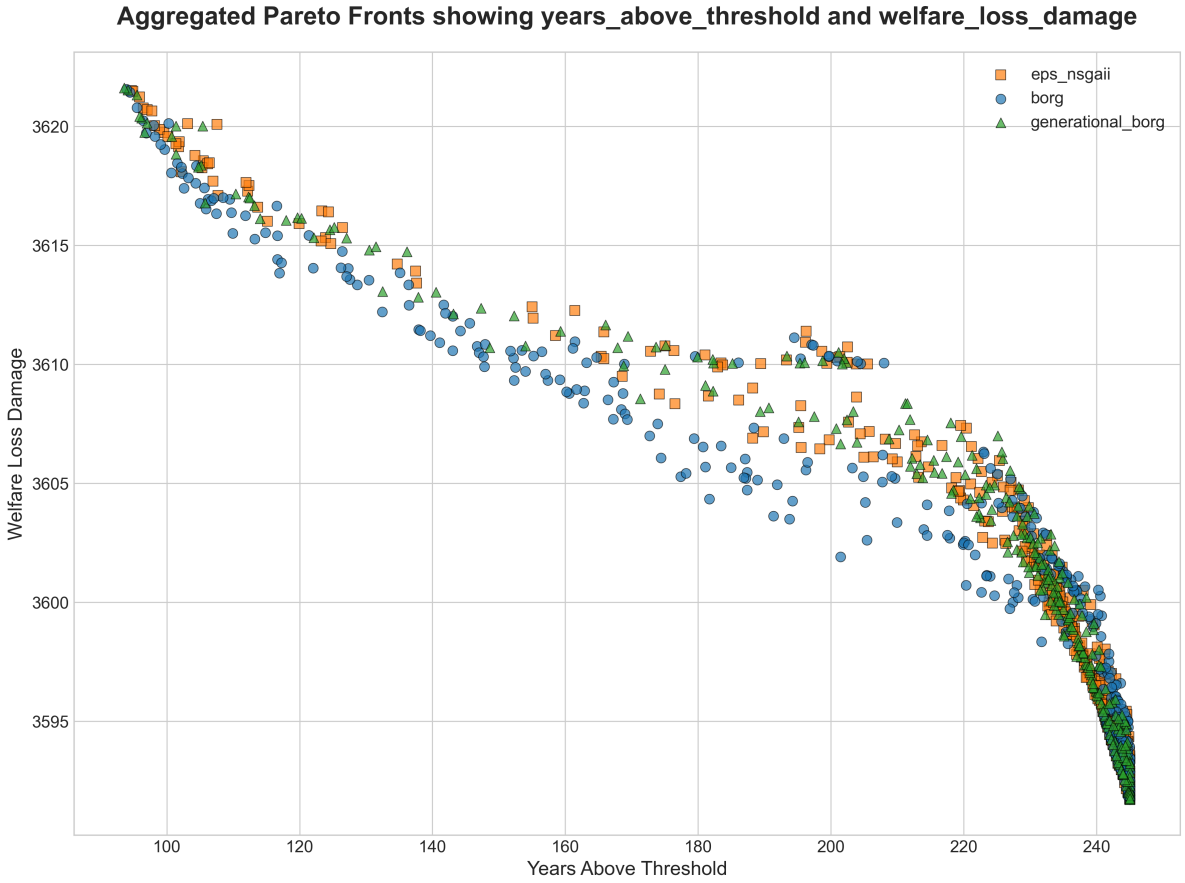

Figure 6.2: 'Years above threshold' versus 'Welfare loss damage' trade-offs found by the optimal solutions per MOEA

Per MOEA, solutions across all seeds and core counts were combined into one large set and subjected to a non-dominated sorting. The plot thus shows the aggregated Pareto fronts obtained by each MOEA on JUSTICE projected on the two objectives *'years_above_threshold'* and *'welfare_loss_damage'*. Only two objectives are shown here as a 4D-plot is less interpretable and this plot only serves the illustrative purpose of showing the policy relevance of MOEA selection. It is easy to see how the better hypervolume and generational distance Borg achieved compared to the other MOEAs translate to more high-quality solutions with better objective values. Almost all Borg solutions dominate the solutions found by the others, consistently getting closer to the optimal lower left corner (since a low welfare loss

damage and few years above the temperature threshold are desirable). The plot displays the expected relation of implementing less costly climate change abatement policies (lower welfare loss damage) also resulting in many more years of the earth being above the 2°C temperature increase threshold. The takeaway here is that better MOEA performance leads to objectively better policy insights, and rigorous MOEA selection should thus always be an integral part of climate research.

6.1.2. Computational Efficiency & Scalability

As already mentioned in chapter 5 a substantial limitation with regards to evaluating the MOEAs computational efficiency and scalability must be kept in mind. Due to unfortunate circumstances, node allocation for experiment runs ranged from one to four, and no MPI was implemented. This introduces uncontrollable variability in exact wall-clock runtimes and prevents valid direct comparisons between MOEAs and core counts from being made. However, since this happened for all core counts, seeds and MOEAs, the observed trends can still offer some general insights if interpreted with a large amount of caution. Therefore the results will still be briefly discussed in this section. Another important point is that for a selected amount of experiment runs, the exact same seed runs were rerun using exclusive node access. Performance metric results were cross-referenced and either completely identical or very similar. Therefore the obtained performance metric values are deemed to be valid, and we can make the following point. As expected, core count does not have any substantial effect on solution quality, only on the time needed to complete the optimisation processes.

Benchmark Problems

When considering DTLZ2 and figure 5.12, the most striking thing immediately is the substantially longer time Borg needs to complete its 70000 NFE than both ϵ-NSGA-II and Generational Borg. Intuitively one would attribute this to the overhead introduced by the time needed for all its adaptive mechanisms to update, which could be overkill for a simple problem like DTLZ2 (as Generational Borg is also slower than ϵ-NSGA-II). However, Borg shares these features with Generational Borg, making Borg's steady-state nature the only difference possibly explaining Borg's longer runtime. As explained, a steady-state MOEA performs selection and updates its archive every function evaluation. Again, this causes computational overhead, especially on simpler problems, and likely causes the longer runtime seen for Borg in DTLZ2. One conclusion that can therefore be drawn is that a steady-state nature influences runtimes significantly more than auto-adaptive features or ϵ-progress tracking. When considering the effect of core count on the runtimes, the largest decrease can be seen for Borg going from 16 to 32 cores. Given Borg's steady-state nature, which is more fit for parallel computing as outlined in chapter 2, this makes sense. However, one would then also expect a decrease when increasing from 32 to 48 cores. These results are therefore very likely influenced by the multi-node issue mentioned numerous times already.

Looking at the DTLZ3, very similar behaviour is observed. While Borg still is slowest, followed by Generational Borg and finally ϵ-NSGA-II, they all take slightly shorter than for DTLZ3, which is interesting. A potential reason for this could be that due to DTLZ3's increased deceptiveness, less solutions are found and added to the ϵ-archive. Therefore during each function evaluation, the solution needs to be compared to fewer others, reducing the wall-clock time needed per function evaluation. Again, we see a slight decrease in Borg runtime when increasing the core count, but no real conclusions can be drawn from this.

JUSTICE

Figure 5.8 shows really peculiar runtime behaviour for all MOEAs. Every MOEA shows an increase in runtime for an increase in core count. This is very unintuitive and is very likely caused by the multi-node and lack of MPI limitation. No further conclusions will be drawn from this plot. On the other hand given that the general trends with regards to hypervolume development over wall-clock time seen in figure 5.9 are similar for every core count, this behaviour is deemed to be valid. It confirms the point previously made about how Borg is slower in its initial hypervolume gains, both NFE- as wall-clock-time-wise, but always overtakes the other MOEAs slightly later in the search. The reason for this, as justified prior, likely is a lower initial search pressure for Borg due to its steady-state nature.

6.1.3. Notable Observations

This section will discuss two additional notable observations not previously mentioned in subsection 6.1.1 yet. Firstly, figure 5.15 shows a very steep and unusual decline in DTLZ3 archive size for two Borg seeds on 16 cores, which is not observed anywhere else. A possible reason for this can normally be a solution breakthrough, where the MOEA was caught in a certain area within the solution space, with many solutions just marginally different. If the MOEA suddenly has a search breakthrough and finds a small number of solutions which dominate a large number of other solutions, the archive size can see a large decrease as the worse solutions are removed from the archive. Given that DTLZ3 is a deceptive problem, this would not be unthinkable. However, for the two specific runs where this happened, figure 5.13 does not show any increase in hypervolume. This makes it unlikely that the archive size reduction was due to a solution breakthrough. The other reason coming to mind would be an archive management issue by the MOEA possibly introduced through the multi-node and lack of MPI issue. Though this too seems unlikely, as it has not happened in any other run. Additionally, it was double-checked and seed 60930, which shows the largest decrease in archive size, was actually run on one single node for the 16 core DTLZ3 experiment, eliminating any suspicion of the multi-node limitation introducing any archive management issues. Uncovering the reason for these observed archive size reductions will therefore be left for future research.

Another interesting observation to discuss is the higher inter-seed variability for Borg and Generational Borg on JUSTICE compared to ϵ-NSGA-II. Even with Borg always obtaining a lower final average spacing value than ϵ-NSGA-II, its seeds differ significantly more than ϵ-NSGA-II. This can be explained by considering the many auto-adaptive features of Borg and Generational Borg. Based on search progress these features can evolve in very different ways. Different random initial seeds can therefore set the MOEAs on relatively different paths from the beginning, leading to larger differences between seeds. On the other hand, ϵ-NSGA-II does not feature the same adaptive features, likely leading to a more deterministic search path. Though it is surprising that this difference in inter-seed variability is mainly observed for the spacing metric for JUSTICE.

6.2. Limitations

Despite many of the major limitations already having been discussed a number of times, this section will briefly synthesise the biggest impact of these limitations on the results and what has to be kept in mind when considering the final recommendations made in this study. For an additional detailed evaluation of the limitations the reader is referred to section 4.6.

Firstly, the uncontrollable variability in wall-clock times due to the non-exclusive node allocation introduced a severe degree of uncertainty in the comparisons of computational efficiency across runs and should be kept in mind. As confirmed, this has not affected the validity of the core performance metrics which can therefore be interpreted at face value. However, any insights drawn from the runtime data should be interpreted with much caution. Secondly, all the tested problems were configured using one single deterministic configuration. While this enabled clear, comparable and reproducible analyses, it is not certain that its conclusions hold for a wider variety of problem settings. Additionally, there is no guarantee that the chosen configuration is the most accurate in capturing real-world climate dynamics. Therefore, the conclusion on which MOEA is best fit to use for IAM-like problems must be interpreted with caution.

Another point of caution is the fact that all of the performance metrics for JUSTICE were calculated using a reference set based on the best solutions found by the MOEAs tested in this study. Therefore, there is no certainty, and little clarity, regarding to what degree the reference set really approaches the true Pareto front. Other MOEAs, not tested here, could potentially explore a much larger area of the solution space, rendering the supposed superiority of Borg over ϵ-NSGA-II and Generational Borg insignificant.

