Doctoral Dissertation (Shinshu University)

Dynamic compartmental and performance models for analysis and configuration of multi-objective evolutionary algorithms

March 2021

Monzón Maldonado Hugo Daniel

I dedicate this thesis to my parents.

Abstract

Evolutionary algorithms can solve complex optimization problems in science and engineering. However, increasing problem size and complexity demand constant improvement. On the other hand, their automatic configuration and selection given a problem instance is also a pending task. Both tasks require a deeper understanding and way to characterize their behavior and performance on a problem instance. This work proposes Dynamic Compartmental Models as a step forward in both tasks. The model allows analyzing, comparing, and configuring evolutionary algorithms. Inspired by epidemiological models, they track population changes modeling them as exchanges between compartments. Each compartment contains part of the population, and the rules to assign them are based on Pareto dominance, recentness, and membership to the Pareto Optimal Set or the Non-dominated solution set. Given the size for each compartment, the model can estimate their change over time. The behavior of an algorithm with its configuration on a problem instance is represented by one set of parameters. Small and large instances of the MNK-landscape problem solved by representative multi- and many-objective algorithms generate the data to train and test the models. In small instances, the trained models' estimations follow the trend of the data. Using the models' parameters and equations to explain how algorithms can keep discovering good solutions when the population is full, gave an example for algorithm analysis. Finding a correlation between one of the compartments' change and the hypervolume, a performance metric, created a way to use them for comparison between algorithms. For algorithm configuration, an instance was created and solved with one algorithm but several configurations. However, only some sample configurations were used to create models. The remaining configurations' models were obtained through interpolating the parameter of the sampled configurations' models. The results showed that both trained and interpolated models obtain estimations that can help decide a user which configuration to choose. In larger instances, a different compartment definition around only the non-dominated set was used. This also demanded the creation of an auxiliary model that links these compartments to the growth of the hypervolume to be able to compare and select between algorithms and configurations. The results indicate that these new features are also able to capture the changes in the population, even on unseen problem instances.

Table of contents

List of figures

List of tables

Chapter 1

Introduction

1.1 Background

In engineering, science, and industry there are times where finding a solution is not enough. For example, in product design, machine scheduling, or in determining the route to make deliveries, it is also desired that costs and time be minimized while quality, output, cargo is maximized. These types of situations fall under optimization problems, and when more than one objective needs to be fulfilled simultaneously then they become multi-objective optimization problems. While the consideration of multiple objectives seems trivial on paper, it has some important effects on the search space. The relationship between objectives can make the search harder, for example, a good solution for one objective can evaluate poorly in another. Several objectives naturally demand the existence of more than one good solution, and they will variate in the degree they fulfill each objective.

To cope with these and other difficulties, multi-objective evolutionary algorithms (MOEAs) have been created and successfully applied to a variety of domains [1, 2], with most of them focusing on 2 and 3 objectives ones [3]. However, industry needs and the interest of the research community to push boundaries have motivated the exploration of problems with a larger number of objectives [4–8]. Although it may seem that any multi-objective algorithm will keep is efficiency and effectiveness regardless of the number of objectives involved, dimensionality of the objective space has shown important effects on both qualities. It is well known in the literature that the performance of several classes of MOEAs reduces in problems with 4 or more objectives [9–13], which made the community make a clear distinction between these two cases, *Multi-Objecive* up to 3 objectives and *Many-Objective* optimization [14] for 4 or more.

Recent years had brought new algorithms, design components and tuning procedures with the aim to improve performance on many-objective problems [15, 16], increasing the

options available for users. To decide which algorithm to apply for a particular problem, quality indicators [17] are used to measure their success on a problem as their ability to reach a good approximation set on different criteria.

While this may be enough for some users, these indicators alone do not provide much insight into how and why a particular algorithm is working or failing in a given problem, which is valuable information to researchers making these algorithms, and users that wish to fine-tune them for their application. Getting the required insight demands looking beyond final results and shift the focus to the dynamics of the search process. An analysis of how different classes of algorithms behave on a problem or even a class of problems could lead to a better understanding of their inner workings, the discovery of strengths and weaknesses, determining the impact of design choices, and improve current algorithm or learn what needs to be done to create better ones. All of these are subjects of high relevance as the number of algorithms keeps increasing and real world problems scale in the number of variables and objectives.

Moreover, more knowledge on the algorithm themselves can be consolidated into frameworks that could, in an automated fashion, properly select and configure them for a particular task; that is, finding the algorithm that best solves a given problem instance. The road to this particular application can be broken into two complementary tasks. The first one is to characterize problems and classify them into different categories using features. This may help to extend matches of a given algorithm and a problem to the sub-class it belongs to or determine the specific features that make the algorithm perform well on it. The second one is to characterize the performance of algorithms in classes of problems, which in recent years has gained traction [18–21].

1.2 Proposal

This work focuses on the task of characterizing the performance of algorithms in instances and classes of problems. It does so by first considering the algorithm's behavior and trying to capture it into models, to later use this information to infer or directly estimate performance. Models are key since they allow to do analysis and can provide estimations to guide selection or configuration of algorithms, in some cases before having to run the algorithm itself.

Dynamic Compartmental Models (DCMs) track population dynamics in Multi- and Many-Objective Evolutionary Algorithm (MOEAs). DCMs are based on the epidemiology SIR model [22, 23]. The SIR model computes an estimate of the spread of an infectious disease in a closed population, by dividing the population in compartments depending on the health status of the individuals, i.e. Susceptible (can contract the disease), Infected (has

the decease) and Recovered (has been cured and cannot be infected again), which gives the model its name. The changes in the sizes of the compartments as time progresses is captured by the model equations and parameters. Similarly, to analyze and understand the dynamics of multi- and many-objective algorithms during the evolutionary process, a more general compartmental model is used, combined with features that divide the population into compartments. Multi-objective algorithms, independently of their inner workings, aim to find a good approximation of the Pareto Optimal Set (POS), which is a set of non dominated solutions, by operating on their population. Thus, measuring these changes through features and using a model to estimate them is a valid approach to this task. Features, at their most basic form, are defined around the dominance or non dominance status of solutions in the current population only. Adding granularity to them, by comparing the non dominated set in the current population with the ones in previous generations, or keep track of when solutions appear during the process, allows exploring the same algorithm with its configuration on a problem from other perspectives. Some particular sets of features can be correlated and used as performance estimators, while others in combination with an auxiliary model provide a direct estimation of a more common performance indicator value such as the hypervolume.

The models and their parameters form a compact representation of the dynamics of an algorithm with its configuration in a problem instance. This can be used to analyze them and explain their behavior in a quantitative fashion, as well as to explore variations in their configuration and the impact on their performance. Estimations on how the composition of the population will change are also possible without having to run the algorithm when the models are available for a given problem instance. In some cases, depending on the features used, ranking and selection between algorithms and their configurations can be made. In this work, it will be explored how these models can be used to characterize the algorithms, perform analysis and configuration in small and large problem instances and motivate their use as part of tools and frameworks that can deliver automatic algorithm configuration and selection and knowledge to improve evolutionary algorithms.

1.3 Contribution

The contributions of this work are:

- Use generational search assessment indices as features capable of capturing algorithms' population dynamics.
- Introduce compartmental models with two and three compartments to track MOEAs population changes.

- Show how to use features and model's parameters for analysis on small instances.
- Relate features indirectly to performance using an example with the accumulated number of Pareto optimal solutions and the Hypervolume.
- Compare representative MOEAs through features and models of population dynamics.
- Introduce interpolation as methodology to extract new models from existing data.
- Use models as a tool to select between configurations for an algorithm.
- Propose a feature set to be used on large instances where the Pareto Optimal Set is not known.
- Create a model to directly estimate performance based on the features changes.

1.4 Related Works

Complementing this introduction, a brief summary on algorithm configuration and selection methods applied in evolutionary computation is presented, followed by some remarks on where the proposal would fall with respect to previous works, as well as some distinctions that can be made with respect to them.

Algorithm Configuration

Evolutionary algorithms require a set of parameters to be tuned by the user depending on the problem to be solved, such as the population size, operator type (selection, mutation, crossover), and the associated probabilities. Although success in a problem instance or class depends on finding a good setting [24], the default settings of some algorithms may provide sufficiently good performance on certain classes of problems [25]. When this is not the case, or resources and time allow for a more precise tuning, the community has developed methods that fall try to solve the per-set algorithm configuration problem which can be stated formally as, given an algorithm A with parameters $\mathbf{p} = (p_1, \ldots, p_k)$, a set C of possible values or configurations for $\mathbf{p}$, a set of problem instances I and a performance metric m, find a configuration $c^* \in C$ of A that achieves optimal performance on I according to m.

This problem is also referred in literature as parameter control. Depending on how the parameters are determined [26] the method can be classified as *deterministic* or *uninformed* if the parameters are set without information from the algorithm, based solely on time, similar to the idea behind simulated annealing [27]. Some examples are the work of Eiben et.al [28] where the population size changes during execution in response to the overall improvement in the individuals. Bäck et.al [29] use a time-dependent mutation rate schedule

on genetic algorithms with better results than setting a fixed value. Mezura-Montes et.al [30] changed differential evolution [31] and constraint handling parameters to solve constrained optimization problems using a mix of scheduling and self-adaptation.

If during the search for the optimal solutions, the algorithm also tries to find its optimal parameter values, the scheme is called *self-adaptive*. Some algorithms include the parameters as part of the solution, taking advantage of the evolutionary operators, good solutions may be the product of a good parameter setting. Evolutionary strategies are good examples of parameter self-adaptation [32, 33]. Bäck et.al [34] built a parameterless genetic algorithm where crossover and mutation rates are self-adapted during the search. Farmani et.al [35] use a self-adaptive fitness formulation to handle constrained problems. Hansen et.al [36] proposed the CMA-ES that uses as the name indicates adapts the covariance matrix of the distribution used to sample new candidate solutions.

Finally, if the method to find the optimal parameters is decoupled from the search, the scheme is called *adaptive* or *dynamic*. Here the method looks at some property of each algorithm run and decides how the parameters should change to obtain improvements. Thierens [37] proposed an adaptive pursuit strategy to change the probability values of a set of evolutionary operators. The algorithm tries to distribute the probabilities of the operators to match the reward from the environment, the quality of generated solutions. Schlierkamp-Voosen et.al [38] modifies the Breeder Genetic Algorithm to change its parameters using competing sub-populations. Each population implements a different strategy (parameters) and the best performing ones are rewarded (can increase their size). The best individuals can migrate from time to time from bad sub-populations to better ones. Igel et.al [39] used online adaptation to change the evolutionary strategy (choice of variation operators, its parameters, population size) to search for neural networks topologies.

Algorithm Selection

While algorithm configuration deals with finding the parameters for a particular algorithm and searches its configuration space, algorithm selection on the other hand chooses only in algorithm space without worrying about their configurations. In fact, algorithm configuration can be seen as a generalization if each possible configuration is taken as a different algorithm.

More formally, given a set I of instances from a problem P, a set $\mathbf{A} = \{A_1, \ldots, A_n\}$ of algorithms that can solve P and a metric $m : \mathbf{A} \times I \to \mathbb{R}$ that gives the performance of $A_i \in \mathbf{A}$ on the set I, build a selector S that for any problem instance $i \in I$ selects an algorithm $S(i) \in \mathbf{A}$ so the overall performance of S on I according to the metric m is optimal [40].

One key aspect of algorithm selection is obtaining information on the problem computing features to determine its characteristics, usually termed as exploratory landscape analysis [41] to feed the selection framework.

Some successful approaches for combinatorial problems are SATzilla by Xu et.al [42] that is used to solve portfolios of SAT (satisfiability) problem using empirical hardness models to decide which solver assign to each instance. AutoFolio by Lindauer et.al [43] where an algorithm configuration scheme SMAC [44] is applied to decide the parameters of the algorithm selection framework CLASPFOLIO2 [45] to solve ASP (answer set programming) problems. On continuous problems, Bischl et.al [46] extracts low-level features by sampling the problem and treats the selection of an algorithm from the portfolio as cost-sensitive learning, using to solve this one-sided support vector regression [47]. Kerschke et.al [20] use exploratory landscape analysis and combine it with machine learning approaches (classification, regression, and pair-wise regression) to select the best solver for instances in the Black-Box Optimization Benchmark(BBOB) [48].

Proposal in context of Algorithm Configuration and Selection

Using the algorithm configuration classification the proposed models in this work can be put under adaptive methods, since a property on the algorithm run, the composition of the population, is used to feed the model and create predictions that can lead to choosing one parameter set over another.

However, the proposed models here differ from previous methods and schemes in the sense that the dynamics of the algorithms is introduced as part of the process. Instead of predicting the performance of the algorithm or a configuration directly, the model predicts the changes in the composition of the population, which is later connected to performance. In terms of algorithm configuration, this opens the proposal of this work to not only focus on the performance but analyze the behavior of the algorithm in the problems (Section 5.3.2). It also allows it to include information on the initial population to make decisions (Section 6.5). The models can also be extended for new configurations (Section 5.4.3) or predict the behavior and performance on larger budgets (Section 5.4.4). In terms of algorithm selection, similar to previous approaches, information about the problem instance needs to be collected first to choose which trained models to use, the ones for the problem sub-class (Section 6.4) or a close enough (Section 6.5) one, to select which algorithm is the best choice.

1.5 Outline

- **Chapter 2** introduces the necessary concepts on multi-objective evolutionary optimization as well as the present challenges in the area.

- **Chapter 3** describes the proposed features and models in this work to capture and analyze MOEA behavior and performance.

- **Chapter 4** covers the algorithms, their experimental settings, as well as the test problem generator and the datasets created to show how the features and models work.
- **Chapter 5** concentrates on the use of the model with features based on the Pareto Optimal Set, exploring how analysis, comparison and configuration algorithms is done.
- **Chapter 6** showcases the model with features based on the Non Dominated Set and its ability to be applied in small and large problems. It also presents the use of the performance estimation model that gives results in a commonly used metric, the hypervolume. A new fitting process is introduced, as well as testing the quality of the estimations on unseen data. How configuration and comparison tasks can be performed is also discussed.
- **Chapter 7** summarizes the obtained results with the models, the different types of features sets and proposes directions on how to expand them.

Chapter 2

Multi- and Many-Objective Evolutionary Optimization

This chapter introduces the basic concepts in many- and multi-objective optimization and evolutionary computation.

2.1 Multi-Objective Optimization

Let x be a decision variables vector and f a collection of functions, where each element represents an objective function f_i that takes x and gives an evaluation in terms of performance criteria described by the function. A Multi-Objective Optimization Problem (MOP) can be defined as finding an x that fulfills any problem restriction while optimizing (maximizes or minimizes) f. In most cases some of f_i in f are in conflict, i.e small improvements in one f_i represents a degradation for the other objectives in f [49, 50]. More formally it can be defined as follows.

Definition 2.1 (Multi-Objective Optimization Problem)**.** Given a vector $x = [x_1, \ldots, x_n]^T$, of n decision variables x_i, that fulfills the following $r = a + b$ restrictions:

$$g(x) = [g_1(x), ..., g_a] \geq 0 \tag{2.1}$$

$$h(x) = [h_1(x), ..., h_b] = 0 \tag{2.2}$$

and optimizes the vector of functions:

$$f(x) = [f_1(x), ..., f_m]^T \tag{2.3}$$

The word *optimize* could indicate minimizing or maximizing the value of each objective function, which will depend on the problem at hand.

A Multi-Objective Optimization Problem becomes *Many-Objective* when, using the above definition, $m \geq 4$. Due to the challenges these problems present to multi-objective solvers, they have been assigned this separate class [14, 51].

Definition 2.2 (Feasible decision space)**.** Is composed by all the vectors x that fulfill restrictions $g(x)$ and $h(x)$. This means that they are contained in the decision variables space.

$$\Omega = \{x \in \mathbb{R}^n \mid g(x) \geq 0 \wedge h(x) = 0\}$$

Definition 2.3 (Feasible solution space)**.** Is composed by the image of the Ω set and is contained inside the objective function space:

$$\Omega_0 = \{f(x) \in \mathbb{R}^m \mid x \in \Omega\}$$

2.2 Pareto Optimality

The origin of Multi-Objective Optimization research begins with the concepts around the Pareto optimum, formulated by Vilfrido Pareto in the XIX century [52, 53]. This section will give the concepts of Pareto dominance, Pareto optimality, Pareto set, and Pareto front as defined in [54, 55].

Definition 2.4 (Pareto dominance)**.** Given two decision vectors $u = (u_1, \ldots, u_n)$, $v = (v_1, \ldots, v_n)$:

$u \preceq v$	if and only if	$f_i(u) \leq f_i(v) \quad \forall i \in (1, \ldots, m)$
u dominates v	and	$f_j(u) < f_j(v) \quad \exists j \in (1, \ldots, m)$
$u \sim v$	if and only if	$f_i(u) \not\leq f_i(v) \wedge f_i(u) \not\leq f_i(v)$
u and v are non dominated		$\forall i \in (1, \ldots, m)$

Although these definitions are expressed in terms of minimization for all objectives, is easy to redefine the operators for maximization problems ($\succeq, \sim$).

Definition 2.5 (Pareto optimum)**.** Given a decision vector u, it can be said that it is Pareto optimal if and only if: $u \in \Omega \mid \neg\exists\, v \in \Omega \mid f(v) \preceq f(u)$

A decision vector that is Pareto optimal cannot be improved in any objective without inflicting a degradation in at least one other objective.

Definition 2.6 (Pareto optimal set)**.** Is defined as $P^* = \{u \in \Omega \mid \neg\exists\, v \in \Omega \mid f(v) \preceq f(u)\}$.

The Pareto optimal set is part of the decision variable space.

Definition 2.7 (Pareto front)**.** Given a Multi-Objective Optimization problem $y = f(x)$ and a Pareto optimal set P^*, the Pareto front (PF) is defined as:

$PF = \{y = f = (f_1(x), ..., f_m(x)) \mid x \in P^*\}$

The Pareto front is part of the solution space.

2.3 Multi-Objective Evolutionary Algorithms (MOEAs)

Simultaneously optimizing for several objectives is not an easy task, and while some traditional techniques can still be applied by transforming the multi-objective problem into a mono-objective version, this not always will yield good results. On the other hand, Evolutionary Algorithms (EAs) had continuously shown their capacity to handle them by solving this type of problems in different areas [1, 2].

Evolutionary algorithms are based on the concepts of Neo-Darwinism, which explains life on the planet by a combination of the following mechanisms [55].

Reproduction The mechanism that allows genetic material to be passed down from one generation to the next one.

Mutation An arbitrary error that can happen when genetic material is copied to the next generation during reproduction, which could be beneficial by introducing a change that improves adaptability.

Selection The survival of the fittest. Individuals that are able to adapt to their environment survive and have a chance to reproduce.

They take these mechanisms and pose the problem of finding an optimal solution or a set of them as, the evolution of a population of solutions over the course of several iterations, referred to as generations. A solution is an individual that with the help of the aforementioned mechanisms evolves toward becoming an optimal solution to the problem. The algorithm goal is to bring the population closer to the Pareto front or a good approximation of it.

Adaptation to the environment or fitness is how the algorithm measures the performance of a solution. Note that in some cases this function and the objective functions are one and the same. More specifically, an objective function defines the optimality of a solution according to the problem, so it belongs to the problem domain, while the fitness function measures how

```
1  t ← 0
2  Initialize population P_0 of size n randomly
3  Evaluate(P_0)
4  while ¬ StoppingCondition() do
5  |  Q_t ← Crossover(P_t)
6  |  Mutation(Q_t)
7  |  Evaluate(Q_t)
8  |  P_{t+1} ← Selection(P_t ∪ Q_t)
9  |  t ← t + 1
10 end
11 return Best individual in P_t
```

Figure 2.1: Pseudo code of a generic evolutionary algorithm.

well that solution satisfies that and belongs to the algorithm domain. Their basic structure can be seen in Figure 2.1 as pseudo-code.

Compared to traditional methods, there are some significant advantages to evolutionary algorithms [55, 54].

- Search of compromise solutions in partially sorted spaces. Solutions that may neglect one objective to be better at another one.
- Produce a set of optimized solutions (population) simultaneously, versus one optimal solution per run in traditional methods.
- Specific knowledge about the problem is not necessary, just a way to validate solutions and evaluated them according to the problem.
- Simple in concept and offer large applicability.
- Due to their nature they are ideal to exploit parallel architectures.
- By searching using a set of solutions instead of a single one they are less susceptible to be trapped in local optima.
- They offer more robustness to dynamic changes, and in some cases the adaptation of their own parameters its part of the evolutionary process.

2.4 Many-Objective Evolutionary Optimization and its challenges

While some algorithms will present no issues on multi-objective problems with up to three objectives, studies have shown that for many-objective problems, i.e. four or more objectives, commonly used EAs that rely on Pareto dominance will present the following issues [51, 56, 57]:

Search ability deterioration: In standard Pareto dominance algorithms such as SPEA [58, 59] or NSGAII [60], when the number of objectives rises, almost all solutions inside the population become non dominated. This reduces the selection pressure towards the Pareto Front, which in turn deteriorates the algorithm's convergence.

Exponential increment of non dominated solutions: Since EA's objective is to find an approximation of the Pareto front, if the hypersurface in the objective space grows in dimensionality, the number of solutions required to approximate it also will. This could mean that thousands of solutions, maybe more than the size of commonly set population sizes, may be required.

Difficulty in the visualization of the solution set: Since usually the final selection of solution relies on the decision maker[1] preference, the increment in number of objectives obfuscates the visualization of the non dominated set of solutions found, which in turn could difficult the selection of a solution.

The previously described issues have been termed as the curse of dimensionality [56, 61], in the context of many-objective problems. Nevertheless, new and improved strategies have been developed to try to avoid or counter these difficulties. Here we list some of the most relevant of them:

Pareto Dominance Relaxation: The goal here is to modify the dominance relation to be able to make decisions around solutions previously seen as incomparable. Some algorithms rely on a mapping function to expand the dominance area of some non dominated solutions. These kinds of methods also induce a spacing between solutions that benefits diversity [62–65].

Decomposition: These algorithms decompose the problem in various single-objective ones using scalarization functions, assuming a priori the distribution of the Pareto front, which are solved simultaneously [66].

[1] The one that takes the final decision, in other words, which solution will be implemented.

Performance Indicators: Relying on the properties of performance indicators and their capacity to include preference information, these algorithms transform the many-objective problem into a single-objective one where the goal is to optimize the indicator [67].

Chapter 3

Models and Features for studying MOEA Behavior

This chapter introduces the concept behind the dynamic compartmental models used in this work to study MOEA's behavior and estimate performance. Here are also presented the features that allow the models to work, and how their combination defines what it is capable of capturing about the algorithms.

3.1 Dynamic Compartmental Models (DCMs)

Dynamic Comparmental Models (DCM) track the changes in the composition of the population of a multi- or many-objective evolutionary algorithm throughout the generations, aim to characterize its behavior, and have a better understanding of their working principles. They achieve this by breaking down the population in two or more types of individuals that belong each to a different compartment, and through simulation using mathematical equations, the change in composition of the population is described as the interaction between the compartments.

DCMs where inspired by and based on epidemiological compartmental models [22, 23], in particular the SIR model. Their goal is to track an infectious decease spread in a population through time. As per the model name, each compartment is defined using the health status of the individuals (Susceptible, Infectious, Recovered) to stratify the population without overlapping. Learning through data the parameters of these models, allows to understand how the decease behaves and make estimations to help in its control.

In order to obtain a similar tool for MOEAs, here compartments are defined in terms of the Pareto dominance status of individuals and its changes throughout several generations,

capturing the evolution of the population and therefore the dynamics of the algorithm. The model is based on the following assumptions:

(i) The population can be split into two or more groups that do not share elements using an appropriate set of rules.

(ii) The size of groups is considered to vary linearly while the total sum (population size) always remains constant.

(iii) The model gives the proportion of individuals from the population that belongs to each group at a given time t.

(iv) The model tracks how many individuals are part of each group, it does not track individuals by themselves.

With them, the parameters of the model offer a compact representation of the interactions between compartments, which can serve as means for algorithm comparison or employed to complement explanations about their behavior. Another useful result of having the parameters is being able to estimate how the compartments sizes and the composition of the population will look in future generations from a given initial state, which could aid in making decisions about the algorithm configuration or its use in a particular problem.

These parameters must be learned by fitting them to the changes observed in the population during the run of a MOEA, with a particular configuration, in a given problem. They are linked to algorithm, configuration and problem instance or if trained on several ones, the problem subclass. Thus, at what point in the algorithm process the data is taken plays an important role of what can be captured.

In this work, were elitist MOEA are studied, the composition of the population is measured on the truncated population obtained after joining the current population and the offspring population. This should make the DCMs capture the collective effects of evolutionary operators, including variation, parent selection and truncation selection.

While Pareto dominance is mentioned as the main rule to create the compartments, other properties of the individuals can be added to study the population from different aspects. However, it must be noted that using dominance as the base does not exclude any of the different classes of MOEA. When the composition of the population is measured for the models, this does not affect the contents, can be performed either online or offline, provided that it has been stored at each generation, and is completely independent of the selection and variation operators used to evolve it. It can be said that DCMs are similar to performance metrics commonly used in the field.

Depending on the number of compartments the chosen set of rules has, several models are possible. The ones considered in this work are as follows:

3.1.1 Two Compartment Model

If the population is split into two different groups or compartments, the obtained simple model can be seen in Figure 3.1.

