Cranfield University

Jevgenijs Butans

Addressing Real-Time Control Problems in Complex Environments Using Dynamic Multi-Objective Evolutionary Approaches

School of Applied Sciences

PhD Thesis

Cranfield University

School of Applied Sciences

PhD Thesis

Academic Year 2011-2012

Jevgenijs Butans

Addressing Real-Time Control Problems in Complex Environments Using Dynamic Multi-Objective Evolutionary Approaches

Supervisors: Prof. A. Tiwari, Dr. E. Hughes

October 2011

This thesis is submitted in partial fulfilment of the requirements
for the degree of Doctor of Philosophy

Abstract

The demand for increased automation of industrial processes generates control problems that are dynamic, multi-objective and noisy at the same time. The primary hypothesis underlying this research is that dynamic evolutionary methods could be used to address dynamic control problems where conflicting control criteria are necessary. The aim of this research is to develop a framework for on-line optimisation of dynamic problems that is capable of a) representing problems in a quantitative way, b) identifying optimal solutions using multi-objective evolutionary algorithms, and c) automatically selecting an optimal solution among alternatives.

A literature review identifies key problems in the area of dynamic multi-objective optimisation, discusses the on-line decision making aspect, analyses existing Multi-Objective Evolutionary Algorithms (MOEA) applications and identifies research gap. Dynamic evolutionary multi-objective search and on-line *a posteriori* decision maker are integrated into an evolutionary multi-objective controller that uses an internal process model to evaluate the fitness of solutions.

Using a benchmark multi-objective optimisation problem, the MOEA ability to track the moving optima is examined with different parameter values, namely, length of pre-execution, frequency of change, length of prediction interval and static mutation rate. A dynamic MOEA with restricted elitism is suggested for noisy environments.

To address the on-line decision making aspect of the dynamic multi-objective optimisation, a novel method for constructing game trees for real-valued multi-objective problems is presented. A novel decision making algorithm based on game trees is proposed along with a baseline random decision maker.

The proposed evolutionary multi-objective controller is systematically analysed using an inverted pendulum problem and its performance is compared to Proportional–Integral–Derivative (PID) and nonlinear Model Predictive Control (MPC) approaches. Finally, the proposed control approach is integrated into a multi-agent framework for coordinated control of multiple entities and validated using a case study of a traffic scheduling problem.

Contents

List of Figures

List of Tables

List of acronyms

ANN Artificial Neural Network

CFD Computational Fluid Dynamics

DES Discrete Event Simulation

DMOEA Dynamic Multi-Objective Evolutionary Algorithms

DM Decision Maker

EA Evolutionary Algorithms

EC Evolution Control

ELDP Experiential Learning Directed Perturbation

EMOO Evolutionary Multi-Objective Optimisation

EMO Evolutionary Multi-objective

ESPEA Extended Strength Pareto Evolutionary Algorithm

FastPGA Fast Pareto Genetic Algorithm

FIFO First In, First Out

FMS Flexible Manufacturing System

GASS Gene Adaptation Selection Strategy

IP Internet Protocol

IS Information System

MAS Multi-Agent System

MCDM Multi-Criteria Decision Making

MIMO Multiple-Input and Multiple-Output

MOEA Multi-Objective Evolutionary Algorithms

MOO Multi-Objective Optimisation

MOP Multi-Objective Problems

MOPSEA Multi-Objective Probabilistic Selection Evolutionary Algorithm

MPC Model Predictive Control

NSGA-II Non-dominated Sorting Genetic Algorithm-II

PAE Performance Assessment Environment

PDT Pareto Decision Tree

PID Proportional–Integral–Derivative

SVM Support Vector Machine

SPEA2 Strength Pareto Evolutionary algorithm 2

VoIP Voice over IP

WPM Weighted Product Model

WSM Weighted Sum Model

XML eXtensible Markup Language

XP eXtreme Programming

Nomenclature

δ_A	Adjustment interval
δ_C	Control interval
λ	Number of offspring individuals
λ	Uncertainty increase constant
μ	Number of parent individuals
ρ	Rate of change
τ_T	Number of fixed generations between states
ξ	Elitism ratio of dNSGAII-C
A	Magnitude of change
c_{HV}	Ratio of hypervolumes of obtained Pareto front to the true Pareto front
CI_2	Two-sided confidence interval
d	Design variable
e_i	Model approximation error
f	Objective function
g	Inequality constraints

h	Equality constraints
K_d	Derivative gain
K_i	Integral gain
K_p	Proportional gain
N	Noise
N_p	Number of pre-execution generations
n_τ	Number of problem states
PF	Pareto front
PS	Pareto set
R	Rank of solution
T_c	Control horizon
T_p	Prediction horizon
u	System input
v_{HV}	Relative variability of hypervolume
w	Control effort
w	Criterion weight
x	Decision variable

Chapter 1

Introduction

An increasing level of automation of various systems poses control engineers with increasingly complex problems to solve. In particular, the complexity may increase quantitatively, e.g., more refined, and, therefore, more computationally expensive process models, increased resolution of sensors and dimensionality of the variable space. However, quantitative changes often lead to qualitative changes in the control problem, that make it markedly different to legacy ones. One change that emerged with the proliferation of intelligent agents [1] is the need of coordinated control of multiple entities.

The term "coordinated control of multiple entities" refers to a set of intelligent agents, each having a set of goals, that are able to act in a coordinated manner. An example of cooperative control of multiple missiles is presented by Hughes [2]. Pereira and De Sousa [3] give an application of coordinated control strategies to autonomous underwater vehicles. Having multiple entities each with a potentially different set of goals presents a number of challenges that makes this type of problem particularly interesting.

- **Conflicting objectives**. Even a single intelligent agent may have a set of goals that contradict each other. Within a single entity, these conflicts can

be usually addressed by a bespoke modelling or an aggregation function that combines the criteria in a way acceptable to the system designer. A procedure that aims to control multiple entities would have to account for each individual's goals. However, dealing with the conflicts between different objectives becomes more difficult.

- **Uncertainty**. The environment observed by an agent depends on the actions of other agents, which makes it uncertain. Explicit handling of multiple objectives increases the dimensionality of the objective space, and, correspondingly, the computational requirements, that must be met. As a result, the models used to predict the system performance may be simplified and no longer reflect the true state. Additional sources of uncertainty come from measurement noise and limited sensor range.

- **Time constraints**. In the aforementioned applications, the interaction between agents happens in real time. The control procedure must execute on-line to supply continuous stream of control actions for the agent. The on-line aspect puts a maximum response time constraint for the operation of the control algorithm.

To summarise, the coordinated control of multiple entities presents a complex control problem, the term 'complex' is understood to be interdependent, uncertain and dynamic. The control procedure is thought to operate in real time, therefore having a definite interval to respond with a controlling action.

1.1 Research aim, objectives and methodology

The primary hypothesis underlying this research is that dynamic evolutionary methods could be used to address dynamic control problems where conflicting control

criteria are necessary.

The aim of this research is to develop a framework for on-line optimisation of dynamic problems that is capable of a) representing problems in a quantitive way, b) identifying optimal solutions using multi-objective evolutionary algorithms, and c) automatically selecting an optimal solution among alternatives.

1.1.1 Research objectives

The research aim stated above indicates the direction of the research. Specific research objectives are set below to structure the work.

1. Develop dynamic problem specification and provide quantitative representation suitable for optimisation using evolutionary techniques.

2. Propose solutions to deal with problem dynamics and create a framework for multi-objective optimisation of dynamic problems.

3. Identify the criteria for decision making and suggest a strategy for making decisions from a set of Pareto-optimal solutions on-line.

4. Identify and select performance metrics for the dynamic optimisation case, and systematically evaluate the performance of the dynamic multi-objective optimisation framework.

5. Validate the proposed framework using a case study.

1.1.2 Research methodology

Methodology documents the process of managing the research project by describing the techniques used to acquire, categorise, store and analyse the information. Veryard [4] urges to view a methodology as an Information System (IS), because

it involves creation, transmission and analysis of information and decisions in the form of models and specifications. That view of a methodology fits well with a general definition of information systems. For example, according to UK Academy For Information Systems, "information systems are the means by which people and organisations, utilising technologies, gather, process, store, use and disseminate information" [5].

Having noted a great deal of similarity between a research methodology and an information system, it can be argued that the methods used to develop an information system may be successfully applied to deal with the complexities of a research project. The challenge in this research is that having multiple intelligent agents, each of which is observed as a 'black box' by the other agents, makes the problem difficult to analyse. Borrowing from Wasserman, who mentions "Modularization — the problem-solving notion of "divide-and-conquer" permits one to subdivide a difficult problem into subproblems and then to divide those problems repeatedly until the resulting problems become intellectually manageable" [6] as the primary concept underlying methods and tools for IS development, the problem could be decomposed into smaller, easier to analyse parts. An obvious solution is to attempt to control a single entity first, modelling the actions of other agents as external constraints.

The key feature of an intelligent agent is its ability to observe and act rationally on environment. Applying the concept of abstraction [6], an intelligent agent can be substituted by any dynamic control problem, as a control method is usually able to observe the environment (system output) and act towards achieving the control signal (system input). Consequently, an intermediate target of research is to be able to handle a dynamic control system with multiple performance criteria and uncertainties. Figure 1.1 presents a conceptual diagram of the project methodology. The project methodology is comprised of three major stages, namely, Analyse, Develop and Implement & Validate.

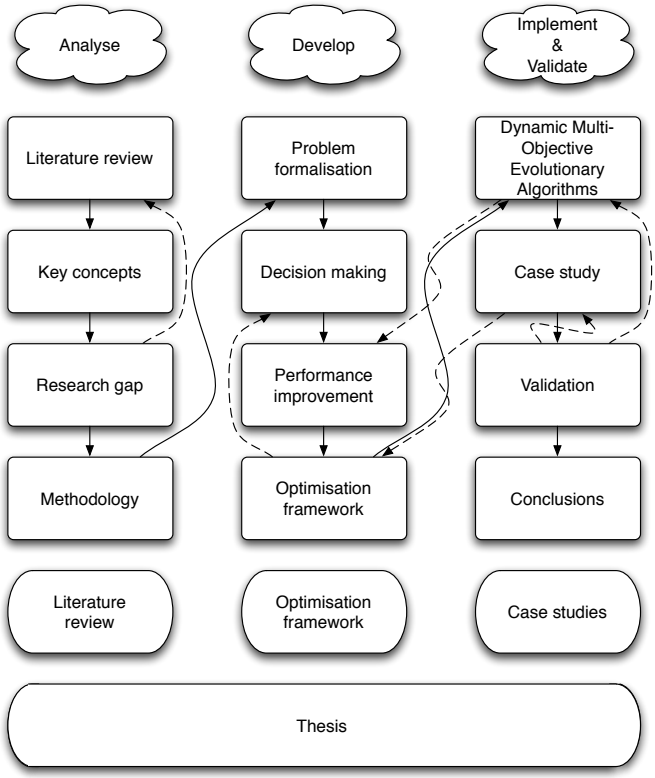


Figure 1.1: Project methodology conceptual diagram

1.1.2.1 Analyse

An extensive literature review is carried out in the areas of dynamic multi-objective optimisation, soft computing, control theory and process modelling. The primary goals of the literature survey are to gain an in-depth understanding of the subject area and acquire terminology, and establish a solid knowledge base for the research by exploring the frontier of current knowledge. The literature review is continuously updated with information about latest developments in the target areas.

The data gathered from current literature is analysed, to identify the key concepts in the subject domain and to identify a research gap. Based on the research gap the methodology of the research is updated and refined.

The literature review should help to identify benchmark problems that are used in the area of control systems and methods used to control them. Also, the current techniques for dynamic decision making should be analysed.

1.1.2.2 Develop

The phase is concerned with four major tasks. First, a multi-objective dynamic control problem should be formalised. In particular, the problem representation and modelling techniques are analysed. Then, the approach of deciding on the operation point in real time is developed along with methods to improve performance of Dynamic Multi-Objective Evolutionary Algorithms (DMOEA) in such conditions. This phase ends with the development of the optimisation framework suitable for application on a set of multi-objective dynamic optimisation problems.

1.1.2.3 Implement and Validate

A software implementation of the algorithms developed in the previous stage is performed in Java programming language. Java is a standard for academic computing projects and includes a number of implementations of the current multi-objective optimisation algorithms [7] that can be used as a "sandbox". The eXtreme Programming (XP) [8, 9, 10] software development methodology will be used to produce the software tool. XP, being an agile software development methodology, takes conventional best practices in software development to the 'extreme', asserting that if a practice is beneficial, than more of the same would be even better. The methodology is chosen due to the strong emphasis on unit tests and incremental changes, which is advantageous for this research. In addition, XP advocates starting development with the simplest solution and adding extra functionality later, if required. That corresponds to "divide-and-conquer" approach used in this research.

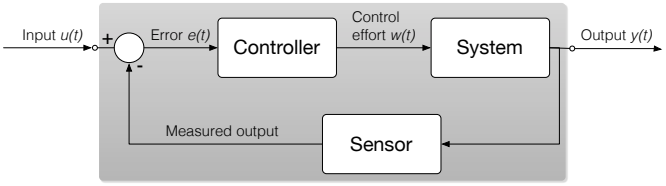


Figure 1.2: A control system

A case study is chosen to demonstrate a wide range of applications and characteristic features of the proposed optimisation framework. A validation is performed using the case study with an analysis of results.

1.2 Introduction to control problems

A control system comprises a device or set of devices (controller) designed to manage the other devices (the system being controlled, or plant), often with a set of sensors to measure the outputs of the plant. Referring to Figure 1.2, which depicts a typical closed-loop control system, the *controller* supplies the *system* with a *control effort* $w(t)$ that directs the system *output* $y(t)$ to match the *input signal* $u(t)$, therefore eliminating the error $e(t)$. The *sensor* observes the instantaneous system output, which is being subtracted from the instantaneous input signal, producing the error signal. It is important to note that at each time instant the controller produces a single control action for the plant.

In order to compute the control effort, an ideal controller must solve an optimisation problem. Optimisation refers to choosing the best solution from a set of feasible alternatives. To be able to optimise the multitude of different real-life systems using a generic optimisation procedure, these problems have to be modelled, i.e., presented using a simplified formal description. In the simplest case, to optimise a process one seeks to minimise or maximise some quantity by adjusting a number

of variables. The manipulated variables are termed as independent and the quantity being optimised is termed as a dependent variable. In other words, the dependent variable is a function of independent variables, such a function is termed as the objective function. In the remaining of this thesis it is assumed that the objective function is always being minimised.

The independent variables usually have certain limitations on the values that can be used. These limitations are expressed as equality and inequality constraints. Having all the components of an optimisation problem defined, it can be expressed in mathematical terms as

$$\begin{aligned} &\min_x f(x) \\ &\text{where } x = [x_1, x_2, ..., x_m]^T \\ &\text{s.t.} \\ &g(x) \leq 0 \\ &h(x) = 0 \\ &x \in [x_{min}, x_{max}], \end{aligned} \tag{1.1}$$

where $f(x)$ is the objective function, x is the vector of independent variables and $g(x)$ and $h(x)$ are constraints.

Equation (1.1) describes a static optimisation problem, i.e., one that has its elements constant over time. In the real world, the parameters of the optimisation problem do not tend to remain constant over time. In steel manufacturing, the changing quality of source materials requires adjustments to the process in order to get the same quality steel. A traffic scheduler in an Internet Protocol (IP) network given a Voice over IP (VoIP) packet has to minimise latency instead of optimisation for throughput. To plan a viable trajectory for a robotic vehicle, the control module has to consider a changing environment. Each of these scenarios describes a dynamic optimisation problem, which has one or more parameters that depend on time. More

formally, the elements of a dynamic optimisation problem are functions of parameter t.

$$\begin{aligned} &\min_x f(x,t) \\ &\text{where } x = [x_1(t), x_2(t), ..., x_m(t)]^T \\ &\text{s.t.} \\ &g(x,t) \leq 0 \\ &h(x,t) = 0 \\ &x_{min}(t) \leq x(t) \leq x_{max}(t), \end{aligned} \tag{1.2}$$

1.3 Model predictive control

The majority of controllers employed today are of a Proportional–Integral–Derivative (PID) variety. A PID controller calculates the control effort from the error signal using three separate parameters: the proportional, the integral and derivative coefficients. The proportional parameter, denoted by P, depends on the current value of the error signal, the integral parameter I depends on the accumulated past errors and the derivative parameter D depends on the current rate of change of the error signal. PID is popular because it is easy to implement and does not require a process model to run, but it experiences difficulties in a complex system with higher order dynamics, noise and time delays.

Complex dynamic systems require a more advanced control method, such as Model Predictive Control (MPC). Like the PID, MPC uses the current plant measurements and the process targets as the input data. In addition, it uses the current dynamic state of the plant and the process models to predict the future changes in the process outputs. At each time instant, the MPC procedure uses the internal process model to explore the process trajectories over a finite horizon. To guide the

process towards the desired state, the MPC uses a cost function that depends on process targets, history of last control actions and current plant state. The MPC control sequence is obtained by minimising the cost function over the future control trajectories. Typically, after the first step of the control sequence is implemented, the prediction horizon is shifted forward and the MPC procedure repeats itself [11]. For this reason the MPC is often referred as *receding horizon control*.

The MPC cost function usually yields a scalar value, so the MPC controller is effectively performing a single-objective optimisation routine. Indeed, the different objectives of the underlying problem are aggregated in the cost function. The aggregation of different objectives implies that the MPC requires information that describes tradeoffs between them. In addition, the cost function, which is predominantly a Weighted Sum Model (WSM), should not be used for problems with non-convex tradeoff surfaces, but the analysis of the tradeoff surface shapes is often lacking in the MPC applications, with one of the rare examples presented in [12].

1.4 Evolutionary multi-objective optimisation

In recent years, Evolutionary Multi-Objective Optimisation (EMOO) has established itself as a reliable tool with a diverse set of applications in the management, manufacturing, design, defence and other sectors [13]. EMOO is characterised by comparatively high computational load and development of a set of optimal solutions, which requires a decision making step to select the final one [14]. For these reasons, the majority of the applications considered the problem to be static over the course of optimisation. A generic dynamic process is usually simplified using a static model with certain assumptions about its dynamics.

A Multi-Objective Optimisation (MOO) procedure that is able to operate in dynamic environments coupled with an on-line decision maker would allow for multi-

criteria control of dynamic processes. A dynamic process model provides greater fidelity and flexibility for a majority of real-life processes. Therefore, the benefits of the dynamic MOO procedure could be classified into two main categories: i) existing applications can be refined to obtain improved results such as precision or response time and ii) the scope of MOO can be extended to the cases that are difficult to represent using single objective or static models [15].

The problems described in Section 1.2 allow for a single universally best solution. In fact, problems in different fields, such as design, scheduling or transportation network optimisation often have trade-offs between two or more conflicting objectives. For example, increasing structural strength of a bridge leads to increased weight, and, consequently, increased costs and build time; minimising fuel consumption of a vehicle reduces dynamic characteristics and increases development costs. The quality of a product and associated profit also bear a trade-off. For a multi-objective problem, there is no single solution that is able to minimise all conflicting performance criteria. In mathematical terms, multi-objective optimisation problems differ from single-objective ones by having a vector of objective values instead of a single objective function

$$\begin{aligned} &\min_x f(x) \\ \text{where } &x = [x_1, x_2, ..., x_m]^T \\ \text{and } &f(x) = [f_1(x), f_2(x), ..., f_n(x)]^T \\ &\text{s.t.} \\ &g(x) \leq 0 \\ &h(x) = 0 \\ &x \in (x_{min}, x_{max}). \end{aligned} \tag{1.3}$$

To be able to minimise $f(x)$ in the multi-objective case, the solution alternatives must be comparable. The predominant comparison method used by the multi-

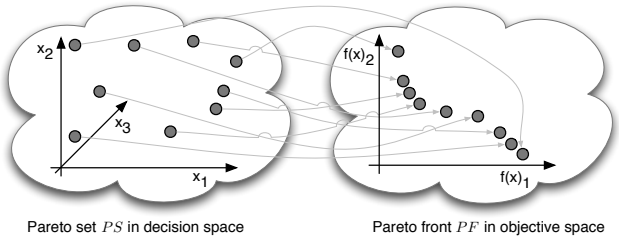


Figure 1.3: Decision and objective spaces.

objective community is Pareto ranking named after Italian economist Vilfredo Pareto [13]. Pareto ranking is based on the concept of Pareto dominance; one alternative x^* is said to (strongly) Pareto dominate another alternative $x^{'}$ if for all objectives x^* is not worse than $x^{'}$ and there exists at least one objective where x^* is better than $x^{'}$

$$\forall i \in \{1, 2, ..., n\} f_i(x^*) \leq f_i(x^{'}) \wedge$$
$$\exists i \in \{1, 2, ..., n\} f_i(x^*) < f_i(x^{'}). \tag{1.4}$$

It follows from (1.3) and (1.4) that the task of a multi-objective algorithm is to find a set of Pareto-optimal points in *decision space* termed as Pareto set PS. Objective values of the solutions that belong to the Pareto set form a Pareto front PF in *objective space* [14]. Figure 1.3 shows an example of Pareto set PS in a three-dimensional decision space mapped to Pareto front PF in a two-dimensional objective space. A solution is part of the Pareto set if there are no other solutions that dominate it.

1.4.1 Importance of multi-objective problem formulation

Multi-objective problems can be modelled as single-objective by using objective ordering (also known as lexicographic techniques, goal programming and aggregate objective functions) [14]. Aggregate functions, in turn, include generic linear ob-

jective aggregation approaches, such as WSM, Weighted Product Model (WPM) or L^p norm-based aggregation and utility functions, that are often non-linear and problem specific. One feature these methods share in common is that the objectives have to be 'prioritised' before the search occurs. For many real life engineering and research problems it may be difficult to establish the objective priority in advance. The cost to the decision maker is, therefore, very high, as the aggregating fitness function directly impacts search, possibly causing it to miss some compromise solutions. In contrast, a multi-objective problem formulation allows the practitioner to delay the choice until the tradeoff surface is obtained, which is especially important in dynamic problem optimisation [16].

From a taxonomical point of view, multi-objective and single-objective problems are connected in a generalisation-specialisation relationship. Typically, a multi-objective problem describes a more complete and therefore complex version of a single-objective case. Consequently, results of an optimisation research with multi-objective problem formulation can be applied to a wider scope of real-life scenarios.

1.4.2 Motivation for evolutionary approach

Multi-Objective Evolutionary Algorithmss (MOEAs) are optimisation algorithms capable of producing the entire set of solutions to the problem (the Pareto front) simultaneously, in a single optimisation run. They are based on Darwin's hypothesis that that life forms adapt to their environment by a process known as natural selection [17]. MOEAs evolve a set of alternative solutions to a problem, termed as population. Because solutions are distributed, the algorithm is inherently robust to small amounts of noise. Likewise, the population concept allows the algorithm to be scaled up or down for tasks of different complexity and improve its performance by means of parallel execution [14, 18].

MOEAs are problem-agnostic, i.e., they can be applied to a wide range of optimisation problems including problems that are difficult to understand and fully formalise. Because the algorithm develops a Pareto-optimal front instead of a single solution, a practitioner is not required to rank or compare different objectives [19].

Evolutionary approaches contain a number of properties that are especially useful in optimisation of dynamic problems. Intermediate solutions can be obtained from the population at any stage of the optimisation run, therefore enabling continuous process control. With slight modifications to existing state-of-the-art evolutionary algorithms, they can adapt to changes in the environment, continuously tracking moving optima [20]. One particularly interesting feature is that an unfeasible solution can potentially be evolved into a good one after a number of generations. Storing previously fit solutions often leads to an increase in performance for problems with cyclic optima transitions, whereby the position of the moving optimum repeatedly visits previous locations [21]. The aforementioned advantages of evolutionary algorithms and their relative ease of implementation make MOEA a dominant approach in the field of multi-objective optimisation, with a multitude of published research works, algorithms and wide range of applications [22]. In this research, Evolutionary Algorithms (EA) were selected as an established method of dealing with uncertain multi-objective problems having a number of applications to dynamic problems.

1.5 Multi-criteria decision making

As noted in Section 1.4, multi-objective problem formulation provides a number of benefits over the single objective aggregation-based approach employed by current dynamic control methods such as MPC. One can wonder if an (evolutionary) multi-objective search algorithm can be integrated into an MPC framework to achieve these benefits.

Applying a dynamic MOEA to a control problem requires an additional element: a dynamic decision maker. At each time instant, the MOEA produces a Pareto front, i.e., the tradeoff surface between objectives. At the same time, as noted in Section 1.2, the controller should produce a single control action for the plant. The decision maker is necessary to select the Pareto-optimal control action from the tradeoff surface.

Hwang and Masud [23] describe the following taxonomy of Multi-Criteria Decision Making (MCDM) methods, based on the stage at which the preference information about objectives is articulated.

1. **No articulation of preference information**. Decision maker is not provided with any ordering or weighting information about the objectives.

2. ***A priori* preference articulation**. The decision takes place before the search phase. Decision maker combines multiple objectives into a scalar-valued cost function, which can be optimised using a single-objective search method.

3. ***Progressive* preference articulation**. Decision and search cycles are interleaved with each other. Decision making at cycle n influences search at cycle $n+1$.

4. ***A posteriori* preference articulation**. Decision maker has to select a solution from a set of provided Pareto-optimal solutions.

It can be argued that the *Global Criterion Method* [24, 25] referred to as belonging to the first category in [26] is, in fact, an *a priori method*. Therefore, only the categories 2, 3 and 4 are to be considered, as in [16].

A priori decision making is widely applied in the domain of dynamic control because of its lower computational requirements, relative simplicity and inherent ability to produce a single 'optimal' point at each time instant. In contrast, current

state-of-the-art MOEA develop the full set of Pareto-optimal solutions in a single pass. Correspondingly, the evolutionary search should be followed with *a posteriori* decision making cycle [20]. Because the decision has to be made on-line as soon as the Pareto-optimal set is found, a decision maker clearly should be an automated procedure, capable of selecting the control action without human interaction.

A majority of existing MCDM methods [27, 23, 28, 29] seem to be targeted for a static one-off execution scenario. Nevertheless, they are actively employed in on-line applications, WPM [28] and WSM [30] in particular enjoying a lot of attention.

A solution using game trees [31, 32] may be better suited for the dynamic selection of a Pareto-optimal control action. A single player scenario with uncertainty, exemplified by a typical control problem, is addressed via the 'move by nature' method [33]. Using the game tree, a *minimax* algorithm is used to select the next move [1]. It should be noted, however, that the game trees are operating on integer games for the most part, i.e., having a finite number of alternatives (tree branches) at each split [1]. A tradeoff surface of a dynamic optimisation problem in the general case has real-valued solutions, providing a potentially unlimited number of alternatives.

1.6 Thesis structure and layout

This thesis consists of 9 chapters. Chapter 2 surveys the current state-of-the-art research in the areas of classic control methods, modern control methods exemplified by MPC, dynamic single and multi-objective evolutionary optimisation algorithms and modelling uncertainties. Chapter 3 describes the proposed approach to dynamic optimisation of control problems using an Evolutionary Multi-objective (EMO) MPC method. Chapter 4 details the design aspects of MOEA that enable them to be used in real-time dynamic search. A novel approach to make decisions in

dynamic environments by using real-valued game trees is presented in Chapter 5. In addition, a stochastic method for decision making is detailed with results presented in Chapter 6. Building upon the preceding chapters, Chapter 7 outlines a Multi-Agent System (MAS) that uses coevolution MOEA coupled with a bespoke dynamic decision maker to simultaneously control multiple agents. A case study of coordinated control of multiple vehicles is presented in Chapter 8. The thesis concludes with Chapter 9, that summarises key findings, presents conclusions, discusses limitation of the present work and gives future research directions.

1.7 Statement of original work

- Chapters 3 and 4 detail the theory of a novel application of MOEAs to dynamic control problems.
- Chapter 5 details novel dynamic decision making approaches based on game trees.
- Chapter 6 presents an evaluation of the proposed EMO MPC approach using an inverted pendulum problem.
- Chapter 7 details a novel MAS framework for coordinated control of multiple entities using the EMO MPC approach.
- Chapter 8 presents a case study of the EMO MPC approach integrated into a MAS for distributed dynamic traffic scheduling.

Chapter 2

Literature review

2.1 Motivation

According to [34], a majority of approaches (90%) to solving multi-objective problems try to approximate the true Pareto front. About 70% of these approaches use EA as the primary meta-heuristics, compared to simulated annealing 24% and tabu search 6%. The importance of EA can be attributed to several factors, one of them being an abundance of algorithms for finding the non-dominated solution set. The earliest MOEA known to the authors is presented in the works of Schaffer [35]. During the early 1990s, a number of MOEA were developed, including [36] and [37]. The success of such approaches is supported by the observation that the population-based nature of an EA naturally provides multiple solutions to an optimisation problem with multiple criteria.

An off-the-shelf single-objective EA could be readily modified to find a front of non-dominated solutions in a single run. This is in contrast to earlier generations of optimisation techniques that usually used a weighted sum approach to fitness evaluation [38] or optimised a single criterion treating other objectives as constraints. EAs perform simultaneous search in multiple parts of the objective space, producing a

diverse set of results in multi-modal, discontinuous and non-convex objective spaces. Because EA do not rely on aggregation approaches to assess an individual's fitness, the user is not required to weight or rank objectives. Finally, unlike the methods that obtain the Pareto front by analytical means, EA are able to provide useful intermediate information for the decision maker in cases, e.g., where the algorithm was terminated prematurely.

The discrete nature of EA allows them to handle models with both continuous variables and discrete or binary variables. Such versatility guaranteed EA widespread applications in different areas. A review [34] estimated 79.1% of papers in this area to be of an application nature (Figure 2.1), meaning that multi-objective optimisation techniques, the majority of which use EA are used a lot in real-world scenarios. This view is further supported by analysing data of electronic publication databases. Referring to Figure 2.2 the majority of papers are in the applied areas like engineering, environmental sciences and biochemistry testifying to the high practical value of EA.

In the context of Figures 2.1 and 2.2, several important points have to be noted. Considering the data is 6 years apart, the figures provide a clear view on how the situation has evolved in the field of multi-objective meta-heuristics. The spread by application area has significantly changed and new application areas have appeared. The most obvious difference is the number of publications, which shows a healthy growth in multi-objective meta-heuristics applications.

Real-world optimisation problems often contain a degree of uncertainty. Example problems include gas turbine combustion rig burner optimisation [39] and machining of gradient materials [40, 41]. Several types of uncertainties have been categorised [21], including noise in the fitness function which usually comes in the form of measurement noise, modelling uncertainty and uncertainty imposed by the dynamic nature of the problem. Initial research has mainly focussed on single-

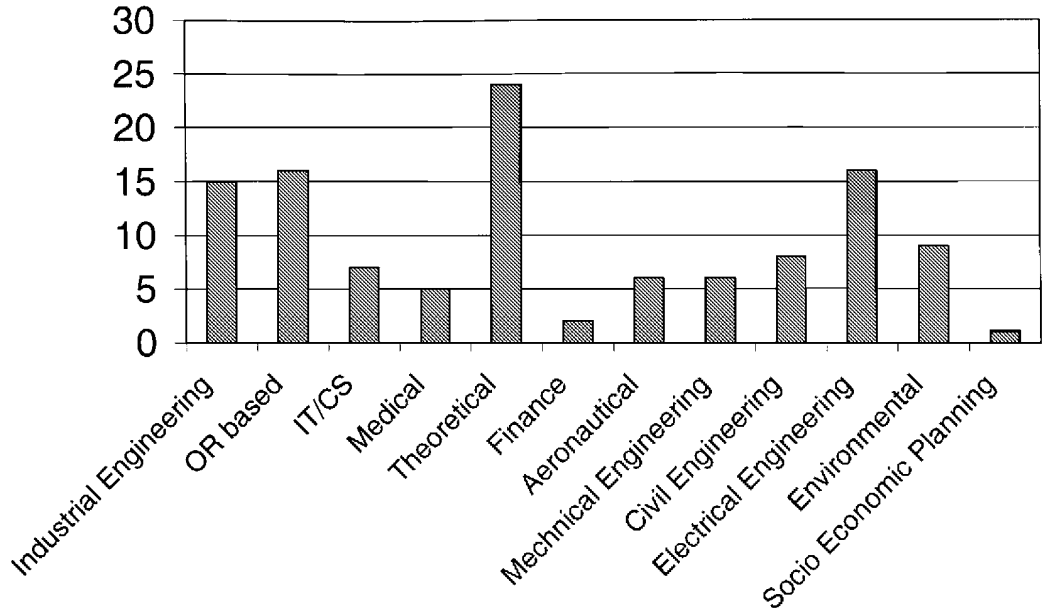


Figure 2.1: Publications in multi-objective optimisation by application area, as presented in [34]

objective optimisation tasks [42, 43, 44] with a few exceptions [45, 46, 2, 47, 39, 48].

The population-based nature of EA provides some inherent ability to work in noisy conditions [21]. However, it has been noted that noise impacts on the convergence rate of the EA [50]. Following comparative studies have indicated that EA enhanced with noise-handling techniques consistently perform better in the presence of noise [48, 51, 52]. Likewise, an observation that the EA inclination to converge over time reduces their ability to find emerging optima [42] has spurred a wave of research in adaptation techniques to dynamic multi-objective optimisation problems [53, 54, 55, 56].

An interesting tendency can be obtained by analysing the number of publications per year in an area. Referring to Figures 2.3 and 2.4, which show how the number of

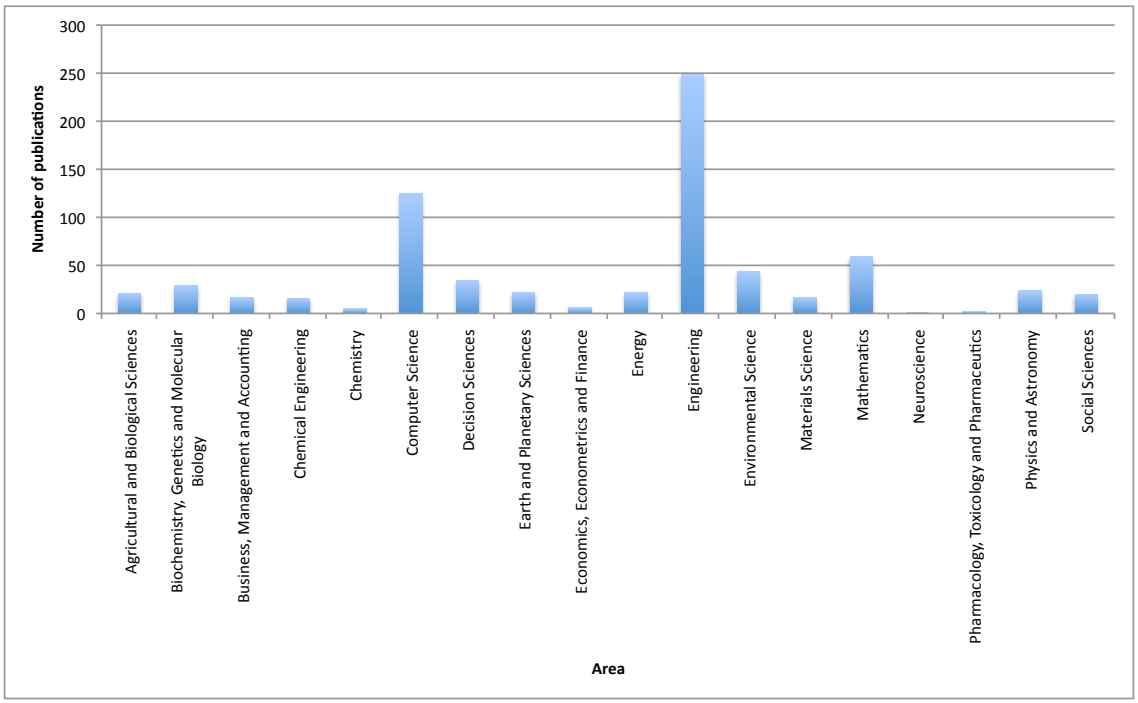


Figure 2.2: Publications in multi-objective optimisation by application area, using data obtained from [49], compiled in 2008.

publications changes year by year in the area of multi-objective optimisation, it can be observed that the number of publications steadily increases reaching respectively 31 and 6 papers in dynamic and noisy areas by 2007.

An important limitation of the majority of the preceding research in the area of uncertainty in multi-objective optimisation, which was first noted in [21], is a concern for only one of the uncertainty types. In contrast, real-world problems may exhibit different uncertainty kinds simultaneously, i.e. have moving optima and have noise in the objective function at the same time. To date, the research in the multi-criterion optimisation of the problems with multiple uncertainty types is limited, with certain exceptions [58, 46, 59].

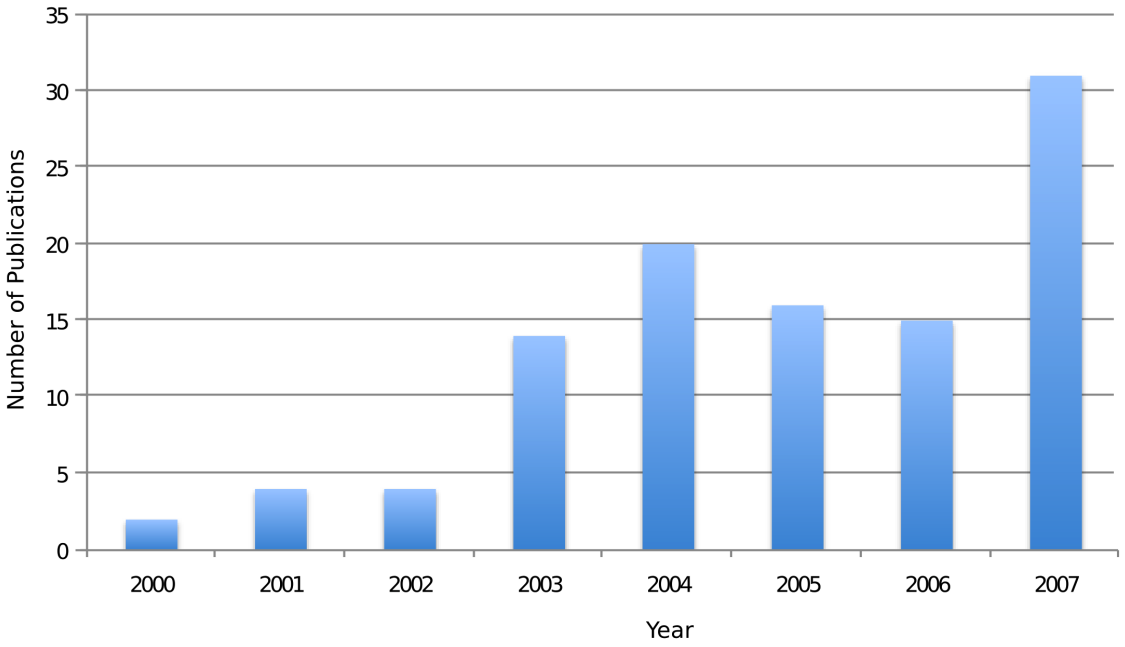


Figure 2.3: Publications in evolutionary multi-objective optimisation in dynamic environments, using data from [57], compiled in 2008.

2.2 Introduction to the key concepts

Over the last two decades, the area of MOO using EA became a well-established research field with its concepts and terminology becoming well defined. However, the same could not be said for optimisation in real-time and uncertain environments. Therefore, the key concepts such as real-time processing or uncertainty are clarified in this section.

2.2.1 Uncertainty

A majority of real-life optimisation problems are uncertain in some way or the other. Starting with early attempts to apply EA to uncertain problems, some authors have tried to coin a definition for uncertainty and make a taxonomy of different uncertainty types. Hughes [60] gives the following classification of uncertainty-related

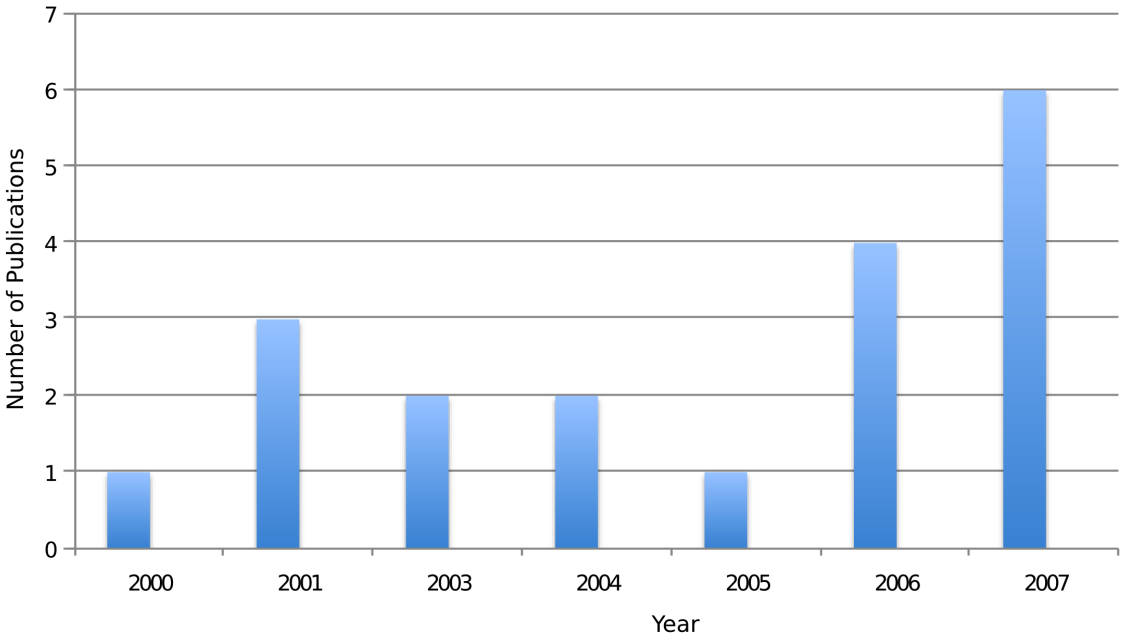


Figure 2.4: Publications in evolutionary multi-objective optimisation in noisy environments, using data from [57], compiled in 2008.

problems. In *noisy* problems two successive evaluations of a chromosome return two different objective vectors. *Uncertain* problems, do not produce different objective vectors on successive evaluations of an individual, but a comparison of objective vectors of two different individuals may give a wrong result because of the approximations and errors in the model. [61] uses the same definition of uncertainty, i.e., uncertainty caused by modelling errors. However, studies on optimisation of dynamic problems [42, 62, 63, 54, 56] have not referred to dynamic problems as being uncertain.

Generally, uncertainty means a lack of certainty about the current environmental state or the future state or the outcome of the process. In other words, the concept of uncertainty involves a set of states or outcomes each having a definite probability. A summary of the previous research and a general taxonomy of uncertainties in

evolutionary optimisation was presented by Jin and Branke [21]. According to Jin, there are four uncertainty types:

- noise in the fitness function,
- robustness,
- fitness approximation, and
- time-varying fitness functions.