6.3. Contributions, Implications & Recommendations

This study has made a number of findings that have relevant implications for both the field of climate-economy modelling as well as the broader field of multi-objective optimisation. Additionally, a number of interesting directions for future research have been uncovered.

6.3.1. Contributions

This study has primarily contributed by providing the first rigorous, empirical performance comparison between generational versus steady-state and auto-adaptive versus non-adaptive MOEAs, namely ϵ-NSGA-II, Borg and Generational Borg, on a complex highly dimensional IAM. This way, it has decoupled the effects of auto-adaptivity and steady-state nature in MOEA architecture. By using Generational Borg as a control MOEA, and contrasting JUSTICE results with those of DTLZ2 and DTLZ3, the contributions of auto-adaptivity and population update scheme have been decoupled. This has shown that multi-modality in problems is best tackled by using both auto-adaptivity and a steady-state nature. When a problem is characterised by high-dimensionality and thus a large solution space, a steady-state nature has been shown to be the most effective feature in tackling the problem. This allows for more justified MOEA selection based on problem characteristics in future work.

Another contribution of this study has been an initial characterisation of the JUSTICE problem landscape. Comparing the performance of each MOEA on DTLZ2 and DTLZ3 with that on JUSTICE suggests two things. Firstly, JUSTICE likely only is moderately deceptive, definitely less so than DTLZ3. This is shown by the more extreme performance differences observed on DTLZ3 compared to JUSTICE. On the other hand, it is likely that JUSTICE features a very large solution space, as theoretically expected due to its high dimensionality. This is suggested by the lower hypervolume values obtained for JUSTICE compared to DTLZ2 (obviously) and the DTLZ3 runs that obtained hypervolume.

6.3.2. Implications

Firstly, the empirical evidence provided in this study advocates for the use of adaptive, steady-state MOEAs like Borg for complex, deceptive and highly dimensional problems such as JUSTICE. Simpler generational MOEAs like ϵ-NSGA-II and Generational Borg remain viable for less complex problems, but for real-world applications where solution quality is a top priority Borg's superior performance makes it the preferred choice. This is especially relevant for climate policy analysis, as the stakes of finding the optimal policy recommendations are very high.

Secondly, despite only being able to provide limited insights into true scalability of the MOEAs, this study has shown that scaling primarily affects runtimes and not solution quality. Therefore, generally, if solution quality is the main concern, increasing the NFE is the most effective consideration. However, in projects with tight time constraints, or heavy and parallelisable applications/problems, it is of course still more effective to run high NFE with higher core counts.

6.3.3. Future Research

First off, future research should address the main limitation present in this study. It would be interesting to see the true scalability of the three MOEAs when increasing the core count while using one exclusive node or implementing MPI. Additionally, the core counts used for optimisation can also be further increased, yielding insights into how scalability gains scale.

Secondly, given that Borg had not yet fully converged after 70000 NFE, it would be interesting to run the same experiments again. However, now run them with an increased computational budget, like 150000 NFE for example, to see how much closer Borg could approach the Pareto front.

A third direction for future research would be to make the results more robust and generalisable. This can be done by running the experiments on benchmark problems with different characteristics and different hyperparameter settings. Most interesting would be also to test additional JUSTICE configurations. Increasing the climate scenario ensemble size, using different welfare functions, different policy function approximators etc. The results could also be made more robust by increasing the number of seeds for each experiment. An additional advantage of doing so is that, at least for the setup used in this study, the resulting reference set used for performance metric calculations would more closely approximate the true (but unknown) Pareto front.

Lastly, as observed in the results, a generational versus a steady-state approach has a noticeable effect on the initial speed with which good solutions propagate in the population. In MOEAs with auto-adaptive mechanisms, which adapt based on high-quality solutions, the speed with which these good solutions propagate in the population thus determines how quickly the auto-adaptive operators can base adaptation on more high-quality solutions. Evaluating the exact interplay between

auto-adaptivity and population update mechanism and quantifying its effect is an interesting direction for future research to further explore.

6.3.4. Societal and Policy Impact

The ability to efficiently optimise complex IAMs, like JUSTICE, directly supports the development of effective, equitable and sustainable climate policies. MOEAs help policymakers make informed trade-offs between conflicting objectives, such as welfare and emission reduction, by finding a high-quality set of Pareto optimal policies (as illustrated by figure 6.2). This study has aimed to provide researchers with a number of insights and recommendations to enable more effective MOEA use in IAM optimisation and thus contribute to better future policy analysis and decision support in the face of global climate challenges.

7

Conclusion

To conclude, this study has researched how and why the three MOEAs ϵ-NSGA-II, Borg and Generational Borg differ regarding their computational efficiency, convergence dynamics and solution quality when optimising a complex IAM like JUSTICE. The findings have been contextualised and validated by running identical experiments for the DTLZ2 and DTLZ3 benchmark problems. This way the knowledge gap identified in chapter 3, being the lack of empirical research regarding MOEA performance in real-world, high-dimensional and policy-relevant problems, has been addressed. Using all the findings and insights discussed in previous sections, we can now first answer the sub-research questions formulated in chapter 3;

1. ***How do ϵ-NSGA-II, Borg and Generational Borg differ in convergence performance metrics and solution quality?***

   The results presented in chapter 5 and discussed in chapter 6 have demonstrated substantial differences between ϵ-NSGA-II, Borg and Generational Borg. Borg was able to consistently obtain the best hypervolume, generational distance and epsilon indicator values across practically all problems, seeds and core counts. This is indicative of a very strong ability to navigate complex problem landscapes. ϵ-NSGA-II showed comparative performance on a simple problem like DTLZ2, and also did reasonably well on JUSTICE, but completely failed on DTLZ3. Showing how its fixed operators and generational nature make it very vulnerable to deceptive multi-modal problems. Lastly, Generational Borg takes an intermediate position. Its lack of steady-state updates inhibit it in reaching similar solution quality to Borg on JUSTICE, but its auto-adaptive features make it generally outperform ϵ-NSGA-II on DTLZ3.

2. ***Are the observed differences in JUSTICE also observed in benchmark problems like DTLZ2 and DTLZ3?***

   Trendwise, the observed differences in JUSTICE are also observed in the benchmark problems DTLZ2 and DTLZ3. This means that Borg's superior performance, and Generational Borg's (mostly) similar performance to ϵ-NSGA-II are observed in every problem, but that performance gaps widen with model complexity. DTLZ2's simple unimodal nature makes the MOEAs obtain final performance metric values that are closer together than is the case with JUSTICE, but retains the same MOEA ranking. For the case of DTLZ3, likely more deceptive than JUSTICE, the obtained final performance metrics are more apart, with ϵ-NSGA-II lacking any hypervolume as most striking example. Though, again, the same ranking is retained, with Borg performing best, followed by Generational Borg and ϵ-NSGA-II.

3. ***Does increasing or decreasing the number of cores used in optimisation influence the differences between the different MOEA performances?***

   The answer to this question is very nuanced. First off, in- or decreasing the core count did not systemically influence the final solution quality. For practically all performance metrics, MOEAs and problems, a differing core count did not show any real effect. However, core count definitely does

influence final runtimes. As theoretically expected, an increase in core count seemed to have the biggest positive impact on Borg's runtime, given its steady-state nature, and was especially effective on more complex and large problems like JUSTICE. However, given the numerously mentioned limitation of lacking node-exclusivity for a large number of runs, as well as the lack of MPI, no exactly quantified conclusions could be drawn from the data obtained in this study.

4. ***What recommendation with regards to MOEA choice can be made based on problem characteristics and available resources?***

   Based on the observed behaviour, a number of recommendations for choice of MOEA can be made. Firstly, Borg, or any steady-state MOEA with auto-adaptive features, always is a good choice when solely considering solution quality. The outcomes showed that Borg was the top performer for practically all runs. However, if the problem to be optimised is known to be simple and unimodal and a short runtime is important given the scope and resources of the study, a simpler and generational MOEA without any advanced auto-adaptive could be the better choice. Its solution quality will not differ much from a MOEA like Borg, while the runtime will be significantly shorter.

   In all other cases a more advanced and steady-state MOEA, like Borg, is the better choice. For problems that are more difficult, like multi-modal or very high dimensional problems such as DTLZ3 and JUSTICE, Borg simply showed that it was able to achieve significantly better performance. For cases where computational resources for example include HPC access, Borg, or any steady-state MOEA, also is the obvious choice. Despite the limitations in this study, steady-state MOEAs are theoretically (and have shown so in practice) much more scalable than generational MOEAs, with better runtime performance than their generational counterparts.

By synthesising the answers to the sub-research questions, the main research question;

How and why do ϵ-NSGA-II, Borg and Generational Borg differ regarding their computational efficiency, convergence dynamics and solution quality when optimising a demanding model such as JUSTICE using the EMODPS framework?

can now be properly answered. The empirical results have led to the conclusion that Borg consistently achieves the highest solution quality, increasing the performance gap with the other MOEAs as problem complexity, due to multi-modality or high dimensionality for example, increases. Its steady-state nature together with auto-adaptive features like operator selection, tournament sizing and ϵ-progress based restarts enabled it to keep outperforming the other MOEAs. The advantage these features give Borg especially materialises in more complex problems, confirming its suitability for use in optimising a demanding real-world problem like JUSTICE. By contrast, ϵ-NSGA-II proved to be a faster and lighter alternative that still holds it own on simple problems like DTLZ2. However, with increasing complexity, especially when introducing multi-modality, its lack of advanced features like those of Borg make its performance drastically decline. This became most apparent on DTLZ3. The adaptive features Generational Borg shares with Borg made it perform mostly on par with or slightly better than ϵ-NSGA-II, at the cost of a slightly increased runtime. However, as problem complexity increased it became more limited by its lack of a steady-state nature making it lag behind the performance of Borg.

In terms of computational efficiency, Borg's steady-state nature did introduce some overhead, mainly on simpler or smaller problems like the DTLZ problems, but it did enable better continued progress on difficult highly-dimensional problems like JUSTICE. The results did hint at Borg being more suited for high scalability, as theoretically expected. However, due to the discussed experimental limitations, mainly the variable node allocation and lack of MPI, the runtime comparisons must further be interpreted very cautiously.

To build on the findings made by this study, future research is suggested to explore the following directions. First, future work should better examine efficiency gains through scalability by running the experiments on truly exclusive nodes, or implementing MPI. Additionally, the MOEAs could be given a higher computational budget to examine what the upper limits of convergence are for problems like JUSTICE, especially considering Borg has not yet fully converged after 70000 NFE for JUSTICE. The experiments could also be run with different MOEAs and different JUSTICE configurations to increase the robustness of currently made observations and findings.

Finally, this research has demonstrated that choice of MOEA critically impacts performance and solution quality, essential when used for real-world policy design under deep uncertainty. The findings have shown that especially steady-state architectures together with auto-adaptive features are essential when navigating complex solution landscapes in IAMs like JUSTICE, aiding the EPA philosophy of model-driven policy design. And while generational MOEAs remain viable options for simpler problems, especially when time is constrained, adaptive steady-state MOEAs like Borg are now shown to better enhance our ability to discover robust and equitable climate policies. Therefore, this study has achieved its goal of aiding researchers in the development of actionable policies combating humanity's most urgent challenge, climate change.