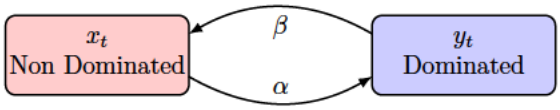


Figure 3.1: A two compartment model.

At each generation t, x_t and y_t represent the proportion of the population that belongs to each compartment. Thus, $x_t + y_t = 1$. The differential equations that describe the two compartment model in a time-discrete form are:

$$\begin{cases} x_{t+1} = (1-\alpha)x_t + \beta y_t \\ y_{t+1} = \alpha x_t + (1-\beta)y_t \\ x_t + y_t = 1, \end{cases} \tag{3.1}$$

where α and β are coefficients that describe the loss in x_t and y_t, respectively. Since the model is closed, a loss in x_t becomes a gain in y_t, and vice versa.

3.1.2 Three Compartment Model

Adding one more compartment as shown in Figure 3.2, gives a more interesting model that can be used to analyze the dynamics in a more detailed manner.

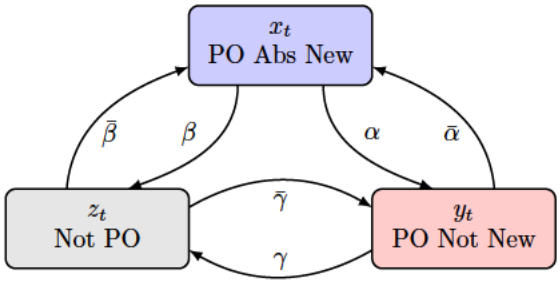


Figure 3.2: A three compartment model.

Similar to the two compartment model, x_t, y_t and z_t represent the proportion of the population that belongs to each state at time t. Thus, $x_t + y_t + z_t = 1$. The differential

equations that describe the three compartment model in a time-discrete form are as follows:

$$\begin{cases} x_{t+1} = (1-(\alpha+\beta))x_t + \bar{\alpha}y_t + \bar{\beta}z_t \\ y_{t+1} = \alpha x_t + (1-(\bar{\alpha}+\gamma))y_t + \bar{\gamma}z_t \\ z_{t+1} = \beta x_t + \gamma y_t + (1-(\bar{\beta}+\bar{\gamma}))z_t \\ x_t + y_t + z_t, = 1 \end{cases} \tag{3.2}$$

where α and β are coefficients that describe the loss in x_t that becomes a gain for y_t and z_t, respectively. Likewise, $\bar{\alpha}$ and γ are coefficients that describe the loss in y_t that becomes a gain for x_t and z_t, respectively. Similarly, $\bar{\beta}$ and $\bar{\gamma}$ are coefficients that describe the loss in z_t that becomes a gain for x_t and y_t, respectively.

3.2 Population Features

Defined as generational search-assessment indices in Aguirre et al. [68], they combine Pareto dominance status and recentness of a solution in the population to create features that can be used to track the search progress in an MOEA. Table 3.1 contains a subset of them that can be paired with DCMs. All features or indices range in the interval $[0,|P|]$, where $|P|$ denotes the population size.

Pareto dominance was introduced in a previous chapter, while the concept of recentness used in this work can be defined as follows:

$$\begin{cases} \{x : x \in \mathscr{R}(t) \wedge x \notin \cup_{i=s}^{t-1}\mathscr{R}(i)\} & \text{is new to the population} \\ \{x : x \in \mathscr{R}(t) \wedge x \in \cup_{i=s}^{t-1}\mathscr{R}(i)\} & \text{has been present before in the population} \end{cases} \tag{3.3}$$

where $\cup_{k=s}^{t-1}\mathscr{R}(k)\}$ is the union of all previous sets of $\mathscr{R}$ found by the algorithm from a generation s to the previous one $t-1$, where the presence of a solution x is checked. Recentness is used to create features that can determine how much of the population is composed of newly discovered solutions, while also looking at the ability of the algorithm to retain previously found solutions or rediscover them during the search.

Before describing what DCMs can be obtained with the above features, for a better organization they are divided into POS or NDS based ones depending if the set of Pareto Optimal or the set of Non Dominated solutions is used to define the feature.

Table 3.1: Generational search-assessment indices. Measures are taken on non dominated population $\mathscr{F}_1(t)$ with respect to $\mathscr{F}_1(t-1)$ and/or the POS.

Name	Abbreviation	Formula
Non Dominated	ND	$\{x : x \in \mathscr{F}_1(t)\}$
Dominated	DOM	$\{x : x \in P \wedge x \notin \mathscr{F}_1(t)\}$
Pareto Optimal	PO	$\{x : x \in \mathscr{F}_1(t) \wedge x \in POS\}$
Pareto Optimal Old	POold	$\{x : x \in \mathscr{F}_1(t) \wedge x \in \mathscr{F}_1(t-1) \wedge x \in POS\}$
PO Possibly New	POpnew	$\{x : x \in \mathscr{F}_1(t) \wedge x \notin \mathscr{F}_1(t-1) \wedge x \in POS\}$
PO Absolutely New	POA	$\{x : x \in \mathscr{F}_1(t) \wedge x \notin \cup_{k=1}^{t-1} \mathscr{F}_1(k) \wedge x \in POS\}$
PO Not Absolutely New	PON	$\{x : x \in \mathscr{F}_1(t) \wedge x \in \cup_{k=1}^{t-1} \mathscr{F}_1(k) \wedge x \in POS\}$
Non Dominated Non PO	NDNP	$\{x : x \in \mathscr{F}_1(t) \wedge x \notin POS\}$
Non Pareto Optimal	NPO	$\{x : x \in P \wedge x \notin POS\}$
Non Dominated New	NDNew	$\{x : x \in \mathscr{F}_1(t) \wedge x \notin \cup_{k=1}^{t-1} \mathscr{F}_1(k)\}$
Non Dominated Old	NDOld	$\{x : x \in \mathscr{F}_1(t) \wedge x \in \cup_{k=1}^{t-1} \mathscr{F}_1(k)\}$

3.2.1 POS based features

As the name implies, if the Pareto Optimal Set for the problem to be solved is available or can be found by enumeration or other method, the features PO, POpnew, POold, POA, PON, NDNP and NPO can be computed.

Pareto Optimal (PO) solutions, count all the solutions in the current generation that are part of the POS. *Pareto Optimal Probably New* (POpnew), checks for recentness and considers solutions that while part of the POS, also appeared in generation t but not in $t-1$. On the other hand, *Pareto Optimal Old* (POold) counts a solution if it appeared in generation t and also $t-1$. Similarly, *Pareto Optimal Absolutely New* (POA) considers solutions that are part of the POS and only appeared in the current generation t, while *Pareto Optimal Not Absolutely New* (PON) counts a solution if it appeared in t and also in any previous generations from 0 to $t-1$, being a more restrictive version of the previous feature.

Non Dominated Non Pareto Optimal (NDNP) counts the solutions that are non dominated but also not part of the POS in the current generation. Finally, *Non Pareto Optimal* (NPO) includes all solutions that are not Pareto optimal, whether they are dominated or non dominated.

With the previous features a two compartment model PO-NPO that can track the presence of PO solutions the population can be created. Other possible and more interesting models are availble in a three compartment configuration such as PO-NDNP-DOM that tracks both the number of Pareto Optimal and regular Non Dominated solutions. The POpnew-POold-NPO can track the ability of the algorithm to retain PO solutions found just one generation before with the POold feature, while the POA-PON-NPO with is more restrictive application of

the recentness of solutions can give a more detailed tracking. PON conveys the ability of the algorithm of keeping Pareto Optimal solutions in its population and POA its ability to discover never previously seen solutions.

One interesting note on POA is that the feature can be used to obtain the total number of found PO solutions of the algorithm, which can later be employed to rank the algorithm against others on the same problem.

3.2.2 NDS based features

For problems where the POS is not available, DCMs can still be employed with features defined around a more general set as one formed by only Non Dominated solutions in the population. Those features are, ND, NDNew, NDOld and DOM.

Non Dominated (ND) solutions, which are simply the solutions that are according to Pareto dominance, not dominated by any other solution in the population. If recentness is introduced, then *Non Dominated New* (NDNew) can be defined as a solution that is non dominated in generation t and is not present in previous generations populations from 0 to $t-1$. Otherwise, the solution is considered *Non Dominated Old* (NDOld). Finally, if the solution is dominated by any other in the population, it will be counted as *Dominated* (DOM).

Using these features one possible two compartmental model is a simple ND-DOM that only tracks the change in non dominated solutions in the population. Going to three compartmental configuration the distinction between when a non dominated solution appeared can be properly track with an NDNew-NDOld-DOM model.

One important note about this last model is what NDNew measures each time. While with POS based features the reference set is unique for the problem, for NDS features the considered set is the most recent set of non dominated solutions in the population. Thus, NDNew will count as new solutions some that eventually will be part of the approximation set found by the algorithm, as well as some solutions that initially appeared as non dominated but later became dominated.

Is possible to create a more robust NDS, by joining all the non dominated solutions found at each generation and eliminated the ones that become dominated, in order to have a more absolute version of NDNew. However, from experiments with the performance estimation model introduced later, it seems the current definition for the NDS manages to give good results while been simple and faster to compute, even allowing its measuring in an online fashion.

All the above mentioned models based on both types of sets can be seen summarized in Table 3.2.

Table 3.2: Possible models from different combinations of generational search assessment indices.

Two and three compartment models			
$\mathbf{x_t}$	$\mathbf{y_t}$	$\mathbf{z_t}$	**Model No.**
ND	DOM	-	2C-1
PO	NPO	-	2C-2
PO	NDNP	DOM	3C-1
POA	PON	NPO	3C-2
POpnew	POold	NPO	3C-3
NDnew	NDold	DOM	3C-4

3.3 Performance Estimation Model

In the previous section it was mentioned that a model with the feature POA can compute the total amount of PO solutions found by the algorithm, and this value later be used to rank algorithms and compare their performance. Using NDS features this is not directly possible, being necessary the introduction of another model that uses them to estimate a performance metric.

3.3.1 Relating Hypervolume to Population Features

Here is proposed the performance estimation model that from the changes in non dominated solutions estimates the hypervolume [58]. More specifically, the hypervolume measured at generation t (HV_t) over all non dominated solutions found so far by the algorithm; that is, from the non dominated set extracted after joining all non dominated sets $F_1(i)$, $i \in \{0, \cdots, t\}$. When measured on such a set at each generation, the hypervolume is known to increase monotonically [69, 70], which is necessary for the proposed model.

A monotonic increment can be represented by a model of the form $\mathrm{HV}_{t+1} = \mathrm{HV}_t + \Delta\mathrm{HV}_{t+1}$. The HV value at generation $t+1$ is estimated as the HV value at generation t, plus the growth of the metric at generation $t+1$, which can be estimated as a function of the newly found non dominated solutions. As such, $\Delta\mathrm{HV}_{t+1}$ can be approximated by a constant parameter μ and some combination of NDS features at generation t.

Searching for an expression that can work as the proposed model is done using Grammatical Evolution (GE) [71]. GE requires an objective function and a grammar. The quality of the expression found by GE is computed as the mean square error between the estimation produced by the model and the hypervolume measured on the joint non dominated set, as explained above. The grammar is defined as shown in Figure 3.3. GE is implemented using

$$
\begin{aligned}
\langle \text{expr} \rangle &\models \langle \text{expr} \rangle \langle \text{op} \rangle \langle \text{expr} \rangle \mid \langle \text{value} \rangle * \langle \text{var} \rangle \\
\langle \text{op} \rangle &\models + \mid - \mid \times \mid \hat{} \mid / \\
\langle \text{var} \rangle &\models \text{NDNew} \mid \text{NDOld} \mid \text{t} \\
\langle \text{value} \rangle &\models \langle \text{cat} \rangle \\
\langle \text{cat} \rangle &\models \langle \text{int} \rangle . \langle \text{int} \rangle \mid \langle \text{int} \rangle \\
\langle \text{int} \rangle &\models \langle \text{int} \rangle \langle \text{number} \rangle \mid \langle \text{number} \rangle \\
\langle \text{number} \rangle &\models [0\text{-}9]
\end{aligned}
$$

Figure 3.3: BNF Grammar used to search for an expression that relates the hypervolume to the features.

`gramEvol` [72], a library in `R`. The obtained expression after some trials and testing is as follows:

$$\text{HV}_{t+1} = \text{HV}_t + \frac{\mu \text{NDNew}_t}{t+1}, \tag{3.4}$$

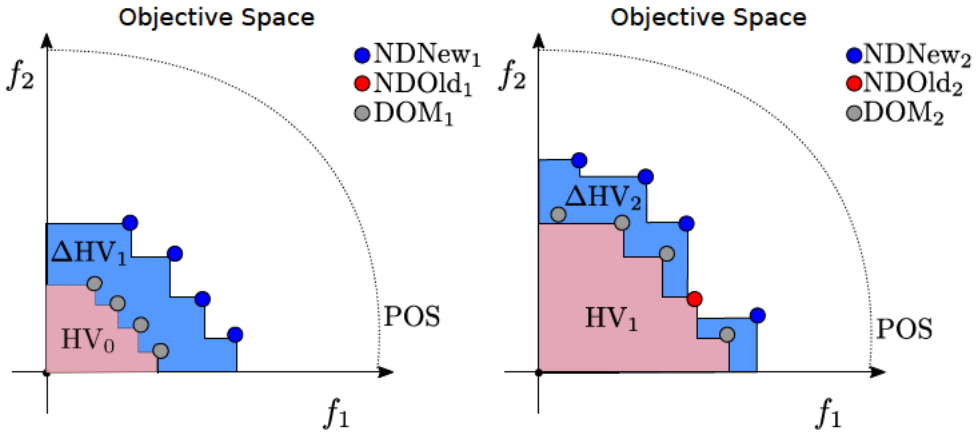


Figure 3.4: Example of how the hypervolume is estimated by using the non dominated new (NDNew) feature

Figure 3.4 illustrates the growth of hypervolume for the two initial generations. This figure will be used to explain how Eq. (3.4) is able to estimate the growth of hypervolume between generations. The algorithm starts from a random initial population, which is typically far from the POS. The growth of HV is expected to be fast during the initial generations, and to gradually slow down as the algorithm evolves the population towards the POS. Eq. (3.4) captures this behavior by dividing the number of newly found non dominated solutions NDNew by t. Thus, for the same number of NDNew, ΔHV would reduce with t,

as illustrated in Figure 3.4. Note that NDNew is also expected to reduce as the algorithm approaches the POS. The parameter μ represents how much the hypervolume increases per newly found non dominated solution, is estimated by fitting the model, and depends on the algorithm and the problem. Note that NDOld is not included in the expression to compute HV_{t+1} because it is already accounted in HV_t, as illustrated in the right side of Figure 3.4.

While for the previous DCM is clear that a estate of the compartments sizes the only input data needed to make estimations, with this new introduced model some extra steps are necessary.

When the model needs to be used to estimate the hypervolume, an initial population is created and on it the hypervolume HV_0 together with the NDS features $NDNew_0$, $NDOld_0$, and DOM_0 is measured. Then, DCM estimates the NDS features, particularly $NDNew_t$ for $t > 0$ until t_{max} generations. Next, the hypervolume estimation model estimates HV_{t+1} for $t > 0$ until t_{max} generations, using as input HV_t estimated by the hypervolume estimation model (except for $t = 0$ where the measured HV_0 is used) and $NDNew_{t+1}$ estimated by DCM.

Comparing with the DCM with POS based features there is one extra model to fit and use, nevertheless the resulting performance estimation is obtained in a more common metric such as the hypervolume.

3.3.2 Considerations about the model

Choosing the hypervolume as the performance metric to estimate forces the user to take into account some considerations when using the model. The first one is related to the reference point used in the computation of the hypervolume. Early empirical results, with other problems not discussed in this work, had brought to attention the importance of choosing an appropriate reference point so the hypervolume value is in a numerical range that eases the fitting process. In some cases, it may be necessary and easier to normalize the hypervolume to a [0,1] range. For a more detailed discussion on how to choose the reference point, the reader is referred to [73].

The second consideration is related to the computational cost of calculating the hypervolume on some problems. Since the model is shown the changes of this value on the accumulated non dominated population until each generation, there is the cost of computing this joint non dominated front for several iterations. The difficulty increases with both the number of objectives and solutions in the population, which is common when dealing with many-objective problems. Interest in the hypervolume metric and its extended use ensures that research on how to overcome these difficulties will arise in time, meanwhile, the reader is advised to look at some implementations and recommendations detailed [74].

Chapter 4

Algorithms and Experimental Settings

This chapter describes the experimental settings used to create the data to train the models. The first section briefly describes the main characteristic of each algorithm and its out-of-the-box configuration. The second section focused on the problem generator and the created instances, which are separated into datasets, that latter chapters will use to show different use cases for the models.

4.1 Representative Evolutionary Algorithms

In this section, the EAs selected for this work will be described briefly. Each one of them represents a different paradigm: Pareto Dominance, Pareto Dominance Relaxation, Decomposition, and Performance Indicators.

4.1.1 NSGAII

The Non dominated Sorting Genetic Algorithm II is an elitist[1] multi-objective evolutionary algorithm that uses Pareto dominance and density estimation, through crowding distance, to determine which solutions are retained and how parents are determined for the next generation.

During each generation, from the current population P_t, the equally sized Q_t that contains the generated offspring is created. After evaluating both sets according to the objectives functions, they are joint to be classified in non dominated $Fronts = \{Front_1, Front_2, \ldots, Front_n\}$. Inside each $Fronts$ a value called crowding distance is calculated that allows estimating for a particular solution the density of solutions surrounding it. To create the new population P_{t+1} solutions in each $Front_i$ are merged into P_{t+1} if this operation does not overfill it. If one of

[1] Preserves the best solutions found so far.

them will cause it, that $Front_i$ is sorted according to its crowding distance, and the necessary number of solutions to complete the population size are copied.

Fitness is determined by a tuple formed by the Front number that the solution belongs, and its crowding distance. Lower Front numbers, which indicate a better rank are preferred, and in case of a tie, higher crowding distance is preferred. This allows the retaining of solutions that cover a zone in the objective space with a low density of solutions. Selection of parents is done by binary tournament between randomly chosen individuals from the population using rank and crowding distance information. In Figure 4.1 the algorithm is presented in pseudo-code. For a more detailed explanation of the algorithm, in particular how the fast non dominated sorting procedure is done, please consult Deb et.al [60].

4.1.2 AεSεH

The Adaptive ε-Selection ε-Hood Genetic Algorithm is a Many-objective optimization algorithm that uses Pareto dominance relaxation in the form of ε-dominance to determine which solutions are retained and how parents are determined for the next generation. There is not an explicit method for fitness assignment in this algorithm.

Similarly to NSGAII, during each generation, the current population P_t and the generated offsprings Q_t are joined and classified in $Fronts$ according to standard non-domination. If $|Front_1| < popSize$ then the procedure continues normally as in NSGAII, where solutions are copied from the $Fronts$, and if a $Front_i$ were to cause an overfill, solutions are selected randomly from this last front until P_t has the correct size.

However, in the more common case of $|Front_1| > popSize$ in Many-objective optimization, ε-sampling with ε_s as parameter is done. This function samples randomly solutions from $Front_1$, copying them into P_{t+1} and eliminating from $Front_1$ all the ε-dominated solutions by the chosen sample. If P_{t+1} were to be overfilled, solutions are randomly eliminated until it reaches the correct size. Otherwise, if P_{t+1} still is not complete, the remaining solutions are randomly chosen from the previously discarded ε-dominated ones.

For parent selection, first ε-hood creation creates a cluster of solutions in objective space, which is done by selecting a solution randomly from the population and creating a neighborhood around it, composed by all the ε-dominated solutions using ε_h. This process is done until all solutions belong to a neighborhood. Then the function ε-mating visits each neighborhood in round-robin, selecting randomly two parents from each one of them. This assures that even solutions in low populated neighborhoods have the same reproduction probability.

At each generation, the ε_s used during selection and the ε_h used during the neighborhood creation are adapted taking into account a step size Δ and the population size. In particular,

```
1  t ← 0
2  P_0 ← InitializePopulation(popSize)
3  Evaluate(P_0)
4  NonDominatedSort(P_0)
5  while ¬ StoppingCondition() do
6      Parents ← SelectParentsByRankAndDistance(P_t)
7      Q_t ← CrossoverAndMutation(Parents)
8      Evaluate(Q_t)
9      R_t ← P_t ∪ Q_t
10     Fronts ←FastNonDominatedSort(R_t)
11     P_{t+1} ← ∅
12     L ← 0
13     while Front_i ∈ Fronts do
14         if Size(P_{t+1})+Size(Front_i) > popSize then
15             L ← i
16             break()
17         end
18         else
19             P_{t+1} ← Merge(P_{t+1}, Front_i)
20         end
21     end
22     if Size(P_{t+1}) < popSize then
23         Front_L ← SortByRankAndDistance(Front_L)
24         for j to popSize-Size(Front_L) do
25             P_{t+1} ← Front_L[i]
26         end
27     end
28     t = t + 1
29 end
30 return Front_1
```

Figure 4.1: Pseudo code of NSGAII.

ε_h is adapted so the number of neighborhoods is closer to the one specified by the user N_h^{Ref}. Pseudo-code of the algorithm can be found in Figure 4.2. For a more detailed explanation of the algorithm, please consult Aguirre et.al [65].

4.1.3 IBEA

The Indicator-Based Evolutionary Algorithm is a Many-objective optimization algorithm that relies on performance indicators to determine which solutions remain in the population. The fitness for each individual represents the loss of quality that will be incurred if this solutions were to be eliminated from the population, calculated as $Fitness(x) = \sum_{x' \in P \setminus \{x\}} -e^{-I(x',x)/k}$, where x is an individual, $I()$ the indicator function and $k > 0$ a scaling factor set by the user.

For each generation, the current population P, and its offspring Q are joint and the fitness for each individual is calculated. Then solutions with the lowest fitness are eliminated until the size of the population reaches the correct size. When a solution is removed from the population, an update is done to reflect the new fitness without the eliminated solution.

During the parent selection, a binary tournament-based on fitness is done between randomly selected solutions from the current population.

Note that IBEA transforms the original optimization problem into one, where the main goal is to obtain an improvement in the overall performance of the population according to the selected indicator.

In this work, the following two indicators are used in conjunction with IBEA. The binary additive ε-indicator ($I_{\varepsilon+}$) and the binary Hypervolume difference-indicator (I_{HD}).

$$I_{\varepsilon+}(x',x) = \max_{i \in \{1,\ldots,n\}} \{f_i(x) - f_i(x')\} \tag{4.1}$$

$$I_{HD}(x,x') = \begin{cases} H(x') - H(x) & \text{if } x' \succeq x \text{ or } x \succeq x' \\ H(x+x') - H(x) & \text{otherwise} \end{cases} \tag{4.2}$$

where $x \succeq x'$ indicates that x dominates x' in a Pareto sense.

$I_{\varepsilon+}(x,x')$ gives the minimum value by which a solution x needs to be translated in the objective space so it can weekly dominate x'. $H(x)$ gives the multidimensional volume of the objective space that is dominated by x. $I_{HD}(x,x')$ gives the hypervolume that x' dominates, but not by x. The algorithm in pseudo-code is presented in Figure 4.3. Further information about the algorithm or the Performance indicators used here can be found in the work done by Zitzler et.al [67].

```
1  t ← 0
2  N_h^Ref ← popSize/H_size^Ref // Set reference number of neighborhoods
3  ε_s, ε_h ← 0
4  Δ_s, Δ_h ← Δ_s0, Δ_h0 // Set step adaptation
5  P = ∅
6  Q ← InitializePopulation(popSize)
7  while ¬ StoppingCondition() do
8      Evaluate(Q)
9      R ← P ∪ Q
10     Fronts ←FastNonDominatedSort(R)
11     if Size(Front_1) < popSize then
12         while Front_i ∈ Fronts do
13             if Size(P_t+1)+Size(Front_i) > popSize then
14                 L ← i
15                 break()
16             end
17             else
18                 P_t+1 ← Merge(Parents, Front_i)
19             end
20         end
21         if Size(P_t+1) < popSize then
22             for j to popSize-Size(Front_L) do
23                 P_t+1 ← Front_L[random()]
24             end
25         end
26     end
27     else
28         P_t+1,N_s ← epsilonSampling(Front_1,ε_s,popSize)
29         adapt(ε_s,Δ_s,popSize,N_s)
30     end
31     H,N_h ← epsilonHoodCreation(P_t+1,ε_h) //H = {H_1,H_2,...,H_N_h}
32     adapt(ε_h,Δ_h,N_h^Ref,N_h)
33     Parents ← epsilonHoodMating(H,popSize)
34     Q ← CrossoverAndMutation(Parents)
35     t = t + 1
36 end
37 return F_1
```

Figure 4.2: Pseudo code of AεSεH.

```
1  t ← 0
2  Initialize population P of size n randomly
3  while ¬ StoppingCondition() do
4  |  EvaluateUsingIndicator(P)
5  |  if Size(P) > popSize then
6  |  |  RemoveLowestFitnessIndividual(P)
7  |  |  UpdateFitness(P)
8  |  end
9  |  Parents ← Selection(P)
10 |  Q ← CrossoverAndMutation(Parents)
11 |  EvaluateUsingIndicator(Q)
12 |  P ← Merge(P ∪ Q)
13 |  t ← t + 1
14 end
15 A ← NonDominated(P)
16 return A
```

Figure 4.3: Pseudo code of IBEA.