Formally, the difference between robustness and noise as defined in [21] is that in the former case the additive noise is usually considered to be applied to the design variables and in the latter case the noise is added to the fitness function. Both uncertainty types are therefore considered in this thesis.

2.2.2 Noisy fitness function

Noise in the fitness function results in uncertain fitness values. The noise may come from different sources such as measurement errors, limited sensor range and randomised calculation parameters. Hughes [45] classifies noise in the fitness function into two distinctive types. *Type A noise* or *process noise* or *systematic noise* [64] is applied to the design variable vector X.

$$f_i' = f_i(X + N)$$

This noise type is analogous to the noise used to explore robust solutions as defined in [21]. *Type B noise* or *measurement noise* or *additive noise* [64] is applied to the result of the fitness function, i.e., the fitness value itself.

$$f_i' = f_i(X) + N$$

While it has been noted that search for robust solutions is similar to handling noise in the fitness function, there are a number of significant differences between

the two. In the case of noisy fitness function, it is not possible to avoid noise and the computed fitness values are not exact. In search for robust solutions it is generally assumed that the fitness function does not contain noise and the noise comes in the form of slight changes in design or environmental variables after the optimisation process has finished [21].

An important feature of *A-noise* is that it can change the topological characteristics of the problem [65, 66]. For example, the presence of *systematic noise* could add additional optima to the fitness landscape.

2.2.3 Approximate fitness function

Despite the fact that EA have enjoyed a great success in applications to the real-world problems, they have encountered a number of significant challenges. In a number of problems the fitness evaluation may be difficult and computationally expensive. To tackle those problems efficient fitness approximation has to be adopted [67, 13].

Approximated fitness function (or meta-model) allows us to estimate the fitness of an individual in a more computationally efficient way than the original fitness function. The fitness value of an individual i obtained using meta-model is:

$$f_i' = f_i + e_i$$

where e_i is an approximation error [67].

An important difference between the meta-models and the noise in the fitness functions is the effect of sampling on the fitness estimation. One of the approaches to deal with noise in the fitness function is to sample the fitness of an individual multiple times. However, the meta-model would always produce systematic error instead of stochastic noise and, therefore, it is impossible to be reduced by sampling. The approximation error could be addressed by using the original fitness function

together with the approximated model, as suggested by Jin [67]. The fitness function would look like:

$$f_i' = \begin{cases} f_i \\ f_i + e_i, \end{cases}$$

where e_i is an approximation error of the model [21].

2.2.4 Dynamic fitness landscape

Problems with dynamic fitness landscape (time-varying fitness) add a new dimension to the multi-objective optimisation. In this scenario, the fitness function depends on the time t [68], i.e.:

$$f_i' = f_{t,i}$$

The dynamic nature of the problem may be addressed by restarting a regular multi-objective EA when the change occurs. However, a more efficient approach would reuse information from optimisation of previous problem states to enable continuous tracking of moving optima without algorithm re-initialisation [40].

The important aspects of dynamic multi-objective optimisation are requirements for the availability of the optimisation results, namely the real-time and on-line performance of the algorithms. The real-time requirement implies fixed deadlines for the optimisation process to produce results, as in [69, 70]. The on-line optimisation runs in parallel to the process being optimised, providing direct control [70, 71, 72, 73].

2.2.5 Real-time constraint

The concept of real-time constraints in multi-objective optimisation means the existence of deadlines for the response of the optimisation algorithm. In this context, the performance of the multi-objective optimisation algorithm depends not only on the quality of the obtained non-dominated set, but also on the execution time of

the algorithm [47]. A fast multi-objective optimisation algorithm or an algorithm with high quality of the obtained Pareto front is not a real-time algorithm as long as there is no deadline set, although the qualities of speed and good Pareto front are usually desirable.

Generally two types of real-time constraints can be defined [74]. *Hard* real-time constraints should always be met, i.e., the solution becomes incorrect if the time required to get the solution extends past the deadline. By contrast, *soft* real-time constraints could sometimes be missed which results in degraded solution quality. A third type of real-time constraint is defined in [74], called *weakly hard* constraints. A system with a weakly hard real-time constraint could miss deadlines only if this happens in a predictable way.

2.2.6 On-line processing

According to [75], on-line refers to "the operation of a functional unit when under the direct control of the system with which it is associated" with a note that "on-line units may not be independently operated". In relation to the multi-objective optimisation, on-line means that the system being optimised is under the direct control of the optimisation algorithm and the decision maker, and that the decisions of the decision maker affect the environmental state of the system and, consequently, the optimisation algorithm.

2.3 Challenges to the Multi-Objective Evolutionary Algorithms

Uncertainties present a serious challenge to Multi-Objective Evolutionary Algorithms (MOEA). A general impact of noise can be twofold [45, 46]:

- a better solution is erroneously evaluated to be worse than an other and consequently lost, and

- a worse solution is erroneously evaluated to be better than an other and consequently survives.

It was demonstrated by various researchers that noise in the objective function significantly degrades the performance of state-of-the-art algorithms such as Non-dominated Sorting Genetic Algorithm-II (NSGA-II) and Strength Pareto Evolutionary algorithm 2 (SPEA2). In [52], it has been shown that conventional algorithms struggle in the presence of noise and their performance deteriorates as noise levels increase. An algorithm with noise-handling features exhibits more even performance across different noise levels. Moreover, both NSGA-II and SPEA2 modified with the noise-handling features improved performance in the presence of noise.

More specifically, additive noise in the fitness function slows down the convergence rate of an EA [50, 76], impacts elitist algorithms by producing false elite individuals [39], meaning that the progress obtained in the preceding generations could be lost and reducing the quality of the resulting non-dominated set [48, 77, 64]. The same could be said for *B-noise*, though it has additional characteristics of interacting with the topological properties of the objective space.

Evolutionary algorithms usually require a large number of fitness evaluations to reach an acceptable result. Published comparative studies of different MOEA usually include examples of tens of thousands and hundreds of thousands of function evaluations [78, 38, 79, 80]. However, in dealing with real-life problems the fitness evaluation may become very expensive or exact fitness function may be unknown. In such cases, a fitness approximation approach is used. An approximate fitness function is also used if additional fitness evaluations are required, for example when dealing with a noisy fitness function [67, 21].

One of the strong points of the evolutionary algorithms is their ability to quickly converge to a set of non-dominated solutions that are close to the true Pareto front. However, once the algorithm has converged to a set of optimal solutions, it is likely to lose its ability to continue searching for the new optima if they would emerge [56]. In fact, one of the main problems encountered by traditional static evolutionary algorithms on dynamic optimisation problem is the loss of diversity in the population [56].

According to [81], to succeed in dynamic problem optimisation, an algorithm should possess the following features:

- continuous adaptation,
- flexibility, and
- robustness.

Continuous adaptation refers to the ability of the algorithm to track the changing environment. Static MOEA have a tendency to stick with the current solution because of the loss of genetic diversity.

Flexibility refers to the ability of a decision maker to make 'responsible' decisions given an unknown future. For a dynamic system with an on-line optimisation algorithm and a decision maker, i.e., when the decisions based on the immediate outcome of the optimisation affect the future state of the system, such decisions should anticipate the future requirements of the system [82]. In other words, the optimisation algorithm should take into account not only the original objective functions of the problem, but also try to improve the flexibility of the solutions.

A solution to a dynamic optimisation problem usually needs to be robust against uncertainties, such as adaptation failures or implementation specifics. These uncertainties may be attributed to different reasons, such as a change in the environment occurred too fast for the algorithm to adapt or various implementation reasons, such

as feasibility and cost of adaptation, changing quality of materials and manufacturing tolerances [21].

2.4 Noise handling

It has been demonstrated that current state-of-the-art EA designed for handling deterministic problems encounter significant challenges when faced with uncertainty. To address these challenges, different strategies have been proposed in the literature. The earliest attempts to improve noise handling in EA were confined to single-objective optimisation [83, 50, 76]. Three main techniques were sizing of the population or so-called *implicit averaging*, resampling or *explicit averaging* and inheritance of re-scaled mutations.

An approach that uses search history to aid an individual's fitness estimation is presented in [84]. Calculated fitness samples are recorded as search history in memory and later used by a stochastic fitness estimation model to estimate fitness values of points of interest in the search space. An improvement of the approach is presented in [85]. These methods assume that noise has the same features in different regions of the search space.

2.4.1 Modified Pareto-ranking and selection

The methods of increasing population size and re-sampling an individual's fitness could be readily extended to multi-objective cases, however, there are some methods that are specific to MOO. Most MOEA employ a ranking and selection scheme based on the concept of Pareto dominance. Hughes [46] and Teich [61] modify the Pareto-ranking procedure to be based on the probability of dominance and replace deterministic selection with probabilistic selection. They present two MOEAs: Multi-Objective Probabilistic Selection Evolutionary Algorithm (MOPSEA) and Ex-

tended Strength Pareto Evolutionary Algorithm (ESPEA), that implement the proposed enhancements respectively. MOPSEA considers an individual's rank to be equal to the sum of probabilities of domination (Equation 2.1).

$$R_i = \sum_{j=1}^{n} P(F_j \succ F_i) \mid_{i \neq j} \tag{2.1}$$

A variant of the ESPEA with a different selection strategy is presented in [39]. To address the issues of unknown and variable noise, Fieldsend and Everson proposed their own variant of probabilistic Pareto-ranking technique [86]. A Bayesian algorithm was presented for estimating the variance of noise without sampling. Another definition of probabilistic dominance concept is presented in [87, 88] with an implementation based on Fast Pareto Genetic Algorithm (FastPGA) [89].

A modification to the NSGA-II algorithm that targeted to improve the original algorithm's performance in noisy conditions was presented in [48]. The proposed algorithm performs ranking of solutions in two steps. First, the unmodified NSGA-II ranking procedure is used to obtain rank 1 front of non-dominated individuals. Next, some adjacent dominated solutions are also included in the rank 1 front according to Equation 2.2:

$$R_j = 1 \quad \text{if} \mid f_k(i) - f_k(j) \mid < K\sqrt{\frac{v_k(i) + v_k(j)}{2}}, \tag{2.2}$$

where $i, j \in P$, $v_k(i)$ is the fitness variance of the k-th objective, $f_k(i)$ is the average fitness value of the k-th objective and K is the *Neighbourhood Restriction Factor* that controls how much of dominated solutions are included in the Pareto front. K would be dynamically adjusted to reduce sampling in later generations (Equation 2.3)

$$K = C \times (1 - e^{-\frac{\beta}{g}}), \tag{2.3}$$

where C and β are constants and g is current generation.

An extensive study of the impact of noise on multi-objective optimisation was performed in [52]. It was noted that the performance of MOEA is significantly

affected by high levels of noise. To counteract these effects, three non-sampling measures were introduced, namely, Experiential Learning Directed Perturbation (ELDP), Gene Adaptation Selection Strategy (GASS) and improved archive update methodology. The first measure, ELDP improves the traditional mutation operator by the use of ordered changes and operation in either genotype or phenotype space. ELDP accelerates convergence and simultaneously improves the algorithm's performance in noisy conditions by restricting unorganised changes to the individuals. The second measure, GASS, defines an operation in the phenotype space that adapts part of an individual's chromosome based on the model of the ideal population behaviour. Finally, an improved archive update methodology uses the probabilistic Pareto-domination concept. A comparative study presented within the same publication indicates that existing algorithms like SPEA2 [90] and NSGA-II [91] show better convergence and improved population density near the Pareto-front if modified with the ELDP and GASS.

2.4.2 Averaging

The approaches based on re-sampling were applied to Multi-Objective Problems (MOP) as well. Di-Pietro *et al.* [77] introduced the concept of a noisy landscape to describe problems with variable noise strength across the search space. Two original resampling techniques were introduced, namely *Standard Error Dynamic Resampling* (SEDR) and *m-Level Resampling* (mLR). SEDR operates by continuously sampling the fitness of an individual until the standard error of the mean fitness value is below a predefined threshold. At this point, the noise impact on an individual is considered to be reduced by the algorithm. Using the same threshold for all individuals in the population allows us to handle uneven noise levels. mLR operates on a different principle, using one of the predefined m sampling rates for different intervals of noise strength. The algorithm performs limited initial resampling

to estimate the noise strength. Afterwards, the algorithm does iterative resampling of an individual until the number of samples is equal to the predefined rate for the calculated standard deviation.

2.5 Fitness approximation

EA require high number of fitness evaluations to obtain optimal performance. However, the exact fitness function might be computationally expensive or unavailable. In these cases, when the evaluation of the exact fitness function becomes difficult the solution is to employ an approximate model of the fitness function [67].

Before implementing the fitness approximation in an EA, a number of issues have to be resolved, such as what model to use, in which part of the algorithm and how to use the model to obtain valid results [67]. In particular, the problem of obtaining an approximation of the fitness landscape with sufficient accuracy is especially important in MOO due to the difficulty of working in higher dimensional spaces. This problem can be addressed in two ways either by improving the quality of the model or by using the original fitness together with an approximated one. Most problems have the original function available, although its evaluation is expensive. The approximate fitness model can be used in conjunction with the original fitness function, which is termed as model management [92, 93] or evolution control [94].

2.5.1 Model usage rationale and integration techniques

Approximate models are used in almost every aspect of the evolutionary algorithm. In the aspect of fitness evaluation, the approximate models are used mainly for four reasons [67]. Firstly, the approximate models are used if each evaluation of the fitness function is computationally expensive. This especially applies to computational fluid dynamics problems [95, 96].

Second motivation to use fitness approximation is provided by the cases where an analytical fitness function is not known (and requires an external estimator) or is provided by means of experiments, i.e., the fitness evaluation may not be available when required [67, 21]. Approximate models enhance interactive EAs that excel in capturing the qualitative knowledge of the problem [97] by reducing the strain being put on the human estimator. In [98, 99], approximate models are used to estimate decision maker preferences, whereby interactive requests to the actual decision maker are triggered based on the utility function estimation. In turn, responses from the decision maker are used to update the model. A similar approach that uses Artificial Neural Network (ANN) to model designer preference is presented in [100].

Thirdly, fitness approximation is also used if the problem is noisy. In contrast to methods based on sampling the fitness of an individual multiple times, which is computationally expensive, the fitness is evaluated by averaging over similar individuals in the neighbourhood. To make the estimation more efficient, a statistical model that incorporates the history of search may be employed [84].

Finally, if the fitness landscape is rugged an approximate fitness model is used to smooth or regularise the solution. Matsui and Kosugi introduced the concept of regularisation in [101]. Unlike the traditional fitness evaluation, where the fitness of an individual depends solely on the individual's phenotype, the proposed approach takes into account phenotypes of the nearby individuals as well.

Approximate models may be used to guide algorithm initialisation, crossover, selection and mutation. For example, an algorithm presented in [80] uses 'informed' evolutionary operators. Such operators evaluate the results of e.g. mutation using the approximate model. In [102], a convergence-based criterion is introduced to choose between the genetical mutation and crossover and modelled offspring generation.

The exact model type used varies across implementations. Jin [67] presents four different model classes, namely *polynomial models*, *kriging models*, *ANN* and *Support Vector Machine (SVM)*. Examples of different model usage in MOO include the use of ANN [95, 96] and the use of SVM [103].

2.5.2 Evolution control

The use of approximate models for fitness evaluation significantly reduces computational requirements of EA. An algorithm with an implementation of an approximate model can provide solutions that are close to the global optimum and at the same time reduce the computational strain as much as possible. However, an approximate model might provide an optimum objective front that differs from the exact optimum fitness front [104]. Low fidelity of the model may be caused by sparse, incomplete or poorly distributed training data. A proposed solution to the problem is to use the approximate model together with the exact fitness data. A set of techniques to combine the exact fitness functions with an approximate model is termed as *evolution control* or *model management* [104, 67, 21].

There are two different types of evolution control, *individual-based* and *generation-based* [94]. Referring to Figure 2.5, *individual-based* evolution control evaluates a percentage of the population using the exact fitness functions each generation. An important consideration when using the individual-based evolution control is which individuals have to be evaluated using the exact fitness functions. Analysis of the literature reveals four predominant strategies to select the individuals [21].

- A naive way is to select random individuals for evaluation using the exact fitness functions [94].

- If the fitness landscape of the approximate model has false optima it is very likely for the optimisation algorithm to converge to a sub-optimal Pareto front.

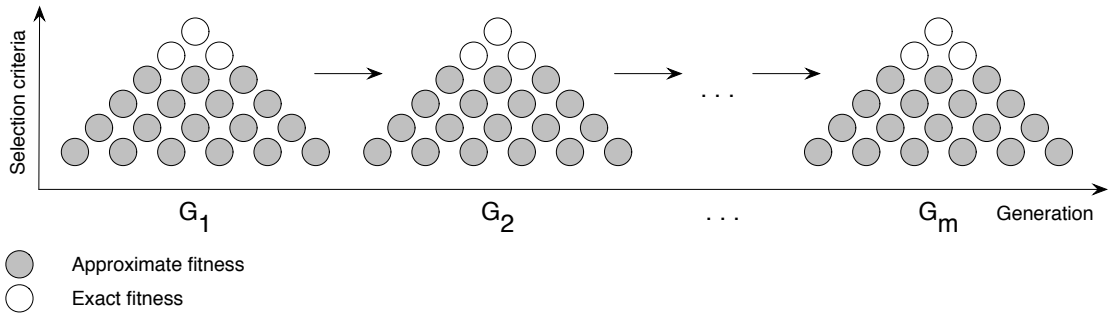


Figure 2.5: Individual-based evolution control

The best individuals in each generation are the most affected by the model inaccuracies. Therefore, an evolution control strategy might choose to evaluate the best individuals in each generation using the exact fitness data. It was found that this strategy performs better than the random strategy [94].

- Another selection strategy is to choose the most uncertain individuals. As noted in [21], selection of the most uncertain individuals is based on two reasons. Firstly, evaluations of the most uncertain members of the population using the exact fitness bring a reduction in the overall degree of uncertainty introduced by the approximate model. Secondly, there is a direct relation between the uncertainty of the individual and the degree of exploration of the surrounding space. Thus, the approach of evaluating the most uncertain individuals encourages exploration.

- Finally, an approach to the problem of evolution control uses so-called preselection [105, 106, 107]. For a regular (μ, λ) EA, exactly λ offspring individuals are created. Preselection methods create λ_{Pre} from μ parent individuals, which have to be subsequently evaluated using the approximate model to produce λ individuals that will be evaluated using the exact fitness functions.

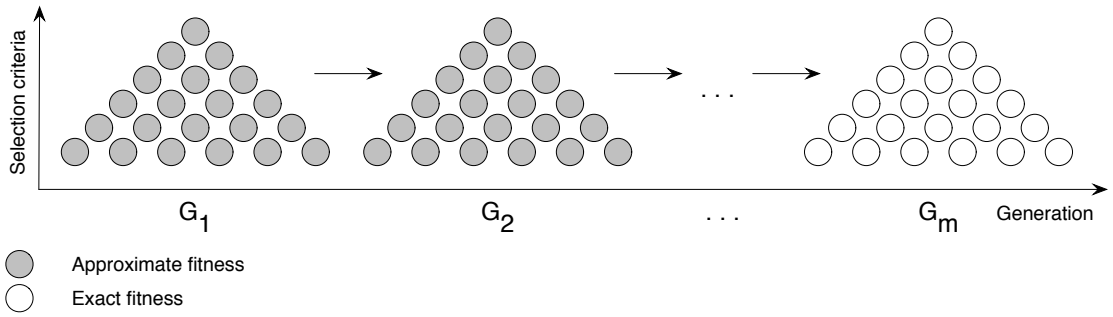


Figure 2.6: Generation-based evolution control

To determine which individuals are the best or the most uncertain, individual-based evolution control methods perform an initial evaluation of individuals using the approximate model. Afterwards a number m of individuals are re-evaluated using the exact fitness functions. Usually the number m stays constant throughout the generations, however, some methods employ an adaptive control of m [21].

Figure 2.6 shows the diagram of *generation-based* evolution control. Generation-based evolution control selects a generation and evaluates all individuals in the generation using the exact fitness data. Generally, each n-th generation is evaluated and the value of n remains constant during the optimisation run [94, 104]. This approach is not optimal, as the accuracy of the model may change during evolution. Improved approaches use adaptive frequency of generation-based evolution control, in particular, [92] uses model error to estimate the local accuracy of the approximate model and choose the control frequency.

2.6 Diversity preservation

EA designed for static problems tend to converge over time to some optima. This behaviour is inappropriate in the situations where the optimum might change its location or be replaced by another optimum. A substantial part of performance

enhancing techniques that address the convergence issue could be categorised into two species: i) management methods and ii) memory methods [53, 108, 54].

2.6.1 Management methods

The management methods employ different strategies for managing two primary tasks of a dynamic EA. Firstly, an EA should converge towards the current global optimum and, secondly, explore decision space for the location of future optima. The management involves choosing a right balance between the two competing functions. A portion of the methods developed in the past prefer to converge on the optimum solution waiting for the change to arrive. With the introduction of change the balance swings towards diversity, thus enabling the algorithm to find new optima. A typical representative example of this technique is the method of hypermutation, described in [109, 43]. The downside of these approaches is the inability to determine the necessary amount of diversity, because large amounts can result in a behaviour close to the full restart of the algorithm, while small amounts will be insufficient to detect the new position of the optimum or the new optima. Another disadvantage of these methods is that hypermutation replaces already found optimal solutions by random individuals, thus decreasing the value of previous generations and accordingly reducing the algorithms' ability to work with changes occurring in cyclic patterns.

Contrary to the previous set of management algorithms, there are some methods that choose to maintain diversity throughout the simulation run [110, 62, 42]. By avoiding convergence there is impact on memory and computational requirements for this group of methods.

Finally, a set of management algorithms uses a multi-objective approach, where the task of maintaining population diversity becomes a separate optimisation objective. An investigation presented in [56], uses NSGAII applied to single-objective

optimisation problems. The second objective is artificially constructed to enable better diversity preservation. Six possibilities to construct the artificial objective were examined, such as *time stamp*, whereby all individuals are time stamped using sequential numbers each time an individual is created. The objective is to minimise the time stamp value during selection phase, i.e., prefer older individuals. Older individuals are generally less converged than the newer ones, therefore minimisation of the timestamp value promotes selection of diverse solutions. The technique is based on the approaches presented in [111, 112] for maintaining diversity in single-objective optimisation of non-stationary functions. A time stamp may be substituted by assigning random numbers to the individuals and minimise these during selection. This technique enables some unfit individuals to survive environmental changes. Another approach takes the original fitness function and substitutes the minimisation task by maximisation and vice versa. Again, this slows down the convergence rate of an algorithm. Finally, an artificial objective could be constructed using an approach based on Euclidean distance, e.g., distance to the closest neighbour, average distance to all individuals and distance to the best individual in the population [56].

The last group of management algorithms have important limitations. The best-performing methods to construct the artificial objective are also the most computationally expensive. The approach was tested on single-objective optimisation problems only. It adds additional objectives to the optimisation problem [56]. In the case of multi-objective optimisation problems, it is desirable to minimise the number of objectives, as the cases with small number of objectives (typically two or three) are better studied and state-of-the-art MOEA such as NSGA-II tend to perform better on these. In addition, the computational effort required for the optimisation of problems with high number of objectives may be prohibitively high. Some methods, such as inversion of the fitness function, require adding a separate objective

for each of the original objectives of the problem, and therefore are not suitable multi-objective optimisation problems.

2.6.2 Memory methods

Memory methods choose to reuse information from past generations instead of introducing random mutations. These algorithms accumulate past fit solutions in the hope to use them as the problem changes [113, 114, 115]. Because of the use of past fit solutions, memory-based methods are particularly useful if the moving optimum repeatedly returns to previously occurred states. Diploid and polyploid chromosome structures are another examples of memory-based methods [116, 117, 118, 119].

Multi-population approaches can be considered as separate kind of memory approaches in the sense that each population represents a single adaptive memory cell. These approaches separate the population into self-contained groups each with its own function. A particular example of these approaches is self-organising scouts presented in [53, 120].

It was noted by [113] that the performance of memory-based algorithms depends on the diversity, therefore memory-based methods should be used in conjunction with other methods that preserve diversity. Moreover, it was stated that a diverse population does not guarantee its ability to adapt to the environment changes [121].

It can be argued that a given algorithm would perform better if it is given knowledge about the nature of the dynamic environment. According to [55, 122], the two principal characteristics of dynamic environments are the frequency and magnitude of change. A change with large magnitude could result in an environmental state that is completely unrelated to the previous one and therefore a complete restart of the optimisation algorithm might be advantageous over diversity increase. However, for changes with small and medium magnitudes, a dynamic optimisation algorithm would arguably yield a better result than a complete restart. A series of small

changes in environmental state may exhibit patterns in their evolution that may be predicted with a certain degree of confidence [122]

2.7 Decision making in dynamic Multi-Objective Optimisation

On-line control problems require a single control action at each time moment, i.e., the input vector is a function of time. At the same time, a multi-objective search develops a set of Pareto-optimal solutions instead of a single optimal point. Consequently, a decision maker is required to select an optimal control action from a Pareto set on-line [20].

Most existing studies of dynamic MOO do not focus on the problems of on-line decision making. Some research consider on-line decision making as problem specific [68]. Others assume a human as a decision maker. For example, Mehnen and Roy [40] employ a DMOEA for the problem of machining gradient materials, i.e., where the properties of the material constantly change, with no approach for on-line decision making. A later study by the same authors uses desirability functions to perform the multi-objective search in the desirability domain and model-based forecasting to aid the decision maker [41]. In [123], a dynamic clonal selection algorithm is proposed and applied to a task of designing a simple gearbox together with DNSGA-II variants from [20]. Again, the selection of a solution from the Pareto set is not described.

In the aforementioned cases, the final decision is thought to be performed by humans having unbounded substantive rationality [124]. The humans are assumed to have infinite information, aptitude and time to perform a decision, which is clearly far from reality, especially in very fast changing scenarios.

Recognising the need to substitute humans in the on-line decision making process, some studies have proposed alternative methods to select a solution from a Pareto set. In [20], a modified NSGA-II algorithm is applied to a problem of hydro-thermal power plant scheduling. The decision maker uses a utility function, which is a problem specific aggregation approach, to perform on-line selection of a solution. A weighted sum aggregation was used by Abe *et al.* [125] to stabilise a human model with a quadratic programming search and by Hughes *et al.* [72] to optimise a swarm trajectory with differential evolution. It can be argued that the aggregation approach has been borrowed from the static cases and *a priori* decision making approaches, e.g., Farina *et al.* [68], consider the on-line decision making issue to be similar to the static case. However, the *a posteriori* on-line decision making case is fundamentally different from the static cases in that it has a time dimension, and, compared to *a priori* decision making, it has a set of optimal alternatives to choose from.

Having a set of Pareto-optimal alternatives at the decision time allows the decision maker a great flexibility in the decisions, but simultaneously raises a number of interesting questions. Suppose the decision maker is comparing two alternatives, **A** and **B**. Would the decision maker's decision change if there is a Pareto-optimal alternative **C**? Some studies that employ heuristics to select a control action from a Pareto set are available, e.g., Hughes [47] *et al.* use a set of rules to select a single trajectory point out of a noisy Pareto set. Having a time dimension would allow us to create a decision maker based on the predicted system states or a history of past control actions.

2.8 Applications

A review of the application of MOEA to real-life uncertain MOP provides several important analysis criteria. In particular, it establishes the current snapshot of

the multi-objective evolutionary optimisation field, allowing to estimate the amount of research coverage in this particular area and if combined with the review of theoretical approaches, the overall 'age' of the research field, i.e., the balance between applications and theoretical studies.

Table 2.1 lists applications of MOEA to uncertain real-life problems. For each problem, specific uncertainty types are listed together with the uncertainty-handling measures that are used in the study.

Two different approaches to groundwater remediation problem use sampling [126] and probabilistic Pareto-ranking and averaging over identical individuals in current and previous generations [127] to handle B-noise in the objective function. The probabilistic Pareto-ranking is also used in [72] to handle B-noise. In [39, 128], A-noise is handled by a modified archiving procedure, ageing of elite individuals and their subsequent re-evaluation.

Both these approaches for handling the approximate fitness use ANN, however, the exact method is different between the applications. In [95, 96], the model is assumed to be of high fidelity, so no evolution control is used, i.e., fitness is evaluated using the model only. In contrast, the study in [129] uses generation-based evolution control with a fixed control frequency.

An algorithm that is using a random immigrants approach to handle dynamics was applied in [20] to the problem of hydro-thermal power scheduling. Two model-based algorithms that use forecasting [41, 40] and weighted fitness with dynamic weights assigned by the model [130] were applied to the problems of machining of materials with changing properties and military force allocation, respectively.

Looking at Table 2.1, it is clear that MOEA have found a diverse range of applications in uncertain domains. However, the absolute number of applications is low in comparison to single-objective or deterministic multi-objective cases.

A prominent way to handle uncertain MOP is to present the problem in a different way that simplifies or eliminates the aspect of uncertainty. In [132], a task of dynamic optimal trajectory planning of robot manipulators is formulated to allow off-line (static) processing. Uncertain (noisy) problems can be transformed to deterministic MOP using non-linear interval number programming methods [133].

2.9 Research gap

The ability to handle various uncertainties can dramatically expand the area of EA applications. At the same time, the field of uncertainty handling in MOO has not yet received sufficient coverage, partially attributed to the fact that accounting even for a single uncertainty type is algorithmically difficult and requires significant computational resources.

Most existing publications describe handling of a single uncertainty type. However, real-life problems often include several uncertainties simultaneously. In particular, the missile guidance problem defined in [2] is clearly both noisy and dynamic. Consequently, a very important issue that should be addressed by future research is the handling of multiple uncertainties simultaneously, e.g., an optimisation algorithm designed to work with noisy data in a dynamic environment. Here, the algorithm might use an approximate fitness landscape model as well.

Certain applications in military and engineering areas require the optimisation and the decision making algorithm to be run on-line in real time. This presents two issues. Firstly, the real-time performance of MOEA should be extensively studied and new ways to radically improve execution speed have to be found. Secondly, the mechanisms of decision making in uncertain environments have not been extensively studied so far. The issue of decision making currently is disconnected from the optimisation algorithms themselves in the evolutionary literature. However, a successful

application to an on-line control problem requires on-line decision making.

2.10 Summary

This chapter attempts to review the latest achievements in the area of multi-objective evolutionary optimisation in the presence of uncertainty. Referring to Section 2.1, it can be said that the existing research in MOO for uncertain environments is insufficient. As a prerequisite for this study, the key concepts in the subject area were identified and described. In particular, it was noted that B-noise is a close analogue of the search for robust solutions. It was demonstrated that current state-of-the-art evolutionary multi-objective algorithms face significant challenges in uncertain environments and consequently require help of special techniques. Existing approaches to handling noise in the objective function, fitness approximation and dynamic fitness landscapes in multi-objective optimisation were categorised and reviewed. Special attention was paid to the decision making in contemporary dynamic multi-objective optimisation research. In addition, some of the existing applications of MOEA to uncertain problems were analysed.

Several possible topics for future research in the area have been suggested. The research topics cover issues of handling different uncertainty types simultaneously, and on-line and real-time performance of MOEA in applications of MOEA to uncertain problems with on-line control. A further research topic might be comparative study of MOEA in dynamic environments, real-time performance studies of MOEA (specifically handling of deadlines), on-line process control by means of evolutionary algorithm and decision-maker, and handling of dynamics and noise within a single algorithm. These research topics can enhance the capacity of MOEA to perform optimisation in approximate, noisy and dynamic environments, therefore greatly increasing the number of applications to real-life problems and quality of obtained

results.

	Problem	Uncertainties	Measures	Notes
[127]	groundwater remediation	B-noise	probabilistic Pareto-ranking, extended averaging over identical individuals in current and past generations	uses NSGAII [91] as the base algorithm
[41, 40]	machining of gradient materials	dynamic properties of material	removes elitism from base algorithm, uses model-based forecasting	NSGA-II-based
[126]	groundwater remediation	B-noise	sampling	
[130]	military force allocation for war simulations	dynamics	weighted fitness with model-based dynamic weights	
[20]	hydro-thermal power scheduling	dynamics	random immigrants, mutation after change is detected	NSGA-II-based
[129]	design of long-span trusses	approximation of user preference	ANN with fixed generation-based evolution control	
[95, 96]	ventilation system design and operation in office environment	surrogate model because of expensive original fitness	ANN with no evolution control	uses WSM, based on framework from [131]
[39, 128]	optimisation of a burner in a gas turbine combustion test	A-noise, outliers	domination dependent lifetime (ageing), re-evaluation of expired solutions, modified archiving	SPEA-based [78]
[72]	multiple missile guidance	B-noise, dynamics	probabilistic Pareto-ranking	

Table 2.1: MOEA applications to real-life problems with uncertainty.

Chapter 3

Dynamic control problems

A control problem (see Figure 1.2 on page 7) is a problem of keeping an output of a dynamic system as close to the desired output as possible. Control problems are studied using control theory, an interdisciplinary branch of science on the border between engineering and mathematics. The different methods designed to solve control problems can be categorised into two groups, 'classic' approaches and 'modern' approaches. The classic approaches, which use system equations transformed into the frequency domain, were invented after the modern methods[1], which use state space equations to describe the system. However, the approach names stem from an increasing number of applications and attention gained by the state-space approaches, contrasted by the gradual decrease of use in the classic methods [136].

An ideal optimal controller (not to be confused with *optimal controller*, that is 'optimal' only in terms of the model that is used to design it) constantly solves a dynamic optimisation problem to minimise the *error*, i.e., the difference between the actual and desired outputs. For example, Model Predictive Control (MPC), a widely

[1]Publications in the middle of XIX century used analysis of motion of the dynamical systems by means of differential equations in time domain, e.g., [134] and [135], although it was not until early 1960s that Robert Kalman introduced the concept of 'state'.

employed modern control method, uses a scalar-valued cost function, effectively running a single-objective optimisation [137, 11]. Since the underlying optimisation problem is usually multi-objective, MPC employs an *a priori* decision maker to find a trade-off between objectives. It will be shown in Chapter 5 that the *a priori* approach has a number of shortcomings when compared to *a posteriori* decision making. This chapter details a multi-objective MPC framework with *a posteriori* decision making to be used for dynamic control problems. The multi-objective search is performed using a Dynamic Multi-Objective Evolutionary Algorithms (DMOEA).

3.1 Optimisation challenges

One of the major tasks in control engineering is to find an acceptable compromise between the complexity of the problem and its formal model used to design a controller. This compromise is required because real-life control problems present a number of challenges, that are difficult to solve using existing controller design methods, especially if a particular problem has several different challenges.

Some control problems are of the Multiple-Input and Multiple-Output (MIMO) variety, i.e., have multiple reference signals with corresponding measured outputs. The performance objectives of such systems may be conflicting and incommensurable. For nonlinear problems, the internal state of the plant changes frequently in a nonlinear manner. These features are difficult and computationally expensive to model, therefore the internal process model used by MPC controller is always different to the real process. The differences between the process model and real process, coupled with measurement noise and limited sensor range, make dynamic control problems uncertain.

Another feature of control problems that makes them difficult is the real-time constraint [47]. For example, an inverted pendulum described in Chapter 6 requires

the controller to make decisions every 5 milliseconds, otherwise it is very difficult to keep this particular pendulum stable. The rapid update requirements may explain why existing MPC methods use single-objective search, as it requires less processing resources. Also, MOEAs produce a whole set of alternative solutions, which would require a dedicated decision maker to produce a single Pareto-optimal control action at each time instant.

3.2 Evolutionary multi-objective model predictive control

Traditional MPC performs an iterative, finite horizon optimisation of a plant model using an internal model of the dynamic process, history of past control moves and an optimisation cost function [11]. A dynamic plant model is in the form described by Equation 3.1, where x_k is the k-th system state, u_k is the system input at k and $f_k(x_k, u_k)$ is the function that maps the system state and input to the next system state.

$$x_{k+1} = f_k(x_k, u_k), \quad x_k, u_k, f_k(x_k, u_k) \in \mathbb{R} \tag{3.1}$$

Referring to Figure 3.1, an open-loop input $\bar{u}$ over control horizon T_c is optimised such that the predicted system state $\bar{x}$ would reach the set point over prediction horizon T_p. Then, the open-loop input $\bar{u}$ is applied for the time δ and the optimisation procedure repeats itself.

The approach presented in Figure 3.1 makes an important assumption about the plant, namely, the algorithm optimises a scalar-valued cost function that should be set in advance. Finding a good cost function is a key element in MPC applications. A cost function, which is usually an aggregation function, depends on the shape of the problem's Pareto front, which makes it difficult to be set in advance. Also, stability of the obtained solutions may be affected as a result.

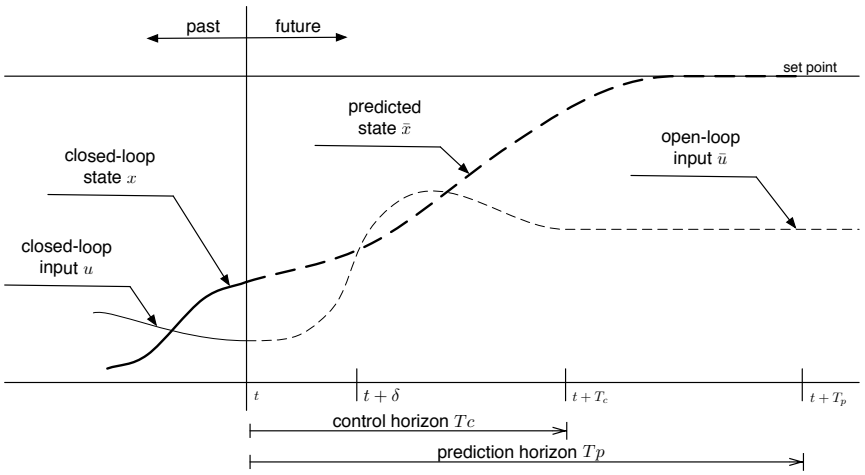


Figure 3.1: Model predictive control (adapted from [11])

To address the issues stemming from 'early' decision making in traditional MPC, it is proposed to make the MPC optimisation stage multi-objective and use an *a posteriori* decision maker to obtain the Pareto-optimal control action. The iterative multi-objective search is performed by a DMOEA. Referring to Figure 3.2, the measured output from the sensor is used by a DMOEA to continuously track the optimal control surface for the plant. A dynamic decision maker is used at regular intervals to select a control strategy from the set of Pareto-optimal solutions. The Pareto-optimal control strategy is used by the control conditioner to generate the control effort $w(t)$.

EA have been applied to MPC before, e.g., [138, 139]. For example, Yan *et al.* [140] use an EA in the MPC loop to optimise battery charging in vehicles. However, the EA used is static and single objective, and the control effort is encoded in the chromosome directly.

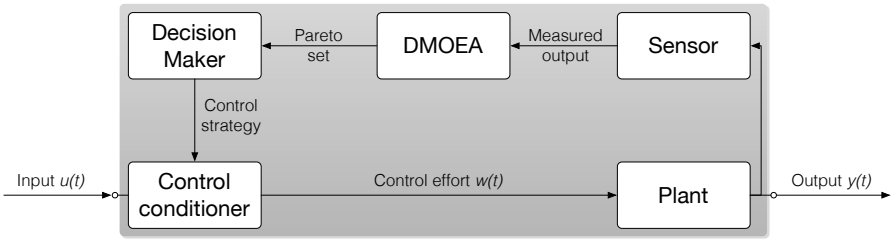


Figure 3.2: Evolutionary multi-objective model predictive control

3.3 Control conditioner

One of the problems faced by DMOEA in real life applications is their comparatively high computational cost caused by a large number of fitness evaluations required for their larger population and slower convergence than single-objective EA. In contrast, some plants require very fast response from the controller in order to remain stable. To address the conflict between long DMOEA iteration times and required fast response, the proposed approach encodes an entire control strategy within an individual rather than a single control effort value. A similar approach is used in [2, 71], where the missile trajectory is encoded in a series of waypoints, acting as the control strategy for the proportional guidance algorithm and in [73], where the chromosome describes a complete sequence of moves in the game of checkers.

A control strategy is decoded by the control conditioner and used to generate intermediate control efforts. Figure 3.3 shows timing diagrams of EMO MPC evolving control effort values directly (**I**) and using a control conditioner to produce the intermediate control effort values (**II**). The algorithm (**I**) is able to change the control effort in δ intervals, where δ is limited by the computational requirements of the DMOEA and dynamic decision maker. In contrast, the algorithm (**II**) is able to make changes to the control effort independently of the processing time required by the DMOEA. Starting from the initial state, a new Pareto-optimal solution is selected every *control interval* δ_C, i.e., the control strategy is changed in δ_C incre-

ments. The changes in the control effort happen every *adjustment interval* $\delta_A \leq \delta_C$.

Two different control conditioners are proposed in this thesis. A PID control conditioner uses a three-parameter control strategy for proportional, integral and derivative gains. The EMO MPC algorithm evolves and dynamically adjusts the PID controller gains. Another example could be a control conditioner that uses B-splines to calculate actual control effort from a control strategy encoded by spline control points. The proposed implementation uses a B-spline of third order $n = 3$ to generate a control strategy for a control interval. The chromosome encoding uses three genes to encode the Y coordinates of the control points. The X coordinates are evenly spaced to cover the whole control interval.

3.4 Variable prediction horizon

MPC performs open-loop input optimisation over a finite horizon T_c, which is shifted forward after each iteration of the algorithm. For this reason, MPC is also called *receding horizon* control. The length of the prediction horizon is an important parameter of the algorithm [141]. Ideally, the prediction horizon should be equal to the system's lifetime. However, for practical reasons this is generally not possible, so various approaches exist to shorten the prediction horizon without affecting stability and optimality of the system [137].

In the context of uncertainty that is characteristic to the real-life control problems, the choice of the prediction horizon length becomes an important issue. A longer prediction horizon makes the system more optimal by avoiding the local extrema on the Pareto surface and ensuring better closed-loop stability. At the same time, making the prediction horizon longer increases uncertainty in the objective functions used to estimate the performance of solutions. To address this issue, a

modification to MPC and a corresponding MOEA chromosome structure is suggested to enable a variable-length prediction horizon.

The proposed modification applies to the way a control strategy is encoded within a chromosome. As used in [73], the control horizon T_c is split into homogeneous intervals $T_{ci}, i \in [1, n]$, each containing a portion of the overall control strategy (see Figure 3.5). The intervals T_{ci} are sequentially encoded in the chromosome with the leftmost gene corresponding to T_{c1} and the rightmost gene corresponding to T_{cn}. In T_{ci} time intervals, the EMO MPC shifts the genes in the chromosome to the left by one, thus, for example, T_{c2} becomes T_{c1}, etc. The rightmost gene that corresponds to T_{cn} is initialised with a random value. Figure 3.4 shows an example of the proposed chromosome structure and cyclic gene shift within a chromosome, assuming a three-parameter control strategy, i.e., each control interval T_{ci} is encoded with three genes. With this encoding, the uncertainty in gene values increases from left to right. Each control horizon interval is separately evaluated using an objective function vector $\vec{f_i}$. The new approach modifies the existing method by incorporating a factor that captures the increasing uncertainty in the right-hand-side of the chromosome structure. The overall fitness function vector is now obtained by piecewise aggregation using Equation 3.2.

$$\vec{f} = \sum_{i=1}^{n} \vec{f_i} e^{-\lambda i}, \tag{3.2}$$

where $\vec{f_i}$ is a fitness vector corresponding to the control horizon interval T_{ci} and λ is the uncertainty increase constant, specific to a particular problem. The effect of such a chromosome structure is twofold. Firstly, the gene ordering by their uncertainty enables the more certain (left) sections of the control horizon to have greater impact on the overall fitness than the more uncertain (right) sections. In essence, this gives a variable prediction horizon that is automatically limited by the level of uncertainty in the chromosome (controlled by λ). Secondly, the gene

shifting supplies the fitness functions that have the most impact on the overall fitness with partially pre-converged genes, which improves the convergence speed of the algorithm.