References

Ackerman, F., DeCanio, S. J., Howarth, R. B., & Sheeran, K. (2009). Limitations of integrated assessment models of climate change. *Climatic change*, *95*, 297–315.

Biswas, P., Akoluk, D., Zatarain Salazar, J., Kwakkel, J., & Verbraeck, A. (2023). Exploring the impact of modelling assumptions on distributive justice using justice.

Biswas, P., Salazar, J. Z., Kwakkel, J., & Marangoni, G. (2025). Prioritizing the welfare of vulnerable nations promotes equitable achievement of paris target.

Branke, J., & Deb, K. (2005). Integrating user preferences into evolutionary multi-objective optimization. In *Knowledge incorporation in evolutionary computation* (pp. 461–477). Springer.

Carlino, A., Gazzotti, P., Tavoni, M., & Castelletti, A. (2022). Cooperative adaptive climate policies: An application of the EMODPS algorithm to deal with multiple objectives and deep uncertainty in the RICE50+ integrated assessment model.

Deb, K., Agrawal, R. B., et al. (1995). Simulated binary crossover for continuous search space. *Complex systems*, *9*(2), 115–148.

Deb, K., Mohan, M., & Mishra, S. (2005). Evaluating the ϵ-domination based multi-objective evolutionary algorithm for a quick computation of pareto-optimal solutions. *Evolutionary computation*, *13*(4), 501–525.

Deb, K., Pratap, A., Agarwal, S., & Meyarivan, T. (2002). A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE transactions on evolutionary computation*, *6*(2), 182–197.

Deb, K., Thiele, L., Laumanns, M., & Zitzler, E. (2005). Scalable test problems for evolutionary multiobjective optimization. In *Evolutionary multiobjective optimization: Theoretical advances and applications* (pp. 105–145). Springer.

Delgado, M., Cuellar, M. P., & Pegalajar, M. C. (2008). Multiobjective hybrid optimization and training of recurrent neural networks. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, *38*(2), 381–403.

Durillo, J. J., Nebro, A. J., Luna, F., & Alba, E. (2008). A study of master-slave approaches to parallelize nsga-ii. *2008 IEEE international symposium on parallel and distributed processing*, 1–8.

ProjectPlatypus. (2024). Platypus: A python library for multi-objective optimization. https://github.com/Project-Platypus/Platypus

Giuliani, M., Castelletti, A., Pianosi, F., Mason, E., & Reed, P. M. (2016). Curses, tradeoffs, and scalable management: Advancing evolutionary multiobjective direct policy search to improve water reservoir operations. *Journal of Water Resources Planning and Management*, *142*(2), 04015050.

Giuliani, M., Quinn, J. D., Herman, J. D., Castelletti, A., & Reed, P. M. (2017). Scalable multiobjective control for large-scale water resources systems under uncertainty. *IEEE Transactions on Control Systems Technology*, *26*(4), 1492–1499.

Guerreiro, A. P., Fonseca, C. M., & Paquete, L. (2021). The hypervolume indicator: Computational problems and algorithms. *ACM Computing Surveys (CSUR)*, *54*(6), 1–42.

Gupta, A. (2020). *Exploring simulation-based decision making frameworks for energy management in electrical networks of the future*. Cornell University.

Hadka, D., Madduri, K., & Reed, P. (2013). Scalability analysis of the asynchronous, master-slave borg multiobjective evolutionary algorithm. *2013 IEEE International Symposium on Parallel & Distributed Processing, Workshops and Phd Forum*, 425–434.

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

Hadka, D., & Reed, P. (2015). Large-scale parallelization of the borg multiobjective evolutionary algorithm to enhance the management of complex environmental systems. *Environmental Modelling & Software*, *69*, 353–369.

Hamdan, M. (2010). On the disruption-level of polynomial mutation for evolutionary multi-objective optimisation algorithms. *Computing and Informatics*, *29*(5), 783–800.

Huband, S., Hingston, P., Barone, L., & While, L. (2006). A review of multiobjective test problems and a scalable test problem toolkit. *IEEE Transactions on Evolutionary Computation*, *10*(5), 477–506.

IPCC. (2023). Synthesis report of the IPCC sixth assessment report (AR6) summary for policymakers. https://www.ipcc.ch/report/ar6/syr/downloads/report/IPCC_AR6_SYR_SPM.pdf

Ishibuchi, H., Imada, R., Setoguchi, Y., & Nojima, Y. (2016). Performance comparison of nsga-ii and nsga-iii on various many-objective test problems. *2016 IEEE Congress on Evolutionary Computation (CEC)*, 3045–3052.

Ishibuchi, H., Tsukamoto, N., Sakane, Y., & Nojima, Y. (2010). Indicator-based evolutionary algorithm with hypervolume approximation by achievement scalarizing functions. *Proceedings of the 12th annual conference on Genetic and evolutionary computation*, 527–534.

Kim, G. J., & Kim, Y.-O. (2021). How does the coupling of real-world policies with optimization models expand the practicality of solutions in reservoir operation problems? *Water Resources Management*, *35*(10), 3121–3137.

Kita, H., Ono, I., & Kobayashi, S. (1999). Theoretical analysis of the unimodal normal distribution crossover for real-coded genetic algorithms. *Transactions of the Society of Instrument and Control Engineers*, *35*(11), 1333–1339.

Knowles, J., & Corne, D. (2002). On metrics for comparing nondominated sets. *Proceedings of the 2002 Congress on Evolutionary Computation. CEC'02 (Cat. No. 02TH8600)*, *1*, 711–716.

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

Kwakkel, J. H., & contributors. (2024). Exploratory modeling and analysis workbench.

Li, J., Zhang, W., & Yeh, W. W.-G. (2021). A proposed multi-objective, multi-stage stochastic programming with recourse model for reservoir management and operation. *Water Resources Research*, *57*(10), e2020WR029200.

Li, M. (2021). Is our archiving reliable? multiobjective archiving methods on "simple" artificial input sequences. *ACM Transactions on Evolutionary Learning and Optimization*, *1*(3), 1–19.

Limmer, S., & Fey, D. (2017). Comparison of common parallel architectures for the execution of the island model and the global parallelization of evolutionary algorithms. *Concurrency and Computation: Practice and Experience*, *29*(9), e3797.

Marangoni, G., Lamontagne, J. R., Quinn, J. D., Reed, P. M., & Keller, K. (2021). Adaptive mitigation strategies hedge against extreme climate futures. *Climatic Change*, *166*(3), 37.

Martí, L., Segredo, E., Sánchez-Pi, N., & Hart, E. (2018). Selection methods and diversity preservation in many-objective evolutionary algorithms. *Data Technologies and Applications*, *52*(4), 502–519.

Mashwani, W. K., Salhi, A., Jan, M. A., Khanum, R. A., & Sulaiman, M. (2015). Impact analysis of crossovers in a multi-objective evolutionary algorithm. *Science International*, *27*(6), 4943–4956.

Nordhaus, W. D. (1993). Optimal greenhouse-gas reductions and tax policy in the ”DICE” model. *The American Economic Review*, *83*(2), 313–317.

Nordhaus, W. D., & Yang, Z. (1996). A regional dynamic general-equilibrium model of alternative climate-change strategies. *The American Economic Review*, 741–765.

Rajakumar, B. (2013). Static and adaptive mutation techniques for genetic algorithm: A systematic comparative analysis. *International Journal of Computational Science and Engineering*, *8*(2), 180–193.

Salazar, J. Z., Hadka, D., Reed, P., Seada, H., & Deb, K. (2024). Diagnostic benchmarking of many-objective evolutionary algorithms for real-world problems. *Engineering Optimization*, 1–22.

Salazar, J. Z., Kwakkel, J., & Witvliet, M. (2022). Exploring global approximators for multiobjective reservoir control. *IFAC-PapersOnLine*, *55*(33), 34–41.

Salazar, J. Z., Reed, P. M., Herman, J. D., Giuliani, M., & Castelletti, A. (2016). A diagnostic assessment of evolutionary algorithms for multi-objective surface water reservoir control. *Advances in water resources*, *92*, 172–185.

Shin, S.-Y., Lee, I.-H., Kim, D., & Zhang, B.-T. (2005). Multiobjective evolutionary optimization of dna sequences for reliable dna computing. *IEEE transactions on evolutionary computation*, *9*(2), 143–158.

Smith, G. D., Steele, N. C., Albrecht, R. F., Dahal, K., & McDonald, J. (1998). Generational and steady-state genetic algorithms for generator maintenance scheduling problems. *Artificial Neural Nets and Genetic Algorithms: Proceedings of the International Conference in Norwich, UK, 1997*, 259–263.

Stehfest, E., van Vuuren, D., Bouwman, L., Kram, T., et al. (2014). *Integrated assessment of global environmental change with IMAGE 3.0: Model description and policy applications*. Netherlands Environmental Assessment Agency (PBL).

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

Yen, G. G., & He, Z. (2013). Performance metric ensemble for multiobjective evolutionary algorithms. *IEEE Transactions on Evolutionary Computation*, *18*(1), 131–144.

Zăvoianu, A.-C., Lughofer, E., Koppelstätter, W., Weidenholzer, G., Amrhein, W., & Klement, E. P. (2013). On the performance of master-slave parallelization methods for multi-objective evolutionary algorithms. *Artificial Intelligence and Soft Computing: 12th International Conference, ICAISC 2013, Zakopane, Poland, June 9-13, 2013, Proceedings, Part II 12*, 122–134.

Zhang, Q., & Li, H. (2007). Moea/d: A multiobjective evolutionary algorithm based on decomposition. *IEEE Transactions on evolutionary computation*, *11*(6), 712–731.

Zhang, Z. (2008). Multiobjective optimization immune algorithm in dynamic environments and its application to greenhouse control. *Applied Soft Computing*, *8*(2), 959–971.

Zhou, A., Qu, B.-Y., Li, H., Zhao, S.-Z., Suganthan, P. N., & Zhang, Q. (2011). Multiobjective evolutionary algorithms: A survey of the state of the art. *Swarm and evolutionary computation*, *1*(1), 32–49.

Zitzler, E., & Künzli, S. (2004). Indicator-based selection in multiobjective search. *International conference on parallel problem solving from nature*, 832–842.