4.1.4 MOEA/D

The Multi-objective Evolutionary Algorithm Based on Decomposition, as the name suggests, decomposes the problem into single-objective ones using scalarization functions and optimizing them simultaneously. Subproblems are optimized using the surrounding solutions that belong to their neighborhood.

The algorithm requires a set of weigh vectors $\Lambda \leftarrow \{\lambda_1, \lambda_2, \ldots, \lambda_N\}$ defined by the user, a scalarization function, in this case is considered the Tchebycheff function, and a $Neighborhood_{Size}$. First, a population of P of N individuals is created and evaluated, followed by assigning each $x_i \in P$ to a particular weigh vector λ_i. Neighborhoods are created by calculating the Euclidean distance between any pair of weigh vectors and clustering the $Neighborhood_{Size}$ ones that are closer. The reference point needed for the scalarization function is initialized, using a problem specific technique or setting the value for each objective as the best one according to the current population.

During each generation, for each subproblem with a weigh vector λ_i, a new individual x'_i is created by applying genetic operators to parents selected randomly inside the neighborhood $Neig(i)$. If the solution dominates some in the neighborhood a replacement is done, otherwise, it is discarded. The reference point is updated, as the external population. This process is repeated until all the subproblems have being visited and updated. The algorithm terminates when its stopping condition is met.

```
1  t ← 0
2  externalP ← ∅
3  Λ ← {λ1, λ2, ..., λN}
4  Initialize population P of size N randomly
5  Evaluate(P)
6  Randomly assign each weight λi a solution from P = {x1, ..., xN}
7  Neig ← CreateNeighborhoods(λ,Neighborhood_Size)
8  Initialize the reference point z = (z1, ..., zm)
9  while ¬ StoppingCondition() do
10     for λi ∈ Λ do
11         Parents ← SelectParents(Neig(i))
12         x'i ← CrossoverMutation(Parents)
13         Evaluate(x'i)
14         UpdateReferencePoint(z)
15         UpdateNeighbors(x'i,Neig(i))
16         Update(externalP)
17     end
18     t ← t + 1
19 end
20 return externalP
```

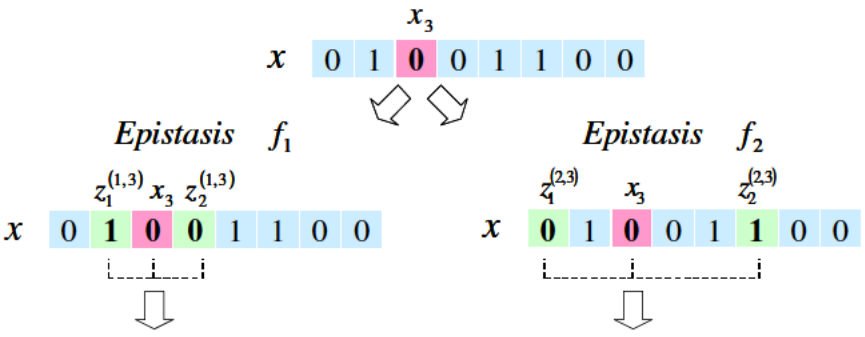
Figure 4.4: Pseudo code of a MOEA/D.

An important note here is that the replacement is done the moment a better solution is found and it could replace any of the ones in its neighborhoods. This means that, since neighborhoods overlap, the next subproblem is working with the best solutions found so far.

The pseudo-code of the algorithm is shown in Figure 4.4. More details of the algorithm and other possible scalarization functions can be found in Zhang et.al [66].

4.2 Test Problem Generator

The dynamic compartmental models study the behavior of an algorithm while solving a problem instance, so is important to choose a versatile problem generator to create instances that can test the model abilities. MNK-Landscapes [75] is a multiobjective extension of Kauffman's NK-Landscapes[76] that can generate adjustable instances of multi- and many-objective optimization problems given some parameters. The number of objectives can be controlled with M, N defines the number of bits of the string that represents the decision variables, while K controls the ruggedness of the landscape by determining the epistatic interactions between the decision variables.



$z_1x_3z_2$	$f_{1,3}$
000	0.83
001	0.34
010	0.68
011	0.10
100	0.67
101	0.24
110	0.60
111	0.64

$z_1x_3z_2$	$f_{2,3}$
000	0.21
001	0.48
010	0.81
011	0.23
100	0.19
101	0.44
110	0.30
111	0.72

Figure 4.5: Epistatic Interactions example.

The last parameter epistasis comes from biology where this term refers to the expression of one gene being masked by the genotypic effect of another one, due to the non-linear interdependence found in genes. In terms of a combinatorial optimization problem, this translates to fitness functions and the non-linear interdependence between variables that can make a single decision variable impact how other variables contribute to the overall fitness.

For a small example, suppose a binary decision vector, where all variables can be on or off, 0 or 1. Given a flip in one variable, whereas this could represent a gain in the contribution made by it, this change can also have a side-effect of decreasing the contribution made by other variables that interact with it. In the context of solving an optimization problem these interactions, which are commonly unknown beforehand, can impact the performance of the search for an optimal solution. To get a better idea of how this occurs, in Figure 4.5 there is a more detailed example of epistasis and epistatic interaction.

Let $f_{1,3}(x_3, z_1^{(1,3)}, z_2^{(1,3)})$ and $f_{2,3}(x_3, z_1^{(2,3)}, z_2^{(2,3)})$ be the fitness functions associated to bit x_3 contributing to the first objective function $f_1(\cdot)$ and the second one $f_2(\cdot)$, respectively, based on different epistatic models for each objective. $z_1^{(1,3)}$, $z_2^{(1,3)}$, $z_1^{(2,3)}$ and $z_2^{(2,3)}$ are determined when the instance is created, as well as the possible combinations and effects it has on each fitness function, shown in the table inside Figure 4.5. In $f_{1,3}$, x_3 epistatically interacts with its left and right neighboring bits, $x_2 = z_1^{(1,3)}$ and $x_4 = z_2^{(1,3)}$. While for $f_{2,3}$, x_3 epistatically interacts with its second bit to the left and with its third bit to the right, $x_1 = z_1^{(2,3)}$

and $x_6 = z_2^{(2,3)}$. For the example, $N = 8$ since there are 8 decision variables and $K = 2$ since each x_i effect on the fitness function depends on other two variables.

It can be seen from the table that if x_2 and x_4 keep their values as well as x_1 and x_6, 0 is a good value for x_3, making its contribution to $f_1(\cdot)$ be 0.67 and $f_2(\cdot)$ be 0.48. If a bit flip were to happen, then the contributions will reduce to 0.60 and 0.23, respectively. However, evolutionary algorithms sometimes use operators that affect multiple bits at once that may or not be contiguous, so when they are not aware of these interactions is easy to see how quickly a flip could bring the fitness contribution of x_3 on both objectives down. Learning these interactions, while solving the problem is not always a straightforward task but several techniques exist [77].

Using this problem generator, several small and large instances were generated and solved with the algorithms described in this chapter. To explore different aspects of the models and the features, they were collected into three different datasets, described below. In cases where one or more algorithms were run with different configurations, it will be specified which instances were used.

4.2.1 Datasets

Dataset 1

This dataset has been created to test the models' ability to do analysis and estimation on small instances, specifically with the features related to the Pareto Optimal Set present in models 3C-1 and 3C-2. It consists of four instances of MNK-landscapes problems with different numbers of objectives, M = 3, 4, 5, and 6, but the same number of variables N = 20, and the epistasis set to K = 1. The low number of variables allows finding the Pareto Optimal Set by enumeration, which is key to compute all the POS related features.

All the algorithms were used to solve each instance 30 times, initializing the population with different seeds. For each instance, different configurations, that is, different population sizes were tried as Table 4.1 indicates, and in all runs the maximum number of generations was set to 100. The sizes were chosen so as to follow the indications in Aguirre et.al [78].

Dataset 2

In this second dataset, the goal was to use the model 3C-2 to select a population size between several options, so only one instance was created with M = 5, N = 20, and K = 1 of an MNK-landscape. Again, the choice of a low N allowed the problem to be enumerated and obtain the Pareto Optimal Set.

The algorithm selected to solve the instance is the AεSεH, which was run 30 times with different seeds to initialize its initial population on different configurations. The population sizes considered are in the [150,750] interval, with increments of 50. To train the models,

Table 4.1: Population Sizes considered per instance. A "✓" indicates that the population size was used for the instance.

Instance	Population Size							
M	50	100	200	500	1000	2000	4000	5600
3	✓	✓	✓	-	-	-	-	-
4	✓	✓	✓	✓	✓	-	-	-
5	✓	✓	✓	-	-	✓	✓	-
6	✓	✓	✓	-	-	-	-	✓

only the population sizes 150, 350, 550 and 750 are considered, while the models for the intervals [200,250,300], [400,450,500] and [600,650,700] are obtained by interpolation, as discussed in Chapter 5. Nevertheless, to be able to compare the interpolated models' estimations, the algorithms are run on all the population sizes listed here. One difference with the previous dataset is that all configurations are executed for the same number of function evaluations (FE). In evolutionary algorithms terms, is the number of calls to the function that evaluates the solution. Since the number of solutions varies for each configuration, the maximum number of generations is defined as $\text{MaxGen} = \text{FE}/\text{PopSize}$. The algorithms were run until 15000 FE.

Dataset 3

This third dataset contains landscapes with N = 20 bits, K=1 epistatic bits, and M = 3, 4, 5, objectives. In total 30 instances per sub-class are independently generated and the chosen algorithm, AεSεH, is run for a budget of 200000 (FE), with population sizes 50, 100, and 200. This dataset will be used to test both, DCM with POS and NDS features, which forced the choice of a reasonable number of variables so the POS can be obtained by enumeration.

Dataset 4

The fourth dataset is the most interesting one since it emulates the use of models in a more general setting. The instances created have M = 3, N = 100, and K = 5, so the problem is large and not enumerable, but perfect to test the model 3C-4 features. For this problem 50 instances were created, reserving 30 to train the models and 20 as unseen data to test the model estimations and ability to generalize. As with the previous dataset, four configurations were considered, population sizes of 3000, 5500, 8000, 10500, all run until 800000 FE. Each instance was run a single time per configuration with the AεSεH algorithm. There is an exception for one instance from the testing set, where a total of 30 runs were made to also see how the model copes with the variability that comes from different initial populations. The results obtained on this dataset are discussed in Chapter 6.

Dataset 5

The final dataset created for this work contains more variations in terms of number of objectives, variables, and interaction between variables in order to test if model trained on one problem sub-class are able to obtain good estimations for sub-classes close in those dimensions. To test transferability in the objectives domain the sub-classes with N = 100, K =5, and M = 3, 4, and 5 objectives were generated. For transferability on the variables domain the sub-classes M = 4, K =5 and N = 90, 100 and 110 were generated. Finally, to test the same but on the variable interactions domain, the sub-classes M = 3, N = 100, and K = 3, 5, 10 were also generated. Per sub-class 30 instances were created, and solved using AεSεH with population 200.

4.2.2 On the use with other problems

In this manuscript, models were created on data generated by solving instances of MNK-Landscapes due to its ability to generate easily problems with a different number of objectives, variables, or even number of interactions that directly impact their difficulty. However, since the models themselves learn the behavior through data of the algorithm run, namely the composition of the population, the only limit to apply the model for other types of problem generators, benchmarks, or even real-world instances would be the ability of the chosen algorithm to solve the problem. In particular, is important that the algorithm run data fed to the models shows that it is converging or has converged already. Empirical tests have shown that when this is not the case, the fitted model estimations will deviate from the mean of the runs and misrepresent the algorithm's behavior.

Chapter 5

DCMs based on POS features and estimation of Accumulated PO solutions

This chapter covers the use of dynamic compartmental models on small and enumerable landscapes by using features that require the Pareto Optimal Set to be known. First is given an example of one method to fit the models and how the learned parameters can be used to analyze algorithms. The remaining of this chapter explores how new models can be discovered from learned ones and use this information to guide the algorithm's configuration.

5.1 Model Fitting

Obtaining the model's parameters can be done in several ways, for some models, it is possible to arrive at a closed-form solution and solve it analytically, while for others fitting the equations using experimental data is a better approach. Here, the latter approach is taken with some transformations on the original equations of the model to facilitate the process. The transformation decouples the system of equations, so each equation does not depend on the others' output [79].

The example here is based on Eq. (3.2), but the same steps apply for Eq. (3.1). Rewriting the system in matrix form.

$$X_{t+1} = AX_t \tag{5.1}$$

where

$$A = \begin{bmatrix} 1-(\alpha+\beta) & \bar{\alpha} & \bar{\beta} \\ \alpha & 1-(\bar{\alpha}+\gamma) & \bar{\gamma} \\ \beta & \gamma & 1-(\bar{\beta}+\bar{\gamma}) \end{bmatrix} \qquad X_t = \begin{bmatrix} x_t \\ y_t \\ z_t \end{bmatrix}$$

Assuming that X_∞ is an equilibrium point of the system X_t. This is true if and only if $X_\infty = AX_\infty$ and X_∞ is an eigenvector associated to the eigenvalue 1 of A. Based on this, A has to be diagonalizable, and at least one of its eigenvalue is known and is 1. Parting from this assumption, the system can be decoupled and its eigenvalues and eigenvectors are used to rewrite the system.

Diagonalizing A as KLK^{-1} where L is a diagonal matrix with A's eigenvalues λ_1, λ_2 and λ_3 as the diagonal elements and K is a matrix whose columns are A's eigenvectors $k_1 = [k_{11}, k_{21}, k_{31}]^\intercal$, $k_2 = [k_{12}, k_{22}, k_{32}]^\intercal$ and $k_3 = [k_{13}, k_{23}, k_{33}]^\intercal$.

Therefore, the system is now:

$$\begin{cases} x_t = k_{11}\lambda_1^t + k_{12}\lambda_2^t + k_{13}\lambda_3^t \\ y_t = k_{21}\lambda_1^t + k_{22}\lambda_2^t + k_{23}\lambda_3^t \\ z_t = k_{31}\lambda_1^t + k_{32}\lambda_2^t + k_{33}\lambda_3^t. \end{cases} \tag{5.2}$$

Using Eq. (5.2), the model is fit against the data. To express them again in the original system terms, the estimated values for the eigenvectors k_1, k_2, k_3 and eigenvalues $\lambda_1, \lambda_2, \lambda_3$ are arranged in matrix form.

$$K = \begin{bmatrix} k_{11} & k_{12} & k_{13} \\ k_{21} & k_{22} & k_{23} \\ k_{31} & k_{32} & k_{33} \end{bmatrix} \qquad L = \begin{bmatrix} \lambda_1 & 0 & 0 \\ 0 & \lambda_2 & 0 \\ 0 & 0 & \lambda_3 \end{bmatrix}$$

And A can be obtained back as:

$$KLK^{-1} = A \tag{5.3}$$

For the fitting process, finding the system Eq. (5.2) parameter values is formulated as an optimization problem and using data on the number of individuals that belong to each compartment. The goal is to find the best set of parameters that produces values close to the ones observed in the experimental data obtained by running the algorithms. This is performed by minimizing the mean square error $mse = \frac{1}{n}\sum_{i=1}^{n}(Dp_i - \bar{D}m_i)^2$, where n is the number of observations (generations) in the experimental data, Dp_i is the prediction made by the

model, and $\bar{D}m_i$ is the average measured values across all runs for the i-th generation where $i = 0, 1, \ldots, 100$.

Looking at it in other terms, the model is learning the expected value of each feature. This comes from the fact that MOEAs are stochastic (randomly generated initial population, crossover, mutation probabilities, parent selection strategies), so each run of the algorithm will give different traces in terms of the features. However, the results produced by these algorithms should be of similar quality, so the traces difference are within a reasonable range. If the model fitting is done properly, it is expected that when given the model the features (x_0,y_0,z_0) measured on different initial populations, will produce an estimation that reflects some of the variance found on that particular MOEA.

This optimization problem is solved using the Covariance Matrix Adaptation Evolutionary Strategy (CMA-ES)[36], a recognized single-objective numerical optimizer. The initial parameters are set at random. The process will stop when there is no improvement in the *mse* or the max number of iterations maxiter $= d^2 \times 100$ have been reached, where d is the number of parameters. The coefficient of determination or $R^2 \in [0, 1]$ is computed to measure the goodness of the fit, i.e. the proportion of variability in the dependent variable explained by the independent variable of the model. If the R^2 given by the model is smaller than 0.95, the optimization is resumed until its improvement is smaller than 0.001.

Be noted that each equality in Eq. (5.2) is related to a particular compartment and could be fitted separately. However, since the parameters λ are common in all equations, they are fitted only once, reusing their values for the two remaining groups. To further reduce the number of parameters to fit, using the assumption made earlier, at least one eigenvalue has to be 1, therefore λ_1 is set to 1 when fitting the model.

5.2 Model Quality

We fit models 2C-1, 3C-1 and 3C-2 for problems with 3, 4, 5, 6 objectives and all population sizes described in Table 4.1 from Section 4.2.1. To generate the prediction, we set the initial state of each compartment ($t = 0$) with the data from the initial population (generation 0), for each run of each configuration, and use the model to generate estimates for all generations ($t = 1, \ldots, 100$), using the values estimated at time t as input to estimate the state at $t + 1$. Using these predictions, we again calculate the R^2 for each run and compute its average. Tables 5.1 and 5.2 illustrate the average R^2 for all the models mentioned on the data obtained from the three objective problem solved using a population size of 200. Figures 5.1-5.3 show the measured values in black and the prediction of the model in red. For space reasons, these figures only include results corresponding to three compartment model 3C-2.

From these tables, we can see that in the two compartment model 2C-1 all algorithms, except MOEA/D, show very high R^2 values on all features. In the three compartment models, again MOEA/D shows lower R^2 values on all features. In addition, all algorithms show relatively low R^2 values on features NDNP and POA in models 3C-1 and 3C-2, respectively.

The reason why MOEA/D achieves lower scores is due to its higher variance in the data from run to run. However, note that the model follows the average of the runs without issues. The NDNP feature, not shown here, is characterized by a sharp peak between generations 5 and 10, also with a large variance in the data. For the POA feature, we can see in Figure 5.1 that the change in the data is small and concentrated on the first 20 generations, with a relatively large variance in the data, making the learning of this particular section difficult. The model follows the general shape of the set of runs but may underestimate some. In all cases where R^2 is lower than 0.95, we see that the model still provides a good idea of where each group will end after some generations. This is significant, more so if it is considered that only one actual measured value at $t = 0$ was used to make the estimations.

Notice that the accumulated PO can be obtained by having the PO Abs New values at each generation. By doing a cumulative sum of this value, we obtain Figure 5.4. Note here that the under- and overestimation of our model (red) becomes more noticeable compared to the actual measured data (black). But in each case, we see that the prediction gets closer to the real values by the final generations.

Table 5.1: Two Compartment Model. Average R^2 of individuals runs for M=3, Pop=200.

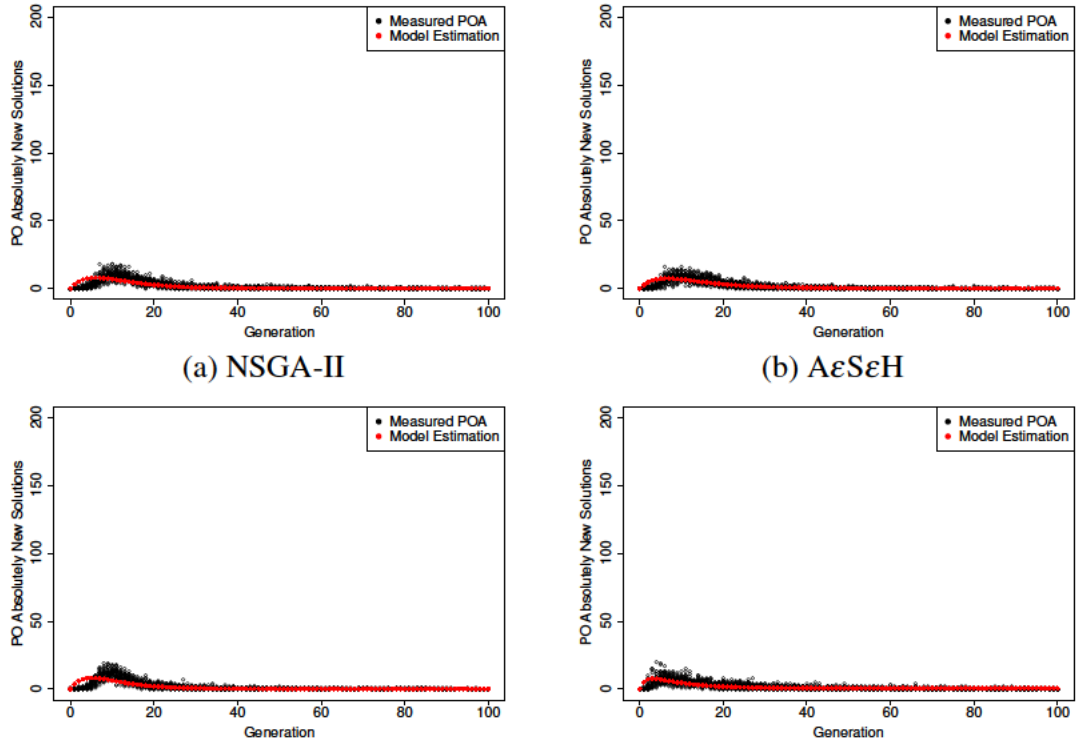
Model 2C-1		
Algorithm	**ND**	**DOM**
NSGA-II	0.963	0.963
AεSεH	0.972	0.972
IBEA$_{HV}$	0.957	0.957
IBEA$_{\varepsilon}+$	0.963	0.963
MOEA/D	0.641	0.641

5.3 Parameter and Feature Analysis

This section will focus on the parameters and their role in different types of analysis. The key element is their ability to encapsulate the dynamics of the population composition in a small set of values.

Table 5.2: Three Compartment Models. Average R^2 of individuals runs for M=3, Pop=200.

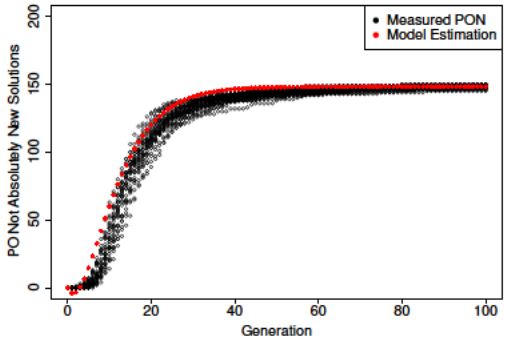
	Model 3C-1			**Model 3C-2**		
Algorithm	**PO**	**NDNP**	**DOM**	**POA**	**PON**	**NPO**
NSGA-II	0.963	0.651	0.964	0.564	0.958	0.958
AεSεH	0.974	0.857	0.968	0.650	0.977	0.977
IBEA$_{HV}$	0.973	0.795	0.970	0.532	0.971	0.971
IBEA$_{\varepsilon+}$	0.958	0.695	0.962	0.518	0.966	0.966
MOEA/D	0.795	0.359	0.645	0.510	0.854	0.789



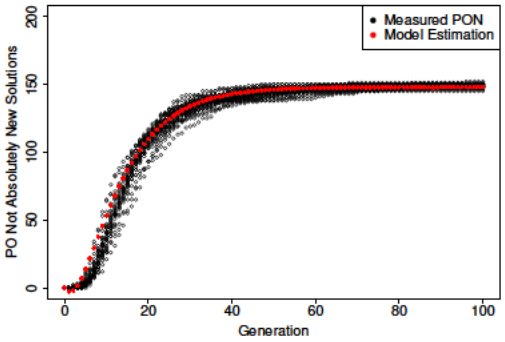
(a) NSGA-II (b) AεSεH

(c) IBEA$_{HV}$ (d) MOEA/D

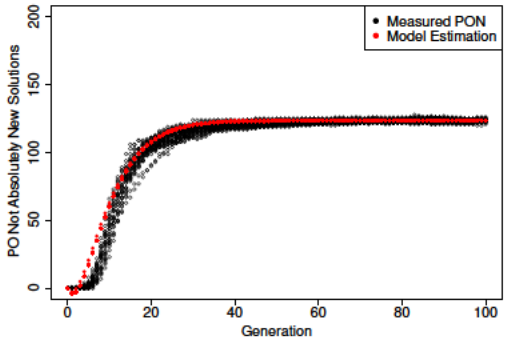
Figure 5.1: Model 3C-2 estimation vs actual PO Absolutely New data.