3.5 Summary

This chapter proposes to modify a traditional MPC by incorporating an evolutionary multi-objective search and *a posteriori decision maker*. Potential on-line performance issues are addressed by evolving a control strategy rather then control effort values directly. The control strategy is decoded by a control conditioner for fast on-line adjustments of the control effort. Two control conditioners, namely, PID and B-spline based control conditioner are proposed. A method to avoid horizon issues characteristic to MPC is suggested, which also captures the effect of uncertainty in future control actions. It is important to note that the process will sill have a maximum prediction horizon length dictated by the maximum chromosome length. However, the effective length of the prediction horizon is dynamically adjusted by the amount of uncertainty in the gene values. An effective length of the prediction horizon can be estimated by comparing the performance of EMO MPC controllers with different values of λ.

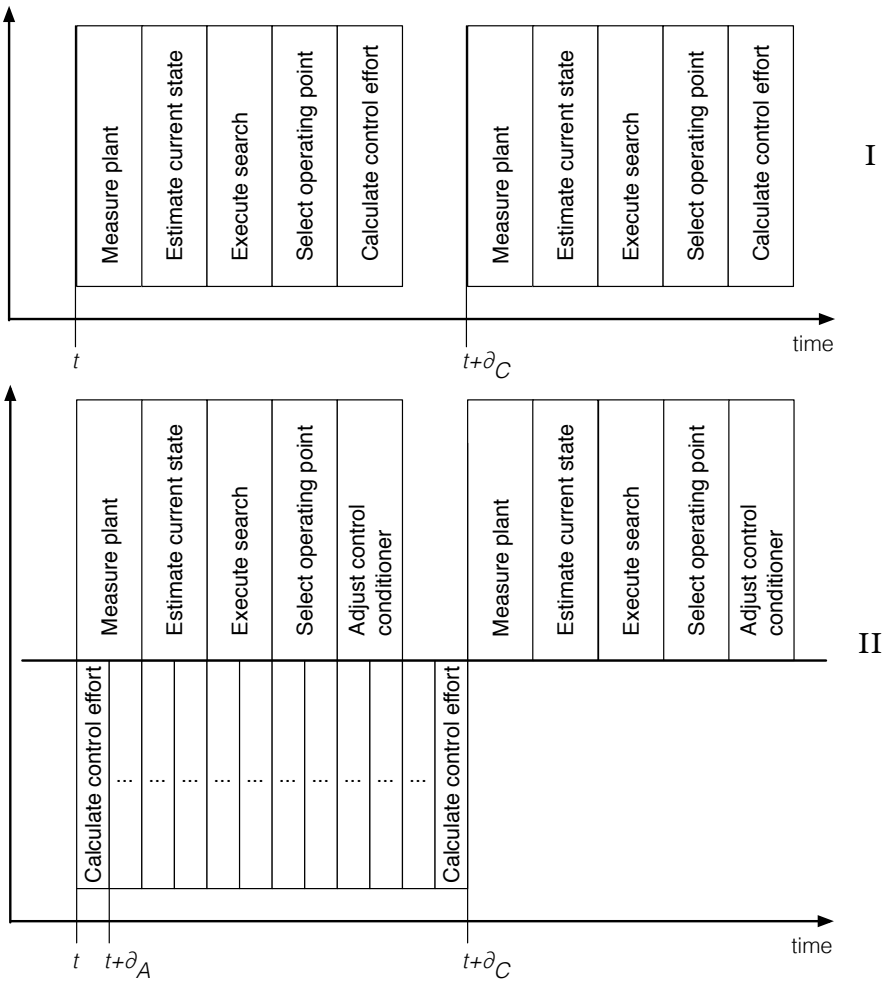


Figure 3.3: Timing diagram of EMO MPC. **I** – control effort is evolved directly by the DMOEA. **II** – control effort is provided by a control conditioner that decodes a control strategy evolved by the DMOEA.

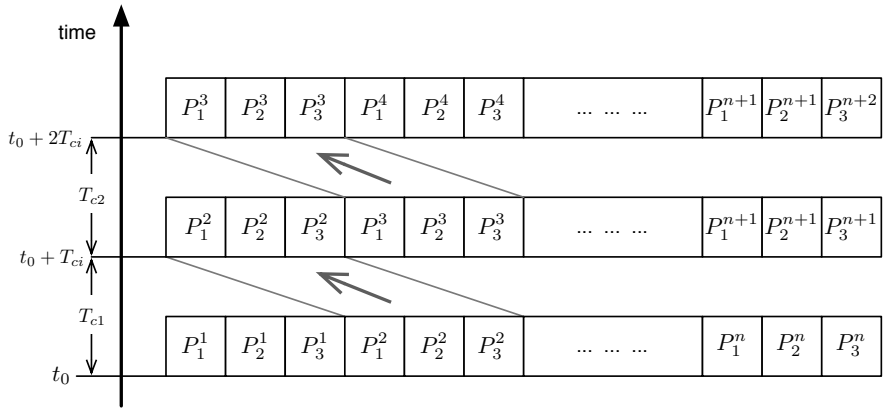


Figure 3.4: Chromosome structure for a three-parameter control strategy. P_j^{n+1} and P_j^{n+2} are initialised with random values, before the optimisation process is re-applied to the new control point sequence.

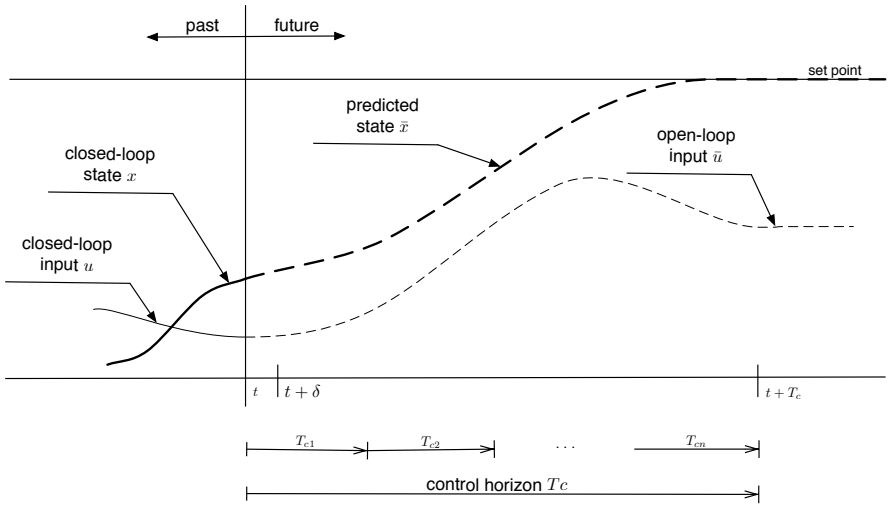


Figure 3.5: Model predictive control with variable length prediction horizon

Chapter 4

Multi-objective evolutionary algorithms for dynamic control problems

A typical control problem presents the evolutionary optimiser with a time-varying search landscape. In particular, the problem could change only in the decision space (PS_{true} changes but PF_{true} does not) or only in the objective space (PF_{true} changes but PS_{true} does not) or simultaneously in the decision and objective spaces (both PS_{true} and PF_{true} change) [68] (see Figure 1.3 for the definition of decision and objective spaces). From the optimiser's point of view, two important characteristics of a change are its frequency and magnitude [20, 122].

The frequency of change μ describes how often the change happens and, therefore, how much time the optimiser has got to converge to a new set point [122]. The frequency of change is expressed in $^1/_{generations}$. The magnitude of change A describes the distance between the previous set point and the current one. It is related to the 'effort' required from a DMOEA to find the new optima. The product of frequency of change with the magnitude is the rate of change $\rho = A\mu$. The rate of

change has direct impact on an evolutionary algorithm's ability to track the moving optima. The higher the rate of change, the more difficult it is to track the PF_{true}.

From Section 2.6, it is evident that MOEAs' ability to adapt to changes is considered to be dependent also on its state of convergence. The implicit assumption is that the convergence level is inversely proportional to the algorithm's ability to adapt to changes. The argument that the special diversity preservation measures are necessary for the dynamic evolutionary algorithm requires verification, as usually those measures require additional processing time, which is limited in the dynamic applications.

Historically, MOEA were applied first to the problems formulated as static over time. Consequently, static MOEA enjoyed the most attention from the scientific community. Adapting regular MOEA to dynamic problems presents a number of issues, most important of them being elitism. Elitism was designed to improve performance of evolutionary algorithms by ensuring that the fittest individuals survive into the future generations without any modifications [38]. In the dynamic case, the fitness information of the elite individuals would degrade over time if no special measures are taken. Current solutions that use elitism either recalculate their fitness in each generation or employ recalculate-on-change approach [20]. The former being computationally expensive, recalculate-on-change is the preferred approach to deal with the elitism issue, but the detection of change could become difficult, especially if the fitness value of an individual is uncertain.

Another challenge to the MOEA in the on-line control applications is the reliability of the obtained results. Evolutionary techniques are stochastic, i.e., the outcome of an optimisation run can not be precisely predicted. The variations in the EA performance over multiple optimisation runs needs to be analysed. To summarise, the following issues must be addressed for a successful application of a MOEA in a dynamic search.

- The balance between convergence and population diversity in dynamic cases. Are special diversity preservation measures necessary?
- Fitness degradation of elite individuals over time.
- Variability of results obtained by the evolutionary search.

4.1 Proposed modifications to NSGA-II

To enable handling of dynamic optimisation problems, several changes are introduced in the NSGA-II algorithm. One way of addressing the fitness degradation of elite individuals is to make a test to identify when a change happens. A test of a random sample from the population similar to that described in [20] is performed each generation. If any of the objective values or the constraints are changed, a change in the problem is assumed. In contrast to [20], all parent solutions are reevaluated before the selection procedure. This approach allows us to generate offspring solutions from the population according to the current state of the problem. The algorithm with this modification is denoted as dNSGAII-A.

The second variant of the algorithm, dNSGAII-B, employs temporary population size increase when a change is detected. The additional individuals are generated using hypermutation from the parent population. A larger population of more diverse solutions can help to overcome the sudden loss of fitness in the event of fast change.

One common problem shared by dNSGAII-A and dNSGAII-B algorithms is the vulnerability of the change detection test to noise found in most of the real life problems. A third variant of the dynamic MOEA, dNSGAII-C is designed to work in noisy environments. dNSGAII-C does not contain any tests to check for changes in the problem. Instead, a sample of the population is reevaluated in each generation and merged with the offspring instead of the whole parent population. This

approach reduces the number of potentially expensive fitness evaluations and simultaneously lowers the degree of elitism in the algorithm. One drawback of dNSGAII-C is potentially slower convergence time compared to the variants with higher degree of elitism.

4.2 Test problem

In [68], Farina *et al.* proposed a set of test problems for dynamic multi-objective optimisers. A Type II problem (both PS_{true} and PF_{true} change), which is based on FDA2 problem from [68], is detailed in Equation 4.1. The problem is selected because it represents a typical control problem where both decision variables and performance vector change over time. In this problem, the Pareto front changes from concave to convex with time. A true Pareto front can be obtained with $\forall x \in X_{II} = 0, \quad \forall x \in X_{III} = H(t)$. Figure 4.1 shows all possible PF_{true} for $n_\tau = 5, \quad \tau_T = 10$, i.e., each 10 generations the problem changes into one of $n_\tau + 1$ states, making the full period of the problem equal to 100 generations.

$$\begin{cases} f_1(X_I) = x_1 \\ g(X_{II}) = 1 + \sum\limits_{x_i \in X_{II}} x_i^2 \\ h(X_{III}, f_1, g) = 1 - (\frac{f_1}{g})^{4(H(t) + \sum\limits_{x_i \in X_{III}} (x_i - \frac{H(t)}{2})^2)} \\ H(t) = 0.75 + 0.7\cos(\pi t), \quad t = \frac{1}{n_t}\lfloor \frac{\tau}{\tau_T} \rfloor \\ f_2 = g \cdot h \\ X_I \in [0,1], \quad X_{II}, X_{III} \in [-1,1] \end{cases} \tag{4.1}$$

In this study, this problem is initialised with the following default parameters: $|X_{II}| = 4$, $|X_{III}| = 3$, $\tau_T = 10$, $n_\tau = 5$.

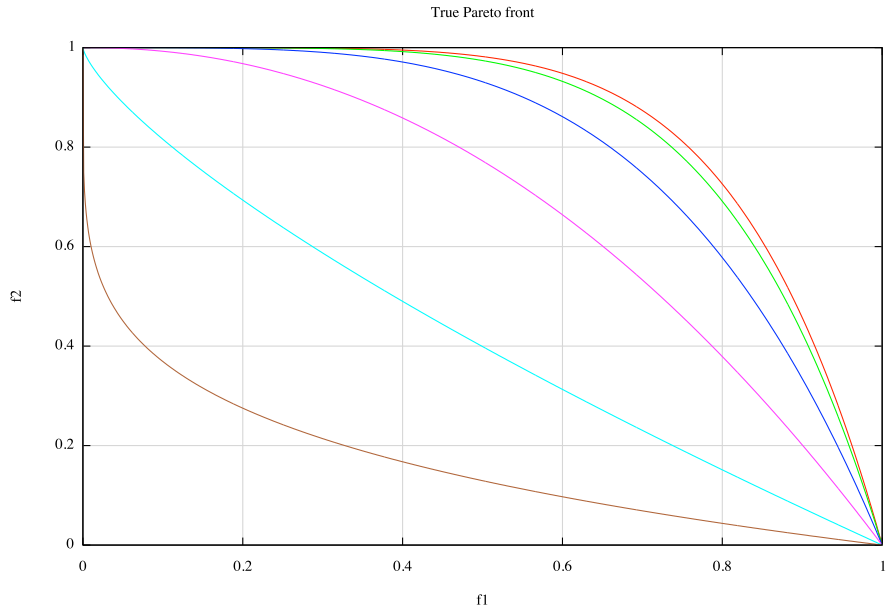


Figure 4.1: True Pareto fronts at different time steps. $n_\tau = 5, \quad \tau_T = 10$.

4.3 Static pre-execution

MOEA search in a problem that is 'frozen' (i.e., does not change during optimisation) shows how fast an evolutionary algorithm can pick the PF_{true} after a sudden large change in the problem that leads to a complete loss of tracking. Also, it allows us to estimate the initial convergence time of an evolutionary algorithm. Figure 4.2 shows example Pareto fronts (diamonds) obtained after 10 generations overlaid on top of the true Pareto fronts (solid lines) for different problem states ($n_\tau = 5, \quad \tau_T = 10$). Example fronts obtained after 30 and 50 generations are presented in Figure 4.3 and Figure 4.4, respectively.

For this problem the performance of MOEA depends on the problem state, with the non-convex Pareto fronts being more difficult to converge to. After 10 generations the algorithm is in an unconverged state, with poor spread of solutions, in particular on the non-convex fronts. After 30 generations, the algorithm has gener-

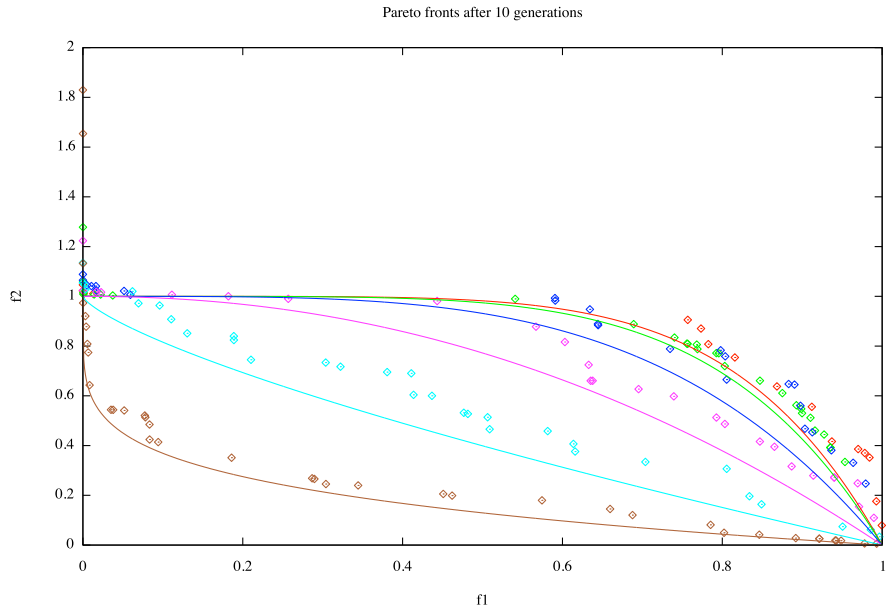


Figure 4.2: Example Pareto fronts at different time steps after 10 generations. MOEA results are shown with diamonds, true Pareto fronts are shown with solid lines.

ally converged close to PF_{true}. The major difference between 30 and 50 generations is a better spread of solutions obtained on the concave fronts.

To analyse the convergence of the evolutionary algorithm, a hypervolume-based convergence metric c_{HV} is defined in Equation 4.2 as the ratio of hypervolumes of obtained Pareto front to the true Pareto front. Figure 4.5 shows the mean value of c_{HV} for time steps 0 and 5 (the outermost concave and the outermost convex fronts in Figure 4.1), with confidence level of 95%. The results support earlier observation that for this particular problem the convergence speed of the optimiser running from start depends on the initial problem state. Considering $c_{HV} = 0.8$[1]

[1]The exact value of the c_{HV} threshold is debatable. In [20] 0.94 is used as a threshold for DMOEA comparison, however, no justification is given why the value is selected. In contrast, the value used in this work roughly corresponds to the border between initial convergence and steady

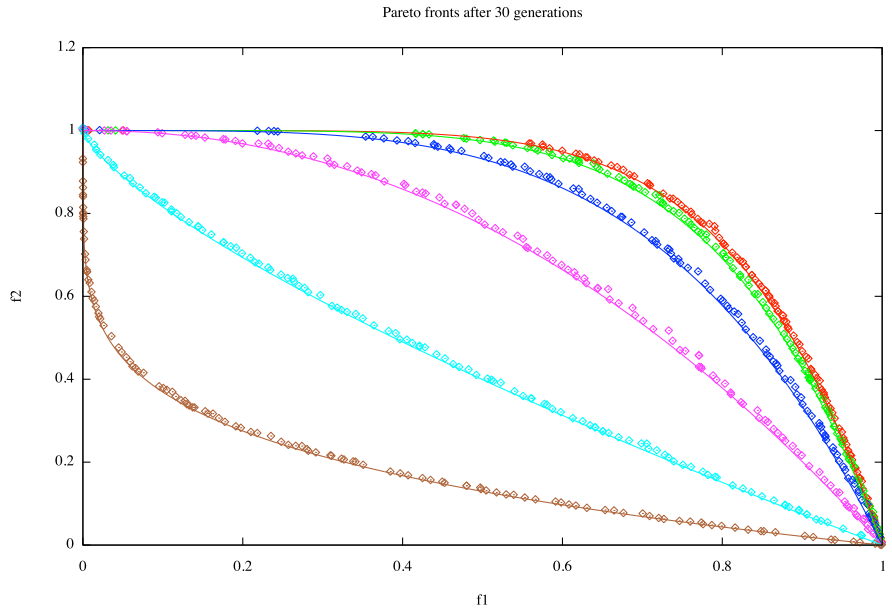


Figure 4.3: Example Pareto fronts at different time steps after 30 generations. MOEA results are shown with diamonds, true Pareto fronts are shown with solid lines.

as the threshold performance, it may require up to 20 generations for MOEA to attain this performance. Note that the length of the confidence interval decreases as the evolutionary algorithm converges, suggesting that the variability of the results decreases over time. Therefore a MOEA, given sufficient time to converge would find the same PF across different runs. In order to obtain more reliable results from a MOEA-infused controller, one should allow it to converge if the MOEA is restarted after each change. That makes the controller slow, but reliable.

$$c_{HV} = \frac{\text{hv}\, PF}{\text{hv}\, PF_{true}} \tag{4.2}$$

state performance of the algorithm, marked by a major decrease of relative variability of results (see Figure 4.8).

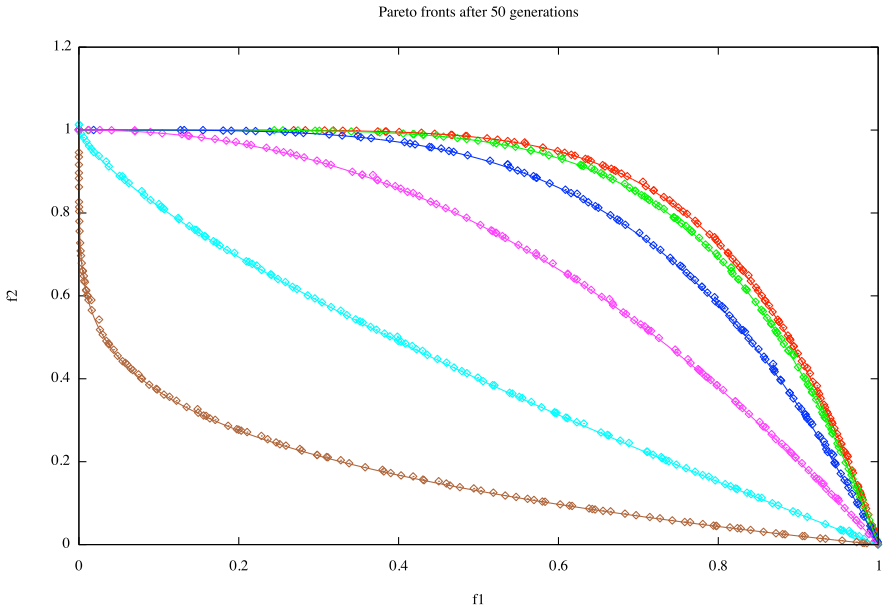
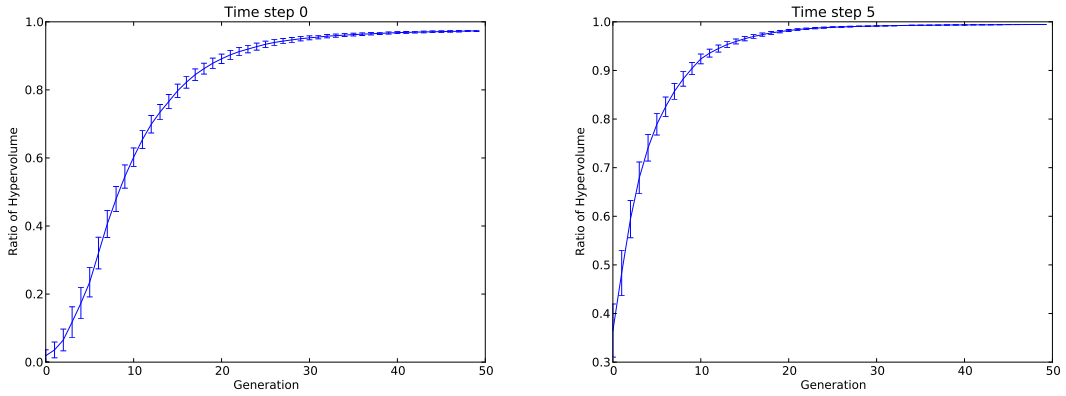


Figure 4.4: Example Pareto fronts at different time steps after 50 generations, MOEA results are shown with diamonds, true Pareto fronts are shown with solid lines.



(a) (b)

Figure 4.5: Static pre-execution. Ratio of hypervolumes of PF to PF_{true}, 25 runs, confidence level $(cl) = 95\%$

population size	100
Crossover	
probability	0.9
distribution index	10
Mutation	
probability	0.125
distribution index	20
dNSGAII-B hypermutation	
probability	0.25
distribution index	4
population increase ratio	0.3

Table 4.1: Default dNSGAII-* parameters

4.4 Dynamic optima tracking

In a tracking scenario, the MOEA continuously adjusts to the moving optima. The studies were performed with the parameters detailed in Table 4.1. dNSGAII-A was used unless explicitly stated otherwise. Figure 4.6 shows the mean value of the c_{HV} parameter for 500 generations. A cyclic pattern can be spotted that corresponds to the period (100 generations) of the problem. Changes in the MOEA performance depend on problem dynamics.

Figure 4.7 presents a detailed view of first 150 generations. The variability of results seem to depend only upon the initial convergence and not on later changes in the problem. Let us define a relative variability measure v_{HV} (4.3), such that

$$v_{HV} = |CI_2|\overline{\text{hv}\, PF}^{-1}, \tag{4.3}$$

where CI_2 is two-sided confidence interval. Figure 4.8 shows the value of v_{HV} for the first 150 generations. After initial period of convergence, multiple runs of the

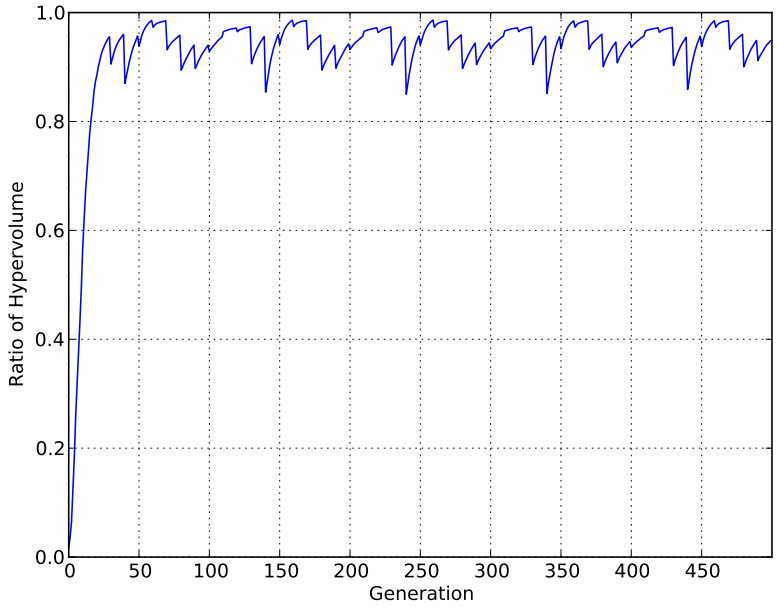


Figure 4.6: Dynamic tracking. Mean ratio of hypervolumes of PF to PF_{true} for 500 generations, 25 runs

optimisation algorithm produce essentially the same results. This means that a controller employing dynamic MOEA produces consistent results after the initial period of convergence.

4.4.1 Static and dynamic mutation rates

Noting the sharp drop in c_{HV} indicator when the problem changes, a study was conducted to see if increasing the level of static mutation and, therefore, increasing population diversity would have any effect. Figure 4.9 shows the c_{HV} indicator values with progressively increasing levels of mutation. Comparing with the default settings shown in Figure 4.7, it can be seen that increasing the mutation level noticeably slows the initial convergence rate. With the default settings, the algorithm has largely converged before generation 30, while with mutation probability = 0.9 and

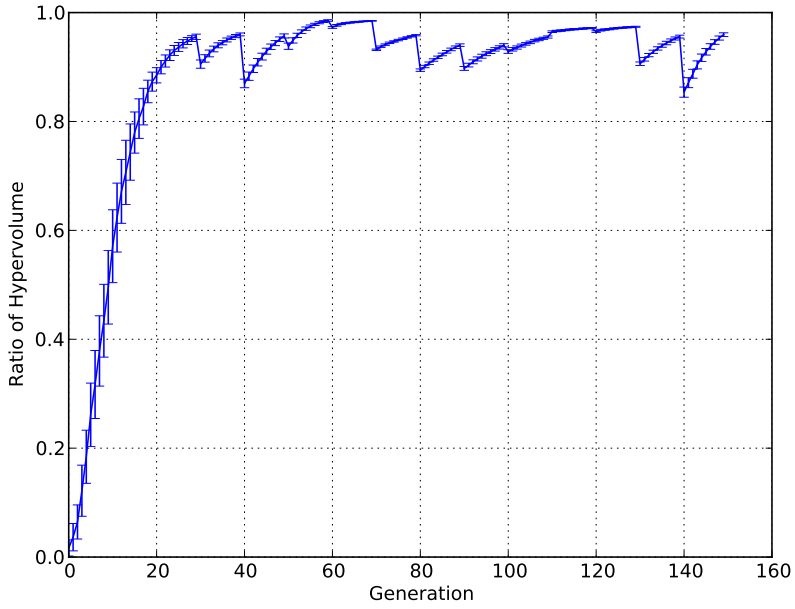


Figure 4.7: Dynamic tracking. Ratio of hypervolumes of PF to PF_{true} for 150 generations, 25 runs, $cl = 95\%$

distribution index = 4 it is still not fully converged at generation 100. However, once the initial convergence is attained, reasonable increases in mutation levels make performance drops relative to the current mean smaller.

Another method to decrease performance drops when a change happens is to use dNSGAII-B, which gives temporary boosts in diversity and population size when a change occurs. In essence, dNSGAII-B provides dynamic mutation rate management. Figure 4.10 shows that the effect of dNSGAII-B is twofold. Firstly, it noticeably speeds the initial convergence in the presence of changes. Secondly, it demonstrates improved reaction to changes, reducing the convergence drops compared to dNSGAII-A (notice the improved convergence around the 140 generations mark).

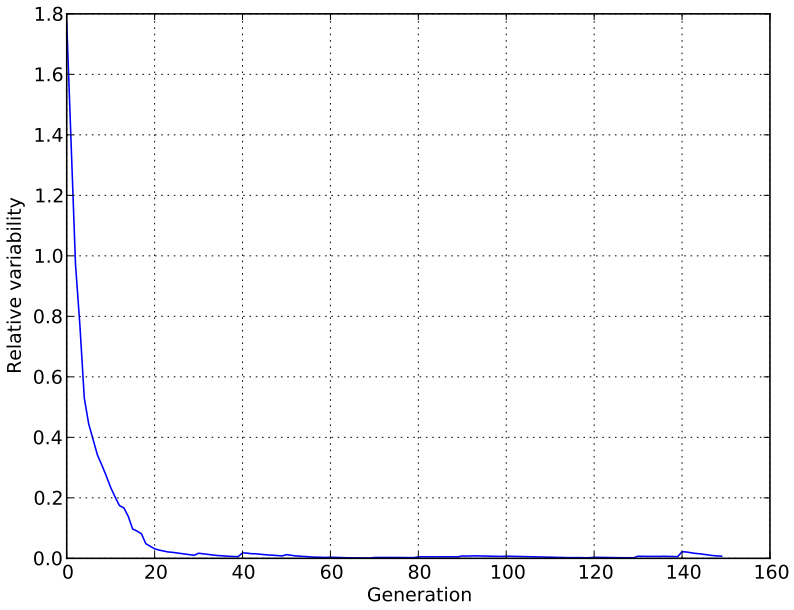


Figure 4.8: Dynamic tracking. Relative variability of hypervolume for 150 generations, 25 runs

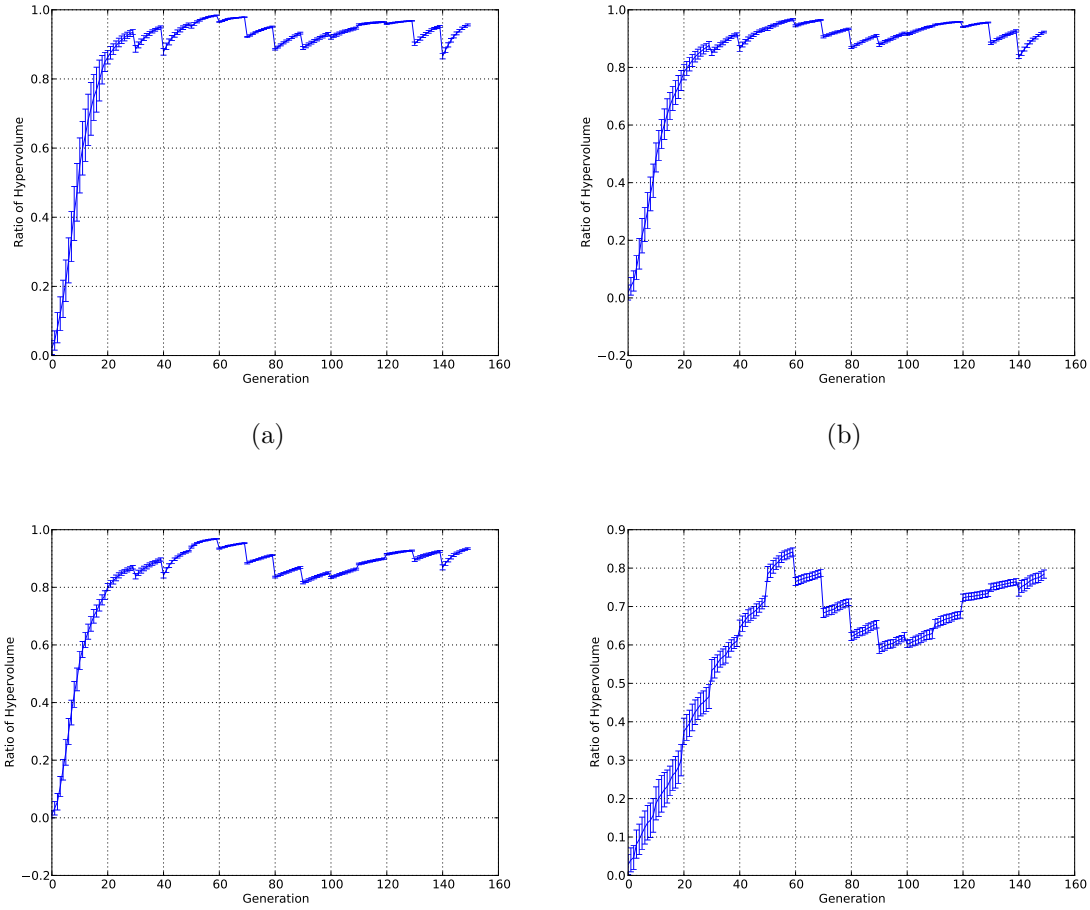


Figure 4.9: c_{HV} for different levels of mutation. a) probability = 0.3, distribution index = 20; b) probability = 0.3, distribution index = 4; c) probability = 0.9, distribution index = 20; d) probability = 0.9, distribution index = 4. 25 runs, $cl = 95\%$

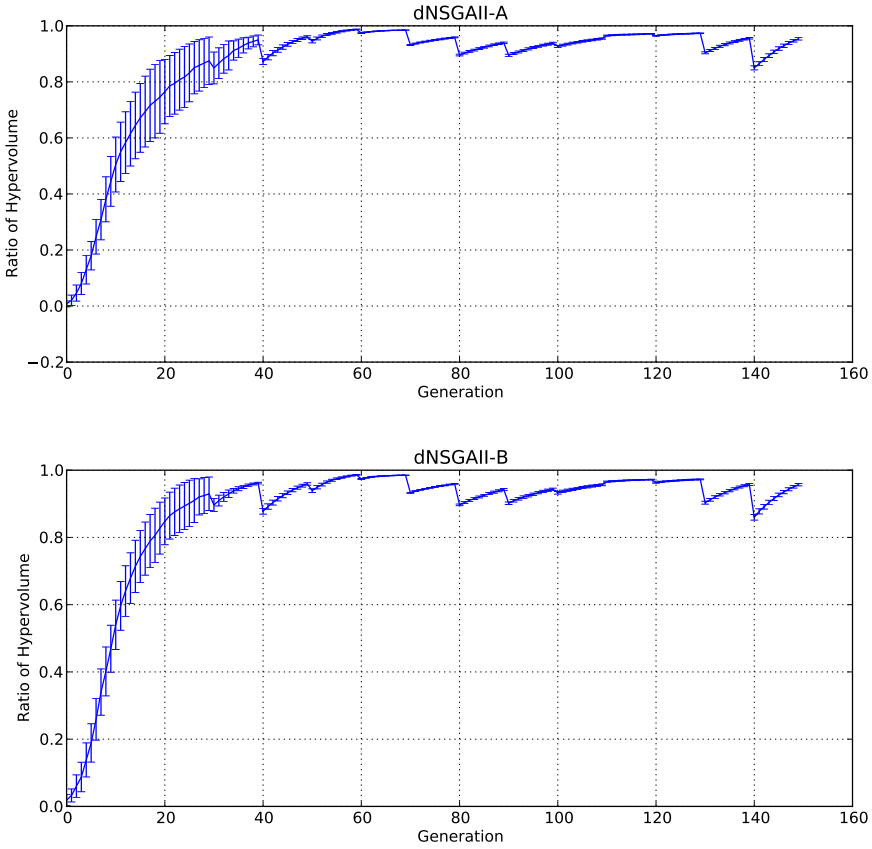


Figure 4.10: c_{HV}, 25 runs, $cl = 95\%$

4.4.2 Effect of rate of change

In order to explore the behaviour of the MOEA with different rates of change R, an additional study was performed for $\tau_T = 30$ and $\tau_T = 2$. Figure 4.11 shows the mean value of the convergence metric obtained using dNSGAII-A with $\tau_T = 30$. A corresponding graph for dNSGAII-B is shown in Figure 4.12. Compared to the regular case (see Figure 4.7; vertical grid lines correspond to the changes in $H(t)$), it is clear, that additional generations allow the algorithm to closely track the PF_{true} at all times, despite the reduced diversity in the population. Addition of hypermutated individuals after a change is detected does not give any appreciable benefits, except in the initial convergence stage, where it may be beneficial because of the temporary population increase. The difference in long-term performance between default $\tau_T = 10$ and $\tau_T = 30$ is much smaller than what could be predicted from the static pre-execution results. Results shown in Figures 4.13 and 4.14, provide c_{HV} values with $\tau_T = 2$. After a prolonged initial convergence period, the evolutionary algorithm achieves good performance. Again, the addition of hypermutated individuals slightly increases the initial convergence speed, but does not give any appreciable benefits afterwards. Considering the additional number of fitness evaluations, the dNSGAII-B remains inferior to the dNSGAII-A even when the algorithm is allowed to converge fully between the changes, as shown in Figure 4.15

4.4.3 dNSGAII-C

Two different strategies to select a sample of the parent population to be reevaluated and merged with the offspring were used. The first strategy randomly selects the individuals from the parent population until a predetermined ratio ξ is reached. The second strategy sequentially selects individuals in order of decreasing fitness. Figure 4.16 shows the two strategies compared with different values of ξ. $\xi = 0$ means that no solution from the parent population is merged with the offspring,

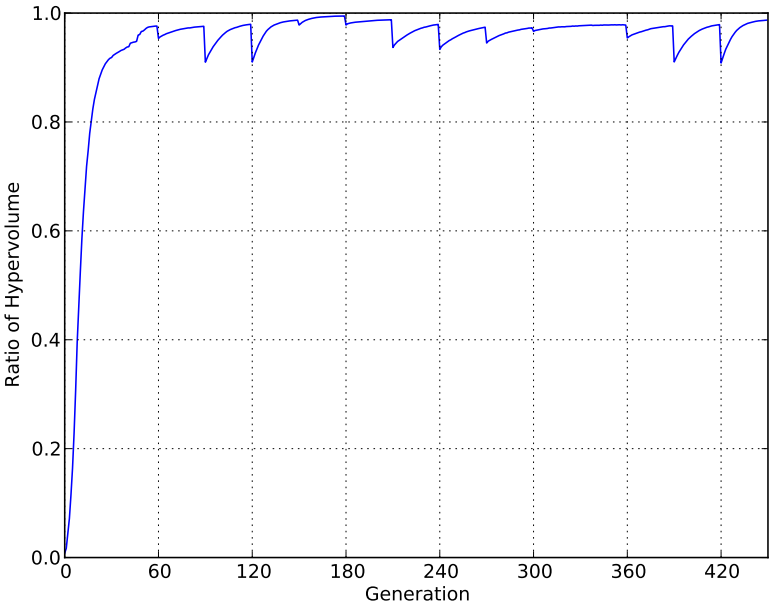


Figure 4.11: Mean c_{HV}, dNSGAII-A, $\tau_T = 30$

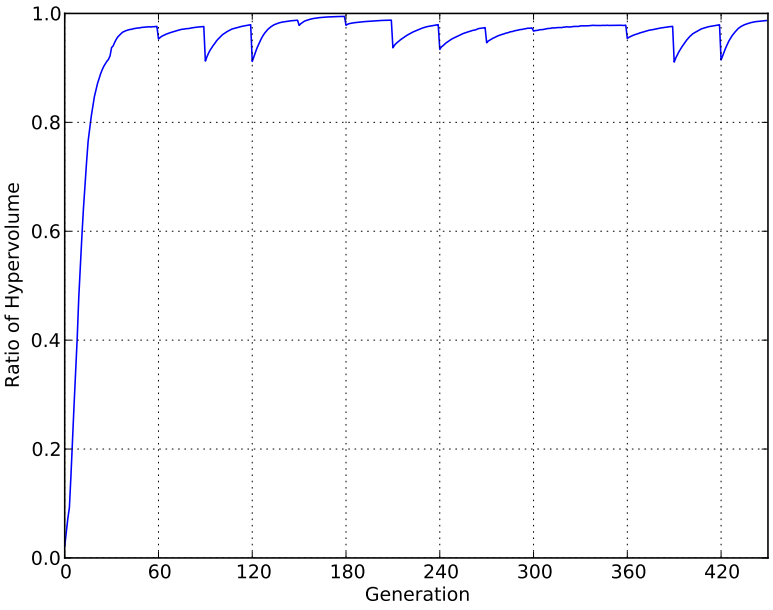
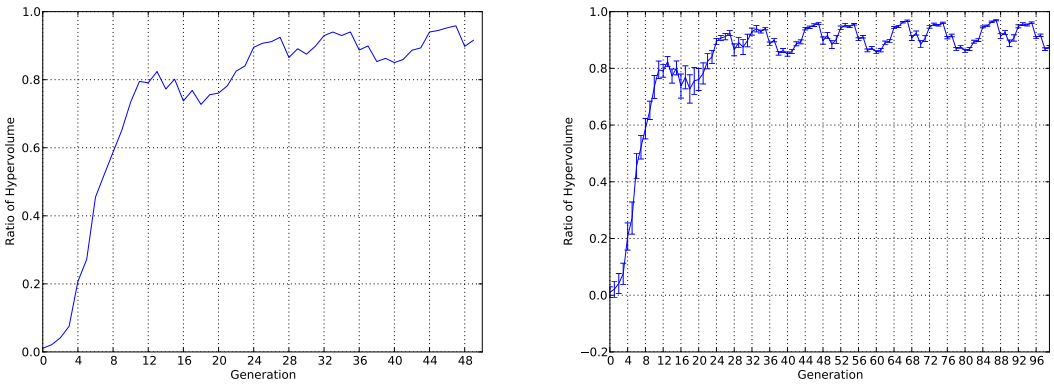


Figure 4.12: Mean c_{HV}, dNSGAII-B, $\tau_T = 30$



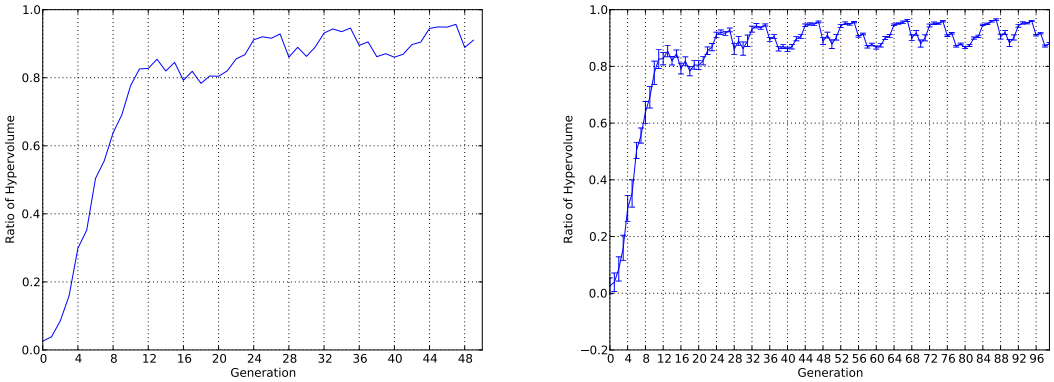
(a) (b)

Figure 4.13: Mean c_{HV}, dNSGAII-A, $\tau_T = 2$, $cl = 95\%$

$\xi = 0.5$ reevaluates and merges half of the parent population with the offspring. The dNSGAII-A algorithm is used as a baseline, i.e., its performance is always considered equal to 1.