A

MOEA Parameter Settings

Table A.1: Detailed Parameter Settings for MOEAs

Parameter	ϵ-NSGA-II	Borg	Generational Borg
Common Parameter Settings			
Stopping Criterion (NFE)	70000	70000	70000
ϵ-values (JUSTICE)	Welfare: 0.01 Years > Threshold: 0.25 Damage Loss: 10 Abatement Loss: 10	Welfare: 0.01 Years > Threshold: 0.25 Damage Loss: 10 Abatement Loss: 10	Welfare: 0.01 Years > Threshold: 0.25 Damage Loss: 10 Abatement Loss: 10
ϵ-values (DTLZ2, DTLZ3)	0.05 (all objectives)	0.05 (all objectives)	0.05 (all objectives)
Initial Population Size (P_{init})	100	100	100
Pop-to-Archive Ratio (λ)	4	4	4
Min Population Size (P_{min})	10	10	10
Max Population Size (P_{max})	10000	10000	10000
Mutator for Pop. Sizing[1]	`UM`(1.0)	`UM`(1.0)	`UM`(1.0)
Core Algorithmic Strategy			
Population Management	Generational	Steady-State	Generational
Parallel Execution Model	Synchronous Master-Slave	Asynchronous Master-Slave	Synchronous Master-Slave
Archive Type	ϵ-Box Archive	ϵ-Box Archive	ϵ-Box Archive
Selection Mechanism	Tournament Selection Size 2	Tournament Selection Size 2	Tournament Selection Size 2
MOEA-Specific Variation Operators			
ϵ-NSGA-II			
Crossover	`SBX`	—	—
Crossover Probability (P_c)	1.0	—	—
Distribution Index (η_c)	15	—	—
Mutation	`PM`	—	—
Mutation Probability (P_m)	1.0	—	—
Distribution Index (η_m)	20	—	—
Borg & Generational Borg			
Operator Selection	—	Adaptive	Adaptive
Initial Operator Probs.	—	Uniform $(1/K)$[2]	Uniform $(1/K)$[2]

Continued on next page

Table A.1 – continued from previous page

Parameter	**ϵ-NSGA-II**	**Borg**	**Generational Borg**
Prob. Update Constant (ζ)	—	N/A[3]	N/A[3]
Operator Suite:			
`SBX`	—	P_c: 1.0[4] η_c: 15	P_c: 1.0[4] η_c: 15
`DE`	—	Crossover Rate: 0.1 Step Size: 0.5	Crossover Rate: 0.1 Step Size: 0.5
`PCX`	—	n parents: 10 n offspring: 2 η: 0.1 ζ: 0.1	n parents: 10 n offspring: 2 η: 0.1 ζ: 0.1
`SPX`	—	n parents: 10 n offspring: 2 Expansion: 0.3	n parents: 10 n offspring: 2 Expansion: 0.3
`UNDX`	—	n parents: 10 n offspring: 2 ζ: 0.5 η: 0.35	n parents: 10 n offspring: 2 ζ: 0.5 η: 0.35
`UM` (as operator)	—	P_m: 1.0[5]	P_m: $1/nvars$
Secondary Mutation with Crossovers	—	`PM` for `SBX` and `DE`	`PM` for all operators
`PM` P_m	—	1.0[6]	$1/nvars$
`PM` η_m	—	20	20
MOEA-Specific Restart Mechanisms			
ϵ-NSGA-II			
Restart Trigger	Stagnation (fixed NFE interval OR Pop/Archive imbalance)	—	—
NFE Interval	1000	—	—
Pop/Archive Imbalance Condition	$P_{size} - (\lambda \cdot \lvert A\rvert) > 0.25 \cdot (\lambda \cdot \lvert A\rvert)$	—	—
Population Reinitialisation	Archive + Mutated Archive Solutions	—	—
Restart Mutation Type	`UM`	—	—
Mutation Probability (P_m)	1.0	—	—
Borg & Generational Borg			
Restart Trigger	—	ϵ-progress stagnation OR Pop/Archive imbalance OR Max Iteration Window	ϵ-progress stagnation OR Pop/Archive imbalance OR Max Iteration Window
Pop/Archive Imbalance Condition	—	$P_{size} - (\lambda \cdot \lvert A\rvert) > 0.25 \cdot (\lambda \cdot \lvert A\rvert)$	$P_{size} - (\lambda \cdot \lvert A\rvert) > 0.25 \cdot (\lambda \cdot \lvert A\rvert)$
ϵ-progress Threshold	—	No new archive improvements	No new archive improvements
No. of Stagnant Windows (K)	—	1 window	1 window
ϵ-progress Window (stagnation check frequency)	—	100 iterations	100 iterations
Max Iteration Window (time-based trigger)	—	1000 iterations	1000 iterations

Continued on next page

Table A.1 – continued from previous page

Parameter	**ϵ-NSGA-II**	**Borg**	**Generational Borg**
Population Reinitialisation	—	Archive + Mutated Archive Solutions	Archive + Mutated Archive Solutions
Restart Mutation Type	`UM`(1.0)	`UM` (Adaptive Prob.)[7]	`UM`(1.0)
Adaptive Tournament Size	—	Yes (ratio: 0.02)	Yes (ratio: 0.02)

[1] Refers to the mutation operator used when filling the population during adaptive population sizing restarts. Default internal probability for 'UM' is 1.0.

[2] K is the number of operators in the suite (6 for both Borg and Generational Borg).

[3] Operator selection probabilities managed by Platypus 'Multimethod''s internal logic; no explicit ζ parameter set.

[4] The probability parameter within SBX itself; selection probability for the SBX operator by 'Multimethod' is adaptive.

[5] Internal mutation probability (P_m) for the 'UM' operator when part of Borg's operator suite. Default for Platypus 'UM()' is 1.0.

[6] Internal mutation probability (P_m) for the 'PM' operator when used as secondary mutation in Borg. Default for Platypus 'PM()' is 1.0.

[7] For Borg, the 'UM' used during restarts (initially 'UM(1.0)') has its internal probability P_m adapted by the formula $P = P_{base} + (1 - P_{base})/nvars$, where P_{base} changes from 0 to 1.0 based on 'base_mutation_index'. This means P_m ranges from approx. $1/nvars$ to 1.0.

B

Results per Seed

B.1. Hypervolume

Table B.1: Final DTLZ2 hypervolume

		Hypervolume		
Cores	**Seed**	**ϵ-NSGA-II**	**Borg**	**Generational Borg**
16	12345	0.597566	0.618505	0.592487
	23403	0.595832	0.619347	0.563799
	39349	0.597660	0.619780	0.588743
	60930	0.599087	0.619788	0.593782
	93489	0.597203	0.619188	0.595135
	Average	0.597470	0.619322	0.586789
32	12345	0.597846	0.618262	0.589443
	23403	0.600720	0.618975	0.589980
	39349	0.598467	0.619772	0.596452
	60930	0.599462	0.618559	0.596288
	93489	0.598307	0.619313	0.593353
	Average	0.598960	0.618976	0.593103
48	12345	0.597967	0.619385	0.587689
	23403	0.595900	0.618809	0.577886
	39349	0.602369	0.619455	0.593747
	60930	0.600007	0.618958	0.595826
	93489	0.600431	0.619336	0.581679
	Average	0.599335	0.619188	0.587365
Total Average		0.598588	0.619162	0.589086

Table B.2: Final DTLZ3 Hypervolume

		Hypervolume		
Cores	**Seed**	**ϵ-NSGA-II**	**Borg**	**Generational Borg**
16	12345	0	0.043492	0
	23403	0	0.609518	0.518801
	39349	0	0.594471	0
	60930	0	0	0
	93489	0	0.586619	0
	Average	0	0.366820	0.103760
32	12345	0	0.582271	0.517945
	23403	0	0.552912	0
	39349	0	0.594028	0.548913
	60930	0	0.612469	0
	93489	0	0.581841	0
	Average	0	0.584704	0.213371
48	12345	0	0.567212	0.560638
	23403	0	0.551911	0.014987
	39349	0	0.595163	0
	60930	0	0.574748	0
	93489	0	0.576914	0
	Average	0	0.573190	0.115125
Total Average		0	0.508238	0.144085

Table B.3: Final JUSTICE Hypervolume

		Hypervolume		
Cores	**Seed**	**ϵ-NSGA-II**	**Borg**	**Generational Borg**
16	12345	0.312357	0.352536	0.303605
	23403	0.321510	0.373744	0.308638
	39349	0.318113	0.385087	0.303087
	60930	0.286710	0.371754	0.332334
	93489	0.318505	0.387342	0.321346
	Average	0.311439	0.374093	0.313802
32	12345	0.312357	0.397780	0.303605
	23403	0.321510	0.365839	0.308638
	39349	0.318113	0.355803	0.303087
	60930	0.286710	0.360913	0.332334
	93489	0.318505	0.352728	0.321346
	Average	0.311439	0.366613	0.313802
48	12345	0.312711	0.357900	0.303605
	23403	0.321510	0.348593	0.308638
	39349	0.318113	0.332404	0.303087
	60930	0.286710	0.392831	0.302281
	93489	0.318505	0.364668	0.321346
	Average	0.311510	0.359279	0.307792
Total Average		0.311463	0.366662	0.311799

B.2. Generational Distance

Table B.4: Final DTLZ2 Generational Distance

		Generational Distance		
Cores	**Seed**	**ϵ-NSGA-II**	**Borg**	**Generational Borg**
16	12345	0.003495	0.003296	0.003315
	23403	0.003515	0.003306	0.003390
	39349	0.003555	0.003401	0.003287
	60930	0.003463	0.003417	0.003278
	93489	0.003462	0.003352	0.003320
	Average	0.003498	0.003354	0.003318
32	12345	0.003509	0.003326	0.003282
	23403	0.003553	0.003346	0.003328
	39349	0.003468	0.003282	0.003290
	60930	0.003576	0.003417	0.003239
	93489	0.003352	0.003281	0.003282
	Average	0.003492	0.003331	0.003284
48	12345	0.003524	0.003299	0.003287
	23403	0.003458	0.003322	0.003293
	39349	0.003489	0.003347	0.003278
	60930	0.003442	0.003413	0.003343
	93489	0.003475	0.003326	0.003240
	Average	0.003478	0.003341	0.003288
Total Average		0.003489	0.003342	0.003297

Table B.5: Final DTLZ3 Generational Distance

		Generational Distance		
Cores	**Seed**	**ϵ-NSGA-II**	**Borg**	**Generational Borg**
16	12345	1.379975	0.034825	0.227291
	23403	1.009488	0.003325	0.003976
	39349	1.164905	0.003449	1.076368
	60930	3.090794	0.186638	0.297484
	93489	2.179744	0.003132	3.249727
	Average	1.764981	0.046274	0.970969
32	12345	3.449566	0.003818	0.004479
	23403	5.260605	0.003348	1.910214
	39349	4.566423	0.003219	0.004118
	60930	2.230648	0.003299	1.744931
	93489	2.352995	0.003242	3.306576
	Average	3.572047	0.003385	1.394063
48	12345	6.147696	0.003046	0.004377
	23403	4.502291	0.003164	0.033674
	39349	3.119187	0.003247	0.626203
	60930	4.122932	0.003189	2.692920
	93489	6.159086	0.003531	1.457703
	Average	4.810238	0.003236	0.962975
Total Average		3.382422	0.017632	1.109336