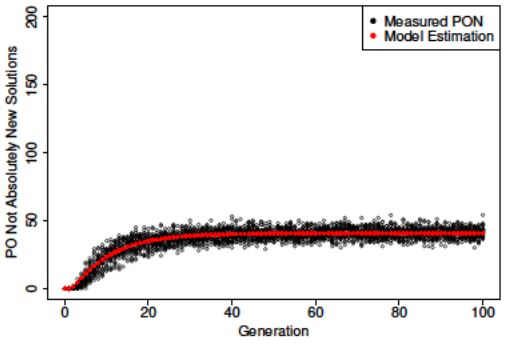
(a) NSGA-II



(b) AεSεH

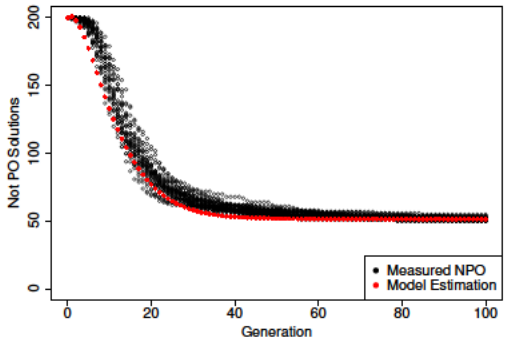


(c) IBEA$_{HV}$

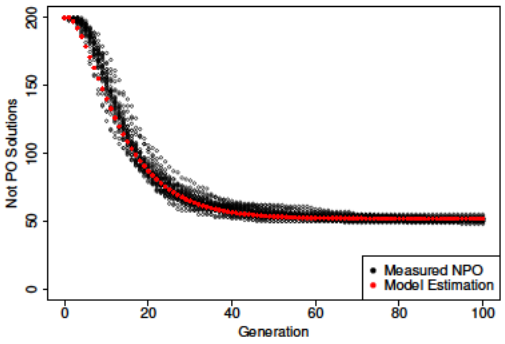


(d) MOEA/D

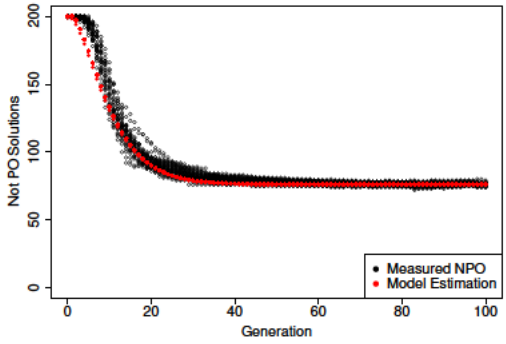
Figure 5.2: Model 3C-2 estimation vs actual PO Not Absolutely New data.



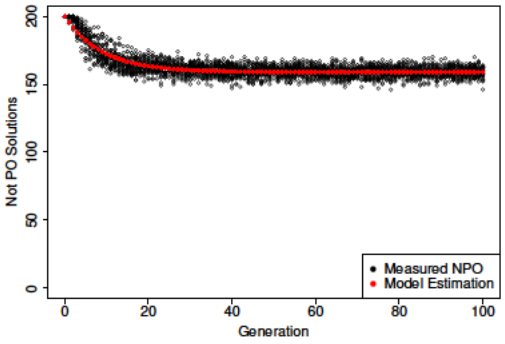
(a) NSGA-II



(b) AεSεH

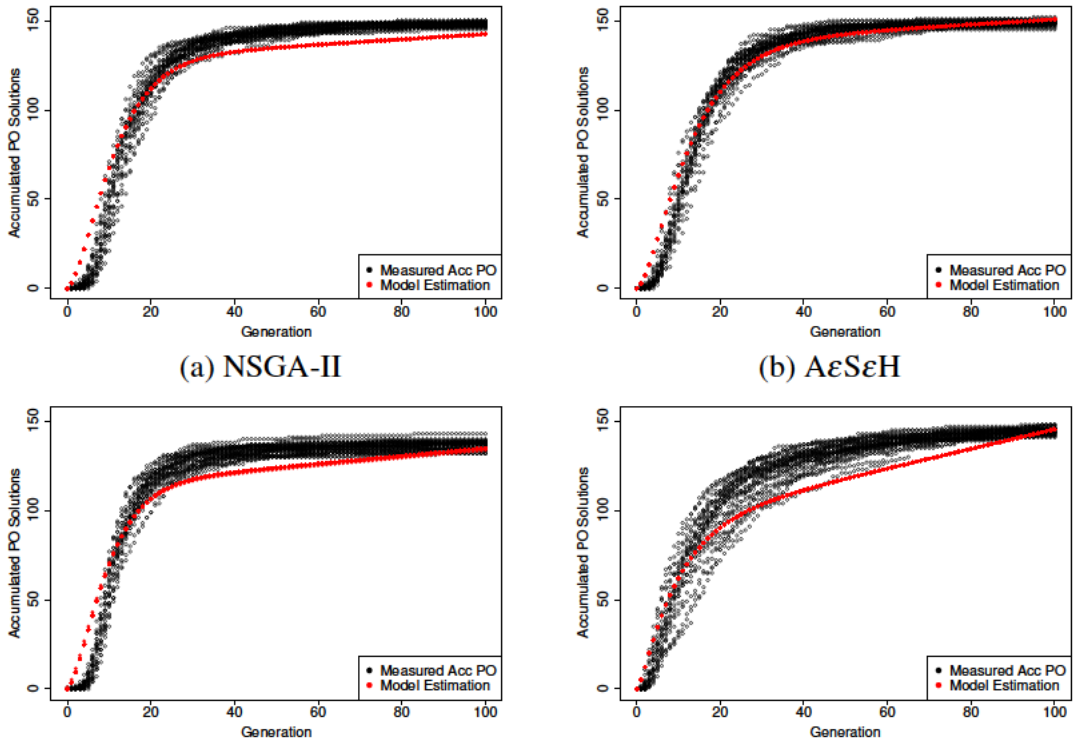


(c) IBEA$_{HV}$



(d) MOEA/D

Figure 5.3: Model 3C-2 estimation vs actual Non PO data.



(a) NSGA-II (b) AεSεH

(c) IBEA$_{HV}$ (d) MOEA/D

Figure 5.4: Model 3C-2 estimation vs actual PO Accumulated data.

5.3.1 Interpreting the Model Parameters

A parameter relates two compartments within the model. In order to interpret them, it is necessary to know what they encapsulate according to the model equations. As an example, looking at the three compartment model 3C-1 listed in Table 3.2 and expanding the first two equations of the system of Eq. (3.2) it can be written:

$$PO_{t+1} = PO_t - \alpha PO_t + \cdots$$

$$NDNP_{t+1} = NDNP_t + \alpha PO_t + \cdots .$$

Focusing on the parameter α that relates the number of PO and NDNP solutions, it can be noted that the values of PO and NDNP at time $t+1$ change in opposite directions, both proportionally to PO at time t by αPO_t. Hence, a loss in one compartment becomes a gain in the other. However, α can take positive or negative values. Thus, when $\alpha > 0$, the model tells that the number of PO solutions reduces with time proportionally to its previous value and the same amount increases in NDNP. On the other hand, when $\alpha < 0$, the model says that the number of PO solutions increases proportionally to its previous value and the same amount

reduces in NDNP. In the above example, $\alpha > 0$ can be interpreted as negative feedback of PO on itself, whereas $\alpha < 0$ has positive feedback of PO on itself.

It is important to clarify that changes in the numbers in each compartment do not imply the transition of the individuals themselves, since one of the assumptions of the model is that individuals are not tracked, only how many are present at each compartment. However, it remains true that a loss (decrease in number) in one compartment is correlated to an equal gain (increase in number) in another compartment by the corresponding parameter of the model.

5.3.2 Model based Population Dynamic's Analysis

The differential equations model the influence between compartments using two parameters per pair of compartments. Thus, in a three compartment model, the equation for each compartment is specified by 2×2 parameters. This allows observing in detail the mutual influence between compartments. It is possible also to capture these influences in a compact and visual form using the graphical representation shown in Figure 3.2.

To illustrate how the model can enrich the analysis, in this section the focus will be on AεSεH and MOEA/D using the Dataset 1 which details can be found in Chapter 4. Using the estimated values of the parameters and simplifying the equations, is shown how each compartment depends on itself and the two other compartments. This allows a better understanding of the overall compartments' dynamics. In the following, the equations marked with *A* should be considered as the ones for AεSεH and with *M* the ones for MOEA/D.

The first analysis is done on the three compartment models 3C-1, composed of compartments for Pareto Optimal (PO), Non Dominated Non PO (NDNP), and Dominated (DOM) solutions for three (M3) and six (M6) objectives with population size 50 (P50) and 200 (P200), respectively.

Starting with the model M3P50 3C-1 which simplified equations are as follows:

M3P50 3C-1

$$\begin{aligned} \mathrm{PO}^A_{t+1} &= 0.998\mathrm{PO}^A_t + 0.105\mathrm{NDNP}^A_t - 0.043\mathrm{DOM}^A_t \\ \mathrm{PO}^M_{t+1} &= 0.984\mathrm{PO}^M_t + 0.136\mathrm{NDNP}^M_t - 0.029\mathrm{DOM}^M_t \end{aligned}$$

$$\begin{aligned} \mathrm{NDNP}^A_{t+1} &= 0.892\mathrm{NDNP}^A_t + 0.003\mathrm{PO}^A_t + 0.165\mathrm{DOM}^A_t \\ \mathrm{NDNP}^M_{t+1} &= 0.706\mathrm{NDNP}^M_t - 0.103\mathrm{PO}^M_t + 0.127\mathrm{DOM}^M_t \end{aligned}$$

$$\text{DOM}^A_{t+1} = 0.878\text{DOM}^A_t - 0.001\text{PO}^A_t + 0.003\text{NDNP}^A_t$$
$$\text{DOM}^M_{t+1} = 0.902\text{DOM}^M_t + 0.119\text{PO}^M_t + 0.158\text{NDNP}^M_t$$

For 3-objectives problems, by randomly sampling a subset of solutions, it is very unlikely to generate a PO solution. Thus, it can be considered that there are no PO solutions in the initial population, $\text{PO}_{t=0}$ is 0, and all solutions are either DOM or NDNP, which is in accordance with the data. Looking at different initial populations it can be observed that for 3 objectives more solutions are DOM than NDNP. In the case of P50, 84% are DOM and 16% are NDNP.

Starting from these initial conditions, an analysis of the compartments' dynamics can be done. From the PO equations, an increase of PO solutions in AεSεH and MOEA/D can be seen to depend mostly on PO itself and in a small fraction on NDNP solutions. Since $\text{PO}_{t=0}$ is 0 and $\text{NDNP}_{t=0}$ is small, it is expected that PO would remain 0 for some generations.

From the NDNP equations, it can be noted in both algorithms that, only a fraction of NDNP remains at each generation, but NDNP gains a fraction of DOM. Since DOM $>$ NDNP the number of solutions gained from DOM is greater than those lost from NDNP, so both algorithms should be gaining NDNP solutions during the initial generations, being the gain faster in AεSεH than in MOEA/D as indicated by the coefficients, i.e. $0.892 > 0.706$ and $0.165 > 0.127$. However, from the DOM equations, notice that during the same initial generations both algorithms should lose DOM solutions (DOM_t coefficients < 1 in both algorithms), being the loss faster in AεSεH than in MOEA/D.

As detailed above, in the initial generations for both algorithms DOM decreases while NDNP is increasing. After a few generations, the situation starts changing, as DOM size becomes so small that is unable to sustain the growth of NDNP. Therefore NDNP will reach a peak and then decrease on both algorithms. The effect should be more noticeable in AεSεH since DOM decreases faster. PO starts increasing when the gain from NDNP surpasses the loss from DOM and remains high as most of PO solutions are retained according to the coefficients in both algorithms.

It is important to note that for AεSεH, the coefficients indicate that DOM would decrease continuously and approach 0. For MOEA/D, on the other hand, the losses from DOM are compensated from the gains from PO and NDNP. Furthermore, NDNP also gets some contribution from DOM. Thus, DOM will reduce up to a point and then stabilize in a value greater than the one in AεSεH. Something similar happens with NDNP, where NDNP in MOEA/D stabilizes in a value larger than in AεSεH.

Now the same compartmental model is considered but with 6 objectives (M6) and population 200 (P200), the equations are the following:

M6P200 3C-1

$$\begin{aligned}\mathrm{PO}^A_{t+1} &= 0.991\mathrm{PO}^A_t + 0.121\mathrm{NDNP}^A_t - 0.346\mathrm{DOM}^A_t\\ \mathrm{PO}^M_{t+1} &= 0.907\mathrm{PO}^M_t + 0.246\mathrm{NDNP}^M_t - 0.107\mathrm{DOM}^M_t\end{aligned}$$

$$\begin{aligned}\mathrm{NDNP}^A_{t+1} &= 0.904\mathrm{NDNP}^A_t + 0.006\mathrm{PO}^A_t + 0.857\mathrm{DOM}^A_t\\ \mathrm{NDNP}^M_{t+1} &= 0.580\mathrm{NDNP}^M_t + 0.06\mathrm{PO}^M_t + 0.458\mathrm{DOM}^M_t\end{aligned}$$

$$\begin{aligned}\mathrm{DOM}^A_{t+1} &= 0.178\mathrm{DOM}^A_t + 0.003\mathrm{PO}^A_t - 0.025\mathrm{NDNP}^A_t\\ \mathrm{DOM}^M_{t+1} &= 0.649\mathrm{DOM}^M_t + 0.033\mathrm{PO}^M_t + 0.174\mathrm{NDNP}^M_t\end{aligned}$$

The likelihood of generating non dominated solutions and finding some PO solutions by random sampling increases in small landscapes with 6 objectives. Indeed, analyzing the composition of different initial populations for M6 and P200 it is observed that 1% are PO solutions, 47% are NDNP solutions and the remaining 52% are DOM solutions.

From the equations of DOM, it can be expected a steady shrink for both algorithms. In particular, AεSεH will be more aggressive since at each generation it will retain less than 18% of the total amount of DOM solutions according to the coefficient of DOM_t. On the opposite, MOEA/D will retain around 65%. NDNP equations show that NDNP solutions will grow in the beginning in both algorithms, while DOM numbers are still able to substitute the loss in this compartment, with AεSεH clearly growing more than MOEA/D. As for PO equations, notice that both algorithms lose some solutions depending on the size of DOM. However, from the DOM equation, it can be known that the number of DOM solutions should go down quickly. After the DOM effect becomes minimal, since AεSεH retains 99% of the already found PO solutions, their increase will be ruled by NDNP solutions in this algorithm. On the other hand, since MOEA/D retains only 91% of the already found PO solutions, their number will increase in this algorithm only if the number of solutions coming from NDNP can compensate those that are not retained from PO. Considering the previous analysis, a possible conclusion is that the faster and larger change from DOM to NDNP in AεSεH allows the algorithm to keep in the population more PO solutions than MOEA/D. The equations of the model aggregated in this way can be used to study these dynamics putting in perspective the effects on the other related features.

For the next example, the model 3C-2 is considered, which includes the features Pareto Optimal Absolutely New (POA), Pareto Optimal Not New (PON) and Non Pareto Optimal (NPO) solutions.

Again, the analysis starts with the 3 objectives (M3) case and Population 50 (P50). The equations are as follows:

M3P50 3C-2

$$\begin{aligned}\text{POA}^A_{t+1} &= 0.7908\text{POA}^A_t + 0.004\text{PON}^A_t + 0.0157\text{NPO}^A_t\\ \text{POA}^M_{t+1} &= 0.8437\text{POA}^M_t - 0.033\text{PON}^M_t + 0.0141\text{NPO}^M_t\end{aligned}$$

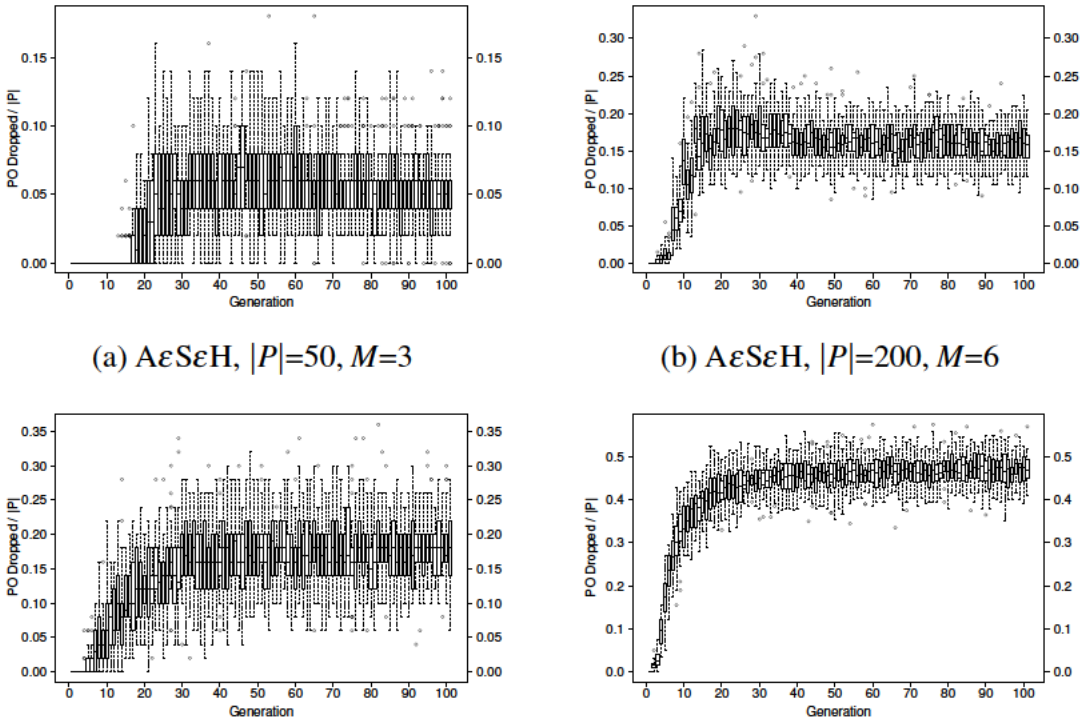
$$\begin{aligned}\text{PON}^A_{t+1} &= 0.9786\text{PON}^A_t + 1.2031\text{POA}^A_t - 0.0188\text{NPO}^A_t\\ \text{PON}^M_{t+1} &= 0.9745\text{PON}^M_t + 0.1039\text{POA}^M_t + 0.0074\text{NPO}^M_t\end{aligned}$$

$$\begin{aligned}\text{NPO}^A_{t+1} &= 1.0031\text{NPO}^A_t - 0.9939\text{POA}^A_t + 0.0174\text{PON}^A_t\\ \text{NPO}^M_{t+1} &= 0.9785\text{NPO}^M_t + 0.0524\text{POA}^M_t + 0.0585\text{PON}^M_t\end{aligned}$$

Similar to the previous case, on 3 objectives is unlikely to find PO solutions as confirmed by the data, so at first, there are only NPO solutions. In the following, the analysis will focus on PON and POA coefficients and their correlation to dropped solutions. These solutions are calculated as the Pareto Optimal solutions found in $t-1$ that are not present in t. In Figure 5.5 how this quantity changes throughout the generations can be seen for AεSεH and MOEA/D.

From the plots, it follows that MOEA/D drops PO solutions in a larger number than AεSεH. Although none of these equations model this feature, PON equations give an insight into this situation. PON feature conveys the number of current PO solutions that the algorithm already found, be it that they are retained in the current population or just found again. In the PON equations, note that the coefficient of PON_t solutions shows that less than 98% PON solutions are retained for the next generation in both algorithms, which suggests that PO solutions are being dropped. Furthermore, the coefficient affecting POA_t indicates the number of POA solutions that will become PON at the next generation. Note that in MOEA/D this number is only 10% whereas in AεSεH is 120%. This clearly indicates that MOEA/D is dropping more PO solutions than AεSεH.

Finally, the 6 objectives (M6) population 200 (P200) model is analyzed. The simplified equations are as follows:



(a) AεSεH, $|P|$=50, M=3

(b) AεSεH, $|P|$=200, M=6

(c) MOEA/D, $|P|$=50, M=3

(d) MOEA/D, $|P|$=200, M=6

Figure 5.5: Pareto optimal solutions dropped from the population. Population sizes 50 and 200 for 3 and 6 objectives, respectively. Algorithms AεSεH and MOEA/D.

M6P200 3C-2

$$\begin{aligned}
\mathrm{POA}^A_{t+1} &= 0.5967\mathrm{POA}^A_t + 0.0489\mathrm{PON}^A_t + 0.0768\mathrm{NPO}^A_t \\
\mathrm{POA}^M_{t+1} &= 0.9621\mathrm{POA}^M_t - 0.0288\mathrm{PON}^M_t + 0.0319\mathrm{NPO}^M_t
\end{aligned}$$

$$\begin{aligned}
\mathrm{PON}^A_{t+1} &= 0.9287\mathrm{PON}^A_t + 0.5305\mathrm{POA}^A_t - 0.0061\mathrm{NPO}^A_t \\
\mathrm{PON}^M_{t+1} &= 0.9684\mathrm{PON}^M_t + 0.0001\mathrm{POA}^M_t + 0.0292\mathrm{NPO}^M_t
\end{aligned}$$

$$\begin{aligned}
\mathrm{NPO}^A_{t+1} &= 0.9293\mathrm{NPO}^A_t - 0.1272\mathrm{POA}^A_t + 0.0224\mathrm{PON}^A_t \\
\mathrm{NPO}^M_{t+1} &= 0.9389\mathrm{NPO}^M_t + 0.0378\mathrm{POA}^M_t + 0.0604\mathrm{PON}^M_t
\end{aligned}$$

As mentioned above, in 6 objectives small landscapes even a random sampling can find some PO solutions, so it is expected to find some of them in the initial population. At $t = 0$, according to the data, less than 1% are PO solutions. For the model, these solutions will be absolutely new since it is the initial population. PON is 0 since there are no previous generations, and more than 99% are NPO.

Similar to the previous 3 objectives case, notice that AεSεH seems to retain more than 53% of POA solutions as PON solutions, as indicated by the coefficient value of the POA_t component in the PON equation, while for MOEA/D this value is very small, less than 0.01%. Thus MOEA/D is dropping substantially more solutions than AεSεH. However, note also in POA equations that POA_t coefficient is 0.59 for AεSεH and 0.96 for MOEA/D, which indicates that MOEA/D finds many more Pareto absolutely new solutions than AεSεH. Looking at Figure 5.5, the above analysis with the equations confirms that, indeed, MOEA/D drops more solutions than AeSeH. Although these model compartments do not directly reference dropped PO solutions, it is possible to identify this trend from the changes of newly found PO and previously found PO.

Finally, it is also worth noting that in M3P50 3C-2 and M6P200 3C-2 that POA solutions are found mostly from previous POA solutions rather than from PON or NPO solutions.

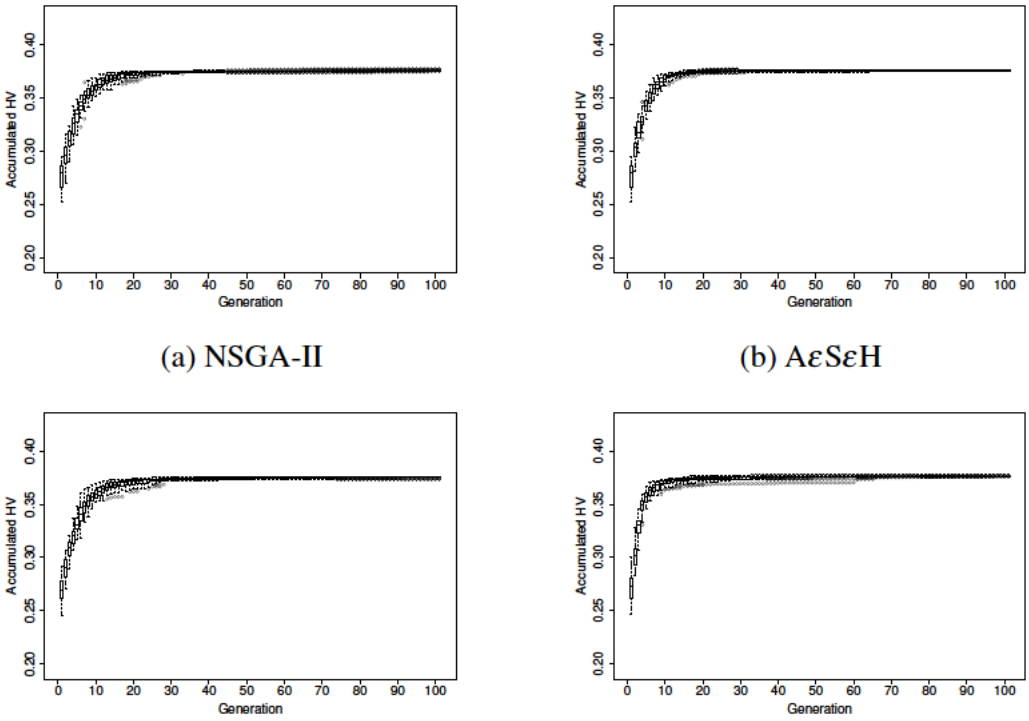
In general, note that this type of analysis can be done only by observing the features changes. Nevertheless, the benefit of using the models is that the parameters capture information on the relationship between these features, which allows easier analysis of the dynamics.

5.3.3 Features vs. Performance Metrics

It was mentioned previously that the accumulated number of PO solutions can be obtained in the 3C-2 POA-PON-DOM model, and it could be used as a performance measure. This section will show how useful is this value as a metric and correlate to a more well known performance indicator, the hypervolume [58]. This metric gives the multi-dimensional volume of the objective space that is dominated by a non dominated set and enclosed by a reference point, which in this case was set to all zeros since the objectives range is [0,1].

For the analysis, the accumulated set of non dominated solutions until each generation t, for each generation and run is computed, followed by the hypervolume for each of these sets. Boxplots for the 3-objective and population 200 case from Dataset 1 are reported in Figure 5.6.