The algorithm with no elitism ($\xi = 0$) exhibits slow convergence speed and the resulting front is about two times worse than that of dNSGAII-A. Allowing some randomly selected individuals to propagate into future generations brings dNSGAII-C performance to about 0.85 – 0.95 that of dNSGAII-A with $\xi = 0.3$ to $\xi = 0.5$. The dNSGAII-B performs equally or better than the dNSGAII-A, except for the first 10 generations.

Ordered selection of individuals to be propagated constantly outperforms the random selection, and in some cases the dNSGAII-A as well. This suggests that with small times between changes and, consequently, a perpetually unconverged population, the dNSGAII-C with ordered selection would prove superior to dNSGAII-A.



(a) (b)

Figure 4.14: Mean c_{HV}, dNSGAII-B, $\tau_T = 2$, $cl = 95\%$

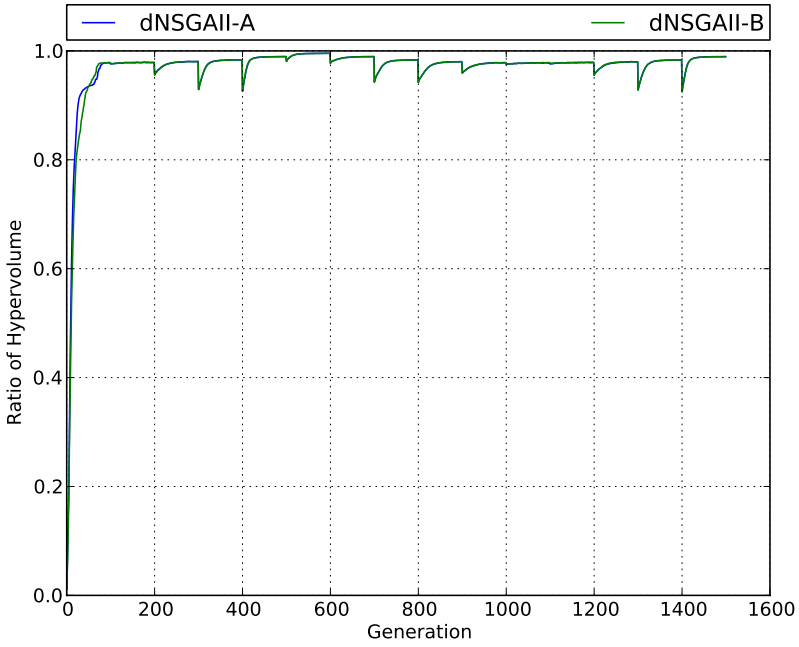


Figure 4.15: Mean c_{HV}, dNSGAII-B, $\tau_T = 100$

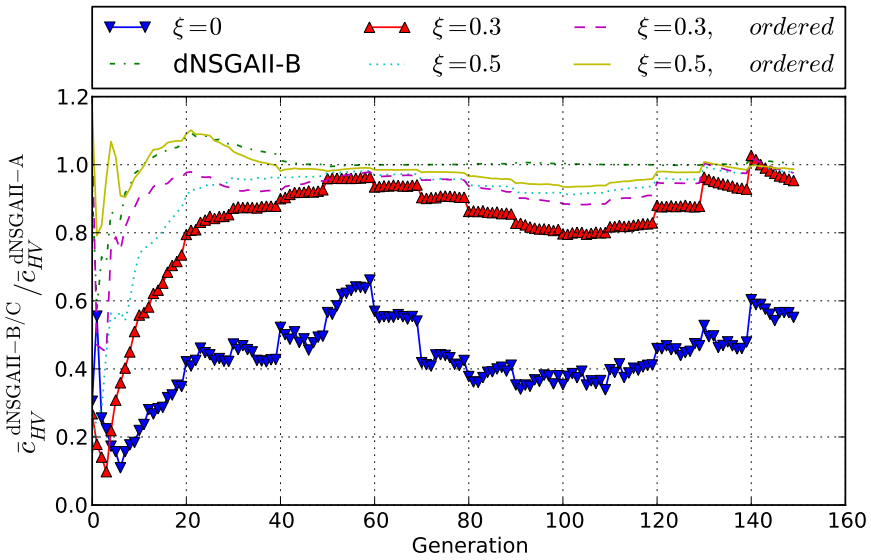


Figure 4.16: Ratio of mean dNSGAII-C and dNSGAII-B c_{HV} to mean dNSGAII-A c_{HV}, 25 runs.

4.5 Summary

This chapter analyses the dynamic performance of the evolutionary multi-objective search. Compared with a restart strategy, the key difference of the DMOEA is the reliability of results, characterised by the length of the confidence interval. In addition, it is possible to identify two distinctive stages in the DMOEA convergence process, namely, the initial convergence and dynamic tracking.

The effect of the frequency of change is less pronounced in the case of DMOEA compared to the restart strategy. Elitism is still very important in the DMOEA, as evidenced by Figure 4.16. The restricted elitism technique can sometimes outperform the baseline dNSGAII-A algorithm. Ordered restricted selection of elite individuals performs better than random selection.

Chapter 5

Decision making in dynamic environments

Decision making is an important aspect of dynamic MOO. MOEA produce a set of Pareto-optimal solutions, so a decision has to be taken as to which solution to select for an application [20]. In static cases, this decision is performed by a human decision maker, possibly with the help of one of the MCDM methods. Solving a control problem using a non-dominated set approach requires performing selection from the Pareto-set in real time, which can be difficult or impossible to achieve with a human decision maker. This chapter analyses the decision making process in dynamic environments and proposes a novel game tree based decision algorithm.

According to [16], MCDM methods can be categorised into three groups according to the stage at which the design and objective preference information is considered. *A priori* methods combine multiple design and objective preferences into a scalar-valued cost function that can be optimised by single-objective search methods. *Progressive* methods use decision making to guide search routine to explore particular areas of the objective space. Finally, *a posteriori* methods select a solution from a Pareto-optimal set of feasible alternatives produced by a multi-

objective optimisation algorithm.

The *a posteriori* approach, which is the subject of this chapter, has several features that distinguish it from the other two. Since multiple alternatives and the general shape of the tradeoff surface are known at the decision time, the decision maker can arguably make a better decision [142]. At the very least the alternatives can be compared based on their performance. The 'cost' of a making suboptimal decision is lowered as the alternatives are Pareto-optimal. Also, the decisions of an *a posteriori* decision maker do not affect search, which becomes important in dynamic applications. In contrast, a new set of weights for an aggregation function can render the population of a single-objective optimiser almost useless.

Notwithstanding the differences in the MCDM methods, they all share the same goal to reduce the dimensionality of the decision space. Making a decision based on several conflicting criteria means selecting a trade-off strategy, either a 'conservative' or an 'aggressive' one [143]. Conservative trade-off strategies seek to improve the under-performing attributes, i.e., the overall preference for an alternative will be based on the value of the least-performing attribute. Aggressive design strategies are willing to reduce the values of under-performing attributes slightly to improve other variables, i.e., they compensate for the lower performing attributes with higher performing attributes. In practice, many approaches combine elements of both conservative and aggressive decision strategies.

Trade-off strategies are implemented using sequential objective ordering [16], aggregation of objective values [144, 143], goal programming [145] or other methods such as rule-based inference [47]. Out of those, the trade-off strategies that use aggregation of objectives are the most widespread, due to their relative simplicity, low computational cost and ease of implementation.

5.1 Decision making by aggregation

Law and Antonsson [144] provide a formal definition of the aggregation approach. Suppose a vector $\vec{d}$ of design variables d_i. All feasible values of $\vec{d}$ constitute a set X of feasible design parameter values in decision space. All feasible values of an individual design variable d_i constitute a set $X_i \subset X$. Performance criteria of an individual solution $\vec{d}$ for a problem are denoted $p_j = f(\vec{d})$. Multiple performance criteria constitute a vector $\vec{p}$. Set Y in objective space contains all possible values of $\vec{p}$, consequently, Y_j corresponds to all possible values of p_j. The map from X to Y can be determined by various approaches, including exhaustive search, closed-form equations, iterative, heuristic or empirical methods. This research uses MOEA to determine the map. To select a single individual from a set of all possible alternatives, a decision maker requires preferences to be defined. According to [144], a design preference for the design variable d_i is a function on set X, such that:

$$\mu_{d_i}(d_i) : X_i \to [0, 1] \subset \mathbb{R}. \tag{5.1}$$

Likewise, a preference for the performance variable p_j is a function on a set Y, such that:

$$\mu_{p_j}(p_j) : Y_j \to [0, 1] \subset \mathbb{R}. \tag{5.2}$$

Overall preference μ_o is defined as a combination of design and performance preferences,

$$\mu_o = \mathcal{P}(\mu(d_1), \mu(d_2), ..., \mu(d_m), \mu(p_1), \mu(p_2), ..., \mu(p_n)). \tag{5.3}$$

The task of a decision maker is to find an element $\vec{d}^*$ that minimises overall preference μ_o.

$$\mu_o(\vec{d}^*) = \mu_o^* = \inf\{\mu_o(\vec{d}) \mid \vec{d} \in X\}. \tag{5.4}$$

5.1.1 Criteria for aggregation functions

It is evident from Equations (5.1), (5.2) and (5.3) that in order to make a decision maker use an aggregation approach, it is necessary to set performance preferences and define an aggregation function, which are application-specific tasks. However, there are certain features that are common to any aggregation-based approaches.

An aggregation function implements a trade-off strategy between several criteria. To reflect the relative importance of conflicting criteria, they are usually assigned individual importance weights.

$$w_{d_i} \in [0,1]; \quad w_{p_j} \in [0,1] \tag{5.5}$$

For simplicity, the weights are usually normalised

$$\sum_{i=1}^{m} w_{d_i} + \sum_{i=1}^{n} w_{p_j} = 1. \tag{5.6}$$

Otto [146] defines four axioms for the aggregation functions. Suppose a value of preference function μ is denoted as α. For an aggregation of two preference values α_1 and α_1 with relative importance weights w_1 and w_2, respectively, an aggregation function $\mathcal{P}(\alpha_1, \alpha_2, w_1, w_2)$ must satisfy the following conditions:
monotonicity

$$\begin{aligned} &\mathcal{P}(\alpha_1, \alpha_2, w_1, w_2) \leq \mathcal{P}(\alpha_1', \alpha_2, w_1, w_2) \quad \forall \quad \alpha_1 \leq \alpha_1' \\ &\mathcal{P}(\alpha_1, \alpha_2, w_1, w_2) \leq \mathcal{P}(\alpha_1, \alpha_2, w_1', w_2) \quad \forall \quad w_1 \leq w_1'; \end{aligned} \tag{5.7}$$

continuity

$$\begin{aligned} \mathcal{P}(\alpha_1, \alpha_2, w_1, w_2) &= \lim_{\alpha_1' \to \alpha_1} \mathcal{P}(\alpha_1', \alpha_2, w_1, w_2) \\ \mathcal{P}(\alpha_1, \alpha_2, w_1, w_2) &= \lim_{w_1' \to w_1} \mathcal{P}(\alpha_1, \alpha_2, w_1', w_2); \end{aligned} \tag{5.8}$$

annihilation

$$\mathcal{P}(0, \alpha, w_1, w_2) = 0 \quad \forall \quad w_1 \neq 0; \tag{5.9}$$

idempotency

$$\mathcal{P}(\alpha, \alpha, w_1, w_2) = \alpha \quad \forall \quad w_1 + w_2 > 0. \tag{5.10}$$

In addition to the above, three new axioms are defined in [142]:
self-scaling weights

$$\mathcal{P}(\alpha_1, \alpha_2, w_1 t, w_2 t) = \mathcal{P}(\alpha_1, \alpha_2, w_1, w_2) \quad \forall \quad t, w_1 + w_2 > 0; \tag{5.11}$$

symmetry

$$\mathcal{P}(\alpha_1, \alpha_2, w_1, w_2) = \mathcal{P}(\alpha_2, \alpha_1, w_2, w_1), \tag{5.12}$$

zero weights

$$\mathcal{P}(\alpha_1, \alpha_2, 0, w_2) = \alpha_1 \quad \forall \quad w_2 \neq 0. \tag{5.13}$$

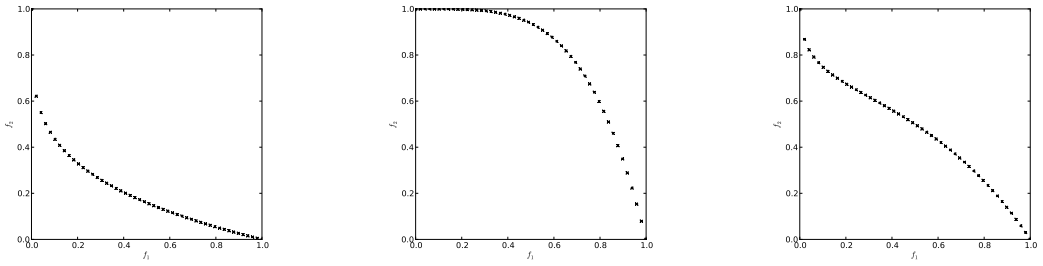
5.1.2 Weighted sum model

The WSM is the most widely used preference aggregation technique, owing to its simplicity and low computational overhead. In particular, it is widely used to perform a priori multiple objective aggregation into a single objective function for use with single-objective EA [147]. Considering a case with two objectives (more than two objectives can be aggregated using a hierarchical pairwise approach presented in [144]), the WSM states that:

$$\mathcal{P}(\alpha_1, \alpha_2, w_1, w_2) = \alpha_1 w_1 + \alpha_2 w_2. \tag{5.14}$$

It is easy to see that WSM does not satisfy the criteria of annihilation and self-scaling weights. However, the main drawback of the WSM lies in its dependency on the Pareto front shape.

Consider Figure 5.1 that shows three Pareto fronts of different shapes. In fact, each front can be decomposed into convex, concave and linear sections. For a Pareto



(a) (b) (c)

Figure 5.1: Different Pareto front shapes a) convex, b) concave, c) mixed

front section bounded by two arbitrary points A and B, the objective values of point C lying between them can be expressed as

$$f_i^C = \mu_i + \frac{1}{2} s_i r_i, \tag{5.15}$$

where

$$\mu_i = \frac{f_i^A + f_i^B}{2} \tag{5.16}$$

is a component of mean objective vector $\vec{\mu}$,

$$r_i = \max(f_i^A, f_i^B) - \min(f_i^A, f_i^B) \tag{5.17}$$

is a component of range vector $\vec{r}$ and $s_i \in [-1, 1]$ is a component of scaling vector $\vec{s}$. Now, the Pareto front section is convex, if $\sum_{i=1}^{n} s_i < 0$. Concave sections correspond to $\sum_{i=1}^{n} s_i > 0$. Finally, linear sections have $\sum_{i=1}^{n} s_i = 0$.

A good trade-off strategy should consider every point on the non-dominated front. This is not always the case with WSM [148]. To prove this, it is enough to find a point that would not be considered by WSM. From Equations (5.14) and (5.15), the WSM metric for point C in two-dimensional objective space is

$$\mathcal{P}(C) = \frac{\mathcal{P}(A) + \mathcal{P}(B)}{2} + \frac{1}{2}(w_1 r_1 s_1 + w_2 r_2 s_2). \tag{5.18}$$

It follows that if $s_1, s_2 \geq 0$, $\mathcal{P}(C)$ will be greater than either $\mathcal{P}(A)$ or $\mathcal{P}(B)$ for any values of w_1 and w_2, which means that the alternative C will never be selected

by the trade-off strategy. The corresponding graphical interpretation is provided in [26].

Various aggregation functions that allow points on the concave sections of the Pareto front to be captured have been developed [149, 142]. The trend is to use L^p norm-derived aggregation [150], such as the global criterion method [151, 26]. Varying the p parameter (often called 'compensation ratio') changes the curvature of the aggregating function, tailoring it to the shape of the Pareto front.

The difficulties in setting weights and p parameter are considerable even in the case of static problems with potentially unlimited time and where human input is possible. In dynamic control problems, it is questionable for a decision maker to rely on a predefined set of weights. An interesting strategy inspired by real life is to select solutions at random. This strategy can only be effective when coupled with dynamic multi-objective search and an *a posteriori* decision maker for the following reasons.

- An *a posteriori* decision maker does not affect search, so the frequent and sudden changes of the Pareto-optimal control strategy would not destroy search performance.

- Selection is performed from a set of Pareto-optimal individuals. The underlying hypothesis is that all individuals are good to be selected.

- A potentially dangerous consequence of selecting an inferior individual is mitigated by the fact that the control strategy selection is iterative.

The random decision maker will be tested in Chapter 6 and compared with the aggregation approach. The question remains if a strategy can do better than random selection without explicitly prioritising the alternatives via weights and compensation ratio.

5.2 Pareto Decision Tree

An *a posteriori* decision maker for an EMO MPC procedure should be able to produce automated decisions in a short time. This makes it similar to the problems considered by mathematical game theory, such as chess and checkers. At each turn, a player is faced with multiple alternatives, each of which leads to a different set of possible moves by the opponent. The sequences of feasible moves from the initial state form a *game tree* for the game. An algorithm to find an optimal move at each branching point is presented in [1].

5.2.1 Minimax algorithm

The minimax algorithm (Appendix A) is a recursive algorithm for choosing the next move in a n-player game. Each *terminal state*, i.e., a state where the game has ended, is assigned a utility value. The minimax algorithm takes the current game state and computes an optimal move by recursively traversing the game tree and examining the utility of each node (minimax value). The minimax value of a node equals the utility for the current player of a particular state occurring. The algorithm assumes that both players play optimally.

In its default form, the computational cost of the minimax algorithm is prohibitively expensive for most practical applications [1]. Most applications choose to evaluate the utility of a node by using a heuristic evaluation function. Using a heuristic evaluation dramatically reduces the depth of the search tree that is needed, effectively making each node a terminal state.

At this point it is worth remembering that the games discussed earlier are deterministic and allow a fixed number of outcomes. Real control problems have various uncertainties and possibly an infinite number of states. Also the evaluation of an individual state is difficult as often there are multiple conflicting performance criteria.

5.2.2 Proposed tree-based decision algorithm

The minimax algorithm operates by examining the utility values of the game tree nodes, representing the states of the environment. To compute the utility value for a state of a control problem, it is proposed to use a metric based on the Pareto front that corresponds to the state. Using a Pareto front metric allows us to quantify a state without explicitly ordering or weighting different performance criteria. A hypervolume indicator [78] is selected in this research as the utility of a state. A hypervolume is an aggregation function based on a set of points (Pareto set) rather than a single point.

The basic idea of the proposed Pareto Decision Tree (PDT) algorithm is to evaluate the implications of selecting a particular Pareto-optimal control action at time $t = m$ on the EMO MPC ability to make decisions at time $t = m+\delta$. Referring to Figure 5.2, the PDT algorithm operates by selecting a potential control action $p^{'}$ from the Pareto front corresponding to the current problem state S. The point is selected at random and each point can be selected only once. The algorithm than uses the point $p^{'}$ to control an internal process model for δ interval and generate a future problem state $S^{'}$. The future state $S^{'}$ is assigned a utility value based on the hypervolume metric of the corresponding Pareto front. If the utility is greater than the maximum utility found so far, the maximum utility value gets updated. Then a next potential Pareto-optimal control action $p^{'}$ is selected leading to another future problem state $S^{'}$. The process continues until there is no spare time left for another iteration of the decision maker. Finally, a control action corresponding to the maximum obtained utility value is returned.

Figure 5.3 shows an example of PDT operation. At time $t = m$, PDT is required to select a single control action from the Pareto front PF_0. To achieve this, PDT selects a random point $p^{'}$ from the Pareto front and predicts a future state of the system $S^{'}$ for time $t = m+\delta$, by using $p^{'}$ to generate the control effort for time δ. In

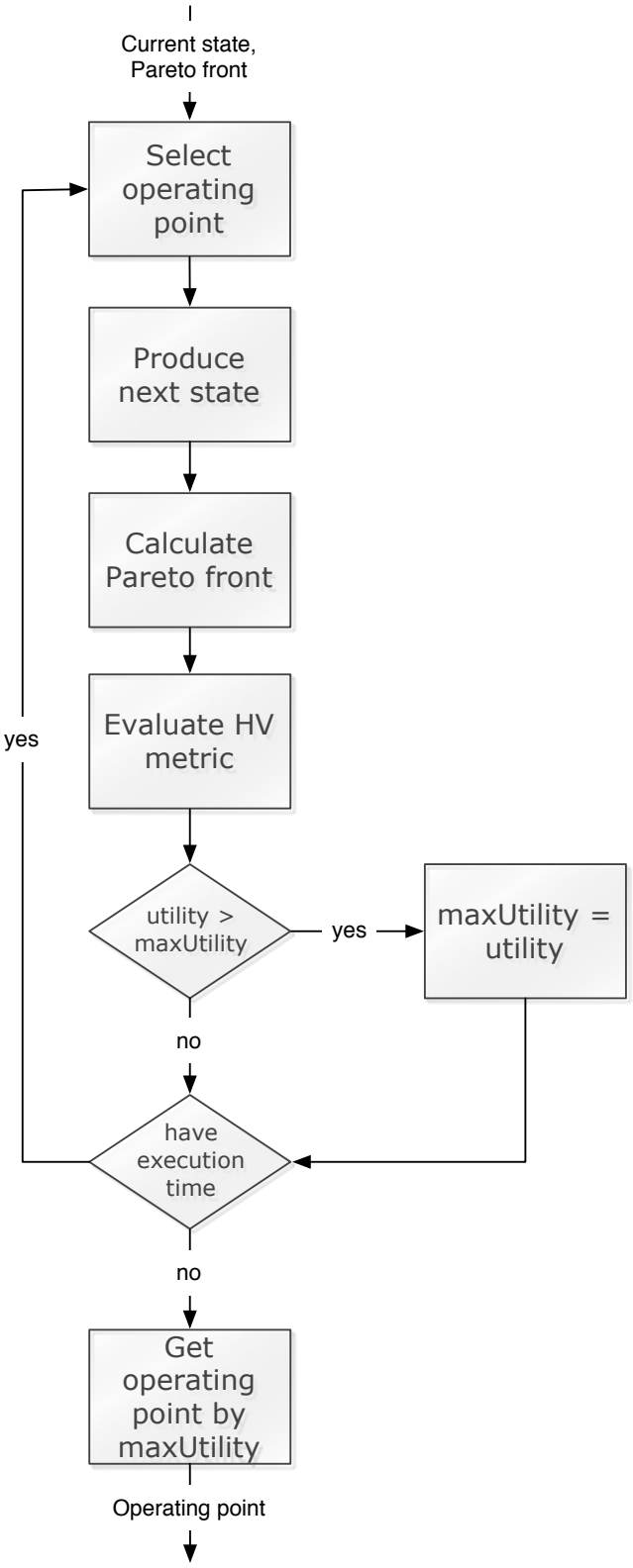


Figure 5.2: Pareto decision tree algorithm

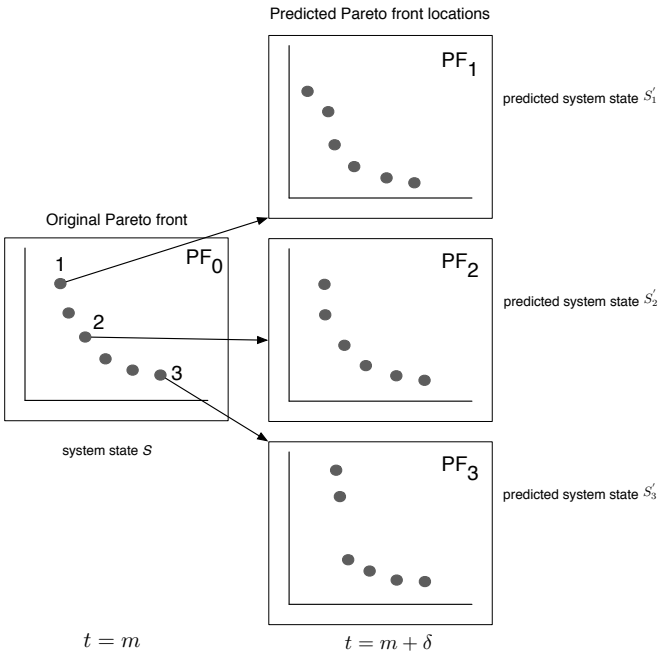


Figure 5.3: Pareto decision tree example, hypervolume $(HV)_1 > HV_2 > HV_3$. Point 1 will be selected by the PDT algorithm.

the example, three states corresponding to three different points on the Pareto front are developed. Each state is assigned a utility value using the hypervolume metric. A point corresponding to the state with the maximum utility, which is point 1 in the example, gets selected as the control action. Appendix B contains a minimal PDT implementation in Python.

5.3 Summary

This chapter addresses the on-line decision making aspect of the dynamic MOO. Analysis of contemporary decision making approaches (aggregation-based) used in dynamic EMOO shows that the results obtained using aggregation functions depend on the shape of the Pareto front. The most widely used dynamic decision

making approach, namely, WSM, is not suited to non-convex Pareto frontiers. Taking into account that an EMO MPC decision maker has to choose from a set of Pareto-optimal alternatives, a random decision maker is proposed as a baseline for comparison.

The author identifies a link between game theory and decision making in on-line multi-objective optimisation, noting that game theory has a number of approaches for on-line decision making. A game tree based decision making algorithm (known as MINIMAX) is recognised as being suitable for decision making in dynamic EMOO. To leverage MINIMAX algorithm, a novel method to build game trees for real-valued problems based on Pareto front metrics is presented. A novel Pareto tree based dynamic decision maker is proposed for on-line decision making within EMO MPC framework.

Chapter 6

Evaluation of single entity control

To evaluate the EMO MPC framework presented in Chapter 3, it was decided to select an established control problem and apply the proposed approach. The rest of this chapter is devoted to the application of the EMO MPC approach to the problem of controlling an inverted pendulum. An inverted pendulum is a well-known control problem, which is widely used for the design, comparison and analysis of control algorithms [152]. An inverted pendulum is used for modelling various processes ranging from walking patterns for robots [153] and crane dynamics [154] to human balancing [155]. Controlling an inverted pendulum is challenging because of the nonlinear dynamic behaviour it has. Also, the inverted pendulum is inherently unstable in its upright position and requires fast response from the controller to maintain equilibrium [156].

6.1 Inverted pendulum model

Referring to Figure 6.1, the inverted pendulum model consists of a point mass m_2 attached to a cart with mass m_1 using a massless rod with length l. The pendulum angle from upright position is defined as α. The pendulum is subjected to tangential

force F_α and the force to the cart F_x. The cart is assumed moving along the x axis with no friction. The dynamics of an inverted pendulum model can be obtained

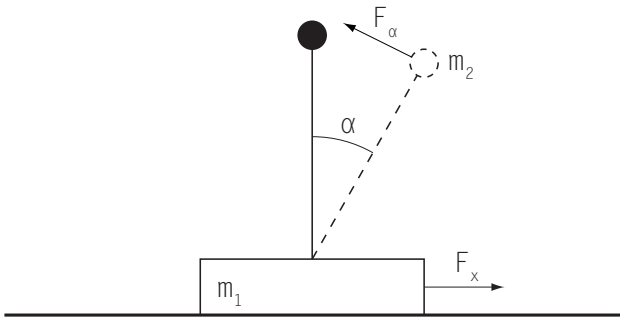


Figure 6.1: Nonlinear inverted pendulum model

using the Lagrangian method. The Lagrange equations for the plant are given by equations (6.1) and (6.2), where g is standard gravity of $9.8m/s^2$.

$$m_1\ddot{x} + m_2 l \sin\alpha\dot{\alpha}^2 - m_2 l\cos\alpha\ddot{\alpha} + m_2\ddot{x} = F_x \tag{6.1}$$

$$-m_2 lg\sin\alpha + m_2 l^2\ddot{\alpha} - m_2 l\ddot{x}\cos\alpha = F_\alpha \tag{6.2}$$

Substituting $\ddot{x}$ with $\dot{v}$ and $\ddot{\alpha}$ with $\dot{\omega}$ and solving for the $\dot{v}$ and $\dot{\omega}$, the dynamic model of the inverted pendulum is obtained in Equation 6.3.

$$\begin{aligned}
\dot{v} =& \frac{lm_2\sin\alpha(g\cos\alpha - l\omega^2) + lF_x + F_\alpha\cos\alpha}{l(m_1 + m_2\sin^2\alpha)} \\
\dot{\omega} =& \frac{N}{l^2 m_2(m_1 + m_2\sin^2\alpha)} \\
N =& m_1(glm_2\sin\alpha + F_\alpha) + \\
& m_2(lm_2\sin\alpha(g - l\omega^2\cos\alpha) + lF_x\cos\alpha + F_\alpha) \\
v =& \dot{x} \\
\omega =& \dot{\alpha}.
\end{aligned} \tag{6.3}$$

The system of equations (6.3) can be solved using the Runge-Kutta method of fourth order integration [157]. However, this method is rather slow, so this full

non-linear form is used only to model the environment. To evaluate the fitness of solutions, a much simpler linear system is used, which is given in Section 6.1.1. The use of the full model to calculate the environment and the simplified model to evaluate fitness introduces some modelling uncertainty, which is a characteristic of real-life applications.

6.1.1 Inverted pendulum state space model

Equation 3.1 defines a canonical form for the MPC models. To obtain a simple model from the system given by Equation 6.3, it needs to be linearised around the pendulum's upright position $\alpha < 0.05$rad. Assuming that $\cos\alpha = 1$, $\sin\alpha = \alpha$, $(\frac{\mathrm{d}\alpha}{\mathrm{d}t})^2 = 0$ and $F_\alpha = 0$, the linearised Lagrange equations are given by (6.4).

$$\begin{aligned} \ddot{x}(m_1 + m_2) - m_2 l\ddot{\alpha} &= F_x \\ -m_2 lg\alpha + m_2 l^2\ddot{\alpha} - m_2 l\ddot{x} &= 0 \end{aligned} \tag{6.4}$$

A state space representation uses x to denote the state vector of the system, so the equation (6.4) needs to be modified further. Let y be the output vector of the system. The inverted pendulum model has two outputs, namely cart position y_1 and pendulum angle y_2. The input of the system $u(t)$ is the motor force F_x. With that in mind, the rearranged equation (6.4) takes the following form.

$$\begin{aligned} m_1\ddot{y_1} - gm_2 y_2 &= u(t) \\ m_1 l\ddot{y_2} - (m_1 + m_2)gy_2 &= u(t). \end{aligned} \tag{6.5}$$

The state vector x is given by Equation 6.6. The first derivative of state vector $\dot{x}$ obtained from (6.5), is given by Equation 6.7

$$x = \begin{bmatrix} y_1 \\ \dot{y_1} \\ y_2 \\ \dot{y_2} \end{bmatrix} \tag{6.6}$$

$$x = \begin{bmatrix} x_2 \\ \frac{u(t)+m_2 g x_3}{m_1} \\ x_4 \\ \frac{u(t)+(m_1+m_2)g x_3}{m_1 l} \end{bmatrix} \tag{6.7}$$

From (6.6) and (6.7), the simplified state space model of the inverted pendulum is

$$\begin{aligned} \dot{x} &= \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & \frac{m_2 g}{m_1} & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & \frac{(m_1+m_2)g}{m_1 l} & 0 \end{bmatrix} x(t) + \begin{bmatrix} 0 \\ \frac{1}{m_1} \\ 0 \\ \frac{1}{m_1 l} \end{bmatrix} u(t) \\ y &= \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} x(t) + \begin{bmatrix} 0 \\ 0 \end{bmatrix} u(t). \end{aligned} \tag{6.8}$$

The state space model (6.8) can be used to approximate the next state of the plant from the current state using Euler's method [158]. Figure 6.2 highlights the differences in impulse response of the simplified and full inverted pendulum models. The simple model diverges rapidly from the upright position with time. The EMO MPC needs to be able to cope with the uncertainty introduced by the modelling differences if it is to produce an effective controller.

6.2 Proportional–Integral–Derivative Controller

PID controllers constitute a majority of the controllers being used and therefore provide a baseline for studying a controller behaviour. A PID controller [159] in a continuous form is given as:

$$u(t) = K_p e(t) + K_i \int_0^t e(\tau)\, d\tau + K_d \frac{d}{dt} e(t), \tag{6.9}$$

where K_p is proportional gain, K_i is integral gain, K_d is derivative gain, e is the difference between the setpoint and the current value (error) and τ is the integration

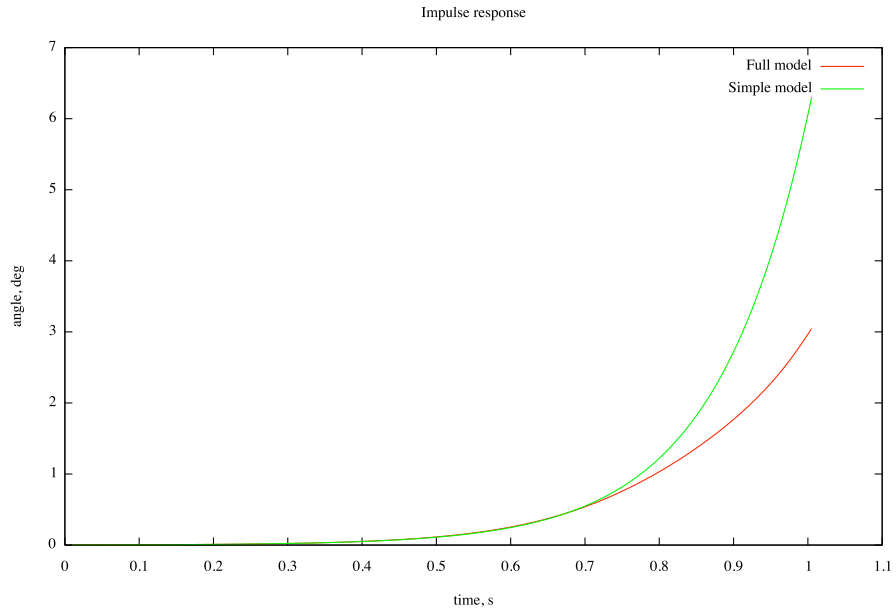


Figure 6.2: Impulse response of simplified linear and full models of inverted pendulum.

variable. In order to use equation (6.9) in an objective function, it needs to be discretised, yielding

$$u(t_n) = K_p e(t_n) + K_i \sum_{i=0}^{n} e(t_i)\Delta t + K_d \frac{e(t_n) - e(t_{n-1})}{\Delta t}, \tag{6.10}$$

where Δt is the sampling step.

6.3 Empirical results

6.3.1 Configuration

The initial parameters of the inverted pendulum model are summarised in Table 6.1. The default parameters of the evolutionary algorithm are presented in Table 6.2.

Parameter	Value
Pendulum mass m_2	0.5 kg
Cart mass m_1	0.5 kg
Rod length l	0.3 m
Cart position y_1	0 m
Cart speed v_1	0 $^{\mathrm{m}}/_{\mathrm{s}}$
Pendulum angle y_2	0 rad
Pendulum speed ω	0 $^{\mathrm{rad}}/_{\mathrm{s}}$
Time t	0 s
Motor force F	0 N
Integration interval	0.0025 s

Table 6.1: Initial parameters of the inverted pendulum model

6.3.2 Static optimisation of PID controller

This section investigates the Pareto surface of PID controllers stabilising the inverted pendulum from initial angle $\theta = 0.25$. The objective functions used are rise time (from $\theta = 0.25$ to $\theta = 0.0125$) and maximum overshoot angle. Figure 6.3 shows tradeoff surfaces between pendulum rise time and maximum overshoot for controllers limited by different maximum demand. It can be seen that the optimal solutions give less than 0.01rad overshoot (or 4 percent) irrespective of the maximum possible controller demand. Conversely, controllers that produce overshoot greater than 0.01rad are suboptimal in the Pareto sense. The slope of the Pareto front allows us to see diminishing returns of selecting very low (10N) and very high (85N) limiting force on the motor drive.

Figures 6.4 and 6.5 show the tradeoff surface for a limiting force of 45N and corresponding controller gains in the decision space. There are three different controller

Parameter	Value
Algorithm	NSGAII
Population size	50
Number of generations	300
Crossover probability	0.9
Crossover distribution index	20
Mutation probability	0.33
Mutation distribution index	20

Table 6.2: Parameters of the evolutionary algorithm

groups within the optimal subset. Referring to Figure 6.5, a majority of controllers appear to have $K_p \in (198, 200)$ with a near zero integral gain. Another group consists of two controllers (highlighted) in the convex region of the Pareto front that have smaller proportional gains, which results in a different impulse response compared to the main group. Finally, a group of controllers with very high derivative and relatively high integral gains correspond to the extreme left part of the Pareto front with longest rise time and lowest overshoot.

In order to explore possible limits for the force demand from the controller, the model was analysed using rise time and maximum controller demand objectives. Referring to Figure 6.6, the minimum force required to make the pendulum stable is approximately 3N. Meanwhile, increasing the force above 25N does not give any significant gains in terms of rise time. Looking in the decision space, which is shown in Figure 6.7, it is important to note that while most of the solutions have $K_i \ll 0.1$, some of them have a considerably higher integral gain (highlighted in Figures 6.6 and 6.7). Note that there are no solutions with high K_d because they are inferior to solutions with small K_d evaluated using rise time alone.

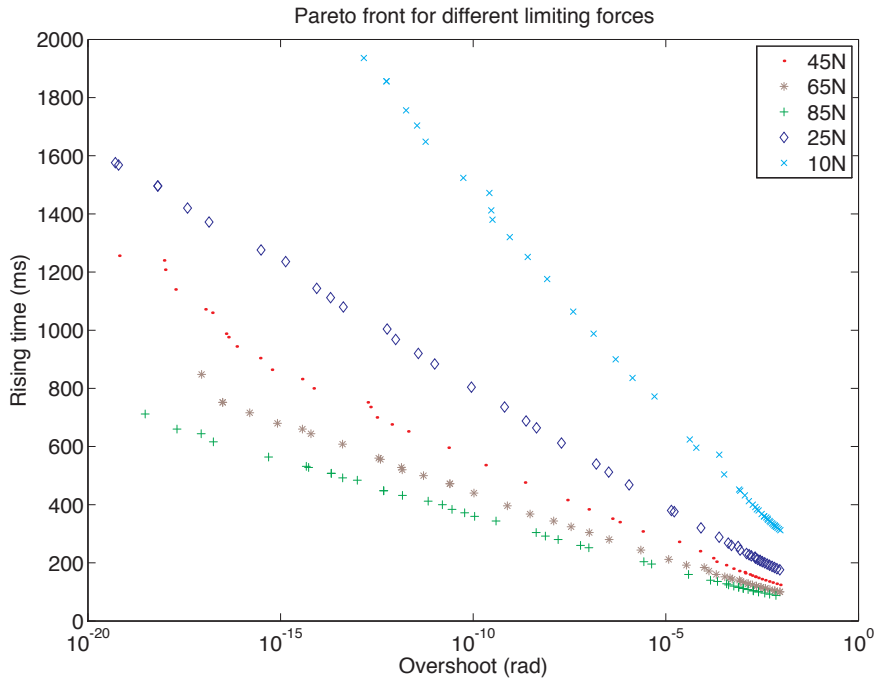


Figure 6.3: Tradeoff between rise time and overshoot for different values of F_{max}. Overshoot is plotted using a logarithmic scale.

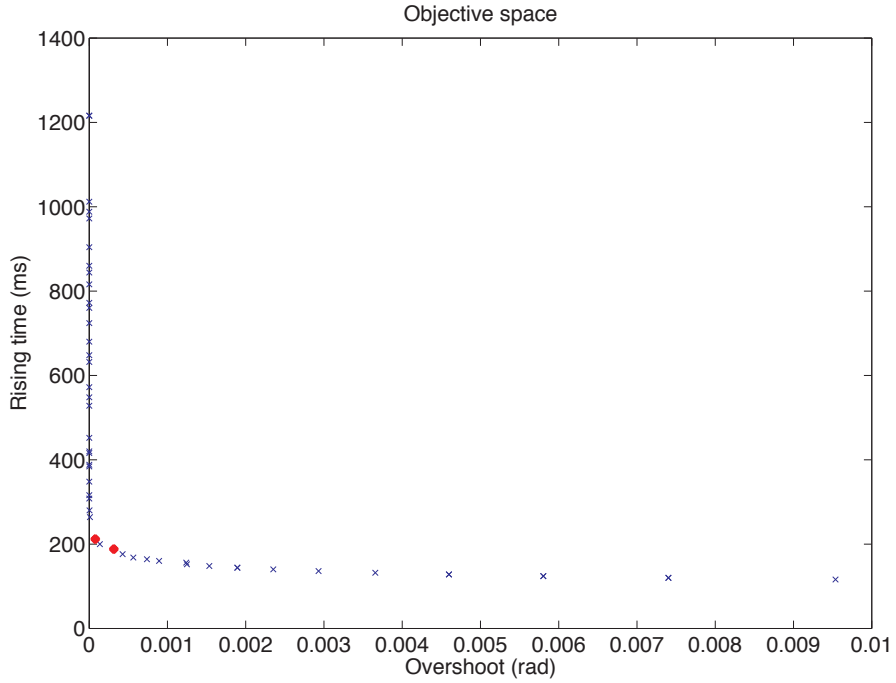


Figure 6.4: Tradeoff between rise time and overshoot, $F_{max} = 45$N (both to be minimised).