Table B.6: Final JUSTICE Generational Distance

		Generational Distance		
Cores	**Seed**	**ϵ-NSGA-II**	**Borg**	**Generational Borg**
16	12345	0.002794	0.001910	0.003144
	23403	0.002308	0.001981	0.002901
	39349	0.002797	0.001855	0.002975
	60930	0.003803	0.001735	0.002411
	93489	0.002571	0.001610	0.002355
	Average	0.002854	0.001818	0.002758
32	12345	0.002794	0.001542	0.003144
	23403	0.002308	0.001658	0.002901
	39349	0.002797	0.001696	0.002975
	60930	0.003803	0.001662	0.002411
	93489	0.002571	0.001901	0.002355
	Average	0.002854	0.001692	0.002758
48	12345	0.002503	0.001859	0.003144
	23403	0.002308	0.002156	0.002901
	39349	0.002797	0.002240	0.002975
	60930	0.003803	0.001272	0.003601
	93489	0.002571	0.001732	0.002355
	Average	0.002796	0.001852	0.002995
Total Average		0.002835	0.001787	0.002837

B.3. Epsilon Indicator

Table B.7: Final DTLZ2 Epsilon Indicator

		Epsilon Indicator		
Cores	**Seed**	**ϵ-NSGA-II**	**Borg**	**Generational Borg**
16	12345	0.055823	0.053433	0.063395
	23403	0.054329	0.050966	0.091709
	39349	0.060740	0.046814	0.063651
	60930	0.053929	0.046923	0.060367
	93489	0.054980	0.067037	0.059801
	Average	0.055960	0.053034	0.067785
32	12345	0.059378	0.052119	0.064971
	23403	0.057865	0.050969	0.056982
	39349	0.069099	0.049092	0.059113
	60930	0.058418	0.052757	0.064051
	93489	0.054992	0.053297	0.076156
	Average	0.059951	0.051647	0.064254
48	12345	0.055960	0.052379	0.066343
	23403	0.064622	0.045496	0.076061
	39349	0.065324	0.052283	0.069194
	60930	0.057030	0.044897	0.059894
	93489	0.059596	0.046414	0.071493
	Average	0.060506	0.048294	0.068597
Total Average		0.058839	0.050992	0.066879

Table B.8: Final DTLZ3 Epsilon Indicator

		Epsilon Indicator		
Cores	**Seed**	**ϵ-NSGA-II**	**Borg**	**Generational Borg**
16	12345	8.726062	0.692393	2.572081
	23403	6.486144	0.057406	0.104258
	39349	3.715129	0.062428	5.211905
	60930	13.126034	2.040228	1.855365
	93489	11.098678	0.070885	12.793205
	Average	8.630409	0.584668	4.507363
32	12345	9.316437	0.089556	0.142555
	23403	13.573567	0.098638	8.240333
	39349	20.118402	0.063936	0.087953
	60930	9.492398	0.058896	5.460742
	93489	9.852280	0.075979	15.794788
	Average	12.470617	0.077401	5.945274
48	12345	16.931851	0.071506	0.098047
	23403	16.787490	0.090818	0.726022
	39349	11.135266	0.060526	3.169642
	60930	18.407931	0.080733	10.924057
	93489	16.153053	0.081406	5.403302
	Average	15.883118	0.076998	4.064214
Total Average		12.328048	0.246356	4.838950

Table B.9: Final JUSTICE Epsilon Indicator

		Epsilon Indicator		
Cores	**Seed**	**ϵ-NSGA-II**	**Borg**	**Generational Borg**
16	12345	0.105828	0.056688	0.086837
	23403	0.094455	0.066369	0.091348
	39349	0.081748	0.055966	0.104837
	60930	0.129602	0.071653	0.088031
	93489	0.085949	0.051806	0.095757
	Average	0.099516	0.060496	0.093362
32	12345	0.105828	0.055345	0.086837
	23403	0.094455	0.055684	0.091348
	39349	0.081748	0.053628	0.104837
	60930	0.129602	0.057526	0.088031
	93489	0.085949	0.057084	0.095757
	Average	0.099516	0.055853	0.093362
48	12345	0.079779	0.061246	0.086837
	23403	0.094455	0.077761	0.091348
	39349	0.081748	0.071383	0.104837
	60930	0.129602	0.043916	0.090969
	93489	0.085949	0.059105	0.095757
	Average	0.094307	0.062682	0.093950
Total Average		0.097780	0.059677	0.093558

B.4. Archive Size

Table B.10: Final DTLZ2 Archive Size

		Archive Size		
Cores	**Seed**	**ϵ-NSGA-II**	**Borg**	**Generational Borg**
16	12345	1166	1386	1239
	23403	1155	1380	1184
	39349	1118	1345	1268
	60930	1175	1321	1264
	93489	1171	1372	1264
	Average	1157	1360	1243
32	12345	1169	1362	1278
	23403	1142	1347	1253
	39349	1149	1389	1287
	60930	1147	1295	1287
	93489	1239	1401	1284
	Average	1169	1358	1277
48	12345	1184	1403	1256
	23403	1165	1374	1222
	39349	1175	1333	1289
	60930	1176	1322	1227
	93489	1190	1380	1242
	Average	1178	1362	1247
Total Average		1168	1360	1255

Table B.11: Final DTLZ3 Archive Size

		Archive Size		
Cores	**Seed**	**ϵ-NSGA-II**	**Borg**	**Generational Borg**
16	12345	109	815	338
	23403	75	1301	754
	39349	61	1203	48
	60930	83	330	76
	93489	93	1362	47
	Average	84	1002	252
32	12345	25	966	585
	23403	19	1135	48
	39349	100	1327	738
	60930	82	1347	51
	93489	42	1275	70
	Average	53	1210	298
48	12345	12	1378	678
	23403	38	1279	982
	39349	52	1297	107
	60930	110	1300	47
	93489	74	1061	23
	Average	57	1263	367
Total Average		64	1158	305

Table B.12: Final JUSTICE Archive Size

		Archive Size		
Cores	**Seed**	**ϵ-NSGA-II**	**Borg**	**Generational Borg**
16	12345	261	337	242
	23403	301	313	237
	39349	255	314	211
	60930	250	365	234
	93489	272	336	254
	Average	267	333	235
32	12345	261	289	242
	23403	301	319	237
	39349	255	288	211
	60930	250	319	234
	93489	272	352	254
	Average	267	313	235
48	12345	247	352	242
	23403	301	318	237
	39349	255	345	211
	60930	250	294	257
	93489	272	364	254
	Average	265	334	240
Total Average		266	326	236

B.5. Spacing

Table B.13: Final DTLZ2 Spacing

		Spacing		
Cores	**Seed**	**ϵ-NSGA-II**	**Borg**	**Generational Borg**
16	12345	0.047558	0.042931	0.051233
	23403	0.047555	0.043361	0.055553
	39349	0.046748	0.042336	0.048944
	60930	0.047371	0.043223	0.050424
	93489	0.047756	0.044253	0.047633
	Average	0.047398	0.043221	0.050758
32	12345	0.046961	0.043617	0.049520
	23403	0.045763	0.043827	0.051563
	39349	0.047907	0.042399	0.053083
	60930	0.047048	0.044679	0.049297
	93489	0.046122	0.043448	0.048753
	Average	0.046760	0.043594	0.050443
48	12345	0.044425	0.043137	0.050138
	23403	0.048464	0.043091	0.050867
	39349	0.048507	0.041591	0.050594
	60930	0.045402	0.041878	0.049968
	93489	0.047510	0.045521	0.051093
	Average	0.046861	0.043044	0.050532
Total Average		0.047006	0.043286	0.050578

Table B.14: Final DTLZ3 Spacing

		Spacing		
Cores	**Seed**	**ϵ-NSGA-II**	**Borg**	**Generational Borg**
16	12345	0.616998	0.086633	0.275961
	23403	0.366937	0.045666	0.063694
	39349	0.455438	0.050104	0.584307
	60930	0.838907	0.252887	0.274019
	93489	0.722037	0.049671	0.639499
	Average	0.600063	0.096992	0.367496
32	12345	0.844823	0.055256	0.068143
	23403	1.012591	0.052309	0.532754
	39349	0.872638	0.046632	0.062453
	60930	0.631056	0.045612	0.598353
	93489	0.718715	0.049173	0.872161
	Average	0.815964	0.049796	0.426773
48	12345	0.549154	0.049069	0.064281
	23403	0.612056	0.051878	0.099573
	39349	0.642924	0.050617	0.375672
	60930	0.921148	0.046654	0.745277
	93489	1.175641	0.053838	0.533455
	Average	0.780184	0.050411	0.363652
Total Average		0.732070	0.065733	0.385974

Table B.15: Final JUSTICE Spacing

		Spacing		
Cores	**Seed**	**ϵ-NSGA-II**	**Borg**	**Generational Borg**
16	12345	4.427365	3.955610	3.967384
	23403	4.282251	3.498025	5.100258
	39349	4.198882	4.159515	5.229442
	60930	4.492479	3.619538	4.756513
	93489	4.173691	4.035434	4.154749
	Average	4.314934	3.853624	4.641669
32	12345	4.427365	2.855222	3.967384
	23403	4.282251	3.313301	5.100258
	39349	4.198882	3.941814	5.229442
	60930	4.492479	3.563038	4.756513
	93489	4.173691	3.437189	4.154749
	Average	4.314934	3.422113	4.641669
48	12345	4.503557	3.410698	3.967384
	23403	4.282251	3.363375	5.100258
	39349	4.198882	3.135706	5.229442
	60930	4.492479	3.904751	4.964810
	93489	4.173691	3.458492	4.154749
	Average	4.330172	3.454604	4.683329
Total Average		4.321482	3.576780	4.655566

B.6. Epsilon Progress

Table B.16: Final DTLZ2 Epsilon Progress

		Epsilon Progress		
Cores	**Seed**	**ϵ-NSGA-II**	**Borg**	**Generational Borg**
16	12345	6359	9901	6681
	23403	6688	9877	7119
	39349	6439	10292	6471
	60930	6395	10844	6456
	93489	6306	10098	6489
	Average	6437	10202	6643
32	12345	6506	10553	6899
	23403	6061	10577	6709
	39349	6254	10135	6134
	60930	6022	10994	6432
	93489	6505	9934	6355
	Average	6269	10438	6505
48	12345	6524	10312	6614
	23403	6979	9728	6971
	39349	6558	10329	6207
	60930	6445	10702	6618
	93489	6567	9977	6187
	Average	6614	10209	6519
Total Average		6440	10283	6555

Table B.17: Final DTLZ3 Epsilon Progress

		Epsilon Progress		
Cores	**Seed**	**ϵ-NSGA-II**	**Borg**	**Generational Borg**
16	12345	3211	9518	3751
	23403	2829	9914	3703
	39349	2642	10558	2426
	60930	2288	9459	3987
	93489	2512	9484	2618
	Average	2696	9786	3297
32	12345	2704	10452	3659
	23403	2451	11643	2852
	39349	2481	10079	3591
	60930	2955	9961	2880
	93489	2690	10425	2566
	Average	2656	10512	3109
48	12345	2361	9441	4292
	23403	2261	11413	4063
	39349	2260	10168	3052
	60930	2458	9767	3059
	93489	2399	10869	2119
	Average	2347	10331	3317
Total Average		2566	10209	3241