By comparing the accumulated number of PO solutions in Figure 5.4 with the hypervolume of the accumulated non dominated set in Figure 5.6, it can be noted that the trend and growth rate in both are similar for each algorithm. From generation 1 to 30, there is an exponential growth of the hypervolume and number of PO solutions found. After that, these values stagnate. Although it cannot be directly said how much the value of the hypervolume will go up with each newly-found PO solution since this model only focuses on their numbers, a link between these two measures is clear, which seems to scale together at least for some generations.



(a) NSGA-II (b) AεSεH

(c) IBEA$_{HV}$ (d) MOEA/D

Figure 5.6: Hypervolume of the accumulated joint non dominated sets.

The correlation between the hypervolume-values for each configuration and the number of PO solutions found until a given generation is also calculated using Spearman's correlation coefficient. For the example shown, the coefficients are 0.897, 0.786, 0.844, 0.795 with $p < 2.2 \times 10^{-16}$ for NSGA-II, AεSεH, IBEA$_{HV}$ and MOEA/D, respectively, which supports the initial assumption that both values are correlated.

These results illustrate the usefulness of the model to estimate the accumulated number of PO solutions in order to estimate the overall hypervolume convergence profile, and whether it will keep improving or will start to slow down.

Knowing that there is a positive correlation between the accumulated number of PO solutions and the hypervolume-value, the next step is to verify if the algorithm comparison using both metrics yield similar results. For this purpose, in Table 5.3 results of both metrics are reported, for the four representative algorithms configured with different population sizes on problems with 3, 4 5 objectives.

The table shows the average values of Accumulated PO solutions found by the algorithm, the estimation of this value by the proposed compartmental model, and the hypervolume of the joint set of non dominated solutions found over all generations. The last three columns, also include a ranking between algorithms from 1 (Best) to 4 (Worst) according to the measured and predicted value of accumulated PO solutions and hypervolume, respectively.

Table 5.3: Algorithm comparison by Accumulated PO Solutions (model feature) and Hypervolume (performance metric)

			Accumulated PO Solutions			Rank		
Algorithm	**Obj.**	**Pop. Size**	**Measured**	**Predicted**	**HV**	**Measured**	**Model**	**HV**
NSGA-II	3	50	105.933	108.805	0.373968	**3**	**3**	**3**
AεSεH			121.967	128.088	0.374248	**1**	**1**	**1**
IBEA$_{HV}$			90.367	92.367	0.372361	**4**	**4**	**4**
MOEA/D			114.000	114.908	0.374201	2	2	1
NSGA-II		100	137.967	137.267	0.375304	2	2	3
AεSεH			141.700	140.943	0.375287	**1**	**1**	**1**
IBEA$_{HV}$			123.700	119.458	0.373812	**4**	**4**	**4**
MOEA/D			135.500	136.994	0.375422	3	3	1
NSGA-II		200	148.033	142.500	0.375640	1	3	3
AεSεH			148.667	150.651	0.375904	**1**	**1**	**1**
IBEA$_{HV}$			136.567	134.788	0.375140	**4**	**4**	**4**
MOEA/D			144.667	145.473	0.375757	3	2	1
NSGA-II	4	50	166.800	168.756	0.194207	4	3	3
AεSεH			346.767	353.793	0.196897	**2**	**2**	**2**
IBEA$_{HV}$			195.500	159.648	0.191004	3	4	4
MOEA/D			436.100	436.990	0.197264	**1**	**1**	**1**
NSGA-II		100	343.233	347.299	0.197154	**3**	**3**	**3**
AεSεH			581.767	582.865	0.198445	2	2	1
IBEA$_{HV}$			302.400	309.600	0.193841	**4**	**4**	**4**
MOEA/D			656.267	656.931	0.198507	**1**	**1**	**1**
NSGA-II		200	608.867	615.447	0.198481	**3**	**3**	**3**
AεSεH			863.767	877.741	0.199031	**1**	**1**	**1**
IBEA$_{HV}$			468.100	477.692	0.197055	**4**	**4**	**4**
MOEA/D			857.500	859.080	0.199034	1	2	1
NSGA-II	5	50	157.800	142.474	0.146900	4	4	3
AεSεH			526.100	533.512	0.153434	**2**	**2**	**2**
IBEA$_{HV}$			190.167	194.093	0.139987	3	3	4
MOEA/D			904.100	735.058	0.154429	**1**	**1**	**1**
NSGA-II		100	341.600	343.592	0.151905	**3**	**3**	**3**
AεSεH			986.433	959.355	0.156632	**2**	**2**	**2**
IBEA$_{HV}$			329.700	336.261	0.145065	3	4	4
MOEA/D			1479.133	1488.790	0.157620	**1**	**1**	**1**
NSGA-II		200	718.500	720.574	0.155398	**3**	**3**	**3**
AεSεH			1631.467	1642.240	0.158275	**2**	**2**	**2**
IBEA$_{HV}$			602.300	614.040	0.150032	**4**	**4**	**4**
MOEA/D			2219.167	2216.400	0.159051	**1**	**1**	**1**

When all rankings agree for a given algorithm this is marked in bold. The hypervolume values for a given number of objectives and population size are checked for a statistically significant difference using a Mann–Whitney test with a 95% confidence interval. If the hypervolume values are not statistically different the corresponding algorithms will be given the same ranking. The variance around the means of the predicted value of accumulated PO solutions is very low and therefore a statistical test is not required to verify its ranking.

Note from this table that there is a very good agreement between the rankings given by the measured and estimated number of PO solutions as well as between the estimated number of solutions and the hypervolume of all non dominated solutions found by the algorithm. Thus, the partial orderings given by this feature of the model are useful, giving a good idea of the relative performance of the algorithms.

5.4 Parameter Interpolation

This last section will cover how to exploit the models' parameters through interpolation to explore other configurations. As mentioned in the model description, each set of model parameters is tied to a specif algorithm, its configuration, and the problem where it was run. If multiple parameters set for the same algorithm and problem but with different configurations, interpolation of the model's parameters can be used to extract configurations in between the available ones. The results presented in this section use the Dataset 2 described in Chapter 4.

5.4.1 Cubic Spline Interpolation

Let $D = \{(x_i, y_i)\}$ be a set of n data points $i = 1, \ldots, n$, and $y_i = f(x_i)$. Then the following expression can be defined.

$$\sum_{i=1}^{n} \{Y_i - \hat{f}(x_i)\}^2 + \lambda \int \hat{f}''(x)^2 dx \tag{5.4}$$

A function $\hat{f}$ that minimizes Eq.(5.4) is called a smoothing spline. Here λ is a parameter that controls the smoothness, i.e trade-off between closeness to the data and roughness of estimation.

The solution for this expression will be piece-wise cubic polynomials as defined by Eq.(5.5)

$$S_i(x) = \frac{z_i(x - x_{i-1})^3}{6h_i} + \frac{z_{i-1}(x_i - x)^3}{6h_i} + \left[\frac{f(x_i)}{h_i} - \frac{z_i h_i}{6}\right] (x - x_{i-1}) + \left[\frac{f(x_{i-1})}{h_i} - \frac{z_{i-1} h_i}{6}\right] (x_i - x) \tag{5.5}$$

where x_{i-1} and $x_i \in D$ are the boundaries between polynomials called knots, $z_i = f''(x_i)$ are the second derivatives at knot ith, $h_i = x_i - x_{i-1}$ and $f(x_i)$ are the values of the function at knot ith.

For a more detailed explanation, please refer to[80]. Here the R implementation of smoothing splines with the default parameters is used for all the interpolations.

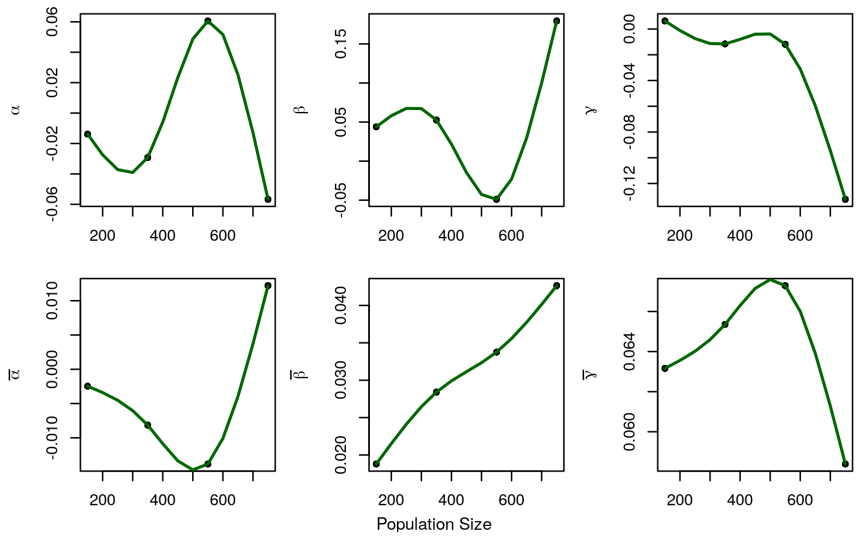


Figure 5.7: Splines for each model parameter for 5 objective MNK-Landscapes test problem.

5.4.2 Parameter Interpolation

The models are composed of six parameters $(\alpha, \beta, \gamma, \bar{\alpha}, \bar{\beta}, \bar{\gamma})$, which values differ depending on the population size, algorithm and problem. In order to interpolate a model, a separete spline for each parameter is needed, taking as data points the pair of population size and parameter value. Then a new model is obtained by using these splines to find each of the parameters for the unknown population sizes, between the ones used to construct the spline.

The first step is to select the population sizes to be used as knots and run the algorithm to generate the data necessary to fit the models. Due to the stochastic nature of the fitting process used, at least 30 fittings are done for each size, preserving the ones with a good R^2. In previous fittings, the R^2 was computed for each feature, and this was the criteria to continue another round of optimization to improve the fitting, while here the main focus was obtaining a good estimation of the accumulated number of PO solutions. Therefore the R^2 computation was done on this value and not the individual features. The chosen population sizes to be used as knots were $\{150, 350, 550, 750\}$, with an R^2 coefficient of 0.858, 0.988, 0.985 and 0.974, respectively.

The next step is to find the function that generates the spline, which results in curves as shown in Figure 5.7, where the selected population sizes appear in black. Finally, splines are used to get the parameters to build new models. Using intervals of 50 between knots, three new population sizes are explored between each pair of knots.

5.4.3 Using interpolated models to explore population sizes

Using the methodology described above for obtaining the interpolated models, in this section, the focus is answering the question "Do interpolated models allow the exploration of population sizes in-between the ones for which data is available?".

To do that, an analysis is performed between the estimated average number of Accumulated PO solutions by a fitted model versus the models obtained through interpolation. A comparison is also made with the measured values by running the algorithm. The goal is to see if the obtained interpolated models help to discover promising configurations in the intervals between the sampled configurations.

Figure 5.8 shows on the left for sample population sizes, the average number of Acc. PO solutions at the end of 10000 FE for 30 runs of the algorithm. These sample population sizes will act later as knots for the spline. On the center, a similar plot is included, with values produced by the fitted models in red, while the ones from interpolated models are in black. Finally on the right side is plotted again the values obtained running the algorithm with the sampled population sizes, and in blue the values obtained running the algorithm with the population sizes not previously sampled, to check against the trends found by the models. In all plots, the error bars show the 95% confidence interval for each mean.

Looking at Figure 5.8 left and center plots, a visual comparison can be done between the fitted model estimation and the measured values for the population sizes used as knots. Notice that for all sizes except 150 the fitted model gets very close to the measured average, in the particular case of 750 is even a little higher. The R^2 for the model used in 150 is under 0.95, which explains the underestimation produced here. Nevertheless, as can be seen in the following results, even with such models as knots, the trend is still correctly replicated.

Now shifting the focus to the center and right figures, looking at the estimations of the models obtained through interpolation. It can be seen that the influence of the first knot, 150, pulls down the estimations of the interpolated models between $[150, 350]$. Nevertheless, in this section, the growing trend is still correctly maintained when compared to the plot for all sizes. Moving to the next interval of $[350, 550]$, the models' estimations get closer to the measured results and also follow the trend detecting that 400 and 500 obtain better values than 450. Finally, in the last interval of $[550, 750]$, the trend is not correctly replicated, as 700 places a bit higher than 650, which according to the right side of the figure should be below. Since the fitted model of 750 actually overestimates a little in this part, it seems to pull the interval making 700 seem a better option. However, the relative position of the knots points is correct, as two of them place higher than 750 as it should be, and neither of the group is higher than the knot on left, 550.

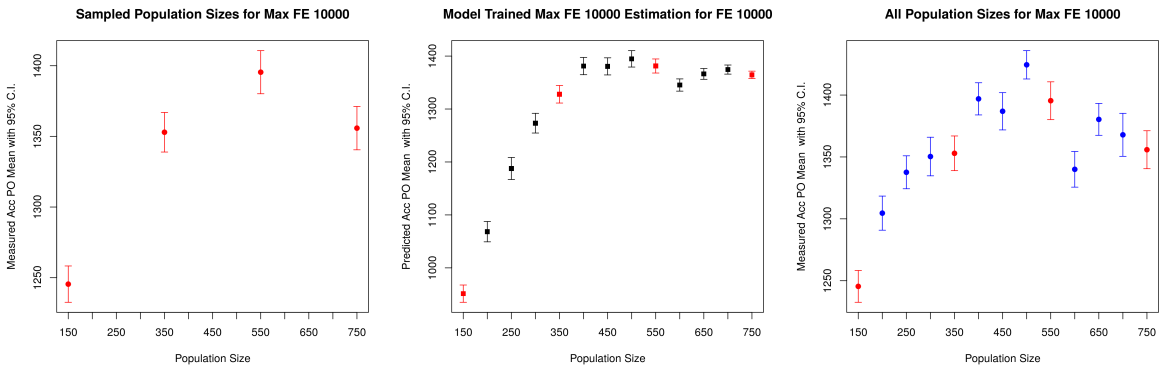


Figure 5.8: [Budget: 10000 FE] Average Acc. PO over population size. Left: Sampled population sizes used as knots for the spline. Center: Model estimation for the knots and interpolated population sizes. Right: Measured values for all population sizes.

Summarizing the above, if only the sampled population and interpolated population model's estimation are available and the user decides to trust this obtained values, then it will be guided to choosing a particular population size in the interval, which as the measured data suggest, gives better use of the available budget than the ones at the sampled configurations. On the other hand, if the user decides to trust partially the model, then it would not choose a population size, but an interval. With this information, it can later explore with experiments the population sizes that the model points to as more interesting. In either case, it still guides the user to the region worth exploring in more detail. From this, and to answer the question at the beginning, it is possible to use the interpolated models, if they follow the trend. In order to ensure this, the sampled configurations must be of good quality, so any error introduced by the interpolation is also minimized.

5.4.4 Re-using models to explore bigger budgets

Another interesting question is, "if there is an extra allowance in the budget, is it possible to reuse all the models obtained for a given budget, to make predictions about population size with larger budgets?". In other words, how far ahead the models can still correctly detect the performance trends of the algorithm.

The methodology for this part is to take the models found for a budget of 10000 FE and make estimations with the same population sizes for budgets of 12000 and 15000 that represent an exploration of the 1.2 and 1.5% of the search space.

For 12000 FE in Figure 5.9, it is plotted on the left the estimation of the model and on the right, the results of running again the algorithms, but now with the new FE limit. Here the

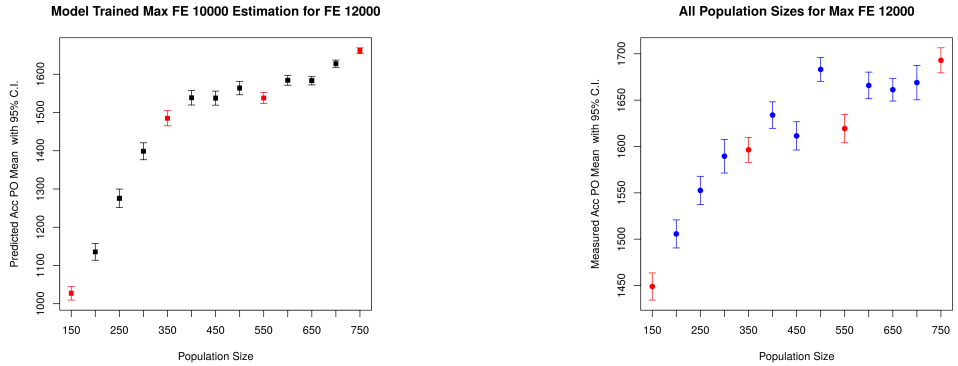


Figure 5.9: [Budget: 12000 FE] Average Acc. PO over population size. Left: Model estimation for the knots and interpolated population sizes. Right: Measured values for all population sizes.

models are forced to estimate 20% more FE than the ones they saw during the fitting, which represents for the lowest population size 14 more generations, and for the biggest population size only 3 more.

The analysis begins with only the fitted models and their position with respect to each other. For 150,350,550 and 750, their relative positions still hold and show an upward trend. Going into the intervals from $[150, 350]$ the interpolated models still are able to follow correctly this trend. For the interval of $[350, 550]$, they are able to replicate even the relative position between themselves, being 400 and 500 a little bit better than 450 and all points in the interval higher than the extremes. As for the last interval of $[550, 750]$, similarly, the relative position inside still holds being 600 and 700 better than 650 and all of them better than the left extreme 550 but not with respect to 750. In particular, on 500, it failed to pick up the trend since looking at the corresponding measured values, it should be better than the last interval $[550, 750]$.

In this situation, from the point of view of the user, all the available data is model estimations. If the user only looks at the fitted models' estimations, the chosen population would be 750. Which, if the populations in the intervals are ignored, is the best option, and the measured data agrees. If the information of the interpolated models is also considered, then the user would ignore 500 as an option, since the model failed to pick up its trend, and instead choose 700, which is the second best option according to the measured values.

Going up to 15000 FE, now the models are forced to estimate 50% more FE than what they have been trained for, which represents 33 more generations for 150 and 6 more for 750. The plots are shown in Figure 5.10.

Starting from the fitted models, it keeps the general trend seen with 12000 FE, where the largest population tends to be better, which is captured by the models. As for the intervals,

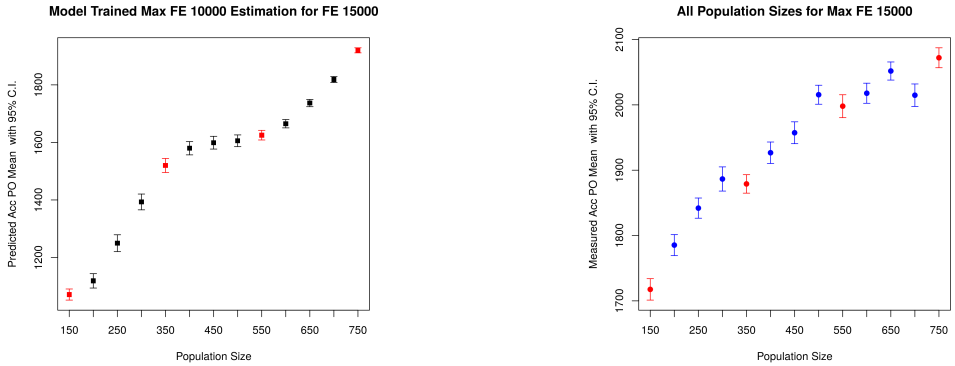


Figure 5.10: [Budget: 15000 FE] Average Acc. PO over population size. Left: Model estimation for the knots and interpolated population sizes. Right: Measured values for all population sizes.

the first two are followed properly if only looked at the relative positions between them, while failing to show when they are better than the extremes. The third interval shows a more noticeable difference between the model and measured values. In the interval, $[550, 750]$ the interpolated models place them in an ascendant order, while from the plot it can be seen that 650 can obtain a better value than 600 and 700, and also 600 is better than 700. This trend is not picked up.

Similar to the previous case, the fitted models still hold and describe the general trend, even though they are asked to estimate farther ahead than what they have seen. The interpolated models, do not perform that well in this case, not being able to describe situations when in the measured values their population obtains a better value than the ones used as knots. Considering that all of them are based on a lower FE budget, the models are still a valuable guide to detect which intervals could be investigated.

To summarize, the fitted models seem to be more reliable than the interpolated models when forced to estimate beyond their training. However, an important consideration to be made is using budgets that allow the population sizes under consideration enough time to converge and properly display their dynamics. In the examples described here, the budget 10000 FE only allowed 13 generations for the biggest population, which made it harder for the model to properly capture its dynamics. This is reflected also in the analysis, as the first two population sizes intervals did behave better, even when their models were used on larger budgets.

The use of the models for algorithm configuration is a very interesting and promising line of research, since it is very cheap to produce estimates once the fitted models are found, and the interpolation does not add much overhead to the whole operation. In general, the biggest cost of this method is generating or obtaining the data. A careful choice of the

sampling points will surely allow taking the most advantage of the models to explore and find interesting configurations.

Chapter 6

DCMs based on NDS features and Hypervolume estimation model

This chapter covers the use of dynamic compartmental models with a new set of features to handle larger landscapes, and the use of the performance estimation model to predict the hypervolume growth. The fitting process is changed and simplified to produce better fittings in an easier way.

6.1 Comparison between NDS and POS based DCMs

Since the aim of the DCMs based on the NDS feature set is to be an alternative to the POS feature set based ones, in this initial section the goal is to assess the accuracy of their estimation in terms of features as well as the performance metric they give, looking also at their ability to rank algorithms.

Given that the POS feature set can be used only when Pareto optimal solutions can be enumerated, Dataset 3 is used, which contains instances where N = 20, making the generation of the POS obtainable by enumeration.

The first part of the analysis focuses on the estimation accuracy of the features. Figure 6.1 and Figure 6.2 show in red the features estimations by DCMs with the POS and the NDS feature sets, respectively, for population size 200 and the problem sub-class with 4 objectives. Similarly, Table 6.1 and Table 6.2 show the R^2 values of the fitting. Results are presented for population size 200 on the 3, 4, and 5 objective problem sub-classes. As can be seen in Figure 6.1 and Figure 6.2, visually there is a good fit for all features. Note that the estimation made by DCM either with POS or NDS features follows the mean of data. However, NDS features seem to better capture the variance than POS features. From Table 6.1 and Table 6.2,

note that DCM with both feature sets present good R^2 values for the respective features. However, note that POA (Pareto Absolutely New) R^2 values are significantly smaller than NDNew (Non dominated New), the corresponding feature in the new feature set. The reason for this is that the number of Pareto optimal solutions found by the algorithm is significantly lower compared to the number of non dominated solutions. This makes it more difficult for the DCM model to capture the POA signal, particularly in the early stages of the search. Note also that there is a lower R^2 score in the instances with 3 objectives, which is explained by the high variance observed in these instances. Overall, the DCM with the new feature set provides a higher estimation accuracy of the NDNew feature related to performance estimation.

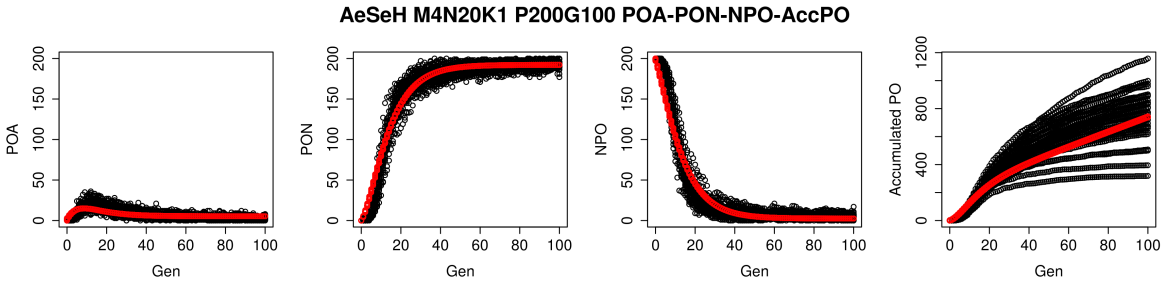


Figure 6.1: Measured (black) and estimated (red) values of the POS feature set POA-PON-NPO on small enumerable instances with N=20 variables, K=1 variable interactions and M=4 objectives. Population size 200.

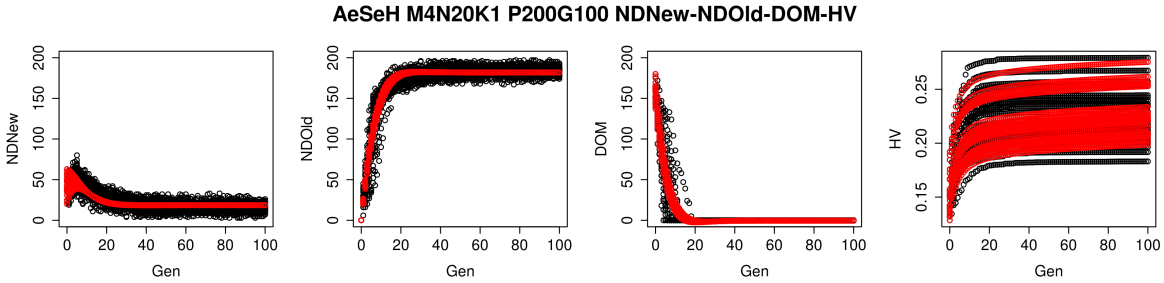


Figure 6.2: Measured (black) and estimated (red) values of the NDS feature set NDNew-NDOld-DOM features on small enumerable instances with N=20 variables, K=1 variable interactions and M=4 objectives. Population size 200.