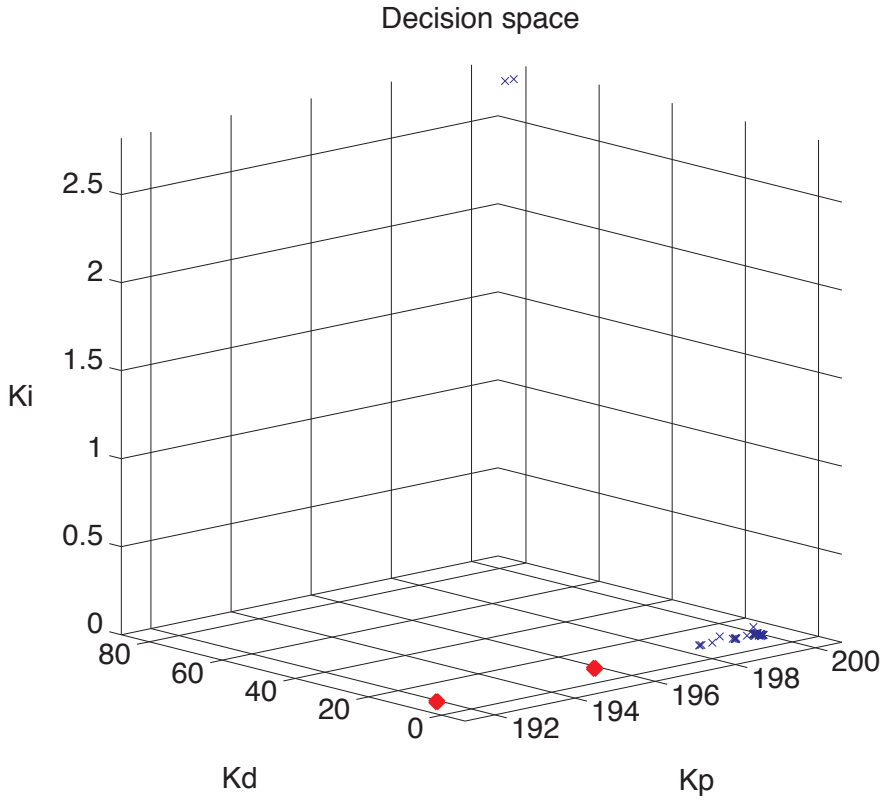


Figure 6.5: Corresponding controller gains for objective values in Figure 6.4. The solutions are grouped lower left, lower right and upper centre.

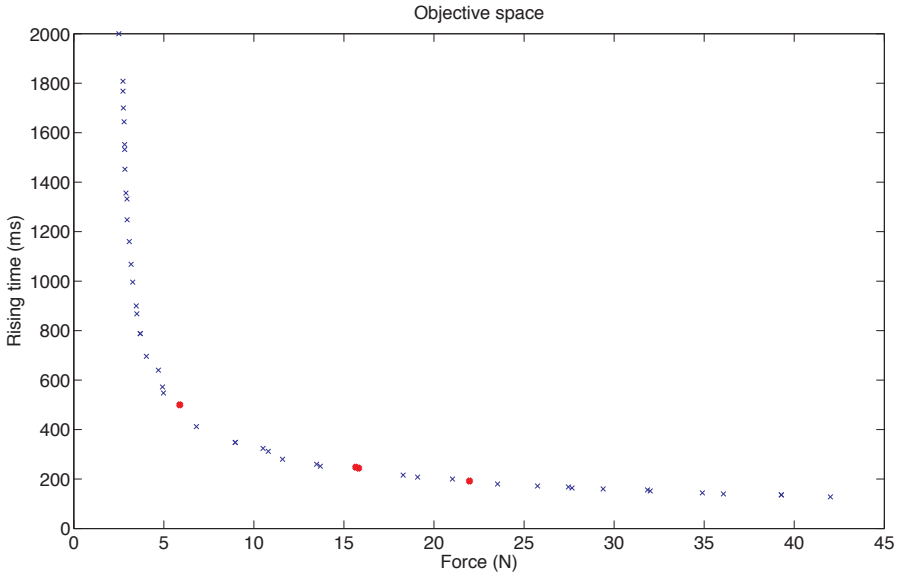


Figure 6.6: Tradeoff between rise time and maximum controller demand

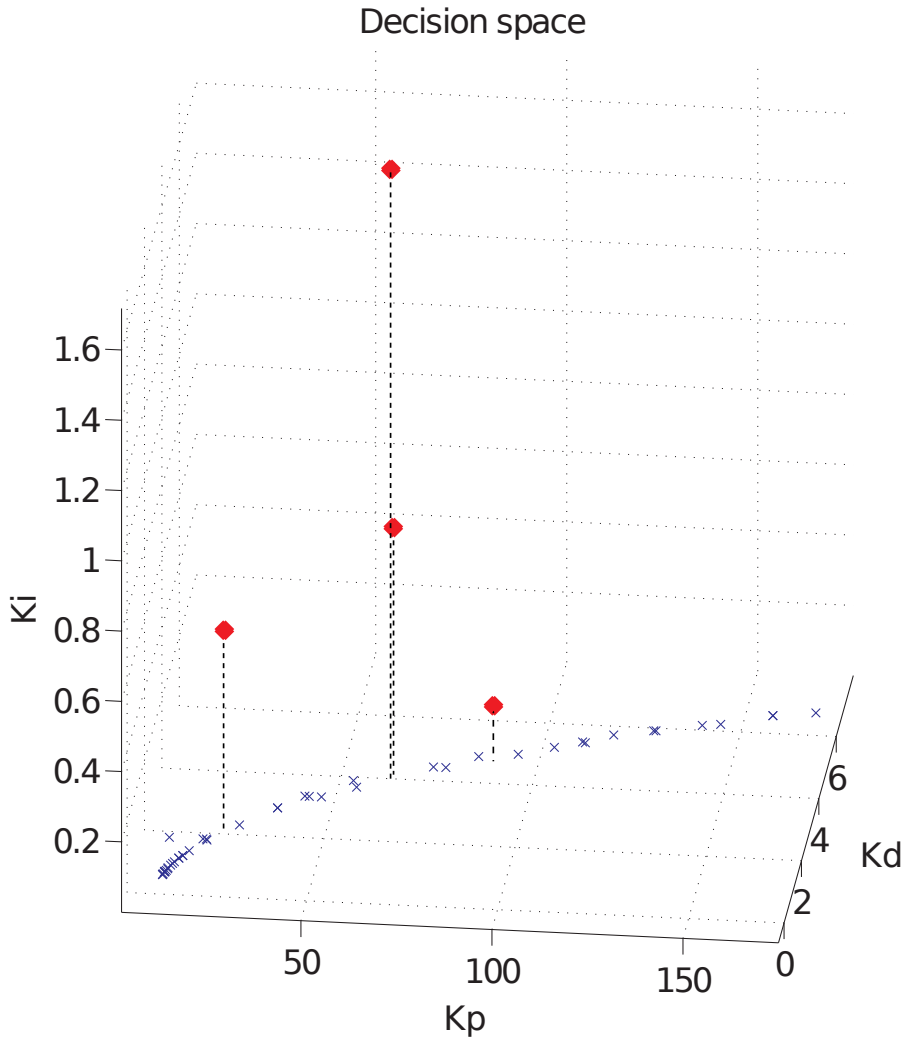


Figure 6.7: Corresponding controller gains for objective values in Figure 6.6

6.3.3 Dynamic control

An aspect that differentiates dynamic and static optimisation cases are the requirements for the objective functions. As discussed in Section 3.4, for practicality reasons, the prediction horizon used in a dynamic optimiser is small compared to the system's lifetime. It often means that the setpoint can not be reached inside the prediction horizon. Consequently, objective functions that use traditional performance metrics, such as settling time and overshoot, are not useful in the dynamic cases, as these metrics can be calculated only when the steady state of the plant has been obtained. It is suggested to classify fitness functions as a) *global* and b) *local*. Global fitness functions are defined in terms of the steady state performance of a control system. The fitness functions used in static optimisation cases are global by definition. Local fitness functions estimate performance inside a prediction horizon. In dynamic cases, both global and local fitness functions could be used.

The local fitness functions are required to be *isotropic*. For example, for the inverted pendulum problem, the fitness values for $+\alpha$ angles should be equal to the fitness values for $-\alpha$ angles. Another important property of local fitness functions is *statelessness*, i.e., individual's fitness over a prediction horizon should be calculated in isolation from the system state at the beginning of the prediction horizon.

A series of preparatory experiments indicated that integrating fitness functions are better suited for dynamic scenarios than functions that use extremum attained values. Uncertainty in dynamic optimisation may produce outliers, that are 'smoothed out' by the integration process. For the inverted pendulum case, two integrating objective functions were selected (6.11), where f_1 is the integral of pendulum deflection, f_2 is the integral of the control effort and τ is an integration variable.

$$\begin{aligned} f_1 &= \int_t^{t+T_p} |\alpha(\tau)| \mathrm{d}\tau \\ f_2 &= \int_t^{t+T_p} |F(\tau)| \mathrm{d}\tau \end{aligned} \tag{6.11}$$

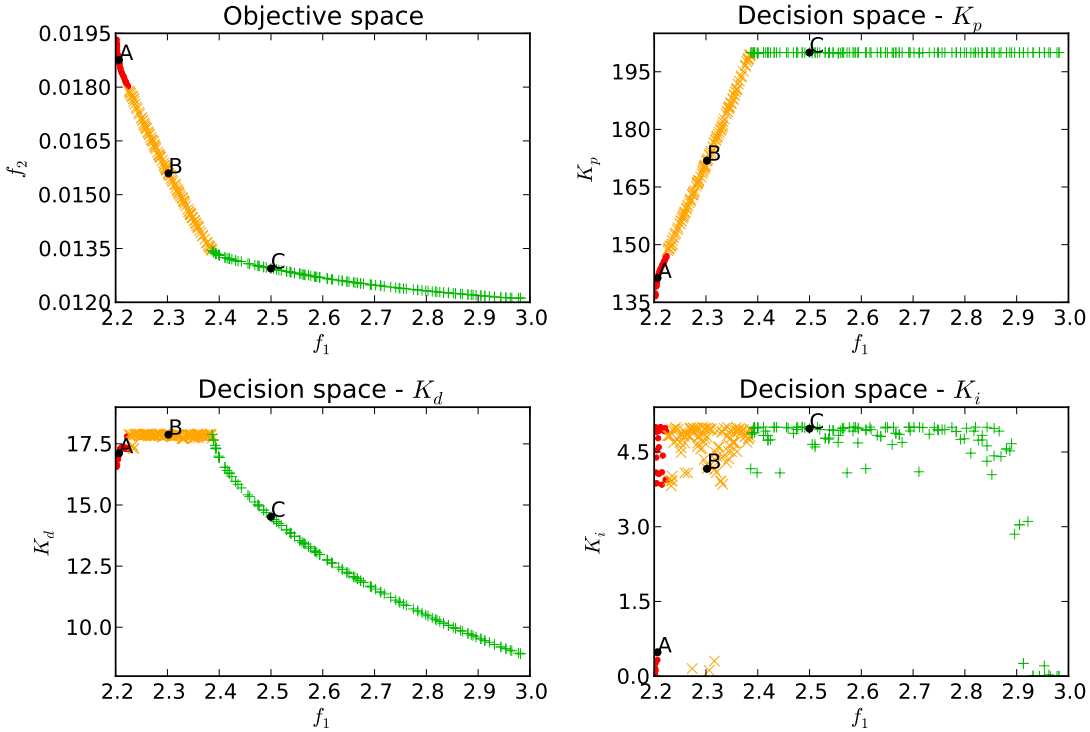


Figure 6.8: Objective and decision spaces for the inverted pendulum problem. Objective functions are defined by (6.11). $T_p = T_c = 0.4$s.

Figure 6.8 shows the objective values with corresponding decision variable values for the task of finding static PID coefficients. There are three distinctive groups of controllers, shown as red '.', orange 'x' and green '+'. The randomness in the integral gain can be attributed to the short prediction horizon, which downplays the role of error integration. The corresponding impulse responses of the plant for points in each controller group are presented in Figure 6.9. The controllers exhibit no overshoot, suggesting that for this problem overshooting control strategies are not optimal.

Figure 6.10 shows an impulse response of the inverted pendulum system with a dynamic EMO MPC controller and PID control conditioner. The decision maker used is WSM with $\{0.5, 0.5\}$ weights. The chromosome consists of three genes en-

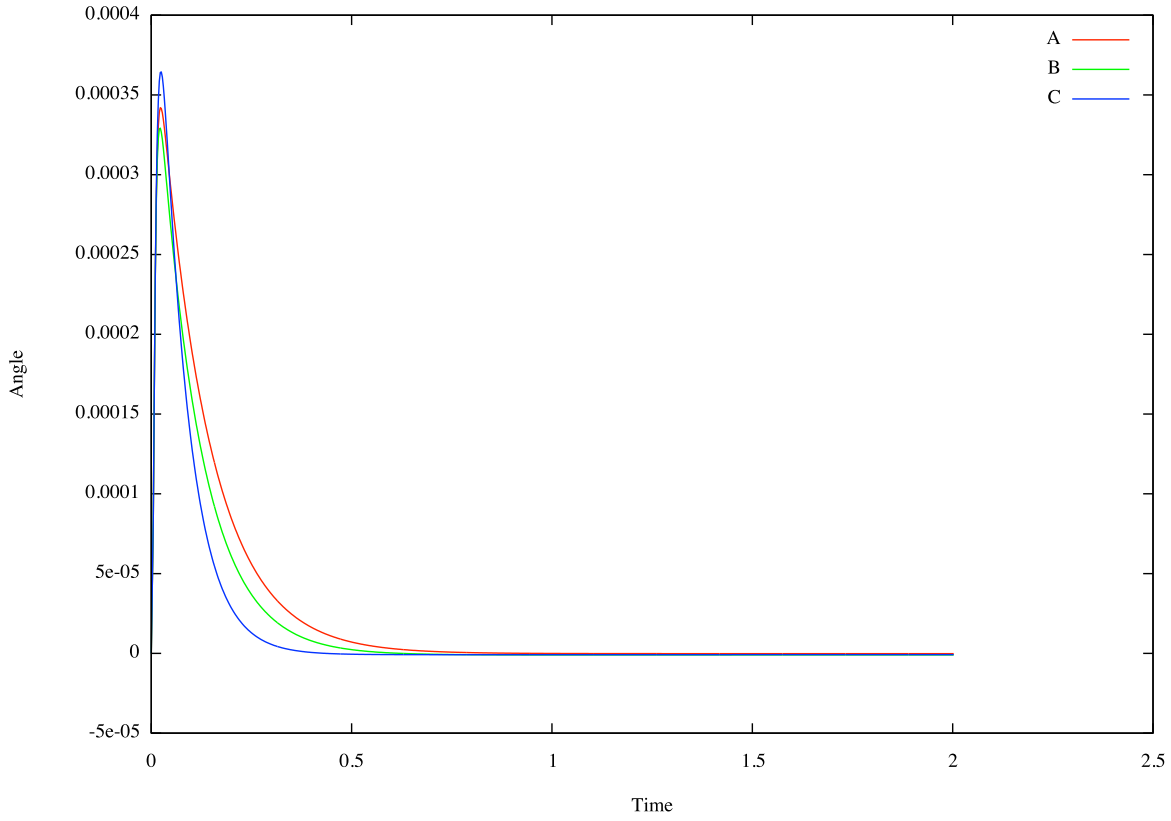


Figure 6.9: Impulse responses for the marked points in Figure 6.8

coding PID coefficients. The experiment was repeated 25 times and 95% confidence intervals of the response are plotted as bars. It is noteworthy that despite having a stochastic optimisation algorithm with only 1 generation to converge after a problem change, the length of the confidence interval is low. Compared to Figure 6.9, the dynamic controller has faster stabilisation time. The control interval δ_C of the dynamic controller is very small at 0.005s. The fitness vector for this particular problem can be calculated faster than 0.005s using a modern computer, i.e., it can run in real-time. In other problems with more difficult fitness functions this may not be the case. The control interval can be increased without sacrificing the system's stability. Figure 6.11 compares impulse responses of the inverted pendulum model

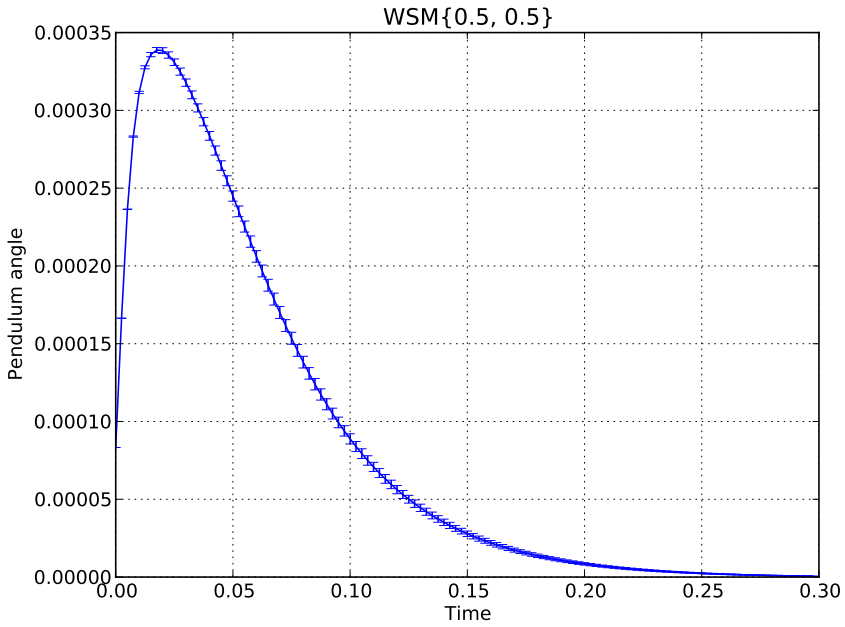


Figure 6.10: Impulse response of a dynamic controller. $T_p = 0.4$s, $\tau_T = 1$, $N_p = 10$, $\delta_C = 0.005$s, 25 tries, $cl = 95\%$. Note: the scale is compressed with respect to Figure 6.9.

with different control intervals. The controller with $\delta_C = 0.005$s can adjust the control strategy 60 times over the plotted interval, while two others with $\delta_C = 0.05$s and $\delta_C = 0.1$s only 6 and 3 times, respectively. Ultimately, all controllers are able to return to the upright position in 0.3 s, which is better than the static PID controllers. The controllers with larger δ_C achieve this by compensating early decisions with a more aggressive control strategy.

It is interesting to see how the Pareto front moves with time. Referring to Figure 6.12, which shows the Pareto front evolution with time for two experiments, the fronts are very close to each other in the beginning, just after the initial unit impulse has been applied. Correspondingly, the decision maker's choices lie close to each other. However, with time the two experiments start to diverge from each

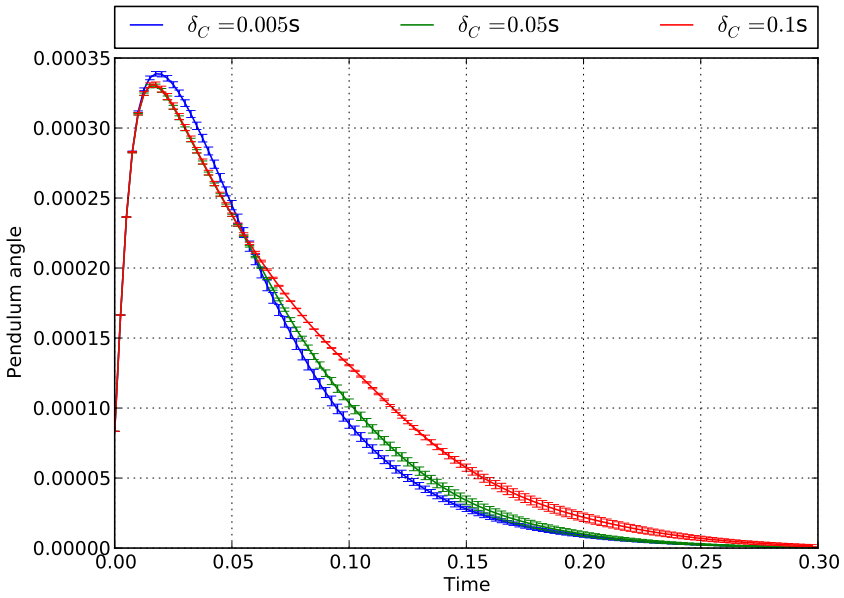


Figure 6.11: Impulse responses of a dynamic controller. $T_p = 0.4$s, $\tau_T = 1$, $N_p = 10$, 25 tries, $cl = 95\%$.

other, with corresponding decision maker's choices lying further away. The two experiments in Figure 6.12 produce different control strategies, yet, according to Figure 6.11, they result in a robust pendulum stabilisation with tight confidence intervals.

To calculate the fitness vector of the inverted pendulum problem, the MPC uses numerical integration of the internal model over prediction horizon T_p. The length of the prediction horizon is, therefore, directly proportional to the computational cost of the objective vector, so to make the fitness evaluation faster, one should lower the length of the prediction horizon. However, the smaller the prediction horizon is, the more divergent is the local fitness from the global one. The responses of the model with different prediction horizon lengths are compared in Figure 6.13 with matching motor force impulse responses in Figure 6.14. There is no difference between $T_p =$

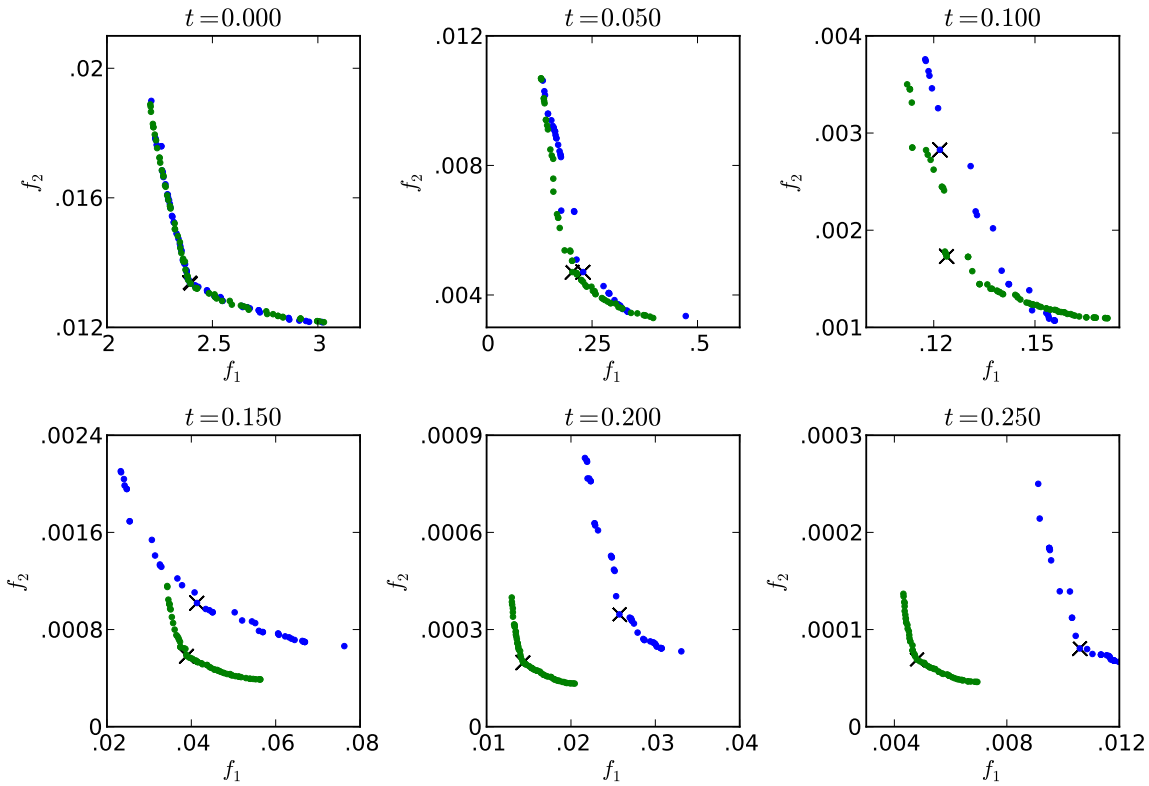


Figure 6.12: Pareto front movement with time for two experiments. Decision maker's choices are marked with 'X'. $T_p = 0.4$s, $\tau_T = 1$, $N_p = 10$, $\delta_C = 0.05$s. Note the progressively smaller scale of f_1 and f_2.

0.4s and $T_p = 0.3$s, as both have a prediction horizon length that is longer than the stabilisation time. A controller with $T_p = 0.2$s does not have to stabilise the pendulum in the early stages, so it can supply a more aggressive strategy to the control conditioner, resulting in a faster stabilisation time. Shortening the prediction horizon further starts to decrease the performance of the controller. In particular, a controller with very short prediction horizon $T_p = 0.05$s, despite reaching the upright position first, overshoots it and then has to apply three times positive force to compensate.

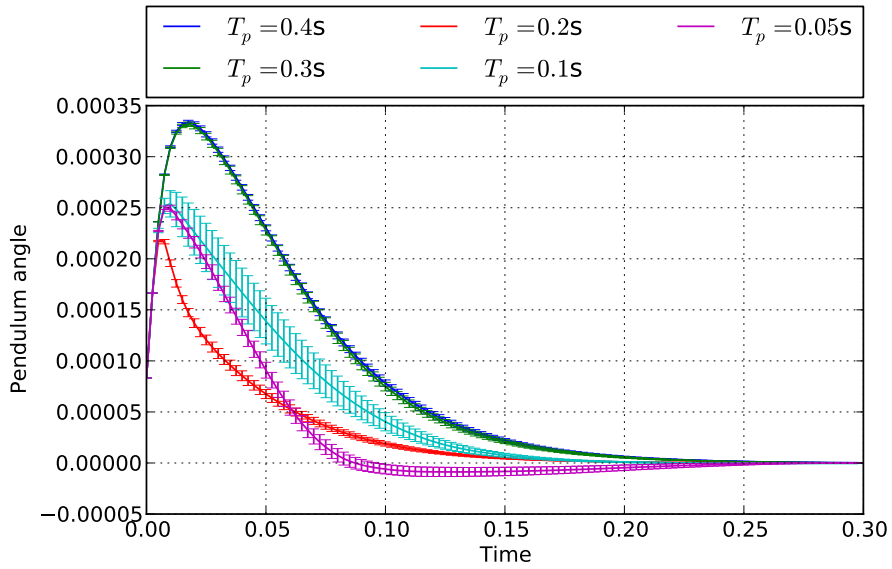


Figure 6.13: Effect of prediction horizon length on controller performance. Pendulum angle impulse response. $\delta_C = 0.01$s, $\tau_T = 1$, $N_p = 10$, 25 tries, $cl = 95\%$.

The impact of the prediction horizon length T_p (observed in Figures 6.13 and 6.14) is also affected by the type of the control conditioner. As the PID control conditioner depends not only on the proportional, derivative and integral gains but also on the process dynamics, it is not possible to achieve arbitrary response curve. Controllers that have prediction horizon length longer than the pendulum stabilisation time are additionally restricted by the control conditioner. To counter the restrictions placed on the objective space by the PID control conditioner, a prediction horizon and chromosome partitioning scheme (as described in Section 3.4) can be applied.

Figure 6.15 contains a comparison of impulse responses similar to the one in Figure 6.13, but with the prediction horizon T_p partitioned into smaller segments $T_{pi}, i \in [1, n]$. The length of the segment T_{pi} is selected to be equal to the control interval δ_C, i.e., $T_{pi} = \delta_C$. Each segment is encoded with $k = 3$ gene values, which correspond to the three gains of the PID control conditioner. Therefore, the overall

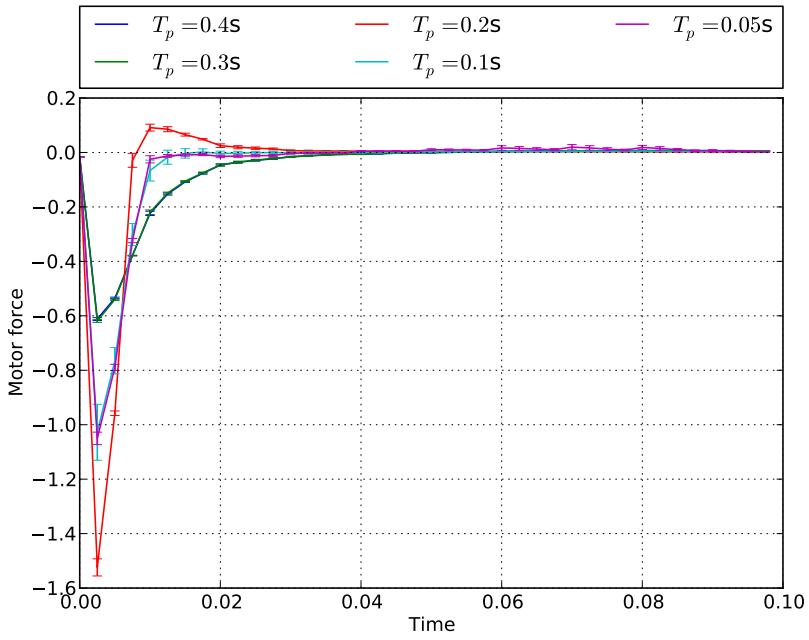


Figure 6.14: Effect of prediction horizon length on controller performance. Motor force impulse response. $\delta_C = 0.01$s, $\tau_T = 1$, $N_p = 10$, 25 tries, $cl = 95\%$.

length of the chromosome is $l = kT_p/T_{pi}$. The chromosome is shifted by k genes to the left each T_{pi} seconds. To compensate for the increased chromosome length, the number of pre-execution generations N_p and the number of fixed generations τ_T are increased. The results show that given enough time to converge, the variable prediction horizon strategy is able to overcome limitations of the PID control conditioner and, consequently, outperform the approach presented in Figure 6.13. The performance difference between the various prediction horizon lengths is minimal. The low performance of the algorithm with $T_p = 0.4$s is because of the very long chromosome of 120 genes; it requires more time to converge than was allotted. The unsatisfactory convergence explanation is supported by the increased width of the confidence interval. The performance of the algorithm with $T_p = 0.4$s could be increased by either increasing the number of pre-execution generations N_p or the

number of fixed generations τ_T, as evidenced by Figure 6.16.

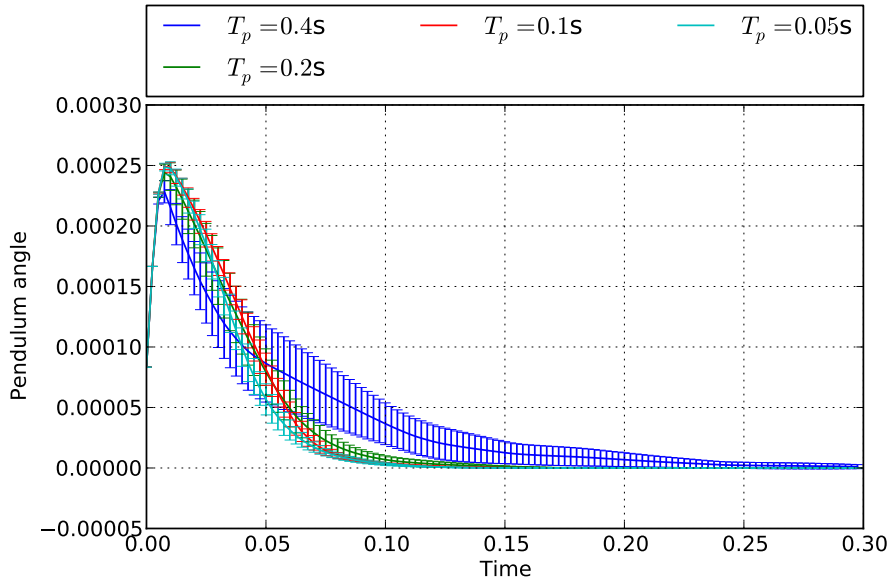


Figure 6.15: Prediction horizon partitioning into segments. Impulse responses of a dynamic controller. T_p is divided into $n = T_p/\delta_C$ segments. $\delta_C = 0.01$s, $\tau_T = 10$, $N_p = 50$, 25 tries, $cl = 95\%$.

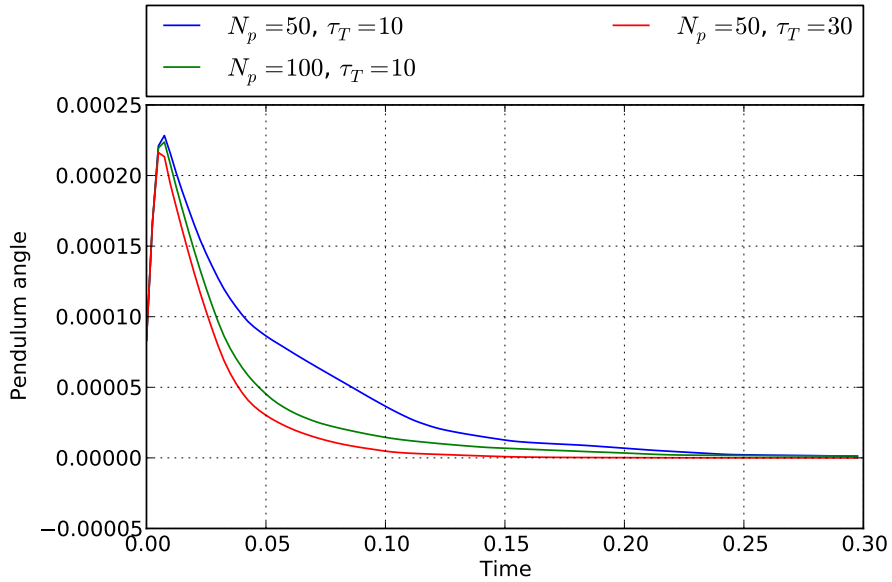


Figure 6.16: Effect of MOEA convergence on controller performance. Impulse responses of a dynamic controller. T_p is divided into $n = T_p/\delta_C$ segments. $\delta_C = 0.01$s, $T_p = 0.4$s, 25 tries, $cl = 95\%$.

6.3.4 Decision makers

The critical role of the prediction horizon length can be seen when comparing the WSM decision makers with different sets of weights. Figure 6.17 shows a comparison of EMO MPC controllers with $T_p = 0.4$s using WSM decision maker with different weights. As the controllers are operating with the global fitness, the stability of the results is not compromised. However, the efficiency of the $\{0.1, 0.9\}$ control strategy, which emphasises fast return to the upright position is reduced compared to the case with shorter prediction horizon $T_p = 0.2$s, shown in Figure 6.18. At the same time, some of the impulse responses in Figure 6.18 are unstable, showing that a control strategy with a strong focus on reducing the motor force coupled with a short prediction horizon may become unstable without explicit stability constraints.

The situation is no different with a random decision maker. Figure 6.19 displays the results of a random decision maker with $T_p = 0.4$s, $T_p = 0.2$s and $T_p = 0.1$s. The number of tries was increased to 100 to get tighter confidence intervals. The results show that the random controller with $T_p = 0.4$s behaves similarly to the WSM $\{0.6, 0.4\}$ controller. With a shorter $T_p = 0.2$s prediction horizon, the performance of random controller degrades and variability between individual attempts rises. However, the controller remains stable in contrast to some WSM combinations. Both random and WSM are capable of choosing any point from a convex Pareto front, but WSM that chooses consistently from one extreme end of Pareto front may go unstable. The performance degrades more drastically with $T_p = 0.1$s, especially regarding the overshoot and consecutive correction phases. Also, the pendulum continues to make small swings around the upright position once the setpoint is attained.

A marked improvement over the WSM and random decision makers can be obtained by using the Pareto Decision Tree (PDT) decision maker. A three point

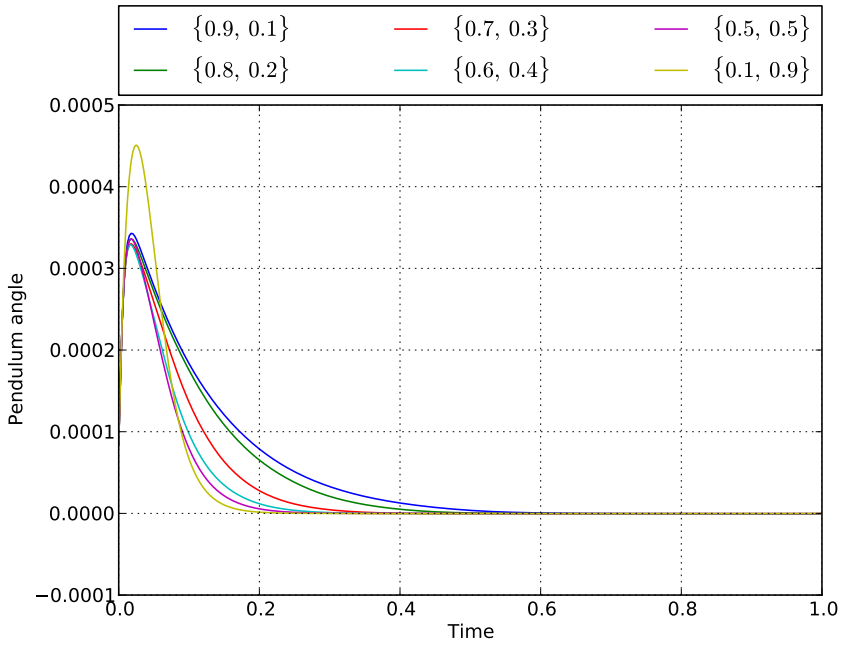


Figure 6.17: Mean pendulum angle. Decision making using weighted sum model with different weights. $T_p = 0.4$s, $\tau_T = 1$, $N_p = 10$, $\delta_C = 0.01$s, 25 tries.

PDT is used, i.e., each generation the decision takes place using tournament selection with hypervolume as the utility measure. Figure 6.20 shows the difference in impulse responses for a PDT decision maker with varying prediction horizon lengths. It can be seen that PDT is more tolerant to the changes in T_p than both WSM and random decision makers. Also, PDT outperforms the random decision maker for all T_p lengths.

6.3.5 Control conditioners

PID control conditioner is simple to implement and, because it receives direct feedback from the process dynamics, it can tolerate modelling imprecisions and noise. However, PID reliance on process dynamics restricts it to a subset of possible control strategies. It can be argued that a B-spline based control conditioner can be more

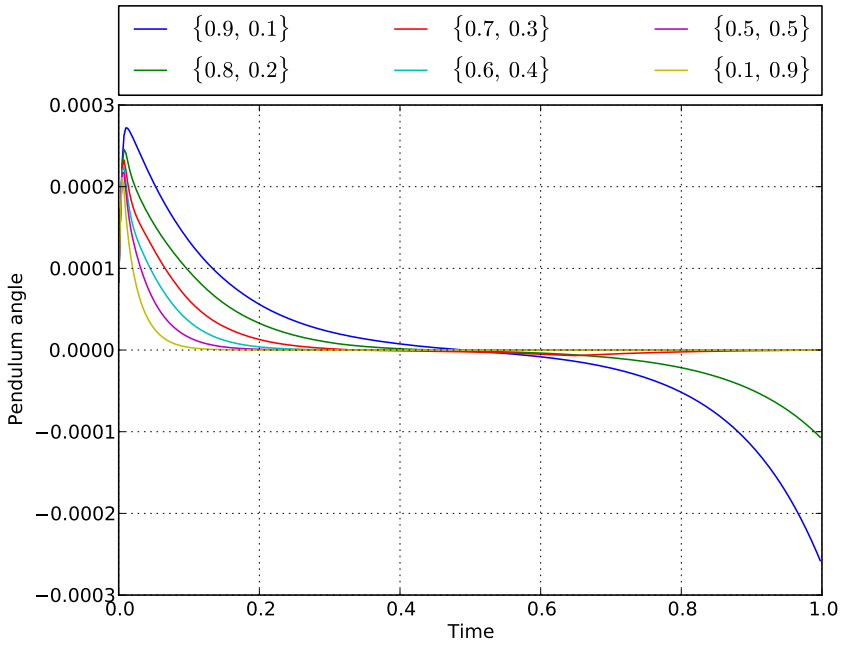


Figure 6.18: Mean pendulum angle. Decision making using weighted sum model with different weights. $T_p = 0.2$s, $\tau_T = 1$, $N_p = 10$, $\delta_C = 0.01$s, 25 tries.

effective than the PID control conditioner. A B-spline based control conditioner encodes the control effort directly using a set of control points. This gives a spline based control conditioner another important advantage over PID, namely, the ease of setting the control effort range. Figure 6.21 shows the impulse response of the EMO MPC controller with B-spline and PID control conditioners. Compared to the PID control conditioner, the performance of the spline based control conditioner might not seem better, but it manages to produce smaller maximum deflection from the upright position. Without receiving direct feedback from the plant, the performance of the spline based control conditioner is limited in the upright position due to the modelling uncertainty.

The PID control conditioner performs well with small deflections from the upright position. However, if the initial deflection is larger, this might not be the case.

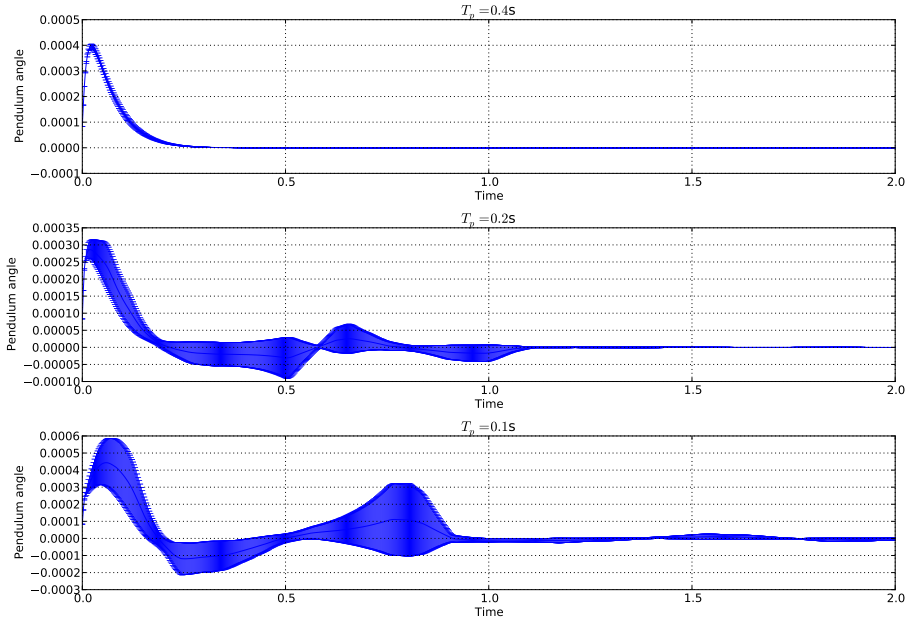


Figure 6.19: Mean pendulum angle. Decision making using random selection. $\tau_T = 1$, $N_p = 10$, $\delta_C = 0.01$s, 100 tries, $cl = 95\%$.