Table B.18: Final JUSTICE Epsilon Progress

		Epsilon Progress		
Cores	**Seed**	**ϵ-NSGA-II**	**Borg**	**Generational Borg**
16	12345	1889	6984	1760
	23403	1963	7452	1626
	39349	1949	6814	1699
	60930	1938	7577	1577
	93489	1864	7489	1629
	Average	1920	7263	1658
32	12345	1889	7583	1760
	23403	1963	8045	1626
	39349	1949	7454	1699
	60930	1938	7150	1577
	93489	1864	7139	1629
	Average	1920	7474	1658
48	12345	1900	7022	1760
	23403	1963	7372	1626
	39349	1949	7517	1699
	60930	1938	7598	1629
	93489	1864	7569	1629
	Average	1922	7415	1668
Total Average		1920	7384	1661

C

DTLZ2 and DTLZ3 Plots

C.1. Additive Epsilon Indicator

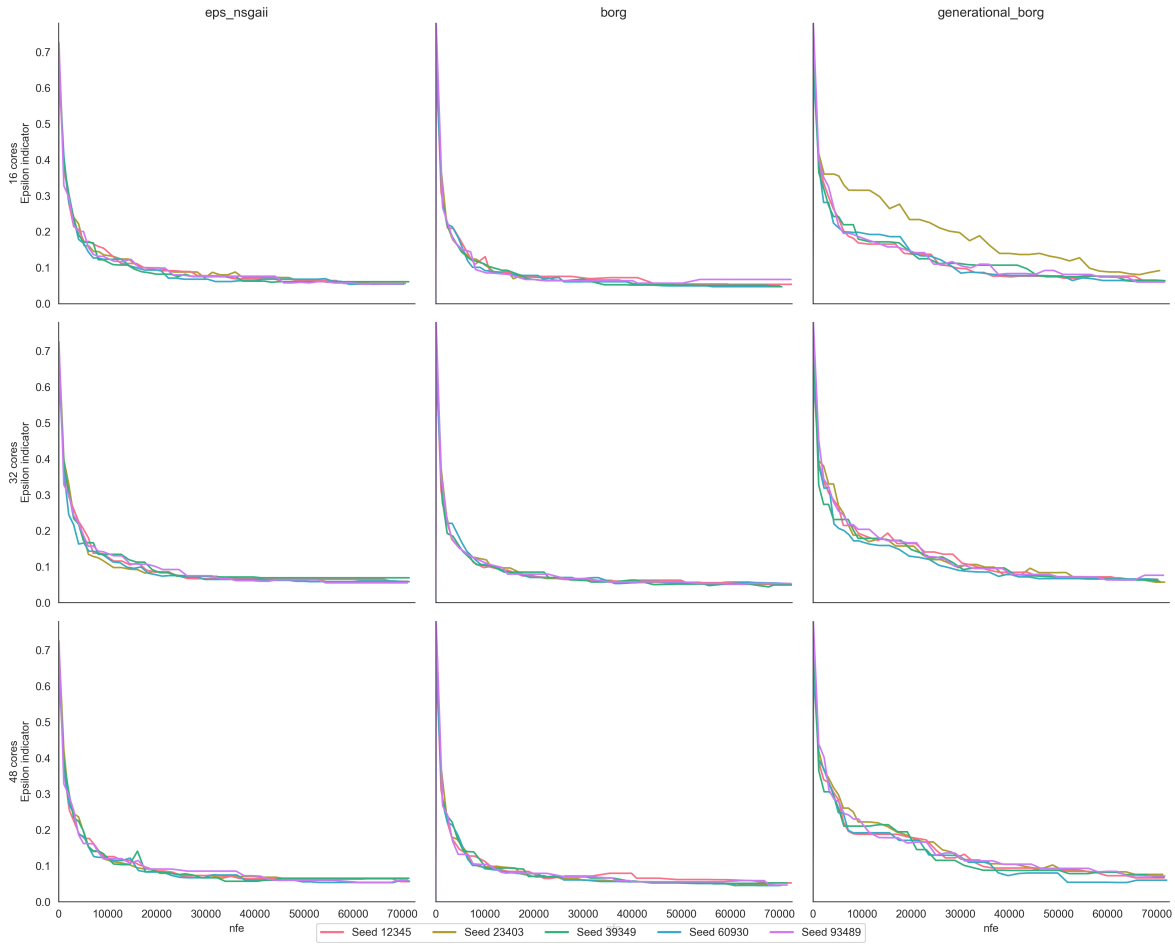

Figure C.1: DTLZ2 Epsilon Indicator

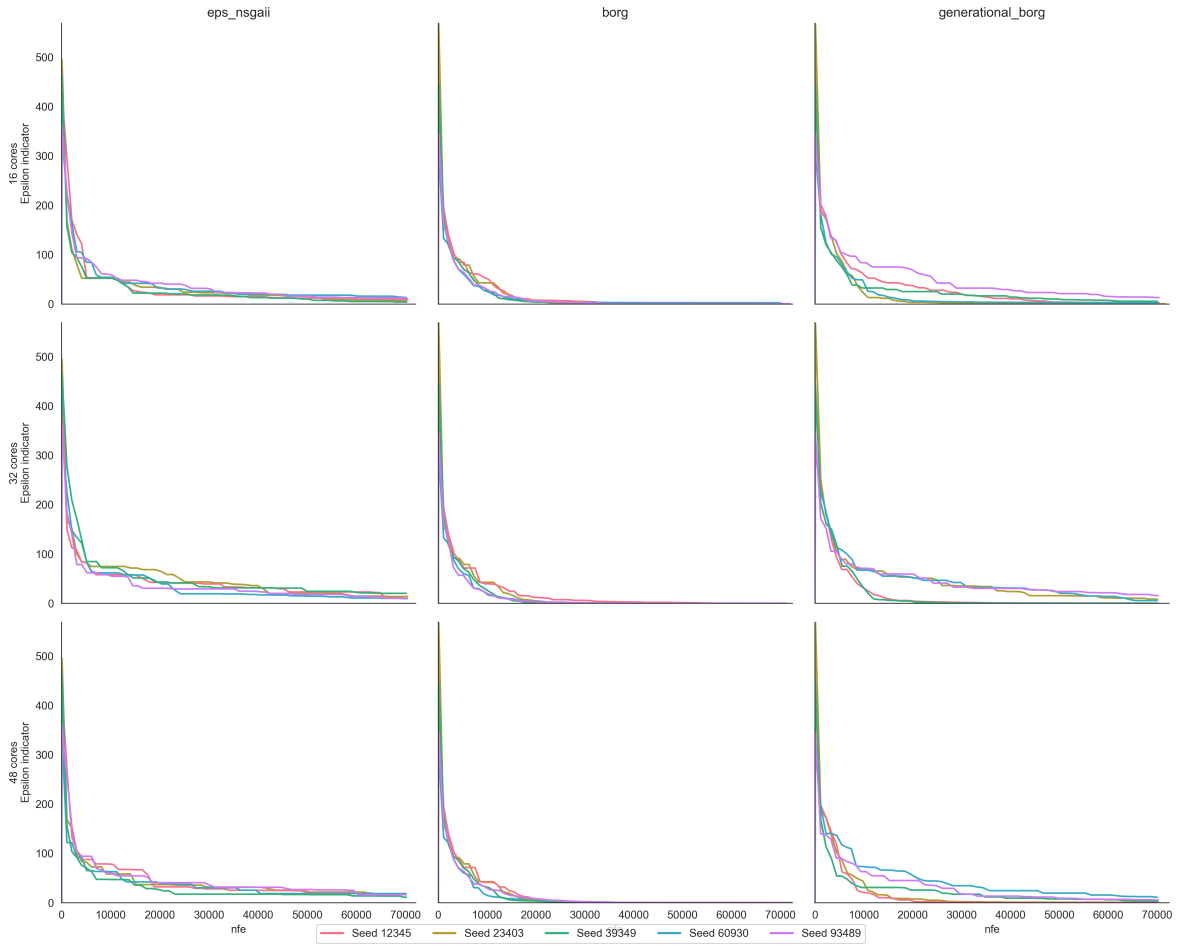

Figure C.2: DTLZ3 Epsilon Indicators

C.2. Archive Size

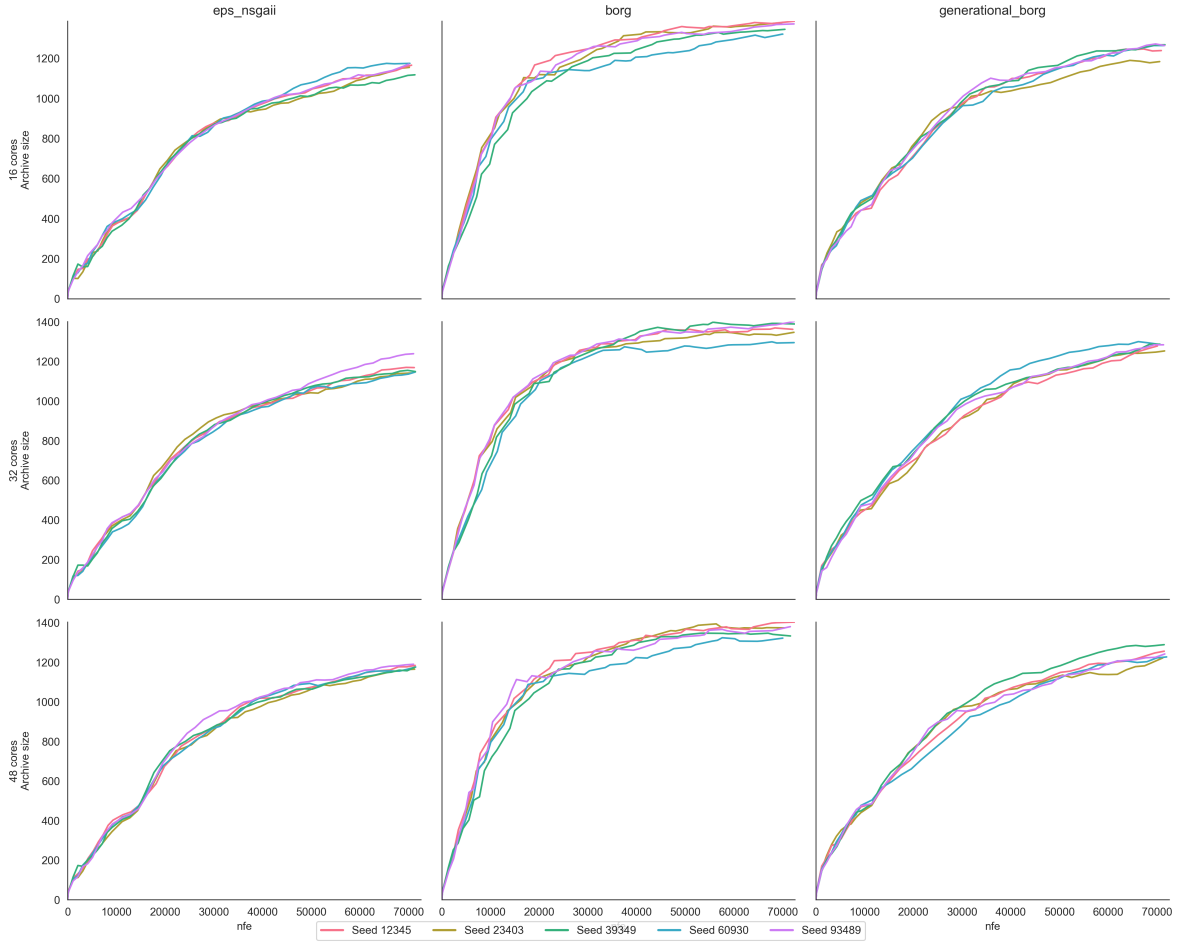

Figure C.3: DTLZ2 Archive Size

C.3. Spacing

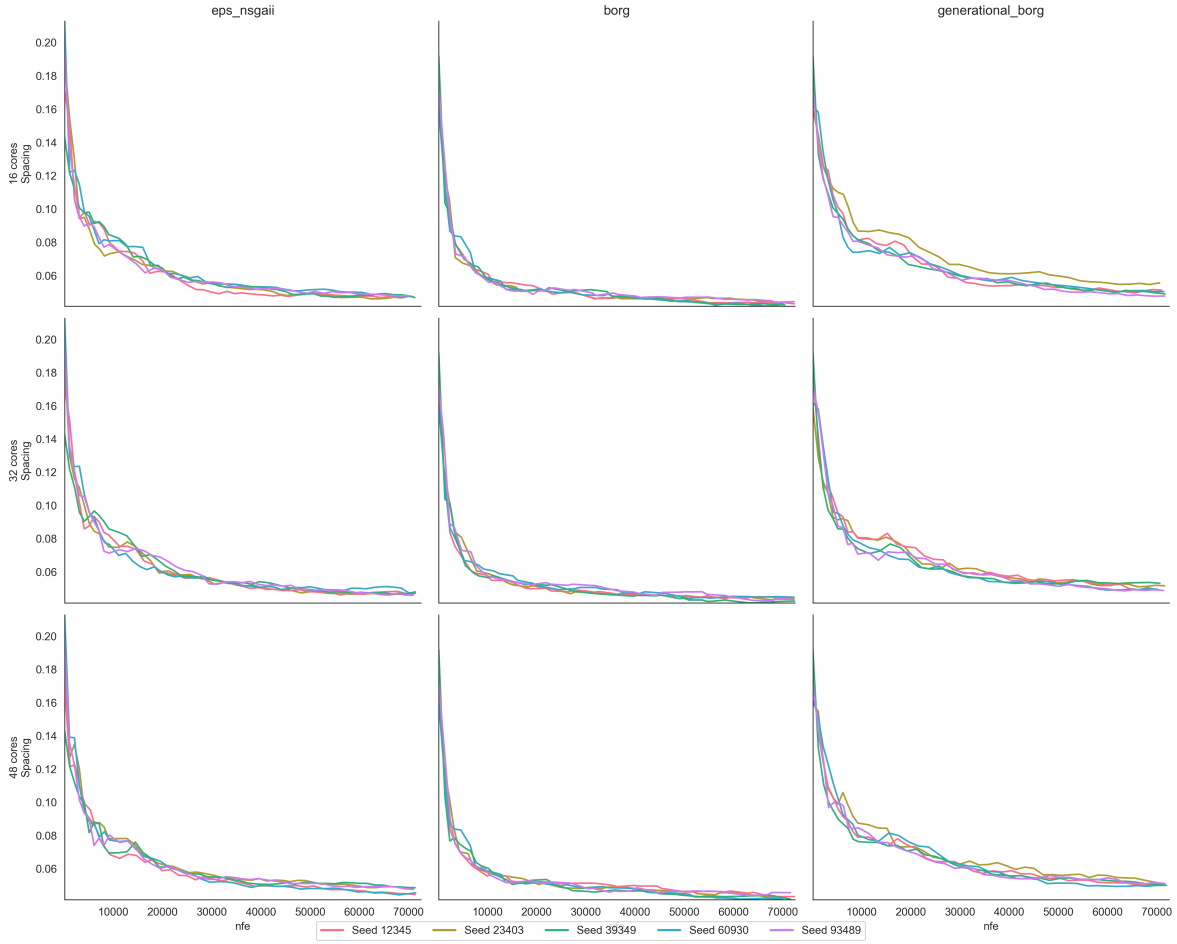

Figure C.4: DTLZ2 Spacing