Following the previous analysis, now the focus moves to their ability to do performance estimation. Figure 6.3 and Figure 6.4 show the measured and estimated values of the accumulated number of PO solutions obtained from the POS features and the hypervolume obtained from the NDS features, respectively, for all problem sub-classes and population sizes. The performance estimation obtained from the POA feature, i.e. the accumulated

Table 6.1: R^2 values for the POS feature set POA-PON-NPO by DCM and the estimated number of accumulated Pareto optimal solutions. Small instances, population size 200.

Instance	POA	PON	NPO	Acc. PO
M3N20K1	0.480	0.755	0.736	0.457
M4N20K1	0.472	0.961	0.951	0.731
M5N20K1	0.376	0.950	0.943	0.801

Table 6.2: R^2 values for the NDS feature set NDNew-NDOld-DOM by DCM and the hypervolume by the HV model. Small instances, population size 200.

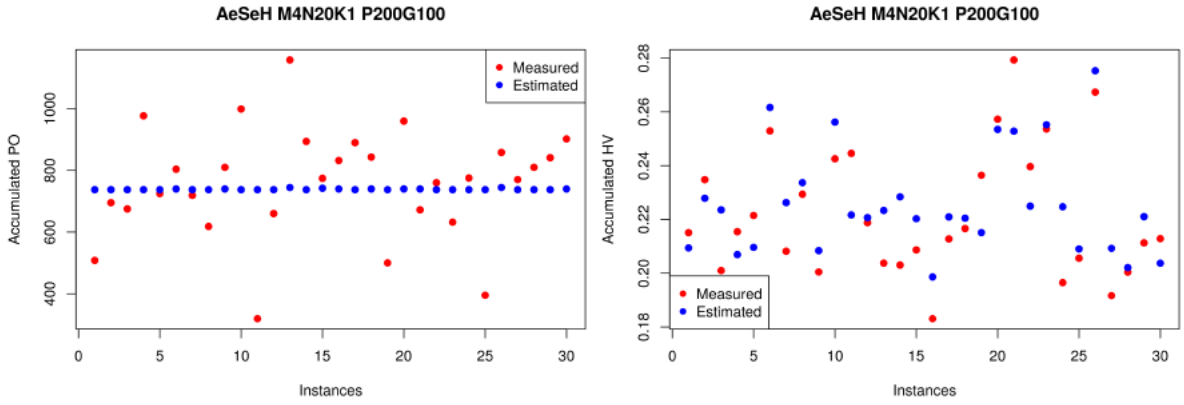
Instance	NDNew	NDOld	DOM	HV
M3N20K1	0.626	0.649	0.548	0.373
M4N20K1	0.657	0.930	0.894	0.695
M5N20K1	0.640	0.866	0.921	0.736

Figure 6.3: Accumulated number of Pareto optimal solutions (M)easured on the data and (E)stimated by DCM using the POS feature set. Small enumerable landscapes with N=20, K=1 and M=3, 4 and 5. Population size 50, 100 and 200.

number of PO solutions, overall approaches the mean of the measured values on population sizes 100 and 200 but underestimates on a population size of 50. Note that in all cases it does not capture the variance. Nonetheless, it ranks the different configurations in each problem sub-class correctly. This particular trait allowed this feature set as seen Chapter 5 to be used in configuration selection.

The performance estimation obtained from the NDNew feature, i.e. hypervolume, is able to capture the variance as well as achieve a mean closer to the one found in the measured data, which also allows this feature set to rank algorithm's configurations with confidence. Not only that, taking a closer look at the estimated vs measured performance for each feature

Figure 6.4: Final hypervolume (M)easured on the data and (E)stimated by the HV Model using the NDS feature set. Small enumerable landscapes with N=20, K=1 and M=3, 4 and 5. Population size 50, 100 and 200.



(a) Pareto optimal solutions

(b) Hypervolume

Figure 6.5: Measured and estimated performance with (a) POS features set and (b) NDS feature set computed at the final generation for all 30 instances of problem sub-class N=20, K=1, and M=4. Population size 200.

set in 6.5, it can be seen that the performance estimation with the new feature set is more accurate, as it is able to follow closer the variance between instances in a given sub-class.

6.2 Improved Model Fitting

While in Chapter 5 a fitting process was already introduced, for the model with the NDS set the approach taken is different based on experimental results. Two important considerations are made for pre-processing the data obtained from the algorithm runs to create the dataset used in the fitting process.

The first consideration is whether to join the data of the features from different landscapes (instances), in which case there will be several entries for the same t, or to take the average of the feature values on all landscapes for each generation. After some initial trials, it was found that the performance estimation model gives better results by using the joined dataset, while the DCM benefits from using the dataset formed by the average of the feature values.

The second consideration is whether to use the measured or the estimated values of the features for the independent variables or predictors. Note that, in the performance estimation model, to compute HV_{t+1} the predictors are NDNew_{t+1} and HV_t. Thus, the performance estimation model can take as input either the NDNew_{t+1} measured on the population returned by the algorithm, or the estimation given by the DCM model. Similarly, it can take as input the HV_t measured for $t \geq 0$ or the value estimated by the model for $t > 0$ (HV_0 for $t = 0$ is always measured in the initial population). In the case of DCM, to compute the value of a feature at time $t + 1$, the predictors are the feature itself and the other features at the previous generation, as shown in Eq. (3.2). It was verified that the fitting of the hypervolume estimation model works better when using estimates rather than measured values for both of its inputs. That is, the dataset for fitting is formed by the NDNew_{t+1} and HV_t values estimated at each generation from DMC and the performance estimation model, respectively. In the case of DCMs, better results were observed when the datasets for fitting are formed by the measured values of the features, particularly for prediction on unseen landscapes.

Once the data is prepared according to which results in a better fitting for the model, instead of the previously introduced process, here the Levenberg-Marquardt Non-Linear Least Squares [81] to fit the data to each equation from the original DCM system of eq. (3.2) as it has been shown to produce better results for both models.

The method expects that a dataset formatted as $(y_t, x_t^1, x_t^2, \ldots)$ for $t = 0$ to $t = t_{max}$ is provided, where y_t denotes the dependent variable and x_t^1, x_t^2 are the independent ones. The data has the following format (HV_t, HV_{t-1}, NDNew_t) for the PEM fitting, and 3 datasets for the three compartment DCM, fitting one per feature where only the first column changes depending on the feature's equation we are solving, e.g for NDNew the data will be (NDNew_t, NDNew_{t-1}, NDOld_{t-1}, DOM_{t-1}).

Note that the models take the previous estimation as the base for the next one, y_t depends on y_{t-1} and so on. In the case of DCM, this dependency on the previous value also encompasses the values estimated by the other equations in the system. Taking into account this fact resulted in a better fitting for the PEM. Its input data was changed so it has the HV measured at the initial generation while the successive HV values are estimates produced by the model. This way, the chosen parameters have a more powerful effect, forcing the fitting process towards good parameters. A similar strategy was attempted to fit the DCMs, while

good results could be seen on the data used for the fitting, a reduction in quality was found when estimating unseen landscapes.

Cross-validation was also introduced, so the obtained parameters are not a product of over-fitting to our generated data and would generalize better in the presence of new and unseen data. k-fold cross-validation was chosen and applied during the DCMs and PEM fitting process. In k-fold cross-validation, the dataset is split into k subsets of equal size, each subset is used only once as a test set and $k-1$ times as part of the training set. For the data from Dataset 4, 30 runs of the algorithm are available for this task, corresponding each one to a different landscape. To ensure an 80/20 split between training and testing data, k is set to 5, a common recommendation for this method as suggested in [82]. So each fold is composed of one subset of 6 landscapes worth of test data, and the remaining 4 subsets, provide 24 landscapes worth of training data. The score obtained on each set is measured by previously introduced goodness of fit or R^2, a value between 0 and 1 that indicates how much of the variance present in the data is explained by the model.

Under cross-validation, the fitting process per configuration is done only with the data from the training set, and the resulting parameters estimation ability is measured on the test set, repeated for each fold. Table 6.3 and Table 6.4 report the average R^2 of the 5 scores obtained for the training and test datasets for all population sizes and considering 800000 FE. Since at the end of the process 5 sets of parameters are found, the average is taken and keep as the parameters found for that configuration and number of FE.

6.3 Model Quality on Seen Landscapes

Table 6.3: R^2 values obtaining during the model training. Obtained doing k-fold cross-validation with $k = 5$.

	Training R^2			
Population	NDNew	NDOld	DOM	HV
3000	0.827232	0.854135	0.834016	0.838772
5500	0.755633	0.852176	0.825499	0.855914
8000	0.669367	0.921005	0.908973	0.833325
10500	0.559348	0.915412	0.896397	0.829055

Table 6.3 shows the R^2 values for training and Table 6.4 for the testing set used in the cross-validation. While for the features NDOld and DOM the obtained R^2 is relatively high, the NDNew feature ends up always with lower scores. However, note that even in this

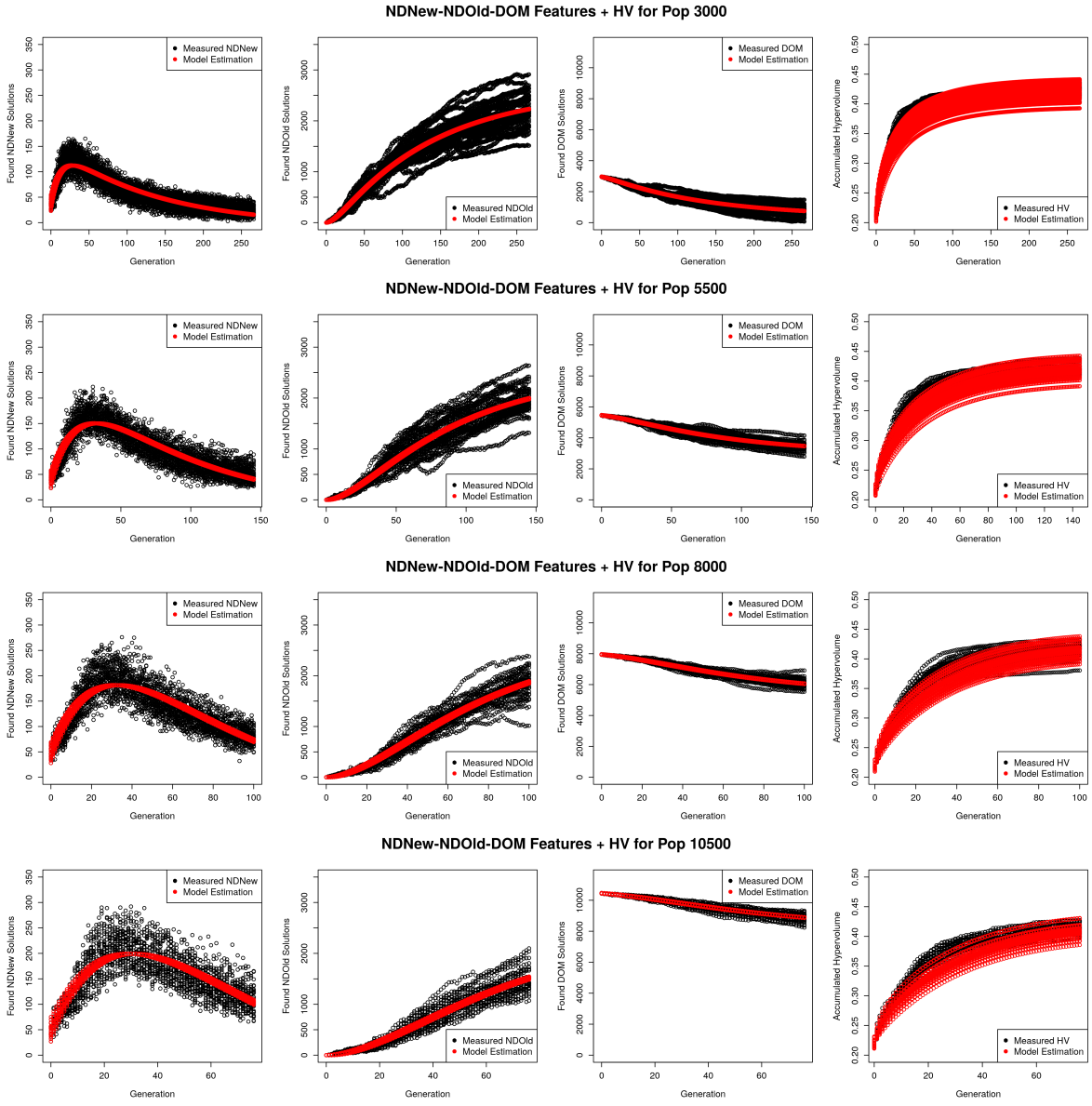


Figure 6.6: [Seen Instances] Comparison between Measured (black) and Estimated (red) features for different population sizes.

Table 6.4: R^2 values obtaining during the model testing. Obtained doing k-fold cross-validation with $k = 5$.

	Testing R^2			
Population	NDNew	NDOld	DOM	HV
3000	0.833518	0.842590	0.820838	0.827245
5500	0.751173	0.842662	0.813348	0.854298
8000	0.670151	0.918018	0.906135	0.829892
10500	0.572069	0.910156	0.888895	0.823724

situation the NDNew values obtained can properly guide the PEM and attain R^2 values above 0.80. NDNew has a large variance (up or down) between each generation, as it conveys the number of new solutions found, which can be better appreciated in Figure 6.6.

For a more visual verification, the DCMs and PEM estimations and the corresponding four features NDNew, NDOld, DOM, and HV measured values are plotted. As can be expected from the R^2, the plot agrees on the ability of the model to follow the changes in each feature. The DCM estimated features' go through the mean of the data for all the population sizes tried with the algorithm as well as the HV estimation, which follows very closely the measured values in all the different instances of the problem.

An interesting observation can be made with respect to the R^2 values of the feature NDNew. When the population size grows, the R^2 reduces. For the other two features, the trend is different. One possible explanation could be the higher variance of the feature NDNew that tends to grow also with the population size, so an impact in the R^2 is not unexpected.

Another comes from the shape of NDNew on the larger population sizes. In population 3000, at the end of the search NDNew has a downward trend but with a soft slope and seems to be converging. Compared to population 10500, where only a downward trend is present, and the slope looks more acute. It is possible that with a few more generations, the more natural converging trend is also picked by the model, which then would be reflected in the estimations. The model expects that the features will converge at some point to a value, so better results can be expected when the data used to train them contains these trends.

6.4 Model Quality on Unseen Landscapes

During cross-validation, the testing R^2 values gave a prospect of how the models will behave when dealing with new instances. In this section, this will be verified using the remaining

Table 6.5: R^2 values for unseen landscapes.

Population	NDNew	NDOld	DOM	HV
3000	0.844574	0.892899	0.883995	0.833870
5500	0.793194	0.915092	0.907057	0.848590
8000	0.732833	0.938995	0.930181	0.810288
10500	0.712733	0.919996	0.913953	0.792546

20 landscapes data from Dataset 4. This can be seen also as simulating the following use case, where a model trained for a problem class and a given algorithm is available, so it can directly be used to check how these algorithms will perform on other instances of the same problem. Going a little further, using landscape analysis techniques the problem can be profiled and see if it is similar enough to be considered of the same class for which trained models are available.

The steps to use the model in the face of new instances are, generate a random population for each one and evaluate according to the instance. Then the number of individuals that match each feature is counted for the DCM and the hypervolume is calculated. After these steps, all the data needed to get the first estimation is ready, and from then on, the models' results become the input for the next estimations.

Looking at Table 6.5 and the results of the testing cross-validation set in Table 6.4, a comparison can be done. For the DCM features, even better scores are found for the NDNew and similar ones for the NDOld and DOM features. The PEM also performed well, with scores similar to the ones according to the cross-validation.

Going to the visual inspection with Figure 6.7 and comparing it to the previous plots, here also the DCM feature estimation goes through the mean of the data. There is a small drop towards the end in NDNew, so the estimation may go more under the mean by the end. As for the PEM results, once again the estimation is very close to the measured values, but for some landscapes, the estimation is higher than the measured values.

The results seen so far seem to point that the models are able to learn how a particular algorithm will perform on a class of problems. Even though relationships between variables are different between each landscape, AεSεH ability to solve it, in terms of a performance metric as the hypervolume and generation of non dominated solutions follows a pattern that is possible to capture with a model.

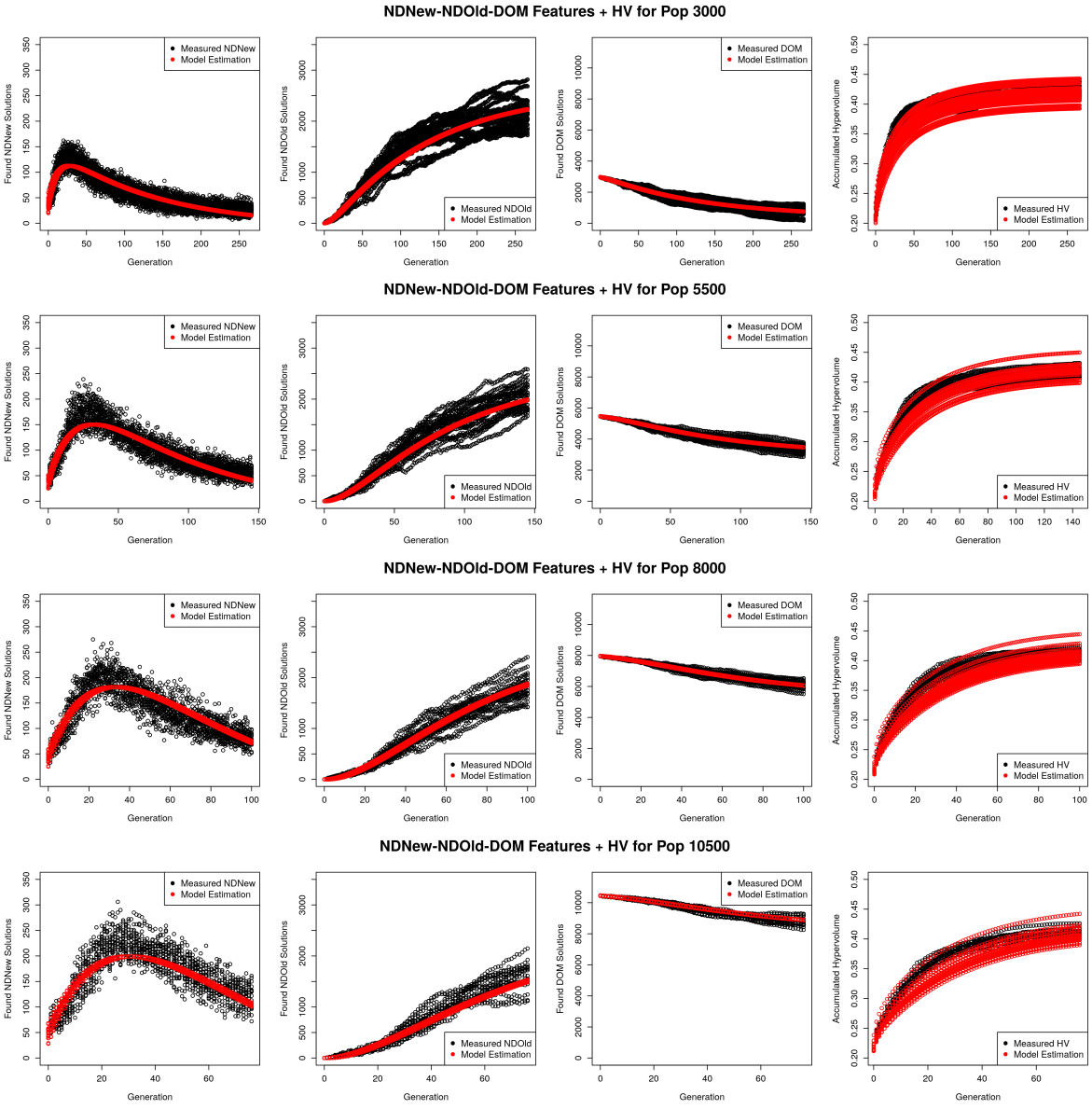


Figure 6.7: [Unseen Instances] Comparison between Measured (black) and Estimated (red) features for different population sizes.

Table 6.6: R^2 values for one unseen landscape multiple seeds.

Population	NDNew	NDOld	DOM	HV
3000	0.836272	0.892363	0.882532	0.812154
5500	0.780932	0.900579	0.888665	0.813902
8000	0.713051	0.930554	0.921455	0.780945
10500	0.680668	0.918793	0.906406	0.806310

6.5 Variability on a Single Landscape

Having seen how the model behaves with several instances, the question that naturally follows is if the model can pick up the algorithm variability. In other words, if several initial populations are generated for a single instance, what kind of estimation these models will produce. To answer this a single landscape was selected from the unseen set in Dataset 4 and 30 initial populations with different seeds were generated, i.e each population composition in terms of the features will be different and dependent on the selected seed.

Looking at Table 6.6 the R^2 scores for all features are similar to the results when all instances were considered. It seems that the model is not only capturing the inherent variance between instances, but also the variation that can occur by starting from different seeds. This happens because the output of the model can be seen as the expected value. Evolutionary algorithms may be stochastic, but we can always expect that the final population composition will be around an average value, which is exactly what the model can and is capturing.

Graphically it can be seen in the plots of Figure 6.8 the NDNew feature of the DCM and the estimation of the hypervolume produced by the PEM for all configurations. The NDNew values produced by the model, as with the previous case, still goes through the mean of the data, which is sufficiently descriptive to guide the PEM and produce close values to the measured hypervolume.

As such, it is possible to train models using several instances of a problem class and capture the expected performance on them. Also, some of the variability shown by algorithms when starting with different initial populations can be reproduced with the models. These results are very useful and promising when combined with the fact that all those estimations are simple mathematical operations that are quick to perform, and at most require spending a few resources creating initial populations for each configuration and measure the features.

Capturing features of dynamics and estimating performance for an algorithm is interesting on its own. However, it can become more useful when combined with problem features and landscape analysis to build a framework for algorithm selection and configuration. Indeed, using the unseen instance reported above, and assuming the problem sub-class has been

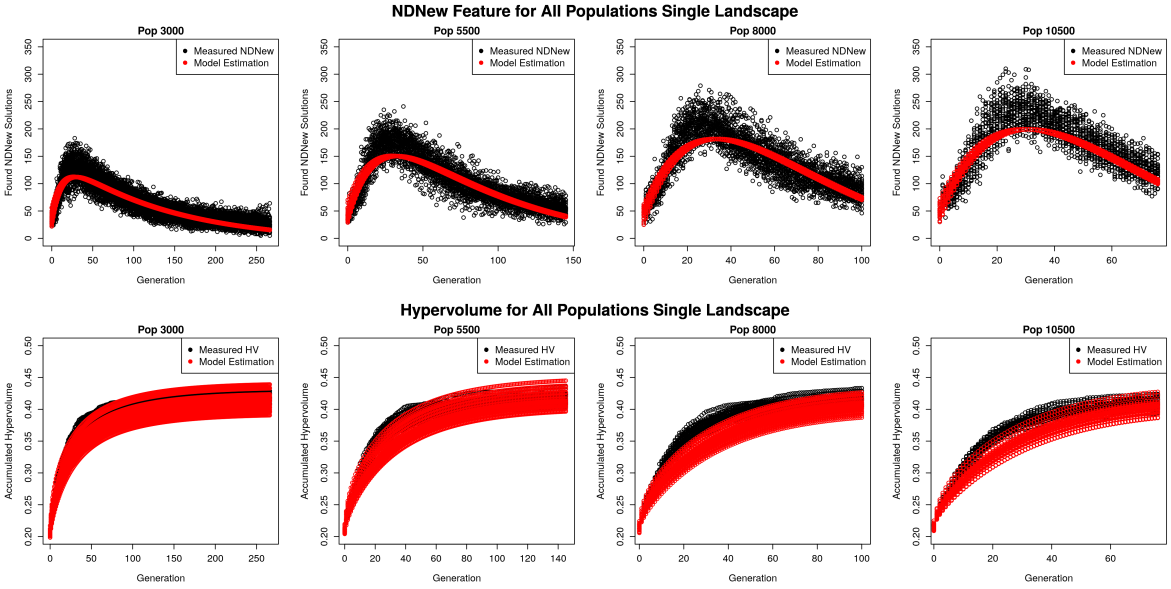


Figure 6.8: Comparison of Features NDNew and HV measured (black) and estimated (red) for a single landscape considering multiple seeds.