Figure 6.22 shows the pendulum angle and motor force graphs for the case where the initial unit impulse duration is 70 times more than in Figure 6.21. As can be seen, the B-spline based control conditioner raises the pendulum into the upright position much faster than the PID method.

6.3.6 Additive noise

To evaluate the piecewise aggregation approach detailed by Equation 3.2, the EMO MPC approach was tested with different values of λ. Figure 6.23 shows the mean impulse response of the EMO MPC controller for different values of λ parameter. With no noise, the response curves stay very close to each other, with the strategy having $\lambda = 1$ marginally outperforming the rest.

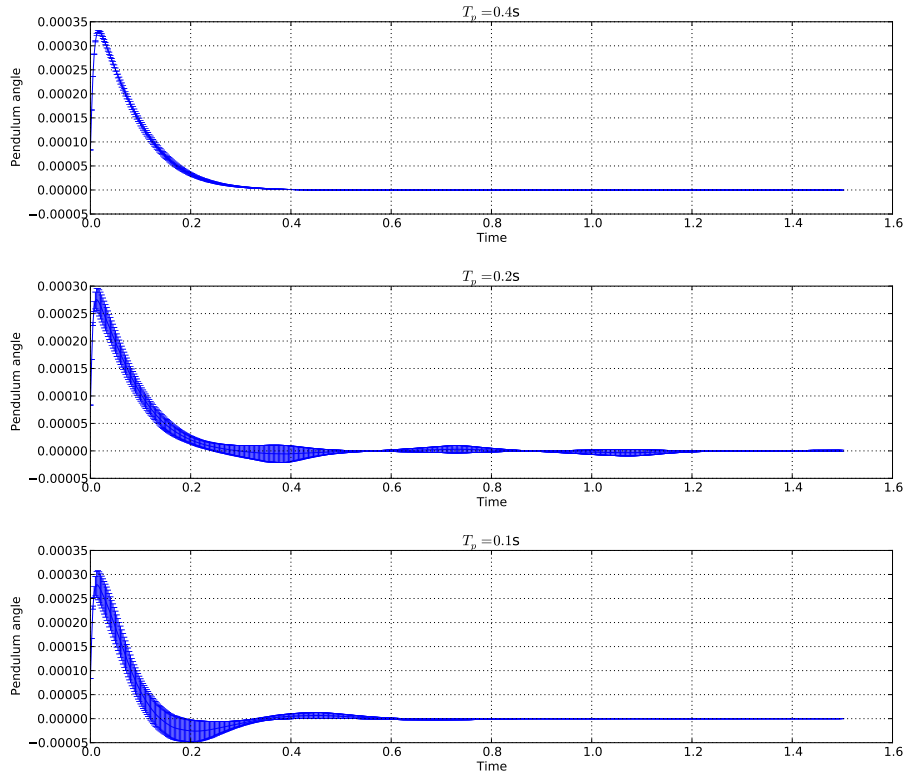


Figure 6.20: Mean pendulum angle. Decision making using PDT decision maker. $\tau_T = 1$, $N_p = 10$, $\delta_C = 0.01$s, 25 tries, $cl = 95\%$.

To increase the level of uncertainty, let us introduce additive noise in the objective vector. Figure 6.24 shows the mean impulse response of the EMO MPC controller with Gaussian noise $(\mu, \sigma) = (0, 0.07)$ added to the objective vector. In this scenario, the response curves are clearly differentiated by λ. The strategy with $\lambda = 2$ outperforms the rest, with the strategy having $\lambda = 3$ in the second place. The observed behaviour matches with the theory, i.e., an increase in uncertainty levels in a problem requires a correspondingly larger value of λ.

6.3.7 Comparison with model predictive control

Model predictive control has been applied to the inverted pendulum problem many times [160, 152, 161]. It is interesting to see how the EMO MPC algorithm com-

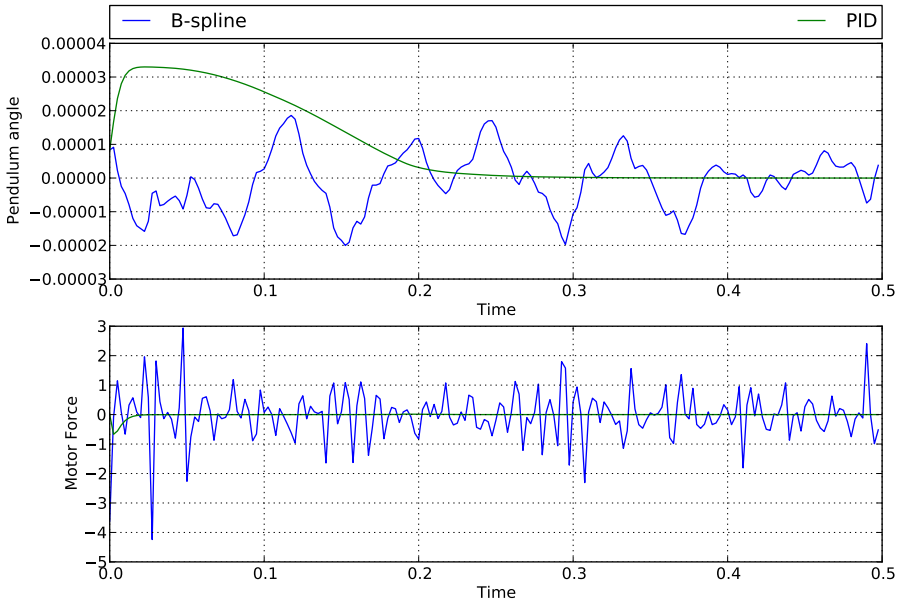


Figure 6.21: Impulse response. Pendulum angle and motor force. Decision making using WSM with $\{0.01, 0.99\}$ weights. $\tau_T = 10$, $N_p = 50$, $\delta_C = 0.01$s, $T_p = 0.1$s.

pares to regular MPC controllers. In [161], nonlinear MPC swing-up and stabilising controllers for the inverted pendulum problem are proposed. Table 6.3 summarises the inverted pendulum parameters used in [161]. Initially the pendulum is in the lower equilibrium (stable) position. The task of the controller is to swing up the pendulum and stabilise it in the upper equilibrium position.

Figure 6.25 shows the original results from [161] for the best performing scenario with a 25 step prediction horizon and the input constrained to $u(t) \in [-15, 15]$. The corresponding performance from the EMO MPC controller with a B-spline control conditioner is shown in Figure 6.26. It can be seen that the performance of the EMO MPC controller closely matches the nonlinear MPC controller presented in [161], despite the approach in [161] not having a modelling uncertainty. Despite very different motor force and cart position profiles, both controllers are able to

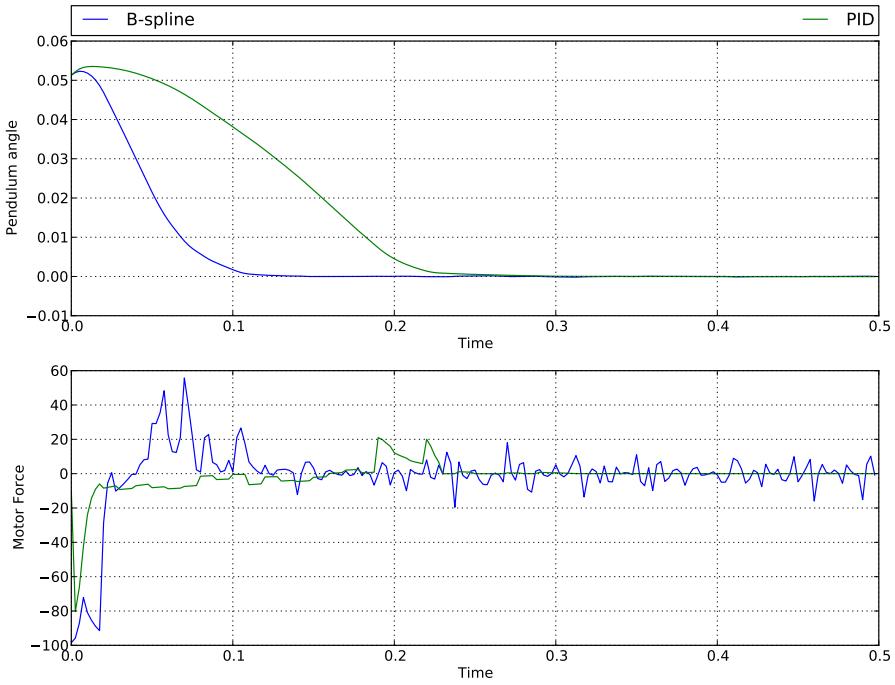


Figure 6.22: Pendulum angle and motor force. Decision making using WSM with $\{0.01,\ 0.99\}$ weights. $\tau_T = 10$, $N_p = 50$, $\delta_C = 0.01$s, $T_p = 0.1$s. Motor force is limited to 100N.

stabilise the inverted pendulum around the same time. However, the cart force demand is far less aggressive with EMO MPC than non-linear MPC.

6.4 Summary

This chapter presents an evaluation of the EMO MPC approach using the problem of stabilising an inverted pendulum. It was observed that for this particular problem the EMO MPC controller is able to stabilise the inverted pendulum in real-time, despite the modelling uncertainties. The performance of the proposed approach is better than a PID controller and comparable to existing nonlinear MPC controllers.

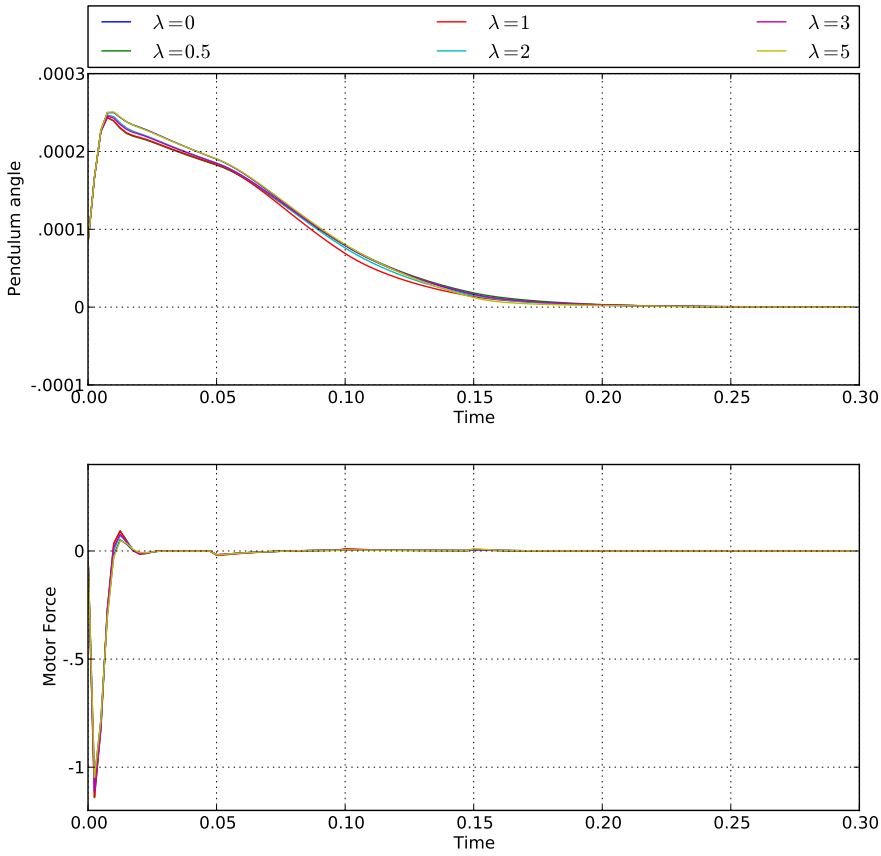


Figure 6.23: Impulse response with varying λ. Pendulum angle and motor force. Decision making using WSM with $\{0.5, 0.5\}$ weights. $\tau_T = 10$, $N_p = 50$, $\delta_C = 0.05$s, $T_p = 0.2$s, 25 tries.

Open loop and closed loop control conditioners were evaluated. It was found that the steady state performance of the closed loop PID control conditioner is better than a spline based one. This can be explained by the inherent benefit of the feedback provided by the PID control conditioner. However, the spline based control conditioner reaches the steady state much faster than the PID control conditioner, as it can explore more of the decision space. Also, the spline based control conditioner provides an easy way to place limits on the possible values of the selected Pareto-

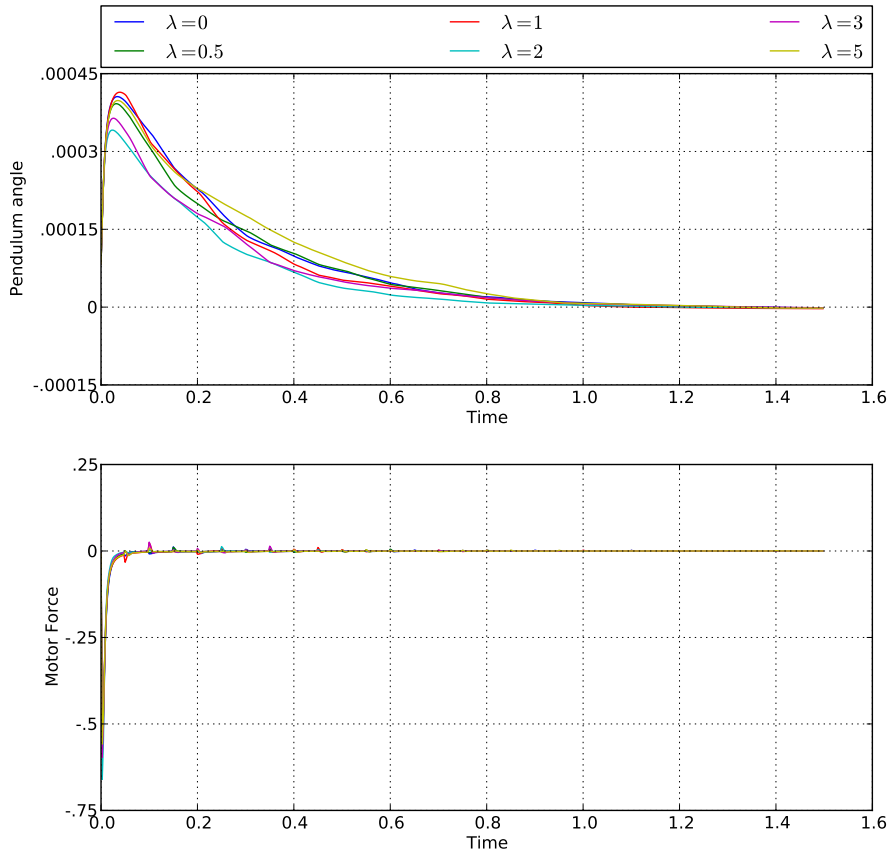


Figure 6.24: Impulse response with varying λ and additive Gaussian noise $(\mu, \sigma) = (0, 0.07)$. Pendulum angle and motor force. Decision making using WSM with $\{0.5, 0.5\}$ weights. $\tau_T = 10$, $N_p = 50$, $\delta_C = 0.05$s, $T_p = 0.2$s, 25 tries.

optimal control strategy.

By comparing different dynamic *a posteriori* decision makers, it was found that the results obtained by the WSM decision maker on the inverted pendulum problem are always stable if the multi-objective search is using a global fitness vector. With local fitness, the stability of the inverted pendulum depends on the WSM weights. In contrast to the static optimisation case, a random decision maker produces stable Pareto-optimal control strategies for inverted pendulum stabilisation. A PDT con-

Parameter	Value
Pendulum mass m_2	0.5 kg
Cart mass m_1	3 kg
Rod length l	0.5 m
Cart position y_1	0 m
Cart speed v_1	0 $^{\mathrm{m}}/_{\mathrm{s}}$
Pendulum angle y_2	π rad
Pendulum speed ω	0 $^{\mathrm{rad}}/_{\mathrm{s}}$
Time t	0 s
Motor force F	0 N

Table 6.3: Inverted pendulum model parameters for the swing-up case [161]

troller outperforms a random decision maker at all prediction horizon lengths. In addition, the PDT controller is more tolerant to the changes in prediction horizon length than both WSM and random decision makers. The effect of PDT can be seen as a stability-enforcing measure, as current decisions causing instability lead to a deterioration of future Pareto fronts and therefore would not be preferred by PDT.

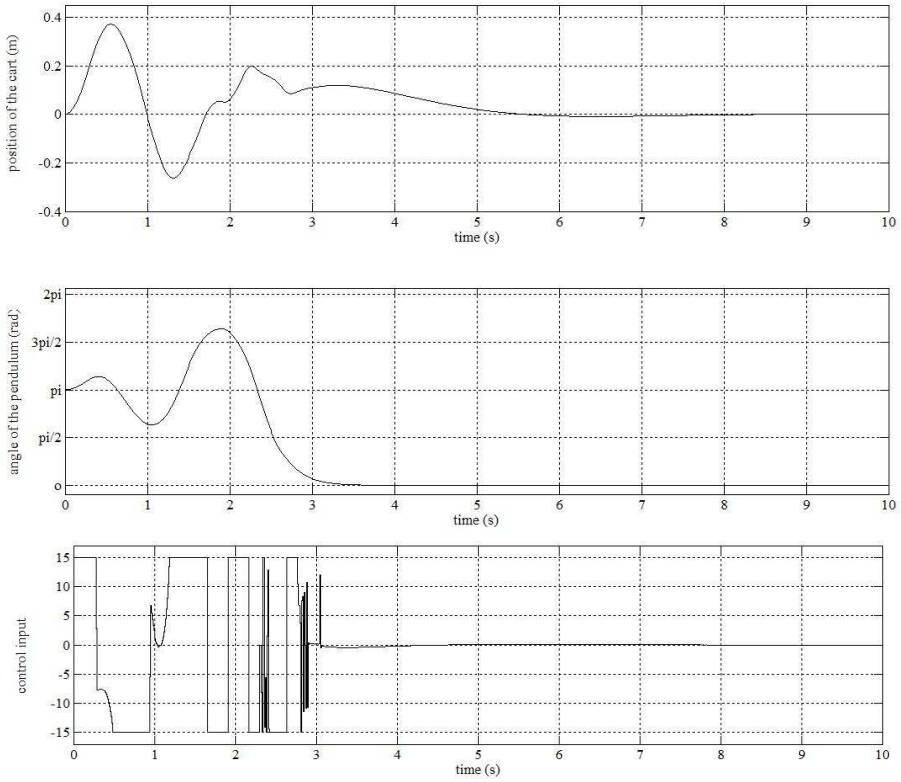


Figure 6.25: Swing-up and stabilisation of the inverted pendulum. Input is constrained to $u(t) \in [-15, 15]$. 25 step prediction horizon. Figure from [161].

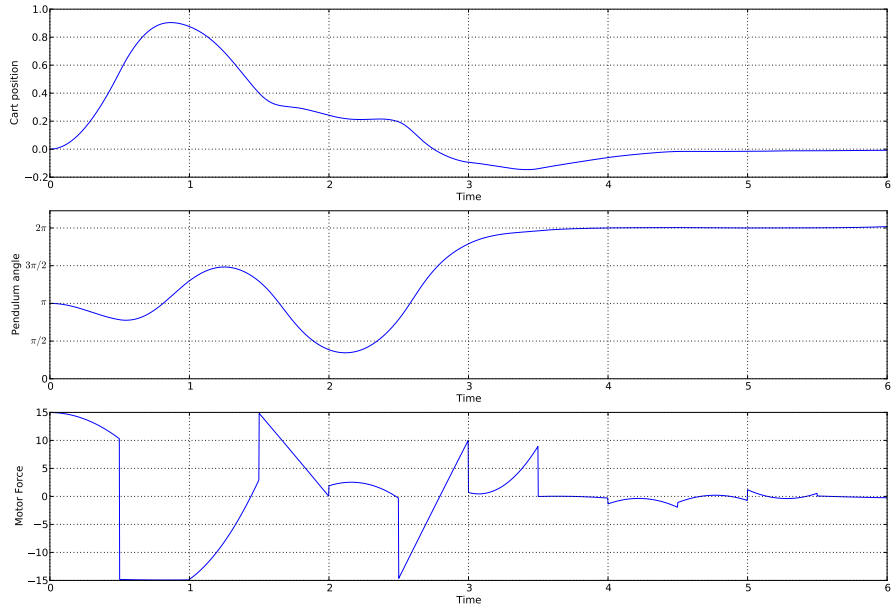


Figure 6.26: Swing-up and stabilisation of the inverted pendulum. Input is constrained to $u(t) \in [-15, 15]$. $\tau_T = 10$, $N_p = 50$, $\delta_C = 0.5$s, $T_p = 1$s

Chapter 7

Multi-agent systems

The key word in coordinated control of multiple entities is 'coordination'. The concept of coordination means that a particular set of intelligent agents has some interaction ties between them with an appropriate interaction protocol. On a general level, the agents are forced to interact because of goals and resources. A goal is a performance objective that an agent or a group of agents is trying to maximise. The former is an individual goal and the latter is a common goal. Resources can be individual (i.e., those that are being allotted to a single agent) or common (that are used by multiple agents cooperatively, e.g., on a time-sharing basis).

The primary application focus for this research is on the shop floor and transportation grid planning tasks, where the agents operate with individual goals using common resources. This chapter outlines the design of a Multi-Agent System (MAS) with intelligent agents that use DMOEA and a dynamic *a posteriori* decision maker to guide their actions.

7.1 Problem representation

In order to facilitate the applications of MAS, the problem has to be represented in an appropriate way. A representation of the problem provides means to capture, refine and communicate the problem-specific knowledge and to express it in a formal

way suitable for application of the EMO MPC approach. This section presents a complete hierarchical representation of the problem that conforms to the following objectives:

- formalise inter-agent communication,
- visualise system structure and data flow, and
- express system design in a formal way suitable to the application of MOEA,

The objectives of the representation task are very diverse and arguably could not be easily achieved using a single representation technique. Therefore, the proposed approach includes two different complementary representations, graphical and parametric. The primary tasks of the graphical representation are to visualise the system design and facilitate system decomposition for modelling purposes. The parametric representation expresses system components through sets of parameters that include decision variables and performance objectives.

The graphical representation contains two orthogonal views of the system, namely the *composition diagram* and *process model diagram*. The composition diagram shows the overall structure of the system and its components and the principal activity vectors in the system. In the composition diagram, entities are depicted as boxes and the activity vectors as arrows.

The process model diagram shows a sequence of processes, actions and decisions within the system and its elements. Both diagram types include several abstraction levels. Abstraction levels are marked with incremental numbers starting from 0. It is assumed that a diagram at level $n + 1$ provides with a more detailed information than the one at level n.

At the top level, a MAS consists of a number of agents and the environment. Referring to Figure 7.1, the system contains a set of agents $A = \{a_1, a_2, ..., a_n\}$ with two primary operations, *act* and *observe*, defined for each agent. The actions of

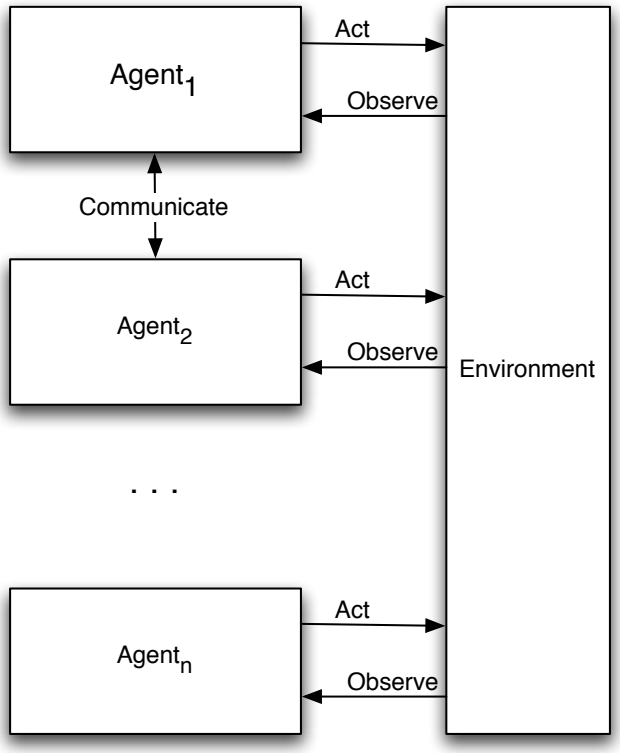
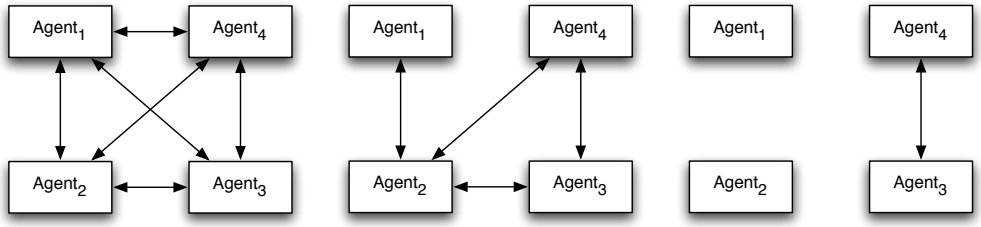


Figure 7.1: Level 0: composition diagram of the system.

the agents result in changes in the environment. The environmental changes affect future actions of the agents by means of observations performed using sensors. This simple model corresponds to the model presented in [1]. An important difference in the present work is the addition of another operation, *communicate*, for the agent. The communication operation is designed to allow the agents to exchange the information about projected resource supply and demand, therefore enabling adaptation and harmonising resource usage (a similar approach is used in [71, 162] for swarm guidance). Communication between agents happens using topologies presented in Figure 7.2.

By adding more details to the picture, the agent becomes a complex entity. Figure 7.3 shows a composition diagram of an agent at level 1. It can be seen that the



(a) Fully meshed (b) Partially meshed (c) Disconnected

Figure 7.2: Different topologies of communication between agents.

agent contains separate functional blocks that correspond to the three main operations defined for an agent at level 0. The fourth block is what enables the agent to act rationally and autonomously. In other words, it contains the "intelligence" of an agent. The following presents a description of the main components of an "intelligence" block. An agent receives information from two primary inputs, namely, the array of sensors and the communication channel with other agents. This information is merged by the unification procedure to produce a coherent view of the outside world. With the help of an internal model, the agent tries to predict the future state of the environment and a corresponding action by using two distinct algorithms, default heuristic and an EMO MPC. The EMO MPC is the agent's primary means to adapt to the environment and other agents' actions and act rationally. The role of the default heuristic is to provide a failsafe baseline solution that can be used if for some reason the EMO MPC solution proves unsatisfactory, there is no solution at all or the produced solution is infeasible. The existence of the default heuristic guarantees that the real-time requirements of the system are met at any time.

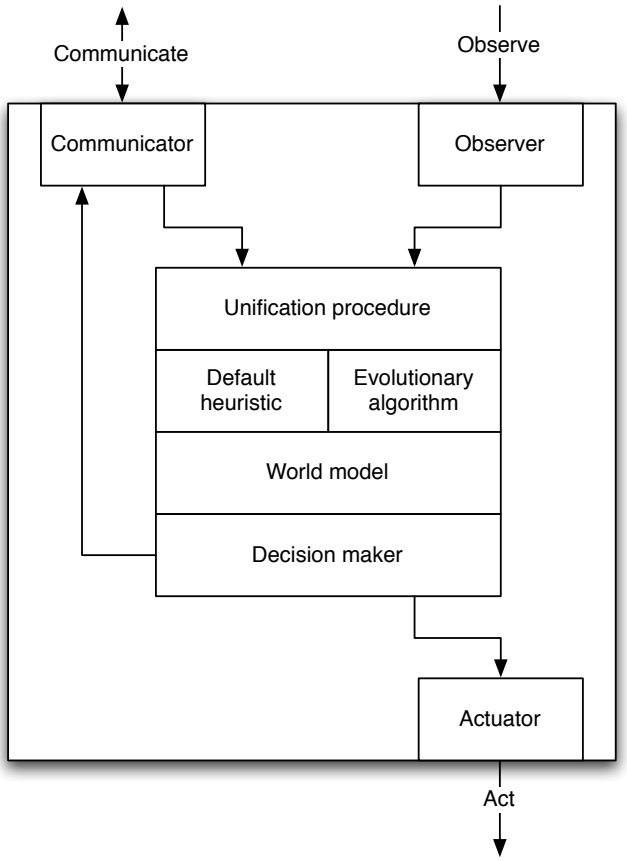


Figure 7.3: Level 1: composition diagram of an agent.

7.1.1 Process model

The preceding discussion introduced the structure of an agent. However, the structure does not tell a full story about the agent. It remains unclear what steps the agent follows in order to arrive to an action. Therefore, a process model diagram of an agent is provided in Figures 7.4 and 7.5. The diagram presents a sequence of processes occurring inside an agent during an iteration. An agent's intelligence consists of two processes running alongside each other, with the main process shown in Figure 7.4. It can be seen that at the beginning of each cycle the agent collects information about the environment and also receives incoming transmissions from

other agents. After that, it merges the information into a coherent view of the outside world. Following the merger of the input information, the agent executes the default heuristic procedure. A strategy is first selected from an internal knowledge database. Using the strategy the agent conducts experiments with an internal world model. For each experiment conducted, the agent evaluates constraints. The process continues repeatedly until the agent conducts an experiment that satisfies all constraints. Note that this condition mandates the existence of "fail-safe" solution in the agent's knowledge base. Finally, when all constraints are satisfied, the agent proceeds to evaluate the performance of the solution.

The EMO MPC process runs in parallel to the main agent's process. The communications between the two are asynchronous, ensuring that the real time response constraint of an agent is always satisfied. Within the scope of the main agent process, the agent only fetches the current solution produced by EMO MPC and compares it with the solution obtained by means of the default heuristic. The dominant solution then proceeds to be implemented using actuators and the projected resource usage associated with the solution is communicated to other agents. In order to understand each other, the agents need to share a common *ontology*, i.e., a complete formal representation of all possible concepts within the communicated domain. In this particular case, the ontology contains all possible resource identifiers and means to express temporal availability of the resources.

As mentioned above, the MOEA runs in a process separate from the main agent process. Referring to Figure 7.5, the MOEA starts its iteration by fetching the current world state from the main process. The algorithm then proceeds to evaluate the current environmental state and shift the chromosome, if necessary. Section 7.1.3 explains the purpose and mechanics of the state changes and associated chromosome shift in detail. Following that, the algorithm executes several breeding cycles that consist of mutation, crossover, selection and ranking to develop a set of Pareto-

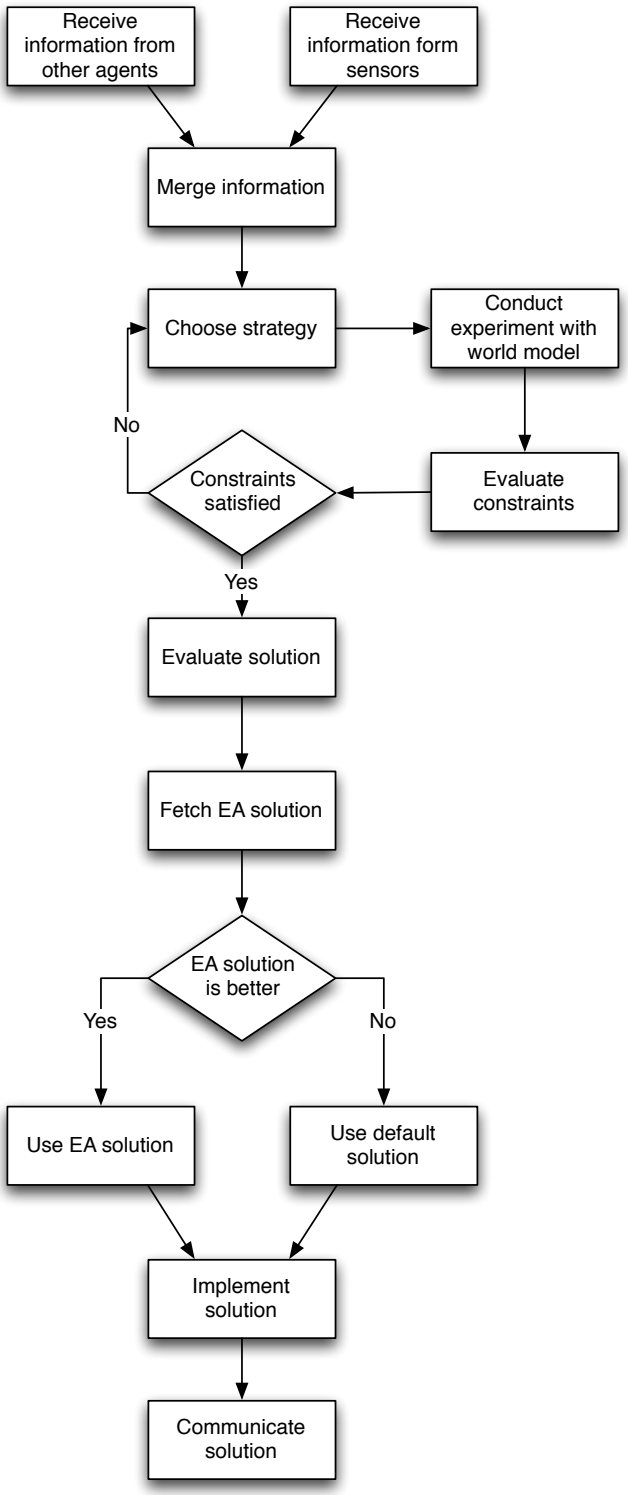


Figure 7.4: Level 1: process diagram of an agent. Main process.

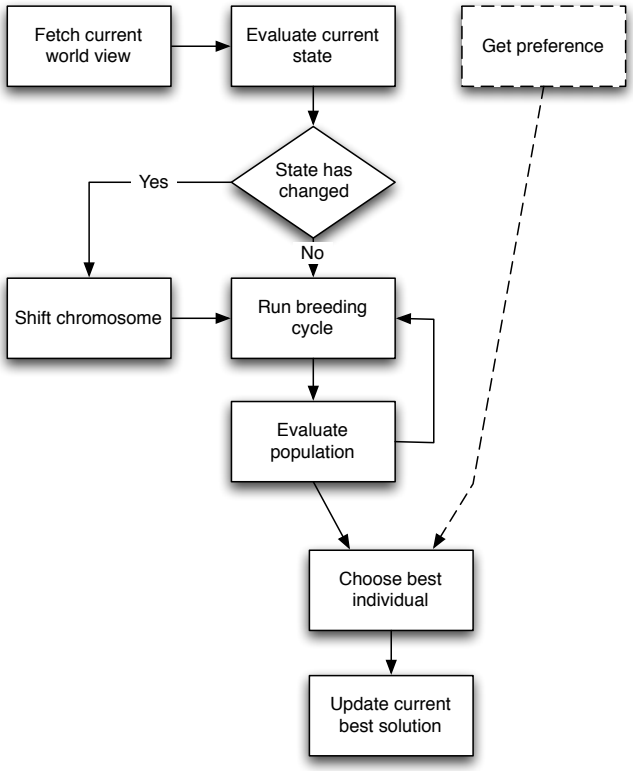


Figure 7.5: Level 1: process diagram of an agent. Evolutionary algorithm.

optimal solutions. A single control action has to be selected from the Pareto set in the next stage. To make a decision, the algorithm may have to rely on optional preference information (either internal or external preference) supplied to the agent. Another option is to use a decision maker that does not require preference information, such as PDT. Because the EA runs in parallel to the main agent process and the EA itself does not have a fixed response time, the obtained solution is then stored as a publicly accessible attribute to enable it to be fetched by the main agent process.

7.1.2 Coevolution of agents

In the proposed framework, multiple agents co-evolve a set of strategies to compete for limited resources. Similar co-evolution techniques have been used in [71, 73]. The agents play a MAX game, i.e., they try to maximise their own gain without minimising the gain of other agents. Consequently, the strategy of an agent has to be evaluated only against the current best strategy selected by other agents.

In order to evaluate a candidate strategy, an agent has to model the actions of other agents using the information about their best strategies that is being broadcast. Based on predicted actions of other agents, the agent can predict the future resource usage and use it as a set of constraints for its current strategy.

In practice, the computational load on an agent that uses the aforementioned approach to evaluate its own strategy grows proportionally to the number of other agents. The computational cost can be reduced bearing in mind that a) an agent needs to know the availability of the resources required for its current control strategy only b) each agent has to model its own strategy and c) the agents are not interested in minimising the gain of other agents. An improved approach makes the agents broadcast their projected resource usage as a function of time instead of the actual action. Therefore, the constraints on evaluation time for a particular control strategy do not depend on the number of other agents in the environment.

7.1.3 Uncertainty vectors and chromosome shift

The agents are naturally uncertain about a number of things because of the limited observation range, sensor precision and changing actions of other agents. This section introduces uncertainty vectors, i.e., directions in which the degree of uncertainty changes in a predictable way. Time vector $\vec{T}$ defines a reduction of uncertainty in time. In other words, the confidence of an agent in its view of the world is inversely proportional to the time left before a decision is required. The space vector $\vec{S}$ de-

notes that uncertainty also decreases in space. In relation to the resources required for an action, the confidence of an agent that a certain resource will be available is inversely proportional to the number of agents in the supply chain or a demand queue on the resource.

In on-line processing, the time allotted for the EA to run is comparatively small. At the same time, the EA as used in the proposed framework is subjected to uncertainty that reduces the convergence rate of the algorithm. An initial population that is tuned to the current environment significantly improves the response time of the EA. Therefore, it is important to exploit possibilities offered by uncertainty vectors. A technique is proposed whereby an uncertainty vector is partitioned into a number of so-called "states". A partitioning scheme is selected to define these states such that an agent's action sequence within a state would be similar to the action sequence within the next state and so on. The possibility of such partitioning follows from the typical grid or lattice-like structure of the MAS under consideration that naturally contains repetitive regions resulting in repetitive agents' actions. After the uncertainty vector is split, typical decision variables are transformed to a vector with the size less than or equal to the number of partitions in the uncertainty vector. Each element of the decision variable vector contains the value of the decision variable that corresponds to a particular state in the uncertainty vector. The proposed composition of the decision variable vector bears an important property that the uncertainty in the decision variable vector follows the rules set for the uncertainty vector.

The uncertainty vectors of a MAS map to the variable prediction horizon scheme of EMO MPC detailed in Section 3.4. The extended decision variable form is accommodated using an additional operation in the EA process of an agent. Consider Figure 7.5, where the EA constantly compares the current environmental state with the state recorded in the uncertainty vector. If the state stays the same, no changes

to the chromosome are required. However, if a change in the state is detected, the chromosome is shifted left by the number of genes corresponding to the system's state, and the rightmost elements in the decision variables are filled with random values.

To illustrate the idea of uncertainty vectors, consider an example of a Flexible Manufacturing System (FMS) in which the capabilities of machines partially overlap [163]. Referring to Figure 7.6, the system consists of a number of resources (machines) $M = \{M_1, M_2, ..., M_m\}$ that are being fed with jobs $J = \{J_1, J_2, ..., J_n\}$. A number of operations $o_{j,k}$ need to be performed with each job J_j, consequently, each job is identified by a set of its operations. Likewise, each resource M_i is identified by a set of operations $o_{j,k}$ that it can perform. If a resource M_i is able to process an operation $o_{j,k}$, it has an associated processing time $p_{i,j,k}$.

Suppose each machine is an intelligent agent that has to plan its load according to some performance criteria, e.g., throughput. Provided the machine has a long First In, First Out (FIFO) queue of jobs, it is certain about the processing targets of the oldest jobs in the queue, i.e., those that would be processed first. Meanwhile, the schedule of more recent jobs contains more uncertainty, as the production plans of other machines may change and, consequently, the schedules would have to be rearranged. It is easy to see that the uncertainty in the processing target of a job increases with the number of preceding jobs in a queue. A variant of the partitioning technique may encode schedules for individual jobs sequentially into a chromosome and shift the genes after a job has been processed.

7.2 Summary

This chapter presents a novel approach to solving complex shop floor scheduling and grid planning problems on-line by using MAS with EMO MPC based intelli-

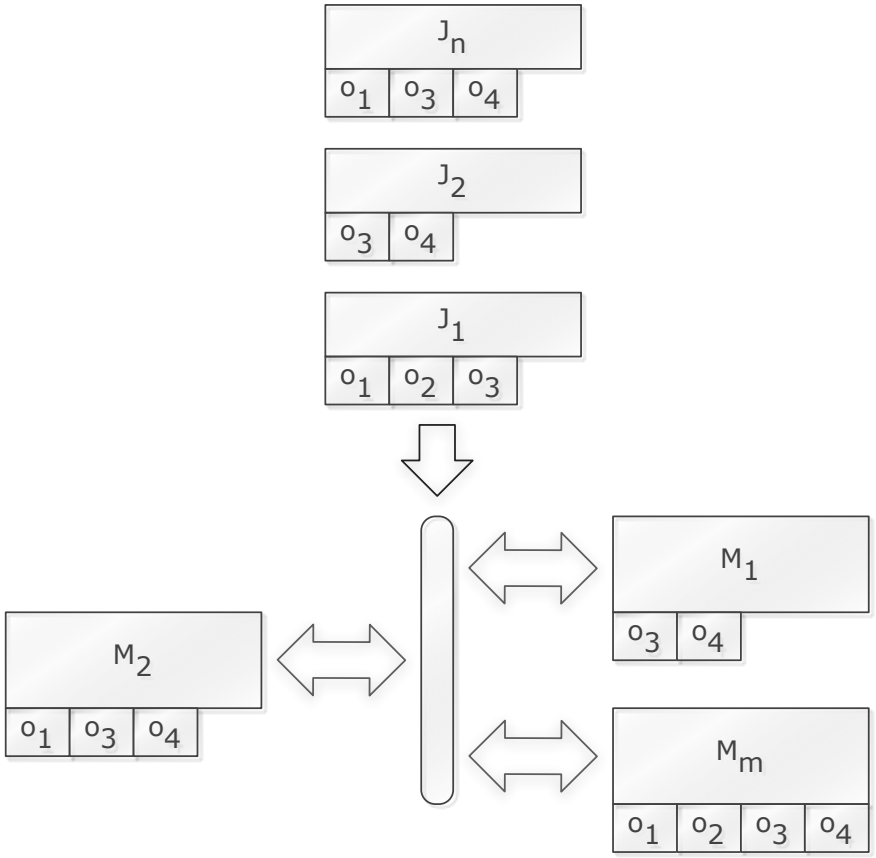


Figure 7.6: Flexible manufacturing system.

gent agents. The real-time constraint inherent to on-line applications is addressed through a default heuristic process running in parallel to the EMO MPC controller. The solutions obtained by the default heuristic are used as a backup if the EMO MPC solution in unfeasible or unsatisfactory.

The coevolution of the agents implies that each agent must be able to take into account the actions of other agents. The proposed MAS defines a concept of shared resources that describe the interaction between agents. Using the shared resource concept, an agent has to broadcast its intended resource usage. Other agents are using the information about availability of the shared resources as constraints to the

DMOEA.