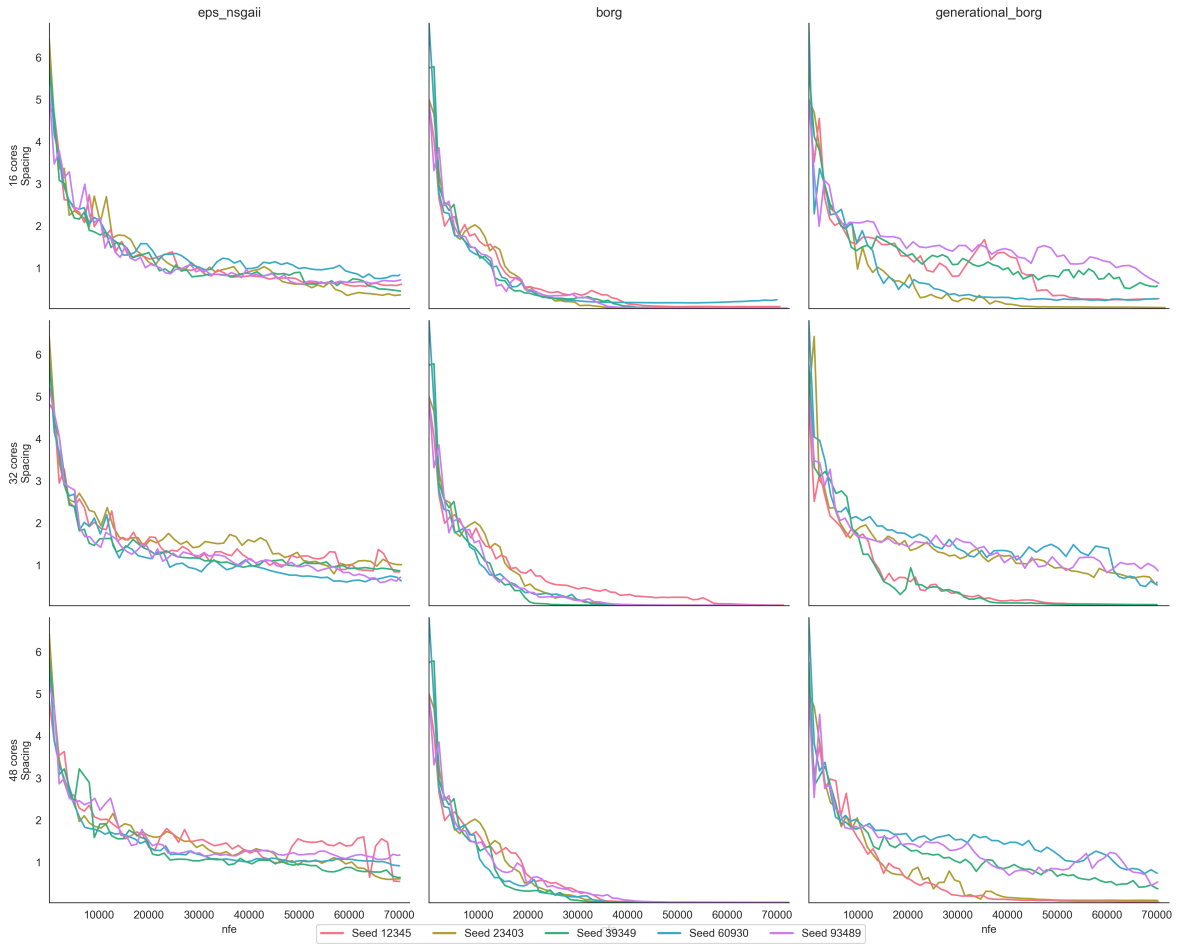

Figure C.5: DTLZ3 Spacing

C.4. Hypervolume Efficiency

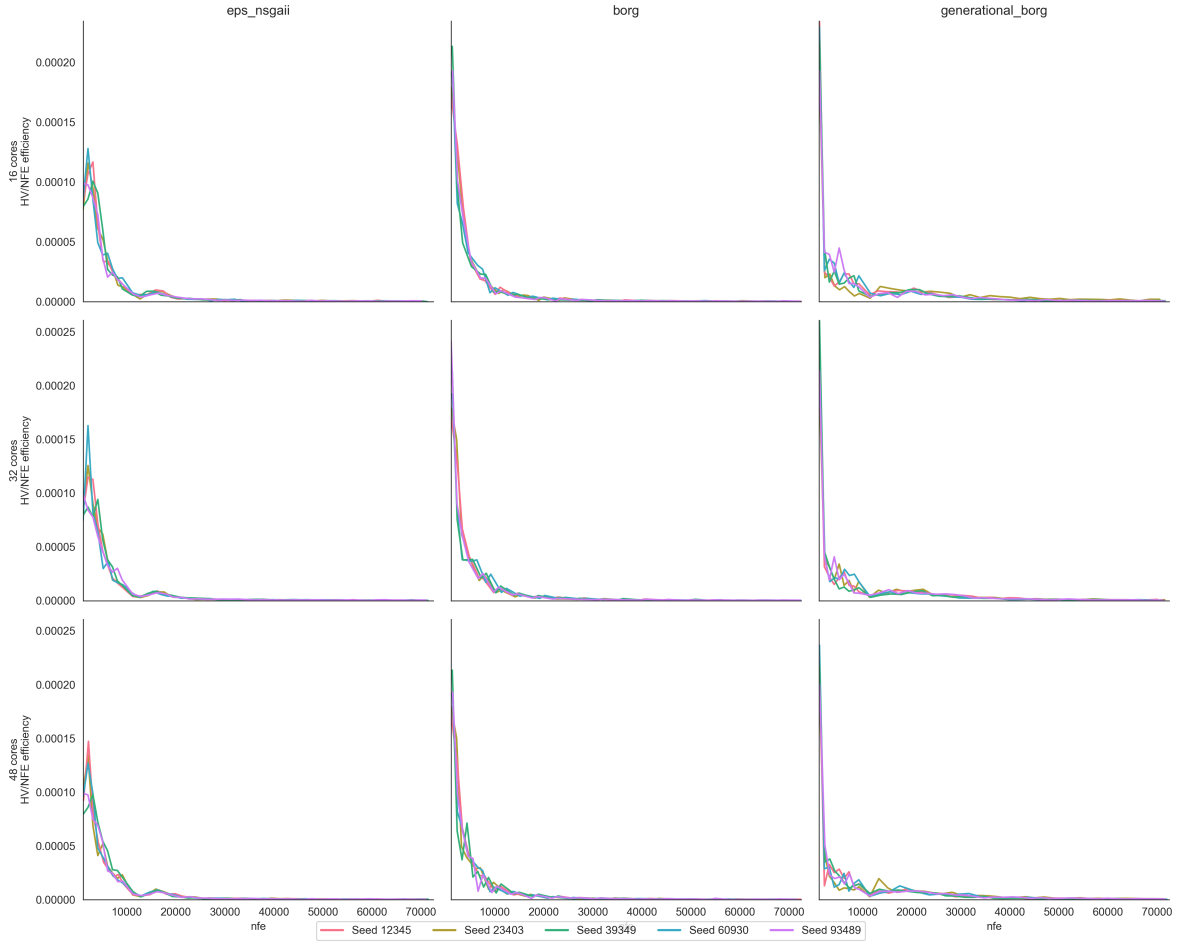

Figure C.6: DTLZ2 Hypervolume Efficiency

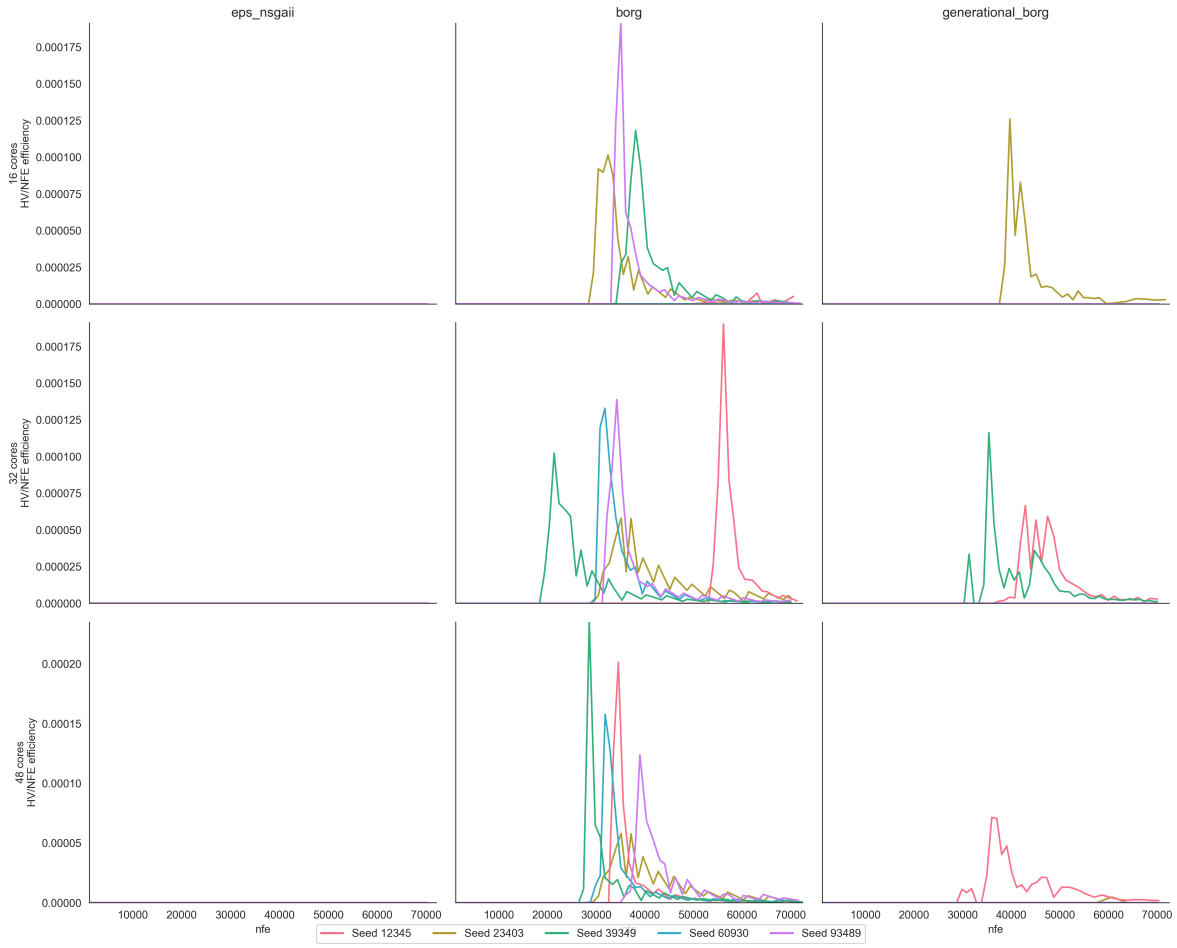

Figure C.7: DTLZ3 Hypervolume Efficiency

C.5. Epsilon Progress

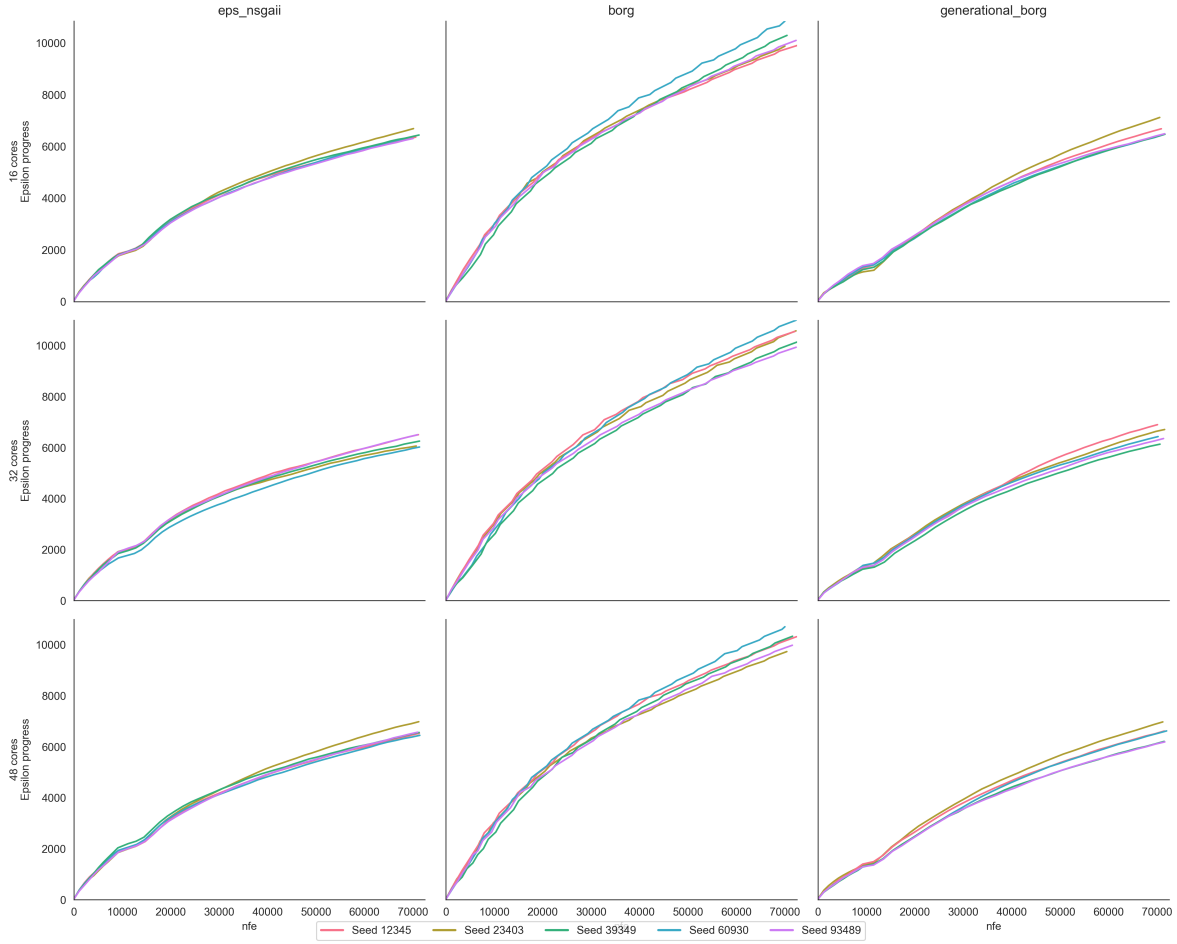

Figure C.8: DTLZ2 Epsilon Progress

C.6. Runtime Comparison

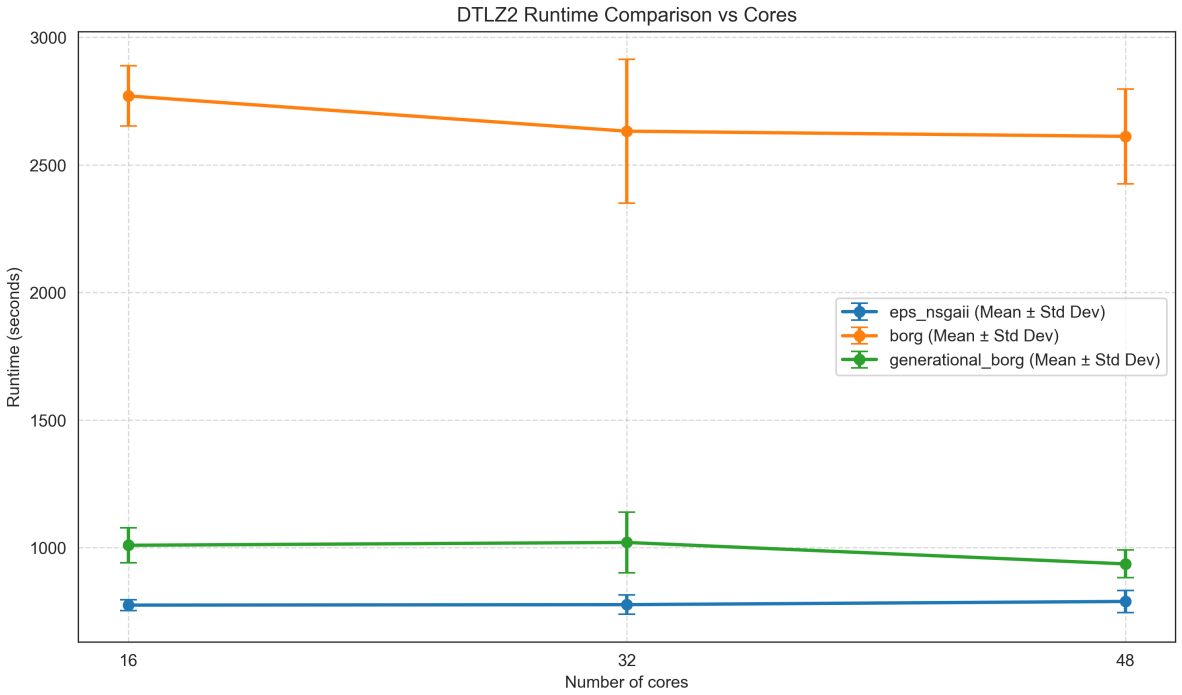

Figure C.9: DTLZ2 Runtime Comparison