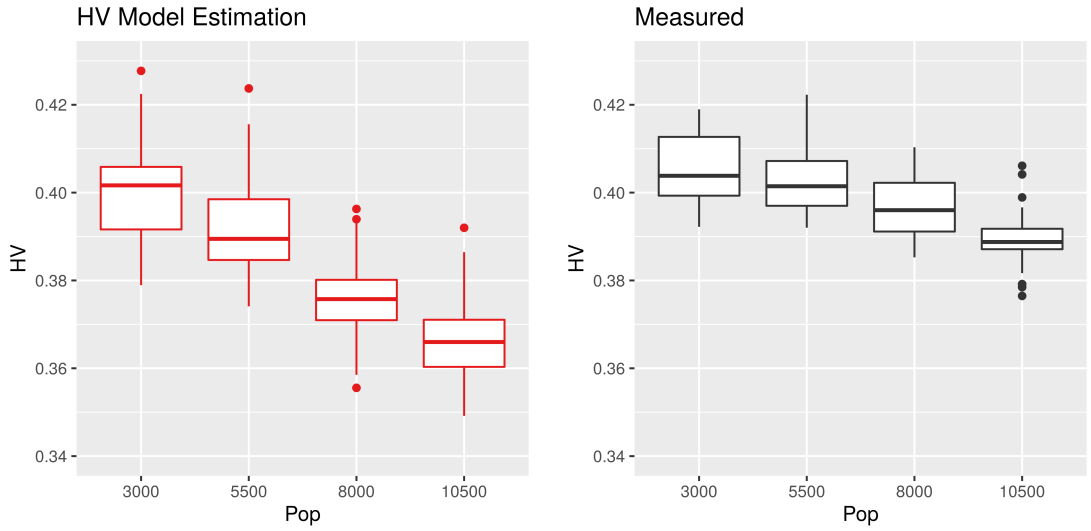


Figure 6.9: Final hypervolume values after 800000 FE for various configurations.

identified using other tools (problem features, landscape analysis), the previously trained models can be used to explore which configuration gives the best performance for a fixed budget of FE.

The expected performance for the 30 different initial populations referred above and 800000 FE is shown in 6.9. Looking at the plot, it can be concluded that a population size of 3000 gives the better performance on this instance and specified budget, without having to run the algorithm.

As more trained models become available, more pairs of algorithms and problems can be considered, and if interpolation, as shown in the previous chapter is added, it becomes easier to explore other configurations not initially considered during the training of the models.

6.6 Transferability

In this last section, one important question is going to be explored. If several of the trained models available do not much completely with the problem instance to be investigated, then is possible or not, to use models trained on similarly enough problem sub-classes that only differ slightly on number of objectives, number of variables, and interaction between variables. In this section, the Dataset 5 will be used.

6.6.1 Same N-K different M

The parameter M determines the number of objectives in MNK-Landscapes, which impacts the dimensionality of objective space and number of non dominated solutions per front. Using the subclass of problems M3N100K5, M4N100K5, and M5N100K5, the behavior of a model trained on M4N100K5 when it tries to estimate for a lower and higher number of objectives will be analyzed. Figure 6.10 shows in black the measured data, in red the estimation done with the model trained on the measured data, and in blue is the estimation for the model trained on instances from the subclass M4N100K5.

As can be noted from the figures, when the model for M4N100K5 is used to estimate instances with lower number of objectives, this model overestimates the NDNew feature which impacts the hypervolume estimation, in turn making this model underestimate the hypervolume. When going to a larger number of objectives, the opposite situation occurs. A closer look to the difference in the estimation can be appreciated in Figure 6.11.

This could be happening because there is a difference in the range of the hypervolume for each subclass. Due to the range of each objective function $\in$ (0 to 1) and how the metric

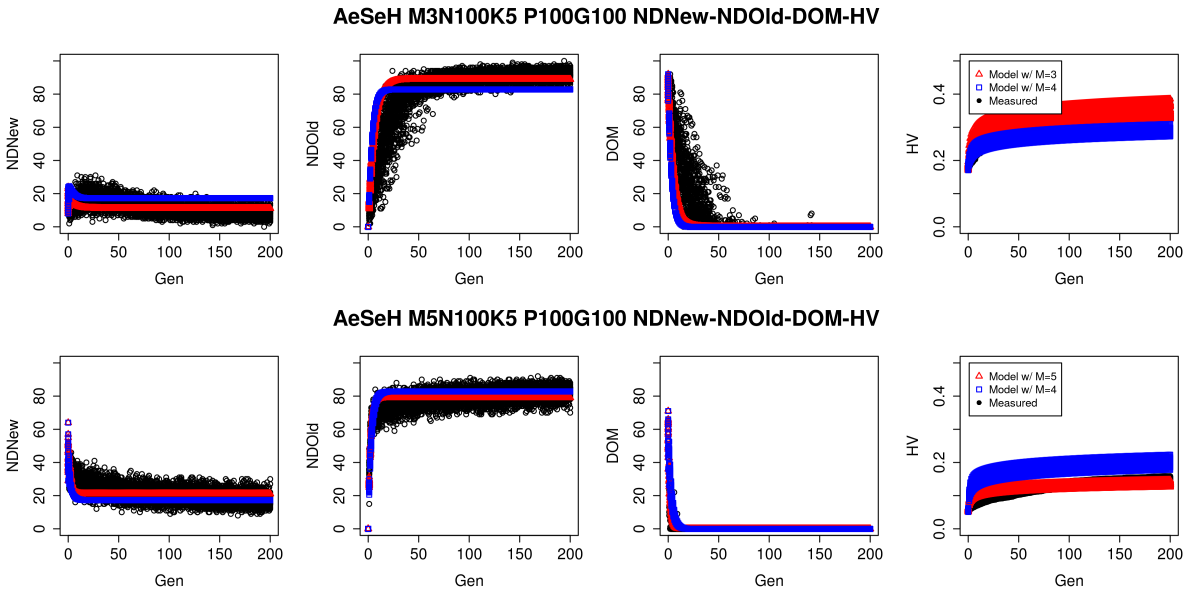


Figure 6.10: Examples of a model trained with M4N100K5 instances estimation ability on instances with lower (M=3) and higher (M=5) objectives.

Figure 6.11: Using model trained on M4N100K5 instances to estimate performance on instances with lower and higher M values.

is calculated, the maximum achievable hypervolume reduces with the number of objectives. This may be avoided by introducing normalization.

Another possible reason is the difference in number of non dominated solutions for each subclass. It is known, that with more objectives, in most cases the number of non dominated solutions increases quickly. Thus, a DCM fitted for a given M will overestimate the values of the features NDNew for a lower M, and underestimate for a larger M. Looking at the coefficients that affect how NDNew is estimated, in equation (3.2), it is expected that the coefficients (1-($\alpha+\beta$)), $\bar{\alpha}$ and $\bar{\beta}$ will take larger values for larger number of objectives, as seen in Table 6.7. Since in these test problems the fitness values are in the range $0 \leq f_i \leq 1$ the absolute value of the hypervolume appears smaller with M. Thus, coefficient μ in the performance estimation model of equation (3.4) should be smaller for larger number of objectives, as the model expects higher counts of NDNew each time, and the hypervolume, in this case, is in a lower range. This suggests that normalization of the counters according to the number of objectives should be introduced.

Table 6.7: Parameters for the NDNew equation of the DCM and the hypervolume model for Subclass N100K5 instances solved with AεSεH with population size 100.

M	**Subclass**	α	β	$\bar{\alpha}$	$\bar{\beta}$	μ
3	N100K5	0.412523	0.002102	0.051730	0.081968	0.002152
4	N100K5	0.395017	-0.010663	0.080203	0.128189	0.000978
5	N100K5	0.378647	-0.012511	0.097756	0.196612	0.000480

Although an example, this illustrates that there are some considerations that should be taken into account to attempt model transferability for different number of objectives, and even then results may not be as good as in other domains like number of variables of interaction between variables.

6.6.2 Same M-K different N

The parameter N determines the number of decision variables in an MNK-Landscape, which modifies the size of the search space. Using the subclass of problems M4N100K5, M4N90K5, and M4N110K5 here it is explored how a model trained on instances with N=100 is able to estimate the features and performance on instances with lower and higher number of variables.

Figure 6.12 shows in black the measured values of the features for each instance, in red is the model estimation trained for each given instance (N=90 or N=110), while in blue is the model trained on M4N100K5. Note that the model trained on the subclass M4N100K5

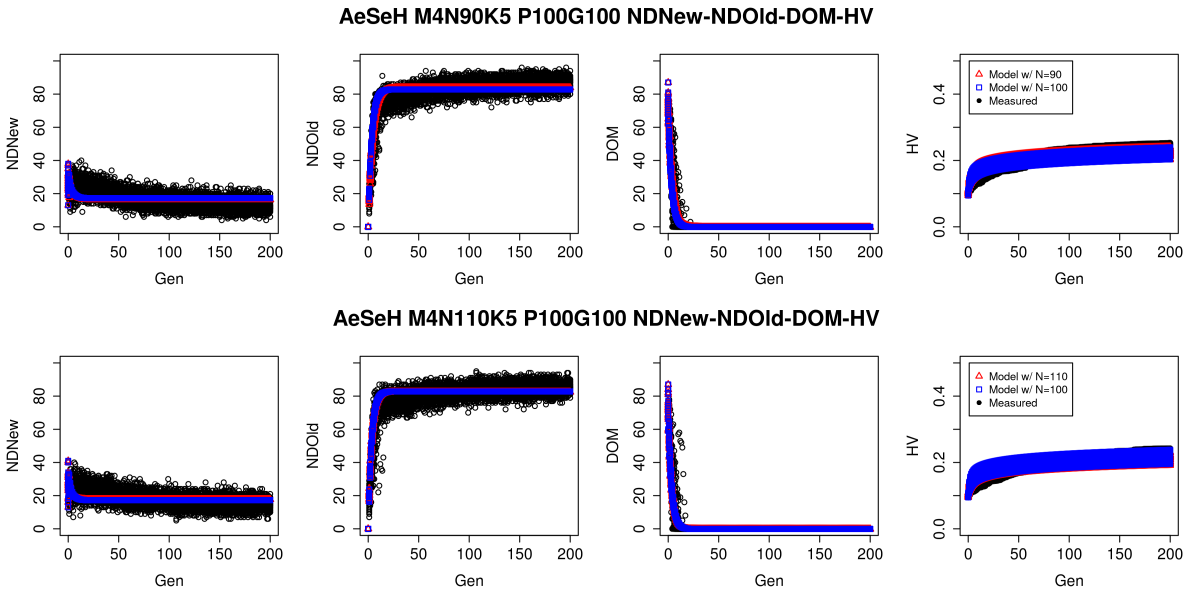


Figure 6.12: Examples of a model trained with M4N100K5 instances estimation ability on instances with lower (N=90) and higher (N=110) decision variables.

is able to follow closely the measured data for the subclasses M4N90K5 and M4N110K5. Taking a closer look at the results of the performance estimation in Figure 6.13 there is a lot of overlap between the models' estimations for each subclass. Looking also at the parameters for both models in Table 6.8, the performance model parameter μ varies slightly between each subclass. This suggests that model transferability for different N could be possible for a given range ΔN on some problems.

Table 6.8: Parameters for the NDNew equation of the DCM and the hypervolume model for Subclass M4 K5 instances solved with AεSεH with population size 100.

Subclass	**N**	α	β	$\bar{\alpha}$	$\bar{\beta}$	μ
M4 K5	90	0.348765	-0.010655	0.066020	0.103543	0.001011
M4 K5	100	0.395017	-0.010663	0.080203	0.128189	0.000978
M4 K5	110	0.412638	-0.017057	0.085076	0.128159	0.000925

6.6.3 Same M-N different K

The parameter K determines the number of correlated variables of each variable of the problem and affects the structure of the problem. For example, it is known that the number of solutions reduces with K, especially in the top local fronts of the landscape [83]. However, these changes are gradual. In addition, within a subclass of problems, there is variability in

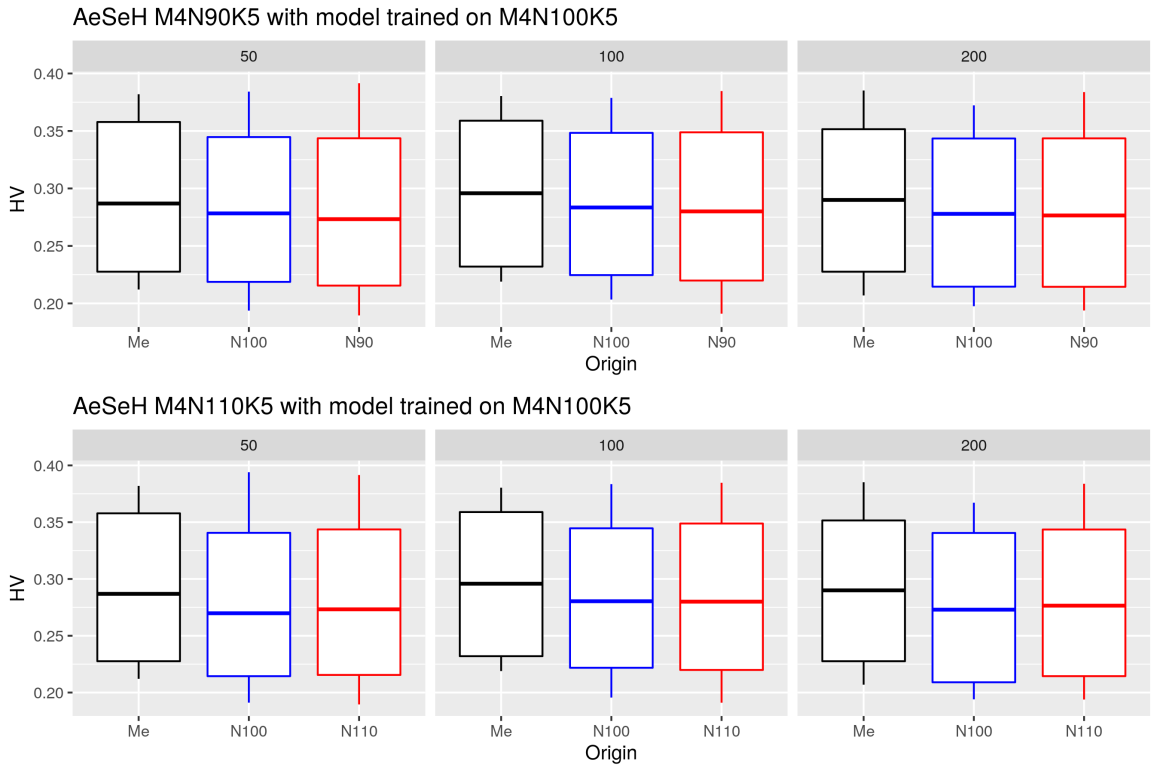


Figure 6.13: Using model trained on M4N100K5 instances to estimate performance on instances with lower and higher N values.

instances, implying that in the problem feature space some instances of a given class may look like ones of another class with a slightly smaller or larger K.

This suggests that to some degree transferability of models could be possible. To illustrate this, here are some example results by a model trained with instances from the class M4N100K5 and tested on the K3 and K10 subclasses. In Figure 6.14 the points in black and red represent the measured data and its estimation by a model fitted with instances of the sub-class K3/K10, respectively, while the points in blue represent the estimation of the model fitted in subclass K5 transferred to subclass K3/K10. These plots correspond to AεSεH with population size 200.

Note that the transferred model in blue is underestimating the value of the feature NDNew when is transferred to instances with a smaller K, and overestimating when is transferred to instances with larger K. Since this feature is also used by the performance estimation model, the hypervolume is slightly over/underestimated in a similar way as shown in Figure 6.15. Is also interesting to look at the parameters for both models in Table 6.9, and see how close they are to each other, even when the subclasses are quite far apart in one case (K=5 and K=10).

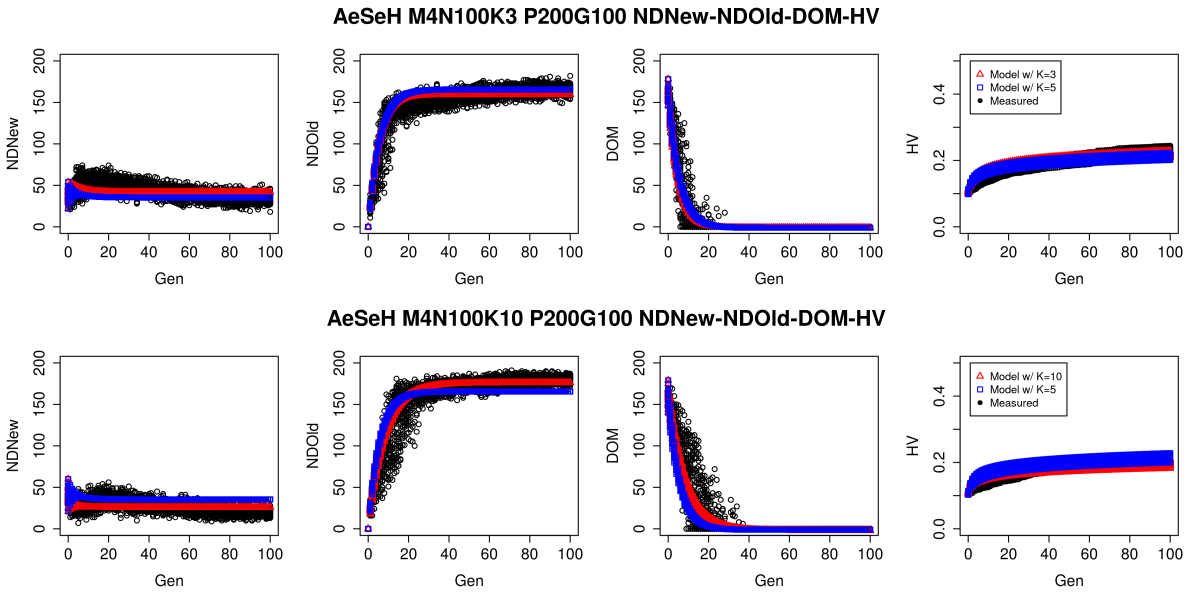


Figure 6.14: Examples of a model trained with M4N100K5 instances estimations ability on instances with lower and higher K values.

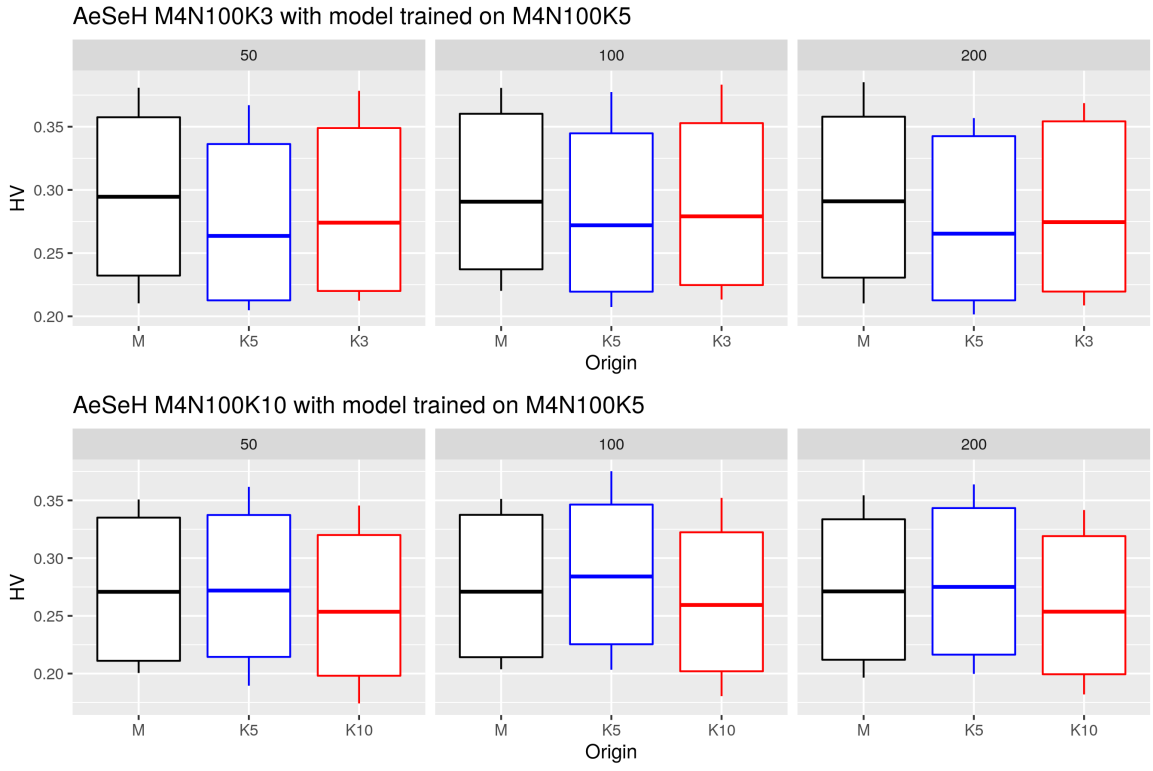


Figure 6.15: Using model trained on M4N100K5 instances to estimate performance on instances with lower and higher K values.

Table 6.9: Parameters for the NDNew equation of the DCM and the hypervolume model for Subclass M3N100 instances solved with AεSεH with population size 200.

Subclass	**K**	α	β	$\bar{\alpha}$	$\bar{\beta}$	μ
M3N100	3	0.293488	-0.039358	0.067216	0.087579	0.000512
M3N100	5	0.330292	-0.026355	0.065760	0.080147	0.000543
M3N100	10	0.375878	-0.025156	0.052120	0.054130	0.000577

From the previous sections on transferability, in summary, it seems that in the case of MNK-Landscape is easier to transfer models that differ in number of variables (N) and variable interactions (K). However, how large the difference can be and still provide a close estimation will require further experiments to test the limits of this use case for these models.

Chapter 7

Conclusions and Future Work

In this work, the population dynamics of some multi- and many-objective optimizers were studied using dynamic and performance models. Through examples, it was shown how to interpret the model parameters to analyze and compare algorithms, extract models for other configurations and use the estimation of the compartments to select between configurations under a budget, in small and large problems.

In small landscapes, the considered two and three compartments models were fitted without issues, achieving high values in the coefficient of determination R^2 which could also be verified in the plots. Although the combination of features and algorithms may produce different dynamics, as seen in the plot shapes, it was verified that the model equations can capture and explain a high percentage of the fitted data. There are cases where the nature of the algorithm produces a high variance in the measured features, but the model is still able to follow the trend of data, producing estimations around the expected value for each feature.

The analysis done with models shows the parameters can capture with high precision the interaction between features. Using the equation forces the user to consider the changes, not in one feature but all of them simultaneously. Thus, given the initial values for each compartment, and the simplified system of equations is possible to see how a particular feature changes and the effect it will produce on the remaining ones, which can later be linked to the way the algorithm is designed or its behavior on the instance.

Focusing on the model that tracks the discovery of new Pareto Optimal Solutions, it was shown that the accumulated number of Pareto Optimal solutions has a high correlation to the hypervolume, a well-known indicator to assess the performance of multi-objective evolutionary algorithms. Using this discovery, the model was used to rank algorithms on the same instance, with similar results to the ordering obtained with the performance metric.

Furthermore, it also allowed the use of models to select between configurations by running the algorithm only in some sample configurations, fitting models with that data, and using

interpolation to extract from the parameters, the models corresponding to configurations between the selected samples. The results indicated that this is a promising methodology that given well-fitted models in the selected samples, can help the user to explore and find which configuration makes better use of its computational budget.

In larger landscapes, the models around the Pareto Optimal Set are not available, which demanded the creation of a new feature set that does not rely on it while allowing the previous task to be done. Here also was introduced a new model to estimate directly performance instead of relying on a correlation with one of the features.

Both the compartmental models with the features set around the most recent non dominated set and the performance estimation model showed good estimations according to the training and testing R^2 values obtained via cross-validation on the training instances. Further results on unseen instances, corroborated the scores obtained for testing. Even with a set of features fixed around a reference that changes constantly as is the most recent non dominated set, the model can follow the trend of the data.

As for the new model, it also showed a good overall result, after finding the proper steps to prepare the data before the fitting. In addition to results on unseen instances, it was tested also variability on a single unseen instance, by making estimations parting from different compositions of the initial population and see how well they fare when compared to measured values. Results have shown that training the model with several instances also prepared it to handle the variability found in a single one. The previous results also were considered to showcase how the model will perform on configuration tasks, achieving good results as was the case with small landscapes.

Although it may seem that models add overhead to the process, they should be seen more as a mathematical approach to certain tasks that evolutionary computation practitioners were doing by experience. Having certain aspects of their dynamics in numbers can help explain more clearly their behavior when analyzing them, while configuration and choosing certain parameters can now be done relying on past data and not only past experiences.

This work presents a step towards developing frameworks and tools that can help characterize how algorithms perform and behave on a problem, and use this information to make decisions on which one should be more appropriate for a given problem, how it can be reconfigured, while also giving clues as to what steps can be taken to make them better.

To keep moving forward on these goals, there is ongoing work to see the behavior of the algorithms considered in this work on the DTLZ benchmark problems, with early results bringing up the importance of choosing a reference point for the hypervolume calculation. Another avenue being explored is changing the model equations to consider only a subset of the relationships between compartments. Limiting their relationships can be used to isolate

which ones are involved in mechanism as elitism, or rediscovery of solutions. Finally, in terms of algorithm configuration is being explored the use of models to decide when a change in population size is needed to avoid early stagnation.

In future works, more algorithms and problems are planned to be considered to build a database of behavior and try to find interesting patterns in this data. To complement this work, it would also be interesting to use exploratory landscape analysis features, or other tools to characterize the problem themselves, find problem classes that are close, and see what the models say about the behavior of an algorithm solving those problems. Finally, it could also prove worthy to change interpolation for other methods, with the hopes of extracting more robust models without depending too much on the quality of sampled ones.