The proposed MAS contains special provisions for handling uncertainty inherent to on-line scenarios in a multi-agent environment. The parameters containing uncertainty are arranged in uncertainty vectors, which are partitioned according to system states. The uncertainty vector arrangement provides a mapping to the variable prediction horizon scheme of EMO MPC.

Chapter 8

Case study: traffic scheduling

This chapter provides an assessment of capabilities and performance evaluation of the proposed framework using a case study.

8.1 Description

The case study presented in this chapter belongs to the category of scheduling problems and describes a problem of traffic optimisation in a rectangular grid-organised traffic network. The case study is inspired by [164]. Referring to Figure 8.1, which displays the elements of a traffic grid, the network is composed of junctions and connection roads. A junction may have between two to four connections. In other words, cul-de-sacs are not allowed. A road that connects two junctions is divided into lanes. There are different kinds of obstacles, including traffic lights, roadblocks and pedestrian crossing points.

It is assumed, that the cars within the grid have their aim and objectives already defined. In this context, the aim denotes the final destination point on the grid that the car should arrive to and the objectives are series of junctions on the path to the destination point. The focus of the optimisation is the movement of a car, i.e., the

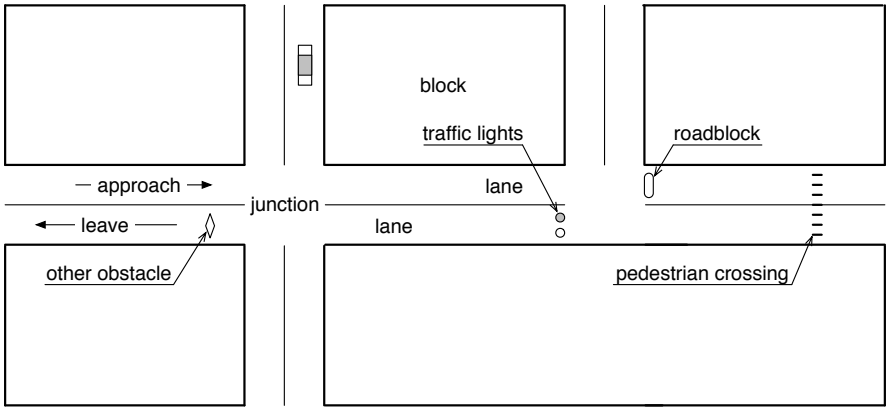


Figure 8.1: Elements of the traffic network

parameters that produce the best result according to a number of criteria, such as overall travel time, efficiency or comfort.

Unlike the traditional traffic optimisation problems, in the given scenario, the driver does not have direct control over his/her car. The mechanisms of the car control will be explained later in Section 8.2. Another characteristic feature is that there are no definite rules as to which car should cross the junction first if there are no traffic lights. The junction crossing priority is handled by the optimisation procedure.

8.2 Mapping and modelling

This section presents mapping between the case study and the framework along with parameter and implementation choices used for modelling. According to the proposed approach, the system is considered as a MAS, where each car is assigned the role of an agent.

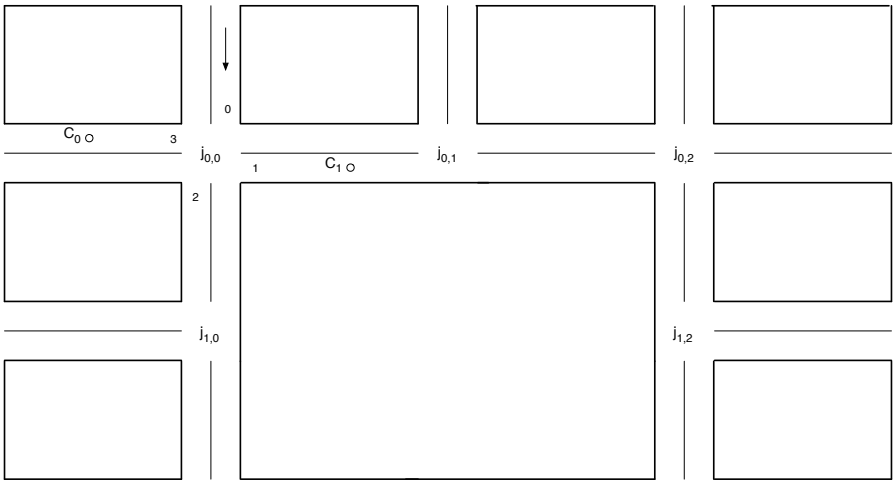


Figure 8.2: System element mapping

The environments consists of a 2-dimensional array of junctions $J = [j_{i,j}]_{i=1,\ldots,m;j=1,\ldots,n}$.

$$J = \begin{bmatrix} j_{0,0} & \cdots & j_{0,n} \\ \ldots & \ldots & \ldots \\ j_{m,0} & \cdots & j_{m,n} \end{bmatrix}$$

The situation in Figure 8.2 corresponds to the following array J.

$$J = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 0 & 1 \end{bmatrix}$$

The directions are marked with positive integers $\{0, 1, 2, 3\}$, where 0 corresponds to the north and 1 to the east direction. The cars participating in each experiment constitute an array $C = [c_i]_k$.

The environmental parameters are summarised in Table 8.1.

Parameter	Description	Value
NL_v	Number of vertical (north-south) lanes	2
NL_h	Number of horizontal (east-west) lanes	2
L_v	Vertical lane length	281m
L_h	Horizontal lane length	61m
W_v	Vertical lane width	20m
W_h	Horizontal lane width	20m
t	System time	
Δt	Simulation time step	100ms

Table 8.1: Parameters of the environment

The parameters of the car are summarised in Table 8.2. Car trajectory T is defined as a vector $T = [t_i \in J']_{i=1,\ldots,len(T)}$, where $J' = [j'_{i,j}]_{i=0,\ldots,m+1;j=0,\ldots,n+1}$. In addition, each pair (t_i, t_{i+1}) forms a consecutive path in J. Trajectory index I_T is defined as the current position of a car on the trajectory, $T \in [1, len(T)]$. Two special values of I_T (1 and $len(T)$) correspond to the car entering and leaving the simulation. Car state S can take one of four the values $S = 0, 1, 2, 3$. State 0 is an *init* state; the car has not started its movement yet. State 1 is an *approach* state; during this state the car approaches the junction t_{I_T}. State 2 indicates that the car is passing through the junction t_{I_T}. Finally, the car that is leaving the simulation has a state 3. Next three parameters describe the physical movement of the car. It should be noted that the position s indicates the current relative position of the car. The relative position is computed depending on the current state. For state 1, the position is measured from the end of the previous junction in the trajectory. For state 2, the position is measured from the start of the junction. Finally, the direction takes values from a set $\{0, 1, 2, 3\}$, which are north, east, south and west directions, respectively.

Parameter	Description
id	Unique identifier of the car
t_s	Car start time
T	Car trajectory
I_T	Trajectory index
S	Car state
a	Acceleration
V	Speed
s	Position
d	Direction

Table 8.2: Car parameters

Parameter	Description	Value
P_m	Mutation probability	0.(3)
P_c	Crossover probability	0.9
N	Population size	100

Table 8.3: Evolutionary algorithm parameters

The parameters related to the evolutionary algorithm running inside a car are given in Table 8.3.

8.2.1 Car motion model

This section describes the rules and border conditions that are used to model the movement of a car.

Table 8.4 describes the constraints used to model the movement of a car. The car is considered to be a point model that is able to accelerate with the acceleration

Constraints	Description	Value
a_{max}	maximum acceleration	2.317 m/s^2
a_{bmax}	maximum braking acceleration	2.317 m/s^2
V_{max}	maximum allowed speed	16.667 m/s
V_{jmax}	maximum allowed speed on junction	8.333 m/s

Table 8.4: Car model constraints

less than or equal to a_{max} and decelerate with acceleration less than or equal to a_{bmax}. V_{max} defines the maximum allowed traffic speed. Junctions are passed at constant speed that is less than or equal to V_{jmax}. The car always moves either with constant speed or constant nonzero acceleration.

Figure 8.3 depicts three possible speed profiles for car movement between the junctions, assuming no obstacles are encountered. Each profile could be divided into three phases, namely, acceleration *1*, constant speed motion *2* and deceleration *3* phases. Profile 1 includes all three phases, while profile 2 does not have constant speed motion phase and profile 3 has only acceleration phase. The following gives time t, speed V and displacement s equations for a car following these profiles.

Equation (8.1) describes how the instantaneous speed on the first phase of the first profile V_{11} changes with time. Equation (8.2) gives displacement from the start of phase 1 of profile 1. The total time required to reach V_{max} from V_0 can be calculated according to equation (8.3).

$$V_{11} = V_0 + at \tag{8.1}$$

$$s_{11} = V_0 t + \frac{at^2}{2} \tag{8.2}$$

$$t_{11} = \frac{V_{max} - V_0}{a} \tag{8.3}$$

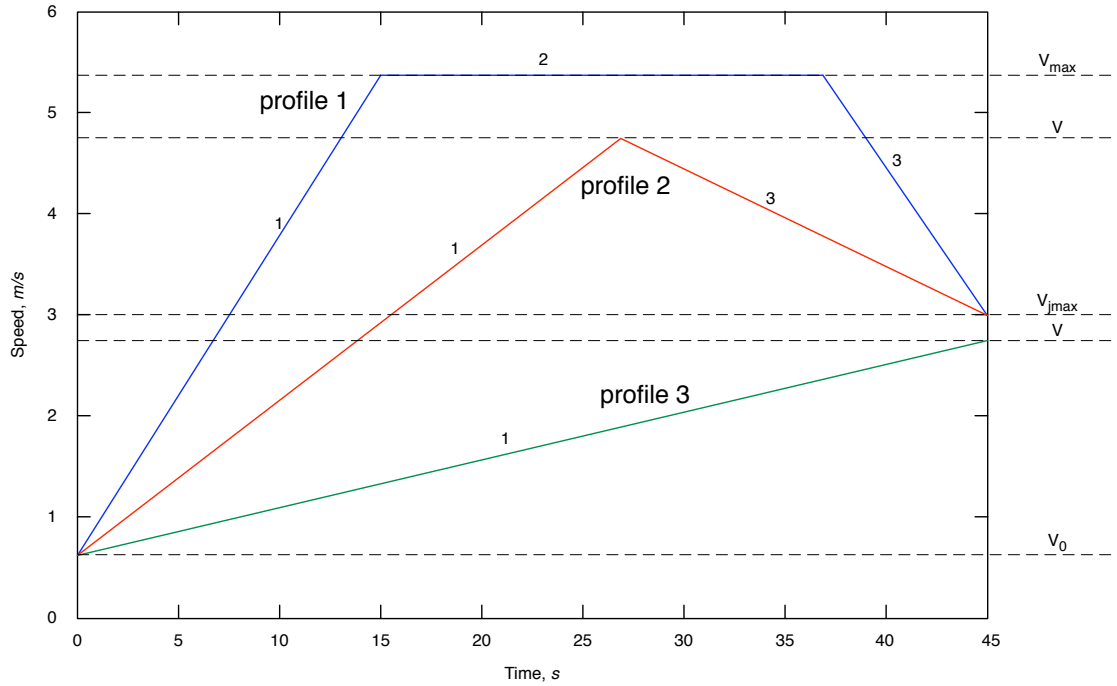


Figure 8.3: Different speed profiles for movement between junctions. 1) Acceleration to the maximum speed V_{max} followed by motion with constant speed followed by braking to the maximum allowed speed on junction V_{jmax}. 2) Acceleration to $V < V_{max}$ followed by braking to V_{jmax}. 3) Motion with constant acceleration to $V \leq V_{jmax}$.

The second phase of profile 1 corresponds to the motion with constant speed. The equations for speed (8.4) and displacement (8.5) are given below.

$$V_{12} = V_{max} \tag{8.4}$$

$$s_{12} = V_{max}t \tag{8.5}$$

Phase 3 of profile 1 corresponds to braking to the maximum junction speed V_{jmax}. The instantaneous speed and displacement are given by equations (8.6) and (8.7), respectively. The total time required for braking can be calculated using equation

(8.8).

$$V_{13} = V_{max} - a_{bmax}t \tag{8.6}$$

$$s_{13} = V_{max}t - \frac{a_{bmax}t^2}{2} \tag{8.7}$$

$$t_{13} = \frac{V_{max} - V_{jmax}}{a_{bmax}} \tag{8.8}$$

Profile 2 describes a scenario containing acceleration and deceleration phases following in succession without the intermediate constant speed motion phase. The time t_{23} required to brake from V to V_{jmax} can be calculated using equation (8.9)

$$t_{23} = \frac{-(\frac{V_{jmax}a_{bmax}}{a} + V_{jmax}) + \sqrt{(\frac{V_{jmax}a_{bmax}}{a} + V_{jmax})^2 - \frac{a_{bmax}(a_{bmax}+a)}{a} \cdot \frac{V_{jmax}^2 - 2as}{a}}}{\frac{2a_b(a_b+a)}{a}}, \tag{8.9}$$

where s is the length of the road between junctions. The corresponding displacement s_{23} and speed attained before deceleration are obtained using equations (8.10) and (8.11). Speed, time and displacement during phase 1 are calculated using the same method as for profile 1.

$$s_{23} = Vt_{23} - \frac{a_{bmax}t_{23}^2}{2} \tag{8.10}$$

$$V = V_{jmax} + a_{bmax}t_{23} \tag{8.11}$$

Profile 3 corresponds to a scenario where a car accelerates to $V < V_{jmax}$. Equations (8.12) and (8.13) allow to calculate travel time t_{31} and attained speed V_{31}.

$$t_{31} = \frac{-V_0 + \sqrt{V_0^2 + 2as}}{a}, \tag{8.12}$$

where s is the length of the road between junctions.

$$V = V_0 + at_{31} \tag{8.13}$$

8.2.2 Car state model

In order to facilitate processing of the environment, the car movement is described by states. Four states are defined, namely, *init*, *approach*, *passing* and *leaving*. The

init state corresponds to a car that has been initialised, but has not started its movement yet because the simulation time t is less than the start time t_s of the car. When the simulation time reaches the start time, the car enters approach state and begins its movement towards the first junction on the trajectory. Once the car enters the junction, the state changes to passing. After the junction is passed, the state either changes back to approach or is set to leaving (if the junction just passed is the last junction on the trajectory).

8.2.3 Chromosome representation

Choosing a suitable chromosome representation is a key factor in successful application of MOEA. It is especially important in on-line applications, as there are additional challenges, such as short response time, dynamics and noise. Suitable implementation of uncertainty vectors (introduced in Section 7.1.3) can help in alleviating these challenges. A strategy could be to divide the trajectory of a car into smaller segments and assign a decision variable to each of these segments. As the trajectory is naturally subdivided into sections between junctions, these sections are used to represent vectors $\vec{T}$ and $\vec{S}$.

Refer to Figure 8.4, which depicts chromosome shift technique for a trajectory of length m. The trajectory points t_1 and t_m indicate approach and leave directions and therefore are not included in the optimisation. The optimisation window is 3 junctions long, i.e., at any point of time the car is optimising its movement for the current junction and two junctions after that. The length of the prediction horizon is determined experimentally to provide good performance while not becoming too computationally expensive, as the additional decision variables tend to slow MOEA convergence. Decision variables are defined as the acceleration values of a car moving towards junctions in the optimisation window. For example, when the car is moving to the first point on the trajectory t_2, it is executing an evolutionary algorithm with

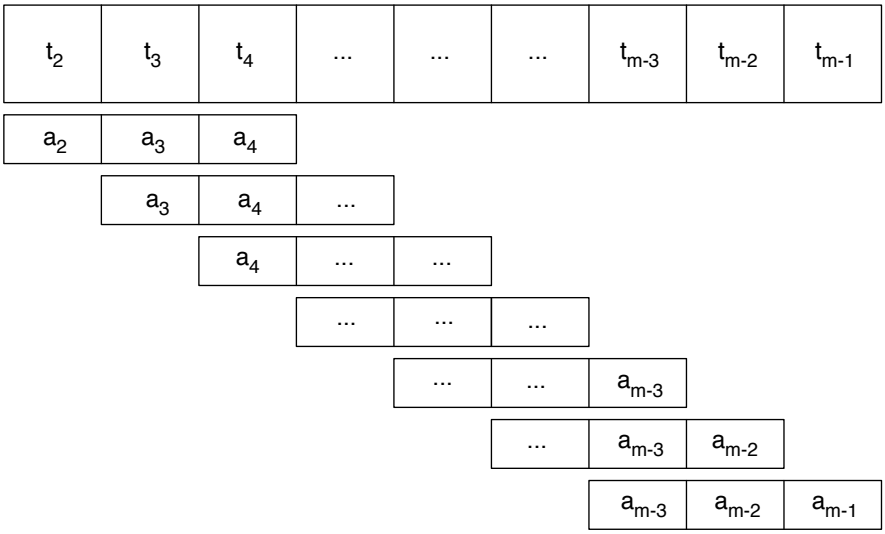


Figure 8.4: Dynamic optimisation window and chromosome shift using trajectory segmentation

a chromosome of length 3 with three genes a_2, a_3 and a_4, which correspond to the acceleration on trajectory segments before t_2, t_3 and t_4, respectively. The choice of acceleration as gene values is motivated by the fact that unlike speed, acceleration is a direct result of the immediate power output of the car's engine and brake force. Immediately after the car has reached t_2, the chromosomes of all population members are shifted to the left by one gene. The rightmost gene is initialised according to the current initialisation procedure, which is usually a random value in the feasibility range. By using the described procedure, a car maintains a set of partially optimised solutions before the start of the optimisation iteration, therefore reducing response time and yielding a higher quality result.

8.2.4 Objective functions

Objective functions provide a way to evaluate individuals in the population of an evolutionary algorithm. For this case study, two objective functions are defined.

- minimise travel time, and
- minimise speed changes.

The objective to minimise travel time is defined by equation (8.14)

$$f_1 = \sum_{i=1}^{min(m,n))} t_j(i), \tag{8.14}$$

where m is length of the chromosome, n is length of the remaining trajectory and $t_j(i)$ is time required to approach junction i.

The objective to minimise speed changes is defined by equation (8.15)

$$f_2 = \sum_{i=1}^{min(m,n))} t_b(i), \tag{8.15}$$

where m is length of the chromosome, n is length of the remaining trajectory and $t_b(i)$ is time required to brake using a_{bmax} before junction i.

The collision between vehicles is avoided by using two constraints, namely, junction occupancy constraint and minimum distance constraint. The junction occupancy constraint prevents a car from entering a junction already occupied by other vehicles. The minimum distance constraint is used on the connection roads and applies to the cars travelling in the same direction. The minimum distance is dynamically calculated to allow for emergency braking with acceleration a_{bmax} from the current speed.

8.2.5 Environment - agent relationship

An important issue in modelling the case study is parallel execution of the environment, default heuristic of the cars and evolutionary algorithms. Figure 8.5 shows the execution model of the case study. A simulation always has one environment block and one or more car blocks. All cars are initialised during the initialisation of the environment and immediately put into init state. A car starts its movement when the simulation time reaches the car start time.

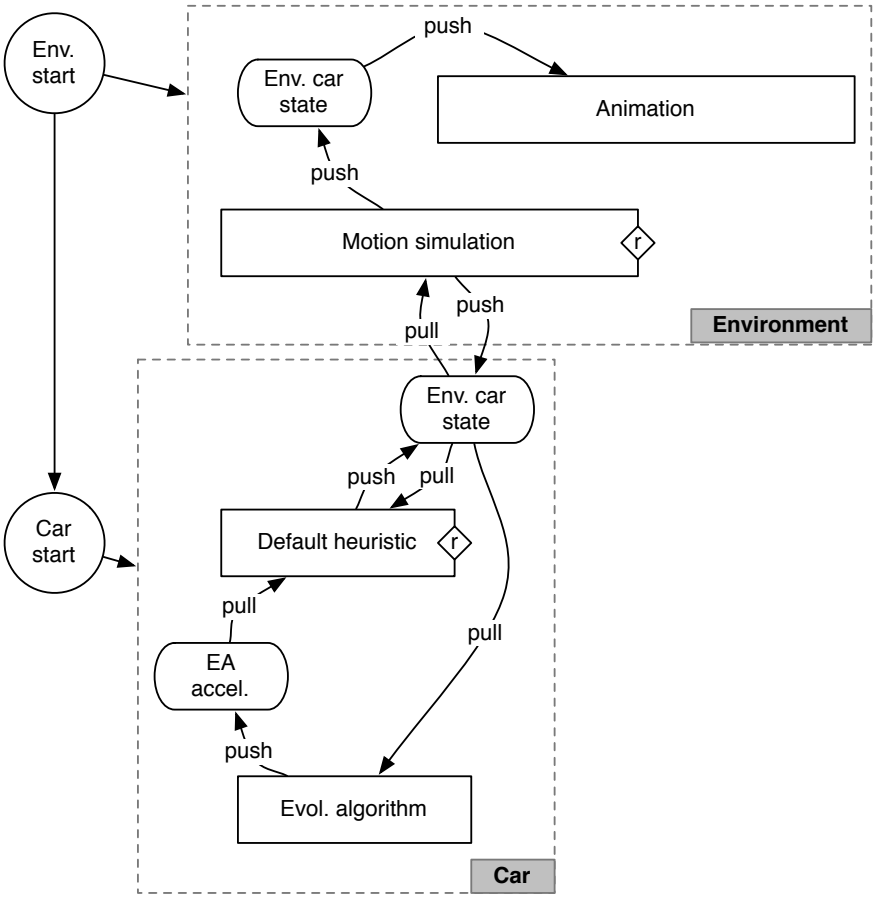


Figure 8.5: Execution model of the transportation grid case study. Note that *default heuristic* and *motion simulation* processes run in real time.

The environment block in Figure 8.5 consists of two processes, namely, the motion simulation and the animation. The motion simulation is the main process in the simulation; it synchronises movement of all cars in the simulation and advances the simulation time. In order to do so, the motion simulation process runs in discrete time, incrementing system time by a small step Δt and performing recalculation of speed and displacement for each car. Once the results are obtained, the environmental state of each car is passed to the animation process that runs with a lower priority to avoid slowing down the simulation. The animation process calculates

instantaneous absolute position of the car on the traffic grid and performs screen update.

The agent representing a car also consists of two processes. The default heuristic process runs in real time and operates in three major steps. Firstly, it generates border-case acceleration values for a car using a set of built-in rules and the world model. Then, it fetches the acceleration obtained by an evolutionary algorithm and compares it to the border-case acceleration values. Finally, the best acceleration value is selected as the current operational value.

An important topic is the synchronisation between different processes. The basic principle is that all processes run asynchronously. The synchronisation between the environment and the car blocks is performed used a shared synchronised data storage termed as *environmental car state*. The data storage can be simultaneously accessed by no more than a single process. In addition, the reads and writes to the data storage are atomic, i.e., can not be subdivided into more granular operations. It is also assumed that the reads and writes to the storage take negligibly small time. The information contained in the data storage is listed in Table 8.2.

The synchronisation within an agent follows the same algorithm as above, except that the synchronised storage *EA acceleration* contains only one field, namely, the acceleration. The evolutionary algorithm pulls the current environmental car state from the environmental car state data storage before the start of each generation, executes the generation and chooses the best solution from non-dominated set according to the preference. Then, the solution is put into the EA acceleration store.

The environment follows a different pattern, because the synchronised data store in the environment block is "active", i.e., it is able to influence the connected processes. It has the same fields as the main environmental car state data store and is used to update the animation. Once data is pushed into the store, the store pushes

it forward to the animation process and notifies it that a change in the system has occurred. The animation process uses the car environmental state data to calculate the position of the car on the grid and then redraw the car.

8.3 Empirical results

8.3.1 Experiment 1

The objective of the experiment is to verify if the evolutionary algorithm is able to match default heuristic in border conditions. In particular, it is assumed that there are no obstacles on the trajectory and $L_h = L_v = 281m$. All other parameters have default values. The preference vector is set to $[0.95, 0.05]$, therefore a car chooses the acceleration values that would guarantee the shortest travel time. Figure 8.6 shows the results of a typical optimisation run. The values obtained using default heuristic are plotted using thick lines and the values obtained using the MOEA are plotted using thin lines. The result is shown for two consecutive approach roads. It can be seen that the evolutionary algorithm is able to adjust fast to the changing environmental conditions and produces results that are very close to those obtained using default heuristic.

8.3.2 Experiment 2

The objective of the experiment is to verify that the cars are able to avoid collisions. It is assumed that there are no obstacles on the trajectory and $L_h = L_v = 281m$. Two cars are defined, car 1 has a trajectory [[-1, 0], [0, 0], [1, 0], [1, 1], [1, 2], [2, 2], [3, 2], [3, 3], [3, 4], [4, 4], [5, 4]]. Car 2 has a trajectory [[0, -1], [0, 0], [0, 1], [1, 1], [2, 1], [2, 2], [2, 3], [3, 3], [4, 3], [4, 4], [4, 5]]. Both cars start at simulation time 0. Constraints are defined in such a way that no two cars occupy the same junction

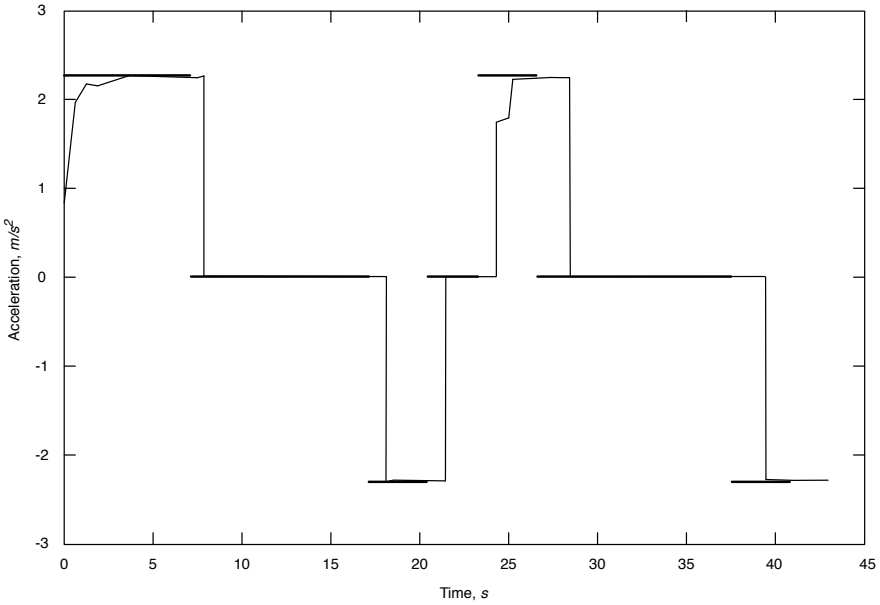


Figure 8.6: Acceleration values computed by default heuristic (thick line) and MOEA (thin line).

simultaneously.

Figure 8.7 shows the positions of the cars at $t = 26100ms$. It can be seen that car 2 reached the junction at the same moment when car 1 left it. This happened because car 2 had to accelerate at a slower rate compared to car 1 to avoid 'collision'. Figure 8.8 displays a moment where car 2 arrived to the second conflicting junction. Because of the longer path it had to take and slowdown before junction [0, 1] it did not have to reduce its rate of acceleration. Finally, Figure 8.9 shows the positions of the cars at $t = 100000ms$, when the simulation ends. Note that because car 1 had to make a right (long) turn, car 2 has decreased its lag with the car 1.

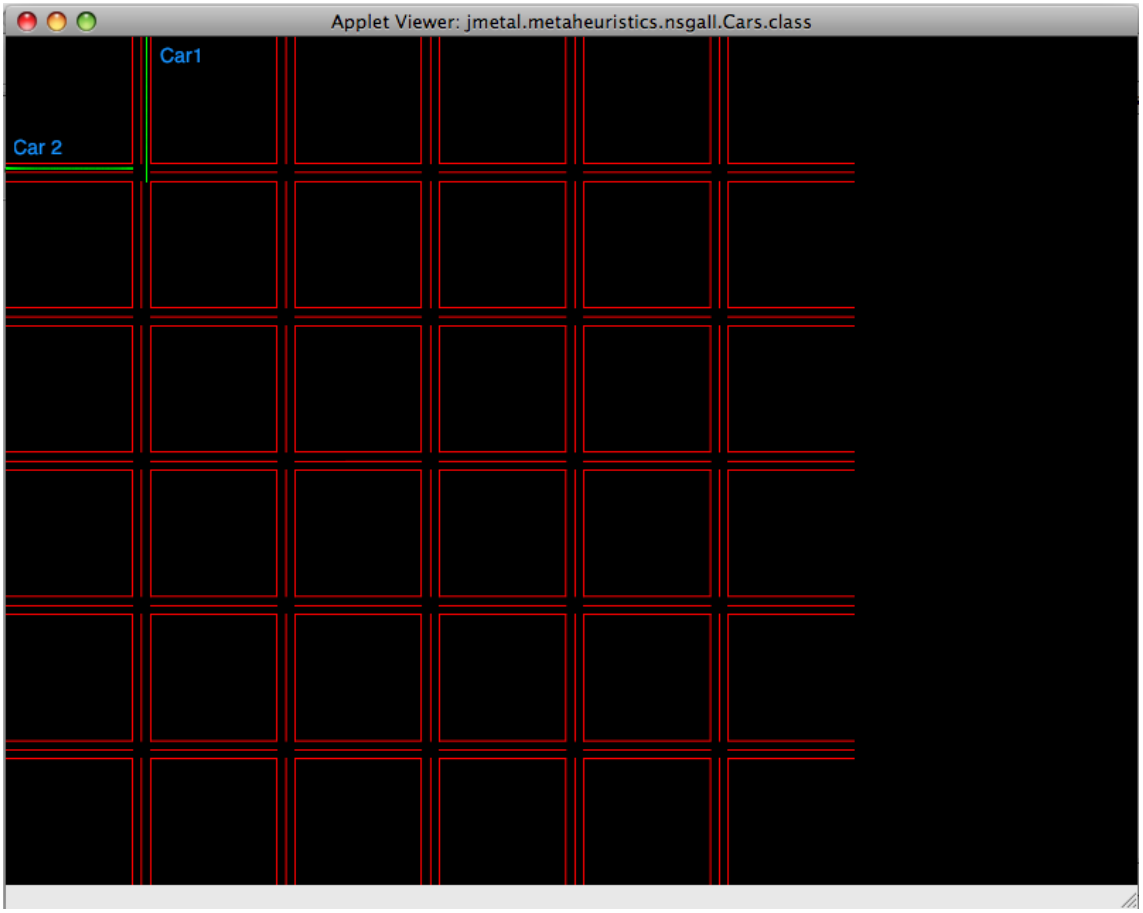


Figure 8.7: Positions of the cars at $t = 26100ms$.

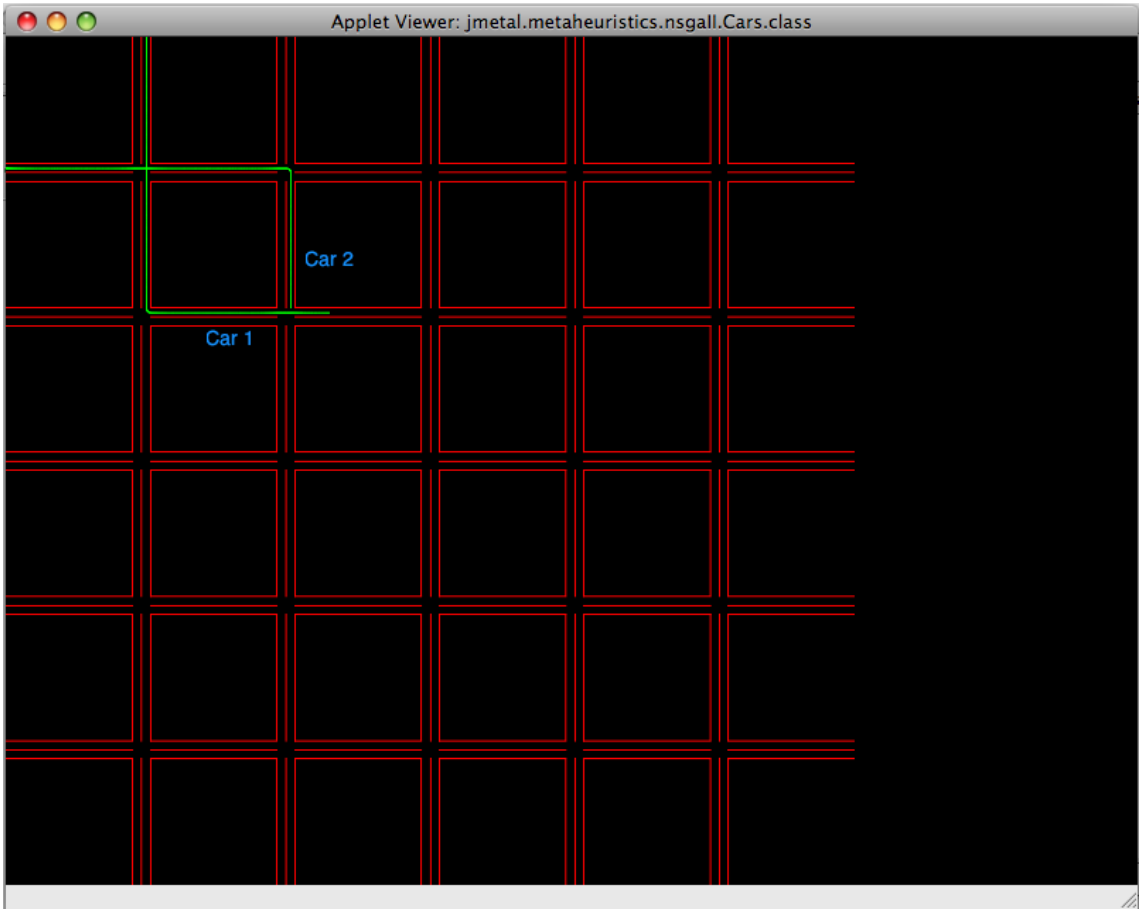


Figure 8.8: Positions of the cars at $t = 69700ms$.

Figure 8.9: Positions of the cars at $t = 100000ms$.

8.4 Summary

This chapter details a traffic scheduling case study that focuses on a close-to-life implementation of a stochastic transportation planning system with intelligent agents representing individual vehicles. By using an asynchronous modelling scheme, the case study emphasises the real-time behaviour of the system and verifies the ability of the proposed approach to control multiple distributed intelligent agents on-line. The experiments confirmed that an agent's behaviour in a complex MAS can be successfully guided by an EMO MPC controller.

Chapter 9

Discussion and conclusions

9.1 Key findings

9.1.1 Literature review

The literature review focused on the characteristics of the uncertainties impacting on the dynamic multi-objective optimisation process and the way they are being addressed by the present-day research. A typical dynamic optimisation problem contains a number of uncertainties, most notably modelling uncertainty caused by the use of streamlined process models, uncertainty induced by the real time constraint and measurement and process noise uncertainties. Papers dealing with uncertainty in multi-objective optimisation typically address a single uncertainty type.

The existing research in dynamic multi-objective evolutionary optimisation is very limited, despite single-objective dynamic optimisation being a well-established area. This is a stark contrast to the field of static optimisation, where there is no such gap between single-objective and multi-objective cases. A large proportion of papers describing themselves as simultaneously dealing with dynamic and multi-objective problems, opt to either model the problem as single-objective by using an *a priori*

decision maker or consider the problem as pseudo-static in the course of optimisation. Most of the research, which is multi-objective and dynamic, comes from the defence sector, reflecting the more uncertain and complex nature of problems faced by the military.

A crucial part of on-line multi-objective optimisation is the ability to repeatedly make automated decisions in order to select a single control action from an array of Pareto-optimal alternatives. Existing methods in MCDM are mainly targeted at off-line situations, where a number of expert opinions are available. Consequently, current dynamic multi-objective applications mainly use simple aggregation-based approaches such as the WSM.

9.1.2 Dynamic control problems

The author Looked for a source of established dynamic optimisation problems, and identified the problems typically considered by control theory as dynamic and multi-objective. Some methods from control theory, such as MPC, are formulated as repeating optimisation problems, albeit single-objective. The focus on single-objective optimisation may be explained by the requirement of a single control action at each time instant.

It was found that some dynamic control problems require very fast response times from the controller in the order of several milliseconds. Such a fast response is not commonly associated with MOEA, which require a large population to be evaluated for a number of generations to produce results. Several methods were found that address the discrepancy between the expected controller response times and the DMOEA performance. Firstly, splitting the controller into an EMO MPC part and a control conditioner allows the EA to evolve a control strategy instead of a momentary control effort value. The control strategy is used for fast and frequent calculations of the control effort until a new strategy could be supplied by

the DMOEA. Secondly, by partitioning the control strategy within an algorithm's chromosome, it is possible to address horizon issues that are characteristic of a classic MPC. Moreover, by introducing a timed shift in the chromosome, it is possible to supply the DMOEA with partially pre-converged solutions during the course of optimisation.

9.1.3 Multi-objective evolutionary algorithms

The performance evaluation of the DMOEA on a benchmark problem was designed to answer two important questions, namely, what are the algorithms' limits in tracking the moving optima and how reliable are the results obtained by a stochastic algorithm. The latter question is important because static EAs have virtually unlimited time to converge and the results are examined by a human with the outliers rejected; this is not possible in the context of on-line applications.

The process of DMOEA convergence can be divided into two different stages, i.e., initial convergence and dynamic tracking. These stages are characterised by the length of the confidence interval within which the mean of the Pareto front metric lies. The initial convergence stage is characterised by the large variability between multiple DMOEA executions, particularly in response to the changes in the problem. After the population has converged, subsequent changes in the problem cause markedly similar EA response between multiple tries. The speed of the initial convergence is greatly affected by the problem state, while the dependency of dynamic tracking on the problem state is less pronounced.

The experiments confirmed that, for the particular benchmark problem, dynamic tracking responds more efficiently to the changes than restarting an algorithm from scratch. When dealing with problem changes, increase in static mutation rates reduces the performance drop relative to the current mean value. In absolute terms, however, increasing static mutation rate does not give an appreciable benefit, as the

algorithm convergence after performance drop is slower. The variability between experiments reducing after the period of initial convergence holds true irrespective of the static mutation rate.

Comparing the different NSGAII variants, the greatest strength of dNSGAII-B versus dNSGAII-A lies in its ability to improve the rate of initial convergence. The additional performance gain in dynamic tracking is insignificant, bearing in mind the increased number of fitness evaluations.

Overall, the performance of DMOEA is much less affected by the frequency of change than can be predicted by observing their static counterparts employed in a restart from scratch strategy. An important role in boosting the DMOEA performance is attributed to elitism. Algorithms with dynamic elitism are up to twice as efficient as those without it in terms of the hypervolume metric. A technique that restricts the degree of elitism was tested in the form of dNSGAII-C, which was found to compete with and sometimes outperform the baseline dNSGAII-A, despite the reduced number of fitness evaluations.

9.1.4 Decision making in dynamic environments

By analysing the aggregation approaches (particularly the WSM), which are the most popular decision making methods in dynamic optimisation, it was found that the results from them depend on the shape of the Pareto front. Because the decision has to be made from a set of Pareto-optimal alternatives and the multi-objective search efficiency is not affected by the decisions of a Decision Maker (DM), a random decision maker was selected as a baseline dynamic DM.

In an attempt to make a better dynamic DM, it was found by the author that the methodology of repeatedly making on-line automated decisions has been explored within the game theory. An algorithm for decision making in integer zero-sum games (known as MINIMAX) is able to make automated decisions without specifying user

preference. The algorithm uses game trees with the nodes consisting of game states to make decisions. However, how to make game trees for real-valued control problems with potentially unlimited number of states remained unclear.

A Pareto front provides a sample of the tradeoff surface of a dynamic system at a given point in time. Consequently, a Pareto front metric can be used to evaluate the state without explicitly ordering or weighting the different performance criteria. A state tree based on the Pareto front was found to allow the application of decision making algorithms from game theory to real-valued control problems.

9.1.5 Evaluation of single entity control

The first important finding during the evaluation was that real control problems are often less difficult than the benchmark dynamic multi-objective optimisation problems for the evolutionary algorithm to track. In particular, this manifests in higher frequencies of change still allowing the DMOEA to maintain a good convergence. It was found that the EMO MPC controller is able to stabilise the inverted pendulum in real-time despite the control response interval of 5ms. In addition, an EMO MPC controller provides better performance than the PID controller used as a baseline. The controllers with long control intervals are able to compensate the seldom changes in behaviour with more aggressive control strategies.

There is a nonlinear relation between the length of the prediction horizon and the performance of an EMO MPC controller. If the prediction horizon is too long and the DMOEA is operating with global fitness, the overall performance of the controller decreases, as only a restricted set of globally stable control strategies is evaluated. Shorter prediction horizons allow the controller to exploit globally unstable control strategies and provide better performance as result. However, very short prediction horizon causes the controller to overshoot (or supply a very large control effort if the control strategy is not limited by constraints).

Regarding the stability of the obtained results, it was observed that the controllers that operate with global fitness functions are stable regardless of the preference vector of the aggregating decision maker. Controllers that evaluate the local fitness require dedicated stability constraints in the DMOEA and the DM in order to be stable in all cases. Without stability constraints, they may become unstable with some preference vectors.

Interesting results were demonstrated by the random decision maker. In contrast with static decision making, in which making a random decision is not generally a good idea even if one has to select from a set of Pareto-optimal alternatives, the random decision maker turned out to be a tough competitor to the WSM. EMO MPC controllers employing random DM were stable with both local and global fitnesses. However, the performance of a controller evaluating local fitness with a random decision maker suffers a lot of degradation as the length of the prediction horizon decreases.

A controller using PDT DM outperforms the random decision maker for all tested lengths of the prediction horizon. In general, the PDT controller is more tolerant to the changes in prediction horizon length than both WSM and random DMs. The effect of PDT can be seen as a stability-enforcing measure, as current decisions causing instability lead to deterioration of future Pareto fronts, and therefore would not be preferred by PDT.

9.1.6 Multi-agent systems

While studying a MAS system that uses EMO MPC controllers to guide agents' actions, two issues were found to have the most impact on the system behaviour. The first one was the need for guaranteed real time response that arises in the interaction between multiple agents. The real time constraint was found to be enforceable through a problem-specific basic default heuristic running alongside the EMO MPC

controller, providing a fallback solution in cases where the evolved solution is poor. In the later stages of the development, is was observed that the EMO MPC controller was always able to supply a satisfactory solution in a timely manner, so that the default heuristic was not needed. However, the presence of the default heuristic was important during the early stages, where it accounted for nearly one third of decisions. The default heuristic is also very important for the problems where the time required to calculate the objective vector can not be reliably estimated, such as when the objectives are calculated using a Discrete Event Simulation (DES) model. The second issue concerns how an agent could model the actions of other agents, so as to adapt to changes in their actions. Seemingly, the computational demands of the entire system rise exponentially with the number of agents. It was found that the actions of an agent can be communicated to other agents in terms of shared resource usage. Therefore, an agent does not need to model actions of other agents, but can directly use the resource availability information as constraints to the DMOEA and within any spatial global objective functions.