Acknowledgements

In the first place, I would like to thank my supervisors Prof. Hernán Aguirre and Prof. Kiyoshi Tanaka, as well Prof. Sébastien Verel, Prof. Arnaud Liefooghe, and Prof. Bilel Derbel. The thesis committee members Prof. Fumihito Sasamori, Prof. Minami Miyakawa, and Prof. Hiroyuki Sato and faculty at Shinshu University for their support. Prof. Aguirre's comments and continued advice throughout these years not only contributed to the realization of this thesis but also to my formation as a researcher. I am grateful to my undergraduate thesis supervisor Prof. Christian von Lücken, the director of the computer engineering career Prof. Diego Pinto and the dean Prof. Abel Bernal of the Polytechnic School, National University of Asunción. Their support, in the finals steps of my undergraduate course, allowed me to apply with confidence to the MEXT scholarship. To the Japanese Embassy in Paraguay, in particular Mrs. Coti Barrios and Mr. Yamagata, that helped me during the application process and believed that I could overcome any challenge during my graduate studies in Japan. I also want to thank the people that I have met throughout these years in Japan. Their advice not only on the academic was key to adapt to a new country and environment. To the members of the Nagano Hokuto Rotary Club, in particular to Etsuyo Doi, Rie Miyazawa. To my friends, both here and in Paraguay, that always reminded me of the importance to take breaks, laugh a little, and relax whenever I can. Finally, I want to thanks my parents for believing and encouraging me to pursue my dreams, work hard for them, and be thankful for the opportunities that I have been given.

I gratefully acknowledge the financial support provided by the Government of Japan through the MEXT scholarship during the master's course, Shinshu University and the Rotary Yoneyama scholarship during the doctoral course.

Bibliography

[1] Carlos A Coello Coello, Gary B Lamont, David A Van Veldhuizen, et al. *Evolutionary algorithms for solving multi-objective problems*, volume 5. Springer, 2007.

[2] Kalyanmoy Deb. *Optimization for engineering design: Algorithms and examples*. PHI Learning Pvt. Ltd., 2012.

[3] C.A.C. Coello and G.B. Lamont. *Applications of Multi-objective Evolutionary Algorithms*. Advances in natural computation. World Scientific, 2004.

[4] Peter J Fleming, Robin C Purshouse, and Robert J Lygoe. Many-objective optimization: An engineering design perspective. In *International conference on evolutionary multi-criterion optimization*, pages 14–32. Springer, 2005.

[5] Jesús García Herrero, Antonio Berlanga, and José Manuel Molina López. Effective evolutionary algorithms for many-specifications attainment: Application to air traffic control tracking filters. *IEEE Transactions on Evolutionary Computation*, 13(1):151–168, 2009.

[6] Robert J Lygoe, Mark Cary, and Peter J Fleming. A real-world application of a many-objective optimisation complexity reduction process. In *International Conference on Evolutionary Multi-Criterion Optimization*, pages 641–655. Springer, 2013.

[7] A. López-Jaimes, A. Oyama, and K. Fujii. Space trajectory design: Analysis of a real-world many-objective optimization problem. In *2013 IEEE Congress on Evolutionary Computation*, pages 2809–2816, June 2013.

[8] Yutaka Nishio, Akira Oyama, Youhei Akimoto, Hernán Aguirre, and Kiyoshi Tanaka. Many-objective optimization of trajectory design for destiny mission. In *Learning and Intelligent Optimization Conference, Lecture Notes in Computer Science, Springer*, 2014.

[9] R. C. Purshouse and P. J. Fleming. Evolutionary many-objective optimisation: an exploratory analysis. In *The 2003 Congress on Evolutionary Computation, 2003. CEC '03.*, volume 3, pages 2066–2073 Vol.3, Dec 2003.

[10] Vineet Khare, Xin Yao, and Kalyanmoy Deb. Performance scaling of multi-objective evolutionary algorithms. In *International conference on evolutionary multi-criterion optimization*, pages 376–390. Springer, 2003.

[11] Hernán Aguirre and Kiyoshi Tanaka. Insights on properties of multiobjective mnk-landscapes. In *Proceedings of the 2004 Congress on Evolutionary Computation (IEEE Cat. No.04TH8753)*, volume 1, pages 196–203 Vol.1, June 2004.

[12] E. J. Hughes. Evolutionary many-objective optimisation: many once or one many? In *2005 IEEE Congress on Evolutionary Computation*, volume 1, pages 222–227 Vol.1, Sep. 2005.

[13] Hisao Ishibuchi, Noritaka Tsukamoto, and Yusuke Nojima. Evolutionary many-objective optimization: A short review. In *2008 IEEE Congress on Evolutionary Computation (IEEE World Congress on Computational Intelligence)*, pages 2419–2426, June 2008.

[14] Marco Farina and Paolo Amato. On the optimal solution definition for many-criteria optimization problems. In *2002 Annual Meeting of the North American Fuzzy Information Processing Society Proceedings. NAFIPS-FLINT 2002 (Cat. No. 02TH8622)*, pages 233–238. IEEE, 2002.

[15] Christian von Lücken, Benjamïn Barän, and Carlos Brizuela. A survey on multi-objective evolutionary algorithms for many-objective problems. *Computational Optimization and Applications*, 58(3):707–756, 2014.

[16] Bingdong Li, Jinlong Li, Ke Tang, and Xin Yao. Many-objective evolutionary algorithms: A survey. *ACM Computing Surveys (CSUR)*, 48(1):13, 2015.

[17] Christian von Lücken, Carlos Brizuela, and Benjamin Barán. An overview on evolutionary algorithms for many-objective optimization problems. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 9(1):e1267, 2019.

[18] M. A. Muñoz, M. Kirley, and S. K. Halgamuge. Exploratory landscape analysis of continuous space optimization problems using information content. *IEEE Transactions on Evolutionary Computation*, 19(1):74–87, 2015.

[19] Fabio Daolio, Arnaud Liefooghe, Sébastien Verel, Hernán Aguirre, and Kiyoshi Tanaka. Problem features versus algorithm performance on rugged multiobjective combinatorial fitness landscapes. *Evolutionary Computation*, 25(4):555–585, 2017.

[20] Pascal Kerschke, Holger H Hoos, Frank Neumann, and Heike Trautmann. Automated algorithm selection: Survey and perspectives. *Evolutionary computation*, 27(1):3–45, 2019.

[21] Arnaud Liefooghe, Fabio Daolio, Sébastien Verel, Bilel Derbel, Hernan Aguirre, and Kiyoshi Tanaka. Landscape-aware performance prediction for evolutionary multi-objective optimization. *IEEE Transactions on Evolutionary Computation*, 2019.

[22] William Ogilvy Kermack and Anderson G McKendrick. A contribution to the mathematical theory of epidemics. *Proceedings of the royal society of london. Series A, Containing papers of a mathematical and physical character*, 115(772):700–721, 1927.

[23] K. Godfrey. *Compartmental Models and Their Application*. Academic Press, 1983.

[24] A. Eiben and S.K. Smit. *Evolutionary Algorithm Parameters and Methods to Tune Them*, pages 15–36. 01 2012.

[25] Kenneth De Jong. *Parameter Setting in EAs: a 30 Year Perspective*, pages 1–18. Springer Berlin Heidelberg, Berlin, Heidelberg, 2007.

[26] A. Eiben, Zbigniew Michalewicz, Marc Schoenauer, and Jim Smith. *Parameter Control in Evolutionary Algorithms*, volume 3, page 19–46. 01 2007.

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

[28] A. E. Eiben, Elena Marchiori, and V. A. Valkó. Evolutionary algorithms with on-the-fly population size adjustment. In Xin Yao, Edmund K. Burke, José A. Lozano, Jim Smith, Juan Julián Merelo-Guervós, John A. Bullinaria, Jonathan E. Rowe, Peter Tiňo, Ata Kabán, and Hans-Paul Schwefel, editors, *Parallel Problem Solving from Nature - PPSN VIII*, pages 41–50, Berlin, Heidelberg, 2004. Springer Berlin Heidelberg.

[29] Thomas Bäck and Martin Schütz. Intelligent mutation rate control in canonical genetic algorithms. pages 158–167, 06 1996.

[30] Efrén Mezura-Montes and Ana Palomeque-Ortiz. *Self-adaptive and Deterministic Parameter Control in Differential Evolution for Constrained Optimization*, volume 198, pages 95–120. 05 2009.

[31] Rainer Storn and Kenneth Price. Differential evolution: A simple and efficient adaptive scheme for global optimization over continuous spaces. *Journal of Global Optimization*, 23, 01 1995.

[32] I. Rechenberg. *Evolutionsstrategie : Optimierung technischer Systeme nach Prinzipien der biologischen Evolution*. Number 15 in Problemata. Frommann-Holzboog, Stuttgart-Bad Cannstatt, 1973.

[33] Hans-Georg Beyer. Toward a theory of evolution strategies: Self-adaptation. *Evol. Comput.*, 3(3):311–347, September 1995.

[34] Thomas Bäck, A. Eiben, and Nikolai Vaart. An emperical study on gas "without parameters". volume 1917, pages 315–324, 09 2000.

[35] R. Farmani and J. A. Wright. Self-adaptive fitness formulation for constrained optimization. *Trans. Evol. Comp*, 7(5):445–455, October 2003.

[36] Nikolaus Hansen and Andreas Ostermeier. Completely derandomized self-adaptation in evolution strategies. *Evol. Comput.*, 9(2):159–195, jun 2001.

[37] Dirk Thierens. An adaptive pursuit strategy for allocating operator probabilities. In *Proceedings of the 7th Annual Conference on Genetic and Evolutionary Computation*, GECCO '05, page 1539–1546, New York, NY, USA, 2005. Association for Computing Machinery.

[38] Dirk Schlierkamp-Voosen and Heinz Mühlenbein. Strategy adaption by competing subpopulations. pages 199–208, 10 1994.

[39] Christian Igel and Martin Kreutz. Operator adaptation in evolutionary computation and its application to structure optimization of neural networks. *Neurocomputing*, 55(1):347 – 361, 2003. Support Vector Machines.

[40] John R. Rice. The algorithm selection problem. volume 15 of *Advances in Computers*, pages 65 – 118. Elsevier, 1976.

[41] Olaf Mersmann, Bernd Bischl, Heike Trautmann, Mike Preuss, Claus Weihs, and Günter Rudolph. Exploratory landscape analysis. In *Proceedings of the 13th Annual Conference on Genetic and Evolutionary Computation*, GECCO '11, page 829–836, New York, NY, USA, 2011. Association for Computing Machinery.

[42] Lin Xu, Frank Hutter, Holger H Hoos, and Kevin Leyton-Brown. Satzilla: portfolio-based algorithm selection for sat. *Journal of artificial intelligence research*, 32:565–606, 2008.

[43] Marius Lindauer, Holger H Hoos, Frank Hutter, and Torsten Schaub. An automatically configured algorithm selector. 2015.

[44] Frank Hutter, Holger H Hoos, and Kevin Leyton-Brown. Sequential model-based optimization for general algorithm configuration. In *International conference on learning and intelligent optimization*, pages 507–523. Springer, 2011.

[45] Holger Hoos, Marius Lindauer, and Torsten Schaub. claspfolio 2: Advances in algorithm selection for answer set programming. *arXiv preprint arXiv:1405.1520*, 2014.

[46] Bernd Bischl, Olaf Mersmann, Heike Trautmann, and Mike Preuß. Algorithm selection based on exploratory landscape analysis and cost-sensitive learning. In *Proceedings of the 14th annual conference on Genetic and evolutionary computation*, pages 313–320, 2012.

[47] Han-Hsing Tu and Hsuan-Tien Lin. One-sided support vector regression for multiclass cost-sensitive classification. In *Proceedings of the 27th International Conference on International Conference on Machine Learning*, ICML'10, page 1095–1102, Madison, WI, USA, 2010. Omnipress.

[48] Nikolaus Hansen, Steffen Finck, Raymond Ros, and Anne Auger. Real-parameter black-box optimization benchmarking 2009: Noisy functions definitions. 01 2009.

[49] Wolfram Stadler. A survey of multicriteria optimization or the vector maximum problem, part i: 1776–1960. *Journal of Optimization Theory and Applications*, 29(1):1–52, 1979.

[50] C.L. Hwang, S.R. Paidy, A.S.M. Masud, and K. Yoon. *Multiple Objective Decision Making — Methods and Applications: A State-of-the-Art Survey*. Lecture Notes in Economics and Mathematical Systems. Springer Berlin Heidelberg, 2012.

[51] Hisao Ishibuchi, Noritaka Tsukamoto, and Yusuke Nojima. Evolutionary many-objective optimization: A short review. In *2008 IEEE Congress on Evolutionary Computation (IEEE World Congress on Computational Intelligence)*, pages 2419–2426, June 2008.

[52] Vilfredo Pareto. *Manuale di economia politica con una introduzione alla scienza sociale*, volume 13. Società editrice libraria, 1919.

[53] Carlos A Coello Coello. An updated survey of evolutionary multiobjective optimization techniques: state of the art and future trends. In *Evolutionary Computation, 1999. CEC 99. Proceedings of the 1999 Congress on*, volume 1, pages –13 Vol. 1, 1999.

[54] David Allen van Veldhuizen. *Multiobjective Evolutionary Algorithms: Classifications, Analyses, and New Innovations*. PhD thesis, Wright Patterson AFB, OH, USA, 1999.

[55] Gregorio Toscano Pulido. Optimización Multiobjetivo Usando un Micro Algoritmo Genético. Master's thesis, Maestría en Inteligencia Artificial, Universidad Veracruzana, Xalapa, Veracruz, México, September 2001. (In Spanish).

[56] Dhish Kumar Saxena, Jo ao A. Duro, Ashutosh Tiwari, Kalyanmoy Deb, and Qingfu Zhang. Objective reduction in many-objective optimization: Linear and nonlinear algorithms. KanGAL Report 2010008, Kanpur Genetic Algorithms Laboratory, Indian Institute of Technology Kanpur, 2010.

[57] Christian von Lücken, Benjamín Barán, and Carlos Brizuela. A survey on multi-objective evolutionary algorithms for many-objective problems. *Computational Optimization and Applications*, 58(3):707–756, 2014.

[58] E. Zitzler and L. Thiele. Multiobjective evolutionary algorithms: a comparative case study and the strength pareto approach. *IEEE Transactions on Evolutionary Computation*, 3(4):257–271, Nov 1999.

[59] Eckart Zitzler, Marco Laumanns, and Lothar Thiele. Spea2: Improving the strength pareto evolutionary algorithm. *TIK-report*, 103, 2001.

[60] Kalyanmoy Deb, Samir Agrawal, Amrit Pratap, and T. Meyarivan. A fast elitist non-dominated sorting genetic algorithm for multi-objective optimization: Nsga-ii. In *Parallel Problem Solving from Nature PPSN VI: 6th International Conference Paris, France, September 18–20, 2000 Proceedings*, pages 849–858, 2000.

[61] Antonio López Jaimes and Carlos Artemio Coello Coello. Study of preference relations in many-objective optimization. In *Proceedings of the 11th Annual Conference on Genetic and Evolutionary Computation*, GECCO '09, pages 611–618, New York, NY, USA, 2009. ACM.

[62] K. Ikeda, H. Kita, and S. Kobayashi. Failure of pareto-based moeas: does non-dominated really mean near to optimal? In *Proceedings of the 2001 Congress on Evolutionary Computation (IEEE Cat. No.01TH8546)*, volume 2, pages 957–962 vol. 2, 2001.

[63] Marco Laumanns, Lothar Thiele, Kalyanmoy Deb, and Eckart Zitzler. Combining convergence and diversity in evolutionary multiobjective optimization. *Evol. Comput.*, 10(3):263–282, September 2002.

[64] Hiroyuki Sato, Hernán E. Aguirre, and Kiyoshi Tanaka. *Controlling Dominance Area of Solutions and Its Impact on the Performance of MOEAs*, pages 5–20. Springer Berlin Heidelberg, Berlin, Heidelberg, 2007.

[65] Hernán Aguirre, Akira Oyama, and Kiyoshi Tanaka. Adaptive ε-sampling and ε-hood for evolutionary many-objective optimization. In *Evolutionary Multi-Criterion Optimization: 7th International Conference, EMO 2013, Sheffield, UK, March 19-22, 2013. Proceedings*, pages 322–336, 2013.

[66] Qingfu Zhang and Hui Li. Moea/d: A multiobjective evolutionary algorithm based on decomposition. *IEEE Transactions on Evolutionary Computation*, 11(6):712–731, Dec 2007.

[67] Eckart Zitzler and Simon Künzli. Indicator-based selection in multiobjective search. In *Parallel Problem Solving from Nature - PPSN VIII: 8th International Conference, Birmingham, UK, September 18-22, 2004. Proceedings*, pages 832–842, 2004.

[68] Hernán Aguirre, Arnaud Liefooghe, Sëbastien Verel, and Kiyoshi Tanaka. An analysis on selection for high-resolution approximations in many-objective optimization. In *Parallel Problem Solving from Nature – PPSN XIII: 13th International Conference, Ljubljana, Slovenia, September 13-17, 2014. Proceedings*, pages 487–497, 2014.

[69] Manuel López-Ibáñez, Joshua Knowles, and Marco Laumanns. On sequential online archiving of objective vectors. In *International Conference on Evolutionary Multi-Criterion Optimization*, pages 46–60. Springer, 2011.

[70] Andreea Radulescu, Manuel López-Ibánez, and Thomas Stützle. Automatically improving the anytime behaviour of multiobjective evolutionary algorithms. In *International Conference on Evolutionary Multi-Criterion Optimization*, pages 825–840. Springer, 2013.

[71] Conor Ryan, J. J. Collins, and Michael O'Neill. Grammatical evolution: Evolving programs for an arbitrary language. In Wolfgang Banzhaf, Riccardo Poli, Marc Schoenauer, and Terence C. Fogarty, editors, *Proceedings of the First European Workshop on Genetic Programming*, volume 1391 of *LNCS*, pages 83–96. Springer-Verlag, 14-15 April 1998.

[72] Farzad Noorian, Anthony M. de Silva, and Philip H. W. Leong. gramEvol: Grammatical evolution in R. *Journal of Statistical Software*, 71(1):1–26, 2016.

[73] H. Ishibuchi, R. Imada, Y. Setoguchi, and Y. Nojima. How to specify a reference point in hypervolume calculation for fair performance comparison. *Evolutionary Computation*, 26(3):411–440, 2018.

[74] Andreia P. Guerreiro, Carlos M. Fonseca, and Luís Paquete. The hypervolume indicator: Problems and algorithms, 2020.

[75] Hernán Aguirre and Kiyoshi Tanaka. Insights on properties of multiobjective mnk-landscapes. In *Proceedings of the 2004 Congress on Evolutionary Computation (IEEE Cat. No.04TH8753)*, volume 1, pages 196–203 Vol.1, June 2004.

[76] S.A. Kauffman. *The Origins of Order: Self-organization and Selection in Evolution*. Oxford University Press, 1993.

[77] YP Chen, Tian-Li Yu, Kumara Sastry, and David E Goldberg. A survey of linkage learning techniques in genetic and evolutionary algorithms. *IlliGAL report*, 2007014, 2007.

[78] Hernán Aguirre, Arnaud Liefooghe, Sébastien Verel, and Kiyoshi Tanaka. An analysis on selection for high-resolution approximations in many-objective optimization. In *Parallel Problem Solving from Nature – PPSN XIII: 13th International Conference, Ljubljana, Slovenia, September 13-17, 2014. Proceedings*, pages 487–497, 2014.

[79] J.H. Hubbard and B.H. West. *Differential Equations: A Dynamical Systems Approach: A Dynamical Systems Approach. Part II: Higher Dimensional Systems*. Differential Equations: A Dynamical Systems Approach. Springer, 1991.

[80] T.J. Hastie and R.J. Tibshirani. *Generalized Additive Models*. Chapman & Hall/CRC Monographs on Statistics & Applied Probability. Taylor & Francis, 1990.

[81] Kenneth Levenberg. A method for the solution of certain non-linear problems in least squares. *Quarterly of applied mathematics*, 2(2):164–168, 1944.

[82] Gareth James, Daniela Witten, Trevor Hastie, and Robert Tibshirani. *An introduction to statistical learning*, volume 112. Springer, 2013.

[83] Hernán E Aguirre and Kiyoshi Tanaka. Working principles, behavior, and performance of moeas on mnk-landscapes. *European Journal of Operational Research*, 181(3):1670–1690, 2007.

Publications

Journal Papers

- Hugo Monzón, Hernán Aguirre, Sébastien Verel, Arnaud Liefooghe, Bilel Derbel, Kiyoshi Tanaka. *Estimating Hypervolume using Population Features from Dynamic Compartmental Models.* In Transactions of the Japanese Society for Evolutionary Computation. Date of Acceptance: 23-Dec-2020, Pages 13.

International Conferences

- Hugo Monzón, Hernán Aguirre, Sébastien Verel, Arnaud Liefooghe, Bilel Derbel, Kiyoshi Tanaka. *Dynamic Compartmental Models for Large Multi-objective Landscapes and Performance Estimation.* In Proceedings of the European Conference on Evolutionary Computation in Combinatorial Optimization (EvoCOP '20), Seville, April, 2020. Pages 15 (99-113).

- Hugo Monzón, Hernán Aguirre, Sébastien Verel, Arnaud Liefooghe, Bilel Derbel, Kiyoshi Tanaka. *Dynamic compartmental models for algorithm analysis and population size estimation.* In Proceedings of the Companion Publication of the Genetic and Evolutionary Computation Conference (GECCO '19), Prague, July, 2019. Pages 4 (2044-2047).

- Hugo Monzón, Hernán Aguirre, Sébastien Verel, Arnaud Liefooghe, Bilel Derbel, Kiyoshi Tanaka. *Studying compartmental models interpolation to estimate MOEAs population size.* In Proceedings of the Companion Publication of the Genetic and Evolutionary Computation Conference (GECCO '19), Prague, July, 2019. Pages 2 (227-228).

- Hugo Monzón, Hernán Aguirre, Sébastien Verel, Arnaud Liefooghe, Bilel Derbel, Kiyoshi Tanaka. *Studying MOEAs Dynamics and their Performance using a Three Compartmental Model.* In Proceedings of the Companion Publication of the Genetic and Evolutionary Computation Conference (GECCO '18), Kyoto, July, 2018. Pages 2 (191-192).

- Hugo Monzón, Hernán Aguirre, Sébastien Verel, Arnaud Liefooghe, Bilel Derbel, Kiyoshi Tanaka. *Closed State Model for Understanding the Dynamics of MOEAs.* In Proceedings of the Genetic and Evolutionary Computation Conference (GECCO '17), Berlin, July, 2017. Pages 8 (606-616).

Local Conferences

- Hugo Monzón, Hernán Aguirre, Sébastien Verel, Arnaud Liefooghe, Bilel Derbel, Kiyoshi Tanaka. *Improved Hypervolume Estimation from Dynamic Compartmental Models.* 17th workshop of the Japanese Society of Evolutionary Computation, Tokyo, February, 2020.
- Hugo Monzón, Hernán Aguirre, Sébastien Verel, Arnaud Liefooghe, Bilel Derbel, Kiyoshi Tanaka. *Estimating Performance with Dynamic Compartmental Models.*Japanese Symposium on Evolutionary Computation. December, 2019, Hyōgo, Japan.
- Hugo Monzón, Hernán Aguirre, Sébastien Verel, Arnaud Liefooghe, Bilel Derbel, Kiyoshi Tanaka. *Selecting MOEAs Population Size using Compartmental Models.* IEEE Session at the IEICE Shin-etsu Section Conference. September, 2019, Niigata, Japan.
- Hugo Monzón, Hernán Aguirre, Sébastien Verel, Arnaud Liefooghe, Bilel Derbel, Kiyoshi Tanaka. *Studying Compartmental Models Interpolation to Configure Population Size.* Japanese Symposium on Evolutionary Computation. December, 2018, Kyushu, Japan.
- Hugo Monzón, Hernán Aguirre, Sébastien Verel, Arnaud Liefooghe, Bilel Derbel, Kiyoshi Tanaka. *Tracking Population Dynamics using Compartmental Models.* IEEE Session at the IEICE Shin-etsu Section Conference. September, 2018, Niigata, Japan.
- Hugo Monzón, Hernán Aguirre, Sébastien Verel, Arnaud Liefooghe, Bilel Derbel, Kiyoshi Tanaka. *Studying MOEAs Population Dynamics by Generational Features Related to Performance.* Japanese Symposium on Evolutionary Computation. December, 2017, Hokkaido, Japan.
- Hugo Monzón, Hernán Aguirre, Sébastien Verel, Arnaud Liefooghe, Bilel Derbel, Kiyoshi Tanaka. *Closed state model for understanding the dynamics of MOEAs.* IEEE Session at the IEICE Shin-etsu Section Conference. October, 2017. Nagano, Japan.
- Hugo Monzón, Hernán Aguirre, Sébastien Verel, Arnaud Liefooghe, Bilel Derbel, Kiyoshi Tanaka. *Initial steps in MOEAs Dynamics Analysis Using Models.* Japanese Symposium on Evolutionary Computation 2016. December, 2016. Chiba, Japan.
- Hugo Monzón, Hernán Aguirre, Sébastien Verel, Arnaud Liefooghe, Bilel Derbel, Kiyoshi Tanaka. *Capturing MOEAs Dynamics.* IEEE Session at the IEICE Shin-etsu Section Conference. October, 2016. Niigata, Japan.