It was also discovered that the uncertainty arising from the interaction of multiple agents can be arranged into uncertainty vectors. The uncertainty vectors are partitioned according to the systems states on which they are based, e.g., on the resource usage. The partitioning of the uncertainty vectors provides a natural mapping to the variable prediction horizon scheme of EMO MPC and allows for chromosome shifts that improve initial convergence.

9.1.7 Case study: traffic scheduling

The traffic scheduling case study focused on a close-to-life implementation of a stochastic transportation planning system with intelligent agents representing individual vehicles. An asynchronous modelling scheme places emphasis on the system's real-time behaviour and ensures that the real-time constraints are always satisfied.

The experiments confirmed that an agent's behaviour in a complex MAS can be successfully guided by an EMO MPC controller.

9.2 Contributions

The aim of this research is to develop a framework for on-line optimisation of dynamic problems that is capable of a) representing problems in a quantitative way, b) identifying optimal solutions using multi-objective evolutionary algorithms, and c) automatically selecting an optimal solution among alternatives.

The overall contribution of this research is a framework for multi-objective optimisation of dynamic problems. The framework comprises problem specification, dynamic multi-objective search and the *a posteriori* dynamic decision maker. The framework is designed to be applied to tasks that require a coordinated control of multiple entities. This section describes the contributions of this research in detail.

A crucial part of this research is the systematic approach to dynamic MOO and recognition of the role that decision making plays in dynamic multi-objective optimisation. In Chapter 3, dynamic evolutionary multi-objective search and dynamic *a posteriori* decision making are considered in the MPC context, which is a well-established method to control complex processes based on iterative process model optimisation, albeit single-objective. Particular attention is devoted to performance improvement techniques that resulted in the concept of a control conditioner and the technique of control strategy evolution. To counter the effects of uncertainty and to improve initial convergence, this research proposes to split the prediction horizon into smaller sections with the corresponding parts of the control strategy sequentially encoded in the chromosome.

Chapter 4 provides a study on how the different parameters of the EMO MPC affect the evolutionary algorithm's ability to track the moving optima. The following

parameters were considered: length of pre-execution, frequency of change, length of prediction interval and static mutation rate. In addition, a DMOEA with restricted elitism is suggested for noisy environments.

To address the decision making aspect of the problem, a novel method for constructing game trees for real-valued multi-objective problems is detailed in Chapter 5. Its use allows us to apply the decision making techniques developed in the area of control theory to control problems. By leveraging the game tree approach, an algorithm for preference-less on-line decision making is proposed.

Chapter 6 contains an empirical evaluation of EMO MPC parameters and comparison of PID and B-spline control conditioners. In addition, the proposed EMO MPC approach is compared with a classic controller on a well-known benchmark problem of balancing an inverted pendulum. The different decision making techniques are examined, and a baseline random decision maker is suggested and tested.

Finally, the EMO MPC approach is integrated into a MAS framework for coordinated control of multiple entities in Chapter 7. The framework is targeted on the applications in shop floor scheduling and traffic planning. A traffic scheduling problem in a rectangular grid environment is detailed and used to validate the framework in Chapter 8.

9.3 Limitations

The limitations of this research are categorised into three groups, namely, decision making, verification and MAS. A major limitation that belongs to the decision making category is that there are no problem-independent metrics to compare dynamic DMs. However, this issue is not specific to the dynamic case, as static decision makers face the same difficulties in comparison [28]. Another limitation within the decision making category is that there is only a basic analysis and motivation for

the selection of a particular control conditioner.

The verification uses a problem with modelling uncertainty, but the incorporation of measurement noise is not tested thoroughly. Consequently, only Pareto ranking is used to compare the alternatives by the DMOEA, which might not be the optimal way in the presence of noise. The evaluation of single entity control is limited to two objectives, although there are no obvious obstacles for the approach to work with three or more objectives. The DMOEA tracking ability is examined using a single benchmark problem, which is a Type II problem in which both the decision variables and the objective front change over time. The EMO MPC controller is compared with the PID and nonlinear MPC controllers, but comparison with other controller types would also be beneficial.

The MAS framework does not detail the handling of intermittent communications between agents, which might be detrimental to the DMOEA performance, as the constraints would alternate rapidly. Also, the issue of handling the resource access priority for agents, especially ones with preference-less DMs, is not considered.

9.4 Future research

Evaluation of the EMO MPC controller demonstrated dynamic multi-objective optimisation as a powerful and versatile tool for solving real-life control problems. At the same time, owing to the relative lack of attention paid to dynamic EMOO, in research literature there are numerous research gaps that cannot be addressed by a single study. These unaddressed research gaps are suggested as possible future research directions below.

There are two principal research directions in the theory surrounding the EMO MPC approach. Firstly, a study into new control conditioners, a methodology for choosing them and a comparison of more control conditioners is required in the

context of EMO MPC. The benefits provided by open and closed loop control conditioners are open to a systematical evaluation. An interesting research question is the complexity split between the dynamic MOEA / DM pair and the control conditioner, i.e., how complex should be the control conditioner in order to maximise it's performance and facilitate the search of the control strategy. Secondly, the use of other non-Pareto ranking techniques for ordering of solutions should be studied for DMOEA, as Pareto ranking is susceptible to noise commonly found in dynamic applications.

A number of future research directions is associated with the dynamic decision making aspect of the multi-objective control. It is important to know what are the optimum depth of the PDT tree, its width at a branching point and the Pareto front metrics that are used to evaluate system states. On the other hand, more research is required in alternative on-line decision makers.

In addition, there are numerous empirical research directions. The EMO MPC controller could be compared against other controller types, such as H_∞ and fuzzy controllers. Another topic of interest is how PDT behaves in higher-objective scenarios. From the performance point of view, the MAS framework should be benchmarked with different communications patterns. The effects of measurement and process noise should be evaluated together with the modelling uncertainty for DMOEA. Additional tests with different control problems are required as well. Finally, better metrics are needed to allow EMO MPC methods to be compared.

9.5 Conclusions

This research demonstrates that the EMO approach can be used to address the complexity found in real-time control problems. The proposed EMO MPC controller is able to provide strong competition to classic controllers in simple cases and

also extend the applications to highly multi-objective, nonlinear and interdependent problems. The proposed decision making approach establishes a link between game theory and decision making in DMOEA, suggesting future exchange of ideas and methods between these areas. Having a Pareto surface available at any point provides a tool for analysis of classic controllers and valuable insight into the characteristics of the problem. Finally, the EMO MPC approach using modern off-the-shelf hardware is sufficiently responsive even for control problems with fast paced dynamics. To concluding this research, the achievements are analysed against the research objectives defined earlier.

1. *Develop dynamic problem specification and provide quantitative representation suitable for optimisation using evolutionary techniques.* Chapter 3 established a link between dynamic control problems and dynamic MOO and provided quantitative problem representation in a form suitable for DMOEA applications. The representation contains a number of features designed to improve the real-time performance of the algorithm in the presence of noise. Chapter 7 describes a MAS that is designed to be integrated with EMO MPC.

2. *Propose solutions to deal with problem dynamics and create a framework for multi-objective optimisation of dynamic problems.* Problem dynamics is addressed by modifications of the state-of-the-art MOEA NSGA-II presented in Chapter 4. An EMO MPC optimisation framework was proposed for dynamic problems in Chapter 3.

3. *Identify the criteria for decision making and suggest a strategy for making decisions from a set of Pareto-optimal solutions on-line.* Chapter 5 outlines a baseline random decision making procedure and a novel game-tree based on-line decision method.

4. *Identify and select performance metrics for the dynamic optimisation case, and systematically evaluate the performance of the dynamic multi-objective optimisation framework.* Performance of the dynamic optimiser is evaluated in Chapter 4. The hypervolume and relative variance metrics are used for the evaluation. The EMO MPC framework is evaluated in Chapter 6.

5. *Validate the proposed framework using a case study* The EMO MPC optimisation framework is evaluated using an inverted pendulum problem in Chapter 6 including comparison to PID and nonlinear MPC methods. MAS for coordinated control of multiple entities was evaluated in Chapter 8 using a traffic scheduling case study.

The aim of this research is to develop a framework for on-line optimisation of dynamic problems that is capable of a) representing problems in a quantitative way, b) identifying optimal solutions using multi-objective evolutionary algorithms, and c) automatically selecting an optimal solution among alternatives. The proposed framework was validated using an inverted pendulum problem and traffic grid scheduling problem. In addition, the multi-objective evolutionary search was verified using a benchmark problem. The results show the multi-objective control approach as a viable alternative to traditional single-objective control methods.

References

[1] S. J. Russell, P. Norvig, Artificial Intelligence: A Modern Approach (Second Edition), Prentice Hall, 2003.

[2] E. J. Hughes, Evolutionary guidance for multiple missiles, in: IFAC World Congress Conference, No. Paper 1801, 2002, pp. 21–26.

[3] F. L. Pereira, J. B. D. Sousa, Coordinated control of networked vehicles: An autonomous underwater system, Autom. and Remote Control 65 (7) (2004) 1037–1045.

[4] R. Veryard, Implementing a methodology, Information and Software Technology 29 (9) (1987) 469–474.

[5] The definition of information systems (July 1999), http://www.turningcourse.com/ukais/isdefn.pdf, (accessed: 21.09.2010).

[6] A. I. Wasserman, Information system design methodology, Journal of the American Society for Information Science 31 (1) (1980) 5–24.

[7] Jmetal - neo, http://mallba10.lcc.uma.es/wiki/index.php/jmetal, (accessed: 27.02.2007).

[8] K. Auer, R. Miller, Extreme programming applied playing to win, Addison-Wesley, Boston, 2002.

[9] K. Beck, Extreme programming eXplained embrace change, Addison-Wesley, Reading, MA, 2000.

[10] K. Beck, M. Fowler, Planning extreme programming, Addison-Wesley, Boston, 2001.

[11] R. Findeisen, F. Allgöwer, An introduction to nonlinear model predictive control, in: Control, 21st Benelux Meeting on Systems and Control, Veidhoven, 2002, pp. 1–23.

[12] R. Subbu, P. Bonissone, N. Eklund, W. Yan, N. Iyer, F. Xue, R. Shah, Management of complex dynamic systems based on model-predictive multi-objective optimization, General Electric Global Research, One Research Circle, Niskayuna, NY 12309, United States, 2006, pp. 64–69.

[13] A. Zhou, B. Y. Qu, H. Li, S. Z. Zhao, P. N. Suganthan, Q. Zhangd, Multiobjective evolutionary algorithms: A survey of the state of the art, Swarm and Evolutionary Computation 1 (1) (2011) 32–49.

[14] C. A. C. Coello, G. B. Lamont, D. A. V. Veldhuizen, Evolutionary algorithms for solving multi-objective problems, Springer, 2007.

[15] L. Huang, I. Suh, A. Abraham, L. Li, Dynamic multi-objective optimization based on membrane computing for control of time-varying unstable plants, Information Sciences 181 (11) (2011) 2370–2391.

[16] D. A. V. Veldhuizen, G. B. Lamont, Multiobjective evolutionary algorithm research: A history and analysis, Tech. Rep. TR-98-03, Dept. Elec. Comput. Eng., Graduate School of Eng., Air Force Inst. Technol. (1998).

[17] C. Darwin, The Origin of Species by Means of Natural Selection, or the Preservation of Favoured Races in the Struggle for Life, 6th Edition, John Murray, London, Retrieved on 22.01.2008.

[18] M. Camara, J. Ortega, F. J. Toro, Parallel processing for multi-objective optimization in dynamic environments, IEEE International Parallel and Distributed Processing Symposium (2007) 243.

[19] A. Konak, D. W. Coit, A. E. Smith, Multi-objective optimization using genetic algorithms: A tutorial, Reliability Engineering & System Safety 91 (9) (2006) 992–1007.

[20] K. Deb, N. Udaya Bhaskara Rao, S. Karthik, Dynamic multi-objective optimization and decision-making using modified NSGA-II: A case study on hydro-thermal power scheduling, in: EMO, 2006, pp. 803–817.

[21] Y. Jin, J. Branke, Evolutionary optimization in uncertain environments - a survey, Evolutionary Computation, IEEE Transactions on 9 (3) (2005) 303–317.

[22] EMOO repository (2011).

[23] C.-L. Hwang, A. S. M. Masud, Multiple objective decision making, methods and applications, Springer-Verlag, 1979.

[24] L. M. Boychuk, V. O. Ovchinnikov, Principal methods for solution of multicriterial optimization problems (survey), in: Soviet Automatic Control, Vol. 6, 1973.

[25] L. S. d. Oliveira, S. F. P. Saramago, Multiobjective optimization techniques applied to engineering problems, Journal of the Brazilian Society of Mechanical Sciences and Engineering 32 (2010) 94–105.

[26] C. L. Hwang, S. R. Paidy, K. Yoon, A. S. M. Masud, Mathematical programming with multiple objectives: A tutorial, Computers and Operations Research 7 (1-2) (1980) 5–31.

[27] C. Bana e Costa, Readings in MCDA, Springer Verlag, Berlin, 1990.

[28] E. Triantaphyllou, S. H. Mann, An examination of the effectiveness of multi-dimensional decision-making methods: a decision-making paradox, Decis. Support Syst. 5 (1989) 303–312.

[29] C. Carlsson, R. Fullér, Fuzzy multiple criteria decision making: Recent developments, Fuzzy Sets and Systems 78 (2) (1996) 139–153.

[30] P. C. Fishburn, Letter to the editor–additive utilities with incomplete product sets: Application to priorities and assignments, Operations Research 15 (3) (1967) 537–542.

[31] V. Allis, Searching for solutions in games and artificial intelligence, Ph.D. thesis, University of Limburg (1994).

[32] S. Hart, R. Aumann, S. Hart, Chapter 2 Games in extensive and strategic forms, Vol. Volume 1, Elsevier, 1992, pp. 19–40.

[33] D. Fudenberg, J. Tirole, Game theory, MIT Press, 1993.

[34] D. F. Jones, S. K. Mirrazavi, M. Tamiz, Multi-objective meta-heuristics: An overview of the current state-of-the-art, European Journal of Operational Research 137 (1) (2002) 1–9.

[35] J. D. Schaffer, Multiple objective optimization with vector evaluated genetic algorithms, in: Proceedings of the 1st International Conference on Genetic Algorithms, Lawrence Erlbaum Associates, Inc., Mahwah, NJ, USA, 1985, pp. 93–100.

[36] C. M. Fonseca, P. J. Fleming, Genetic algorithms for multiobjective optimization: Formulation, discussion and generalization, in: Genetic Algorithms:

Proceedings of the Fifth International Conference, Morgan Kaufmann, 1993, pp. 416–423.

[37] N. Srinivas, K. Deb, Multiobjective Optimization Using Nondominated Sorting in Genetic Algorithms, Evolutionary Computation 2 (3) (1994) 221–248.

[38] E. Zitzler, K. Deb, L. Thiele, Comparison of multiobjective evolutionary algorithms: Empirical results, Evol. Comput. 8 (2) (2000) 173–195.

[39] D. Büche, P. Stoll, R. Dornberger, P. Koumoutsakos, Multiobjective evolutionary algorithm for the optimization of noisy combustion processes, in: IEEE Transactions on Systems, Man, and Cybernetics Part C—Applications and Reviews, Vol. 32, 2002, pp. 460–473.

[40] J. Mehnen, R. Roy, Multi-objective evolutionary algorithms applied to machining of materials with dynamically changing properties, in: A. De Silva, D. K. Harrison (Eds.), 20th International Conference on Computer-Aided, Production Engineering, Glasgow, UK, 2007.

[41] R. Roy, J. Mehnen, Dynamic multi-objective optimisation for machining gradient materials, in: CIRP Annals-Manufacturing Technology, Vol. 57, 2008, pp. 429–432.

[42] J. J. Grefenstette, Genetic algorithms for changing environments, in: R. Maenner, B. Manderick (Eds.), Parallel Problem Solving from Nature 2, North Holland, 1992, pp. 137–144.

[43] H. G. Cobb, J. J. Grefenstette, Genetic algorithms for tracking changing environments, in: 5th international Conference on Genetic Algorithms S. Forrest, Ed. Morgan Kaufmann Publishers, San Francisco, CA, 1993, pp. 523–530.

[44] Y. Monnier, J.-P. Beauvais, A.-M. Déplanche, A genetic algorithm for scheduling tasks in a real-time distributed system, in: EUROMICRO '98: Proceedings

of the 24th Conference on EUROMICRO, IEEE Computer Society, Washington, DC, USA, 1998, p. 20708.

[45] E. J. Hughes, Multi-objective, probabalistic selection evolutionary algorithms (MOPSEA), Tech. Rep. DAPS/EJH/56/2000, Cranfield University (2000).

[46] E. Hughes, Evolutionary multi-objective ranking with uncertainty and noise, in: EMO '01: Proceedings of the First International Conference on Evolutionary Multi-Criterion Optimization, Springer-Verlag, London, UK, 2001, pp. 329–343.

[47] E. J. Hughes, Multi-objective evolutionary guidance for swarms, in: CEC '02: Proceedings of the Evolutionary Computation on 2002. CEC '02. Proceedings of the 2002 Congress, IEEE Computer Society, Washington, DC, USA, 2002, pp. 1127–1132.

[48] M. Babbar, A. Lakshmikantha, D. E. Goldberg, A modified nsga-ii to solve noisy multiobjective problems, in: J. Foster (Ed.), Genetic and Evolutionary Computation Conference. Late-Breaking Papers, 2003, pp. 21–27.

[49] SCOPUS, http://www.scopus.com/, (accessed: 05.04.2008).

[50] B. L. Miller, Noise, sampling, and efficient genetic algorthms, Ph.D. thesis, University of Illinois, Champaign, IL, USA (1997).

[51] C. Goh, K. Tan, Noise handling in evolutionary multi-objective optimization, in: Evolutionary Computation, 2006. CEC 2006. IEEE Congress on, 2006, p. 1354.

[52] C. K. Goh, K. C. Tan, An investigation on noisy environments in evolutionary multiobjective optimization, IEEE Transactions on Evolutionary Computation 11 (3) (2007) 354–381.

[53] J. Branke, T. Kaußler, C. Schmidt, H. Schmeck, A multi-population approach to dynamic optimization problems, in: Adaptive Computing in Design and Manufacturing 2000, Springer, 2000.

[54] J. Branke, Evolutionary approaches to dynamic optimization problems – updated survey, in: GECCO Workshop on Evolutionary Algorithms for Dynamic Optimization Problems, 2001, pp. 27–30.

[55] J. Branke, Evolutionary Optimization in Dynamic Environments, Kluwer Academic Publishers, Boston, MA, 2002.

[56] L. Bui, J. Branke, H. Abbass, Multiobjective optimization for dynamic environments, Technical Report TR-ALAR-200504007, Artificial Life and Adaptive Robotics Laboratory, Canberra (2004).

[57] Engineering Village, http://www.engineeringvillage.org/, (accessed: 05.04.2008).

[58] N. Nariman-zadeh, A. Hajiloo, A. Jamali, A. Bagheri, Reliability-based pareto optimum design of robust compensators for a dynamic system with parametric uncertainty, in: 2006 European Simulation and Modelling Conference. Modelling and Simulation 2006, 23-25 Oct. 2006, EUROSIS-ETI, Dept. of Mech. Eng., Univ. of Guilan, Rasht, Iran, 2006, pp. 83–7.

[59] K. Tan, C. Goh, Handling uncertainties in evolutionary multi-objective optimization, Computational Intelligence: Research Frontiers (2008) 262–292.

[60] E. Hughes, Constraint handling with uncertain and noisy multi-objective evolution, in: Evolutionary Computation, 2001. Proceedings of the 2001 Congress on, Vol. 2, 2001, pp. 963–970.

[61] J. Teich, Pareto-front exploration with uncertain objectives, in: EMO '01:

Proceedings of the First International Conference on Evolutionary Multi-Criterion Optimization, Springer-Verlag, London, UK, 2001, pp. 314–328.

[62] W. Cedeno, V. R. Vemuri, On the use of niching for dynamic landscapes, in: International Conference on Evolutionary Computation, IEEE, 1997.

[63] J. Branke, Evolutionary approaches to dynamic optimization problems – a survey, Technical Report 387, Institute AIFB, University of Karlsruhe (1999).

[64] H. G. Beyer, M. Olhofer, B. Sendhoff, On the impact of systematic noise on the evolutionary optimization performance - a sphere model analysis, Genetic Programming and Evolvable Machines 5 (4) (2004) 327–360.

[65] H. Beyer, B. Sendhoff, Functions with noise-induced multimodality: A test for evolutionary robust optimization – properties and performance analysis, in: IEEE Transactions on Evolutionary Computation, Vol. 10, 2006, pp. 507–526.

[66] B. Sendhoff, H.-G. Beyer, M. Olhofer, On noise induced multi-modality in evolutionary algorithms, in: K. C. T. L. Wang, T. Furuhashi, J.-H. Kim, X. Yao (Eds.), 4th Asia-Pacific Conference on Simulated Evolution And Learning—SEAL, Vol. 1, 2002, pp. 219–224.

[67] Y. Jin, A comprehensive survey of fitness approximation in evolutionary computation, Soft Comput. 9 (1) (2005) 3–12.

[68] M. Farina, K. Deb, P. Amato, Dynamic multiobjective optimization problems: Test cases, approximation, and applications, Evolutionary Computation, IEEE Transactions on 8 (5) (2004) 425–442.

[69] D. Montana, G. Bidwell, S. Moore, Using genetic algorithms for complex, real-time scheduling, in: Network Operations and Management Symposium, IEEE, Vol. 1, 1998, pp. 245–248.

[70] F. Cupertino, E. Mininno, D. Naso, B. Turchiano, L. Salvatore, On-line genetic design of anti-windup unstructured controllers for electric drives with variable load, IEEE Trans. Evolutionary Computation 8 (4) (2004) 347–364.

[71] E. J. Hughes, B. A. White, On-line evolutionary algorithm guidance for multiple missiles against multiple targets, in: 16th IFAC Symposium on Automatic Control in Aerospace, IFAC World Congress Conference, Saint-Petersburg, 2004.

[72] E. J. Hughes, R. Zbikowski, A. Tsourdos, B. A. White, On-line evolutionary algorithm swarm trajectory optimisation, Tech. Rep. DAPS/EJH/114/2005, Cranfield University (2005).

[73] E. J. Hughes, Checkers using a co-evolutionary on-line evolutionary algorithm, in: Congress on Evolutionary Computation, IEEE, 2005, pp. 1899–1905.

[74] G. Bernat, A. Burns, A. Llamosi, Weakly hard real-time systems, IEEE Trans. Comput. 50 (4) (2001) 308–321.

[75] W. J. Ingram, E. Gray, Federal standard 1037c, NTIA report; 99-365, U.S. Dept. of Commerce, National Telecommunications and Information Administration (1999).

[76] H. Georg Beyer, Evolutionary algorithms in noisy environments: theoretical issues and guidelines for practice, in: Computer Methods in Applied Mechanics and Engineering, 2000, pp. 239–267.

[77] A. Di Pietro, L. While, L. Barone, Applying evolutionary algorithms to problems with noisy, time-consuming fitness functions, in: Proceedings of the 2004 Congress on Evolutionary Computation, CEC2004, Vol. 2, Sch. of Comp. Sci./Software Eng., University of Western Australia, Crawley, WA 6009, Australia, 2004, pp. 1254–1261.

[78] E. Zitzler, L. Thiele, Multiobjective Evolutionary Algorithms: A Comparative Case Study and the Strength Pareto Approach, IEEE Transactions on Evolutionary Computation 3 (4) (1999) 257–271.

[79] L. T. Bui, H. A. Abbass, D. Essam, Fitness inheritance for noisy evolutionary multi-objective optimization, in: GECCO '05: Proceedings of the 2005 conference on Genetic and evolutionary computation, ACM, New York, NY, USA, 2005, pp. 779–785.

[80] L. Shi, K. Rasheed, Asaga: an adaptive surrogate-assisted genetic algorithm, in: GECCO '08: Proceedings of the 10th annual conference on Genetic and evolutionary computation, ACM, New York, NY, USA, 2008, pp. 1049–1056.

[81] J. Branke, Evolutionary Optimization in Dynamic Environments, Kluwer Academic Publishers, Norwell, MA, USA, 2001.

[82] J. Branke, D. C. Mattfeld, Anticipation and flexibility in dynamic scheduling, International Journal of Production Research 43 (15) (2005) 3103–3129.

[83] J. M. Fitzpatrick, J. J. Grefenstette, Genetic algorithms in noisy environments, Mach. Learn. 3 (2-3) (1988) 101–120.

[84] Y. Sano, H. Kita, I. Kamihira, M. Yamaguchi, Online optimization of an engine controller by means of a genetic algorithm using history of search, Vol. 4, Interdiscip. Grad. Sch. Sci. Eng., Tokyo Institute of Technology, Nagatsuta, Yokohama 226-8502, Japan, 2000, pp. 2929–2934.

[85] Y. Sano, H. Kita, Online optimization of an engine controller by means of a genetic algorithm using history of search with test of estimation, in: Evolutionary Computation, 2002. CEC '02. Proceedings of the 2002 Congress on, Vol. 1, 2002, pp. 360–365.

[86] J. Fieldsend, R. Everson, Multi-objective optimisation in the presence of uncertainty, in: Evolutionary Computation, 2005. The 2005 IEEE Congress on, Vol. 1, 2005, pp. 243–250.

[87] H. Eskandari, C. Geiger, R. Bird, Handling uncertainty in evolutionary multiobjective optimization: Spga, in: Evolutionary Computation, 2007. CEC 2007. IEEE Congress on, 2007, pp. 4130–4137.

[88] H. Eskandari, C. Geiger, Evolutionary multiobjective optimization in noisy problem environments, Journal of Heuristics 15 (2009) 559–595.

[89] H. Eskandari, C. Geiger, A fast pareto genetic algorithm approach for solving expensive multiobjective optimization problems, Journal of Heuristics 14 (3) (2008) 203–241.

[90] E. Zitzler, M. Laumanns, L. Thiele, SPEA2: Improving the Strength Pareto Evolutionary Algorithm, Tech. Rep. 103, Gloriastrasse 35, CH-8092 Zurich, Switzerland (2001).

[91] K. Deb, A. Pratap, S. Agarwal, T. Meyarivan, A fast and elitist multiobjective genetic algorithm: NSGA-II, IEEE Transactions on Evolutionary Computation 6 (2) (2002) 182–197.

[92] Y. Jin, M. Olhofer, B. Sendhoff, Managing approximate models in evolutionary aerodynamic design optimization, in: In Proceedings of IEEE Congress on Evolutionary Computation, IEEE Press, 2001, pp. 592–599.

[93] J. Dennis, J. E. Dennis, V. Torczon, V. Torczon, Managing approximation models in optimization, in: Multidisciplinary Design Optimization: State-of-the-Art, 1997, pp. 330–347.

[94] Y. Jin, M. Olhofer, B. S. Member, A framework for evolutionary optimiza-

tion with approximate fitness functions, IEEE Transactions on Evolutionary Computation 6 (2000) 2002.

[95] L. Zhou, F. Haghighat, Optimization of ventilation system design and operation in office environment, part i: Methodology, Building and Environment.

[96] L. Zhou, F. Haghighat, Optimization of ventilation systems in office environment, Part II: Results and discussions, in: Building and Environment, Department of Building, Civil and Environmental Engineering, Concordia University, 1455 de Maisonneuve Blvd. W, Montreal, Quebec H3G 1M8, Canada, 2008.

[97] A. M. Brintrup, J. Ramsden, A. Tiwari, An interactive genetic algorithm-based framework for handling qualitative criteria in design optimization, Computers in Industry 58 (3) (2007) 279–291.

[98] S. Phelps, M. Köksalan, An interactive evolutionary metaheuristic for multiobjective combinatorial optimization, Management Science 49 (12) (2003) 1726–1738.

[99] M. Koksalan, S. P. Phelps, An Evolutionary Metaheuristic for Approximating Preference-Nondominated Solutions, INFORMS JOURNAL ON COMPUTING 19 (2) (2007) 291–301.

[100] D. S. Todd, P. Sen, Directed multiple objective search of design spaces using genetic algorithms and neural networks, in: Genetic and Evolutionary Computation Conference, Morgan Kaufmann, 1999, pp. 1738–1743.

[101] K. Matsui, Y. Kosugi, The effect of regularization with macroscopic fitness in a genetic approach to elastic image mapping, IEICE Transactions on Information and Systems E81-D (5) (1998) 472–478.

[102] A. Zhou, Y. Jin, Q. Zhang, B. Sendhoff, E. Tsang, Combining model-based and genetics-based offspring generation for multi-objective optimization using

a convergence criterion, in: Congress on Evolutionary Computation, IEEE Press, 2006, pp. 3234–3240.

[103] M. Bhattacharya, Expensive optimization, uncertain environment: an EA-based solution, in: GECCO '07: Proceedings of the 2007 GECCO conference companion on Genetic and evolutionary computation, ACM, New York, NY, USA, 2007, pp. 2407–2414.

[104] Y. Jin, M. Olhofer, B. Sendhoff, On evolutionary optimization with approximate fitness functions, in: L. D. Whitley, D. E. Goldberg, E. Cantú-Paz, L. Spector, I. C. Parmee, H.-G. Beyer (Eds.), GECCO, Morgan Kaufmann, 2000, pp. 786–793.

[105] H. Ulmer, F. Streichert, A. Zell, Optimization by Gaussian processes assisted evolution strategies, in: Selected Papers of the International Conference on Operations Research (OR 2003), Springer Verlag, Heidelberg, Germany, 2003, pp. 434–442.

[106] H. Ulmer, F. Streichert, A. Zell, Evolution strategies assisted by Gaussian processes with improved pre-selection criterion, in: Proceedings of the 2003 Congress on Evolutionary Computation CEC2003, Canberra, Australia, IEEE Press, 2003, pp. 692–699.

[107] H. Ulmer, F. Streichert, A. Zell, Evolution strategies with controlled model assistance, in: Proceedings of the 2004 Congress on Evolutionary Computation CEC2003, Canberra, Australia, IEEE Press, 2004, pp. 1569–1576.

[108] Y. Jin, B. Sendhoff, Constructing dynamic optimization test problems using the multi-objective optimization concept, in: G. Raidl, et al. (Eds.), Applications of Evolutionary Algorithms, Vol. 3005 of LNCS, Springer, 2004, pp. 525–536.

[109] H. Cobb, An investigation into the use of hypermutation as an adaptive operator in genetic algorithms having continuous, time-dependent nonstationary environments., Tech. rep., NRL Memorandum Report 6760 (1990).

[110] S. Yang, Memory-based immigrants for genetic algorithms in dynamic environments, in: H.-G. Beyer, et al. (Eds.), Genetic and Evolutionary Computation Conference, ACM, 2005, pp. 1115–1122.

[111] A. Ghosh, S. Tsutsui, H. Tanaka, Function optimization in nonstationary environment using steady state genetic algorithms with aging of individuals, Evolutionary Computation Proceedings, 1998. IEEE World Congress on Computational Intelligence., (1998) 666–671.

[112] K. L. Mak, Y. S. Wong, Function optimization by using genetic algorithms with individuals having different birth and survival rates, Engineering Optimization 33 (6) (2001) 749–777.

[113] J. Branke, Memory enhanced evolutionary algorithms for changing optimization problems, in: IEEE Congress on Evolutionary Computation (CEC), Vol. 3, 1999, pp. 1875–1882.

[114] D. E. Goldberg, R. E. Smith, Nonstationary function optimization using genetic algorithm with dominance and diploidy, in: J. J. Grefenstette (Ed.), Second international Conference on Genetic Algorithms on Genetic Algorithms and their Application (Cambridge, Massachusetts, United States), Lawrence Erlbaum Associates, Mahwah, NJ, 1987, pp. 59–68.

[115] K. Yamasaki, Dynamic Pareto optimum GA against the changing environments, in: Genetic Evolutionary Computation Conf. Workshop Program, San Francisco, CA, 2001, pp. 47–50.

[116] M. Kominami, T. Hamagami, A new genetic algorithm with diploid chromosomes using probability decoding for adaptation to various environments, Electronics and Communications in Japan 93 (8) (2010) 38–46.

[117] K. Y. Shao, F. Li, B. Y. Jiang, N. Wang, H. Y. Zhang, W. C. Li, Neural network optimization based on improved diploidic genetic algorithm, Vol. 3, School of Electrical and Information Engineering, Daqing Petroleum Institute, Heilongjiang Daqing 163318, China, 2010, pp. 1470–1475.

[118] A. Ş. Uyar, A. E. Harmanci, A new population based adaptive domination change mechanism for diploid genetic algorithms in dynamic environments, Soft Computing - A Fusion of Foundations, Methodologies and Applications 9 (11) (2005) 777–857.

[119] F. Greene, Method for utilizing diploid/dominance in genetic search, Vol. 1, IEEE, Univ of Washington, Seattle, United States, 1994, pp. 439–444.

[120] J. Branke, H. Schmeck, Designing evolutionary algorithms for dynamic optimization problems, in: S. Tsutsui, A. Ghosh (Eds.), Theory and Application of Evolutionary Computation: Recent Trend s, Springer, 2002, pp. 239–262.

[121] M. Adrews, A. Tuson, Diversity does not necessarily imply adaptability, in: J. Branke (Ed.), GECCO Workshop on Evolutionary Algorithms for Dynamic Optimization Problems, 2003, pp. 24–28.

[122] J. Branke, E. Salihoglu, S. Uyar, Towards an analysis of dynamic environments, in: H. Beyer (Ed.), Conference on Genetic and Evolutionary Computation, ACM Press, New York, NY, Washington DC, USA, 2005, pp. 1433–1440.

[123] Z. Zhang, S. Qian, X. Tu, Dynamic clonal selection algorithm solving constrained multi-objective problems in dynamic environments, Vol. 6, Institute

of System Science and Information Technology, Guizhou University, Guizhou, China, 2010, pp. 2861–2865.

[124] W. C. Stirling, Satisficing games and decision making: with applications to engineering and computer science, Cambridge University Press, 2003.

[125] Y. Abe, M. da Silva, J. Popović, Multiobjective control with frictional contacts, in: Proceedings of the 2007 ACM SIGGRAPH/Eurographics symposium on Computer animation California, 2007, pp. 249–258.

[126] A. Mantoglou, G. Kourakos, Optimal groundwater remediation under uncertainty using multi-objective optimization, Water Resources Management 21 (5) (2007) 835–847.

[127] A. Singh, B. Minsker, Uncertainty based multi-objective optimization of groundwater remediation at the Umatilla chemical depot, Vol. 138, ASCE, 2004, pp. 96–96.

[128] D. Büche, P. Stoll, P. Koumoutsakos, An evolutionary algorithm for multiobjective optimization of combustion processes, in: CTR Annual Research Briefs, Stanford University, 2001.

[129] B. W. Bailey, A. M. Raich, Interactive multi-objective design of long-span trusses, Department of Civil Engineering, Texas A and M - Kingsville, Kingsville, TX 78363, 2007.

[130] Z. Bingul, Adaptive genetic algorithms applied to dynamic multiobjective problems, Appl. Soft Comput. 7 (3) (2007) 791–799.

[131] T. T. Chow, G. Q. Zhang, Z. Lin, C. L. Song, Global optimization of absorption chiller system by genetic algorithm and neural network, Energy and Buildings 34 (1) (2002) 103–109.

[132] S. Ramabalan, R. Saravanan, C. Balamurugan, Multi-objective dynamic optimal trajectory planning of robot manipulators in the presence of obstacles, International Journal of Advanced Manufacturing Technology (2008) 1–15.

[133] C. Jiang, X. Han, F. J. Guan, Y. H. Li, An uncertain structural optimization method based on nonlinear interval number programming and interval analysis method, Engineering Structures 29 (11) (2007) 3168–3177.

[134] G. Airy, On the regulator of the clock-work for effecting uniform movement of equatorials., Memoirs of the Royal Astronomical Society 11 (1840) 249 – 267.

[135] J. Maxwell, On governors, Proceedings of Royal Society 6 (1868) 270 – 283.

[136] W. S. Levine, The control handbook, CRC Press, 1996.

[137] S. J. Qin, T. A. Badgwell, An overview of nonlinear model predictive control applications, in: Nonlinear Predictive Control, Verlag, 2000, pp. 369–392.

[138] M. L. Fravolini, A. Ficola, M. L. Cava, Real-time evolutionary algorithms for constrained predictive control, in: H. Iba (Ed.), Frontiers in Evolutionary Robotics, 2008.

[139] R. Roy, Soft computing and industry: recent applications, Springer, 2002.

[140] J. Yan, G. Xu, H. Qian, Y. Xu, Z. Song, Model predictive control-based fast charging for vehicular batteries, Energies 4 (8) (2011) 1178–1196.

[141] H. Chen, F. Allgöwer, A quasi-infinite horizon nonlinear model predictive control scheme with guaranteed stability, Automatica 34 (10) (1998) 1205 – 1217.

[142] M. J. Scott, E. K. Antonsson, Compensation and weights for trade-offs in engineering design: Beyond the weighted sum, Journal of Mechanical Design 127 (6) (2005) 1045–1055.

[143] K. N. Otto, E. K. Antonsson, Trade-off strategies in engineering design, Research in Engineering Design 3 (1991) 87–103.

[144] W. S. Law, E. K. Antonsson, Anon, Hierarchical imprecise design with weights, Vol. 1, IEEE, California Inst of Technology, Pasadena, United States, 1995, pp. 383–388.

[145] R. Kicinger, T. Arciszewski, K. De Jong, Evolutionary computation and structural design: A survey of the state-of-the-art, Computers and Structures 83 (23-24) (2005) 1943–1978.

[146] K. N. Otto, A formal representational theory for engineering design, Ph.D. thesis, California Institute of Technology (1992).

[147] A. Younes, Adapting evolutionary approaches for optimization in dynamic environments, Ph.D. thesis, University of Waterloo (2006).

[148] I. Das, J. E. Dennis, A closer look at drawbacks of minimizing weighted sums of objectives for pareto set generation in multicriteria optimization problems, Structural Optimization 14 (1) (1997) 63–69.

[149] A. Messac, C. Puemi-Sukam, E. Melachrinoudis, Aggregate objective functions and pareto frontiers: Required relationships and practical implications, Optimization and Engineering 1 (2000) 171–188.

[150] R. C. James, G. James, Mathematics dictionary, Chapman & Hall, 1992.

[151] M. Salukvadze, On the existence of solutions in problems of optimization under vector-valued criteria, Journal of Optimization Theory and Applications 13 (1974) 203–217.

[152] P. Chalupa, V. Bobál, Modelling and predictive control of inverted pendulum,

in: L. S. Louca, Y. Chrysanthou, Z. Oplatková, K. Al-Begain (Eds.), 22nd European Conference on Modelling and Simulation, 2008.

[153] S. Hong, Y. Oh, D. Kim, S. Ra, B. J. You, Walking pattern generation method with feedforward and feedback control for humanoid robots, University of Science and Technology (UST), Center for Cognitive Robotics Research, Korea Institute of Science and Technology (KIST), South Korea, 2009, pp. 263–268.

[154] R. E. Precup, C. Gavriluta, M. B. Radac, S. Preitl, C. A. Dragos, J. K. Tar, E. M. Petriu, Iterative learning control experimental results for inverted pendulum crane mode control, Dept. of Automation and Applied Informatics, Politehnica University of Timisoara, Timisoara, Romania, 2009, pp. 323–328.

[155] J. H. G. Macdonald, Lateral excitation of bridges by balancing pedestrians, Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences 465 (2104) (2009) 1055–1073.

[156] A. Bradshaw, J. Shao, Swing-up control of inverted pendulum systems, Robotica 14 (04) (1996) 397–405.

[157] J. C. Butcher, Numerical Methods for Ordinary Differential Equations, Wiley; 2nd edition, 2003.

[158] Euler's method for o.d.e.'s, http://math.fullerton.edu/mathews/n2003/euler'smethodmod.html (accessed: 02.05.2011).

[159] Y. Li, K. Ang, G. Chong, PID control system analysis and design, IEEE Control Systems Magazine 26 (1) (2006) 32–41.

[160] A. Mills, A. Wills, B. Ninness, Nonlinear model predictive control of an inverted pendulum, in: Proceedings of the 2009 conference on American Control Conference, ACC'09, IEEE Press, Piscataway, NJ, USA, 2009, pp. 2335–2340.

[161] S. Kahvecioglu, A. Karamancioglu, A. Yazici, Nonlinear model predictive swing-up and stabilizing sliding mode controllers, Proceedings of World Academy of Science, Engineering and Technology 57 (2009) 230–235.

[162] E. Hughes, Swarm guidance using a multi-objective co-evolutionary online evolutionary algorithm, in: Evolutionary Computation, 2004. CEC2004. Congress on, Vol. 2, 2004, pp. 2357– 2363.

[163] C. Wang, H. Ghenniwa, W. Shen, Real time distributed shop floor scheduling using an agent-based service-oriented architecture, International Journal of Production Research 46 (9) (2008) 2433–2452.

[164] Zurich Connect, How future mobility could look like, http://www.youtube.com/watch?v=pyphusemoak.

Appendix A

Minimax algorithm implementation

Listing A.1: Mimimax algorithm implementation in Python

```
class State(object):
    def __init__(self, move, children, \
            isTerminalState, terminalUtility = None):
        self._move = move
        self._isTerminal = isTerminalState
        self._utility = terminalUtility
        self._children = children

    def maxValue(self):
        if self.isTerminal:
            _rv = self._utility
        else:
            _rv = self._children[0].minValue()
            for _state in self._children:
```

```
                _rv = max(_rv, _state.minValue())
        return _rv

    def minValue(state):
         if self.isTerminal:
            _rv = self._utility
        else:
            _rv = self._children[0].maxValue()
            for _state in self._children:
                _rv = min(_rv, _state.maxValue())
        return _rv

function getMove(state):
    _utility = state.maxValue()
    return state.getChildByUtility(_utility)
```

Appendix B

Pareto Decision Tree algorithm implementation

Listing B.1: PDT algorithm implementation in Python

```
class PDT(object):
    def __init__(self, dMOEA, internalProcessModel)
        self._refAlgorithm = dMOEA
        self._refModel = internalProcessModel
        self._algorithm = deepcopy(self._refAlgorithm)


    def getOperatingPoint(self, ParetoFront):
        _s1 = ParetoFront.popRandomSolution()
        _s2 = ParetoFront.popRandomSolution()

        _maxUtility, _opPoint = \
            self.tournament(_s1, self.utility(_s1), _s2)
```

```
        _numberOfExecutions = 1
        while _numberOfExecutions < maxExecutions:
            _s = ParetoFront.popRandomSolution()
            _maxUtility, _opPoint = \
                tournament(_opPoint, _maxUtility,  _s)
        return _opPoint

    def tournament(self, solution1, utility1, solution2):
        _utility2 = self.utility(solution2)

        if utility1 > _utility2:
            _rv = (utility1, solution1)
        else:
            _rv = (_utility2, solution2)
        return _rv

    def utility(self, solution):
        _population = deepcopy(algorithm.getPopulation())
        self._algorithm.setPopulation(_population)

        _control = new ControlConditioner(solution)
        _model = ProcessModel(self._refModel, _control)
        self._algorithm.getProblem().setModel(_model)

        self._model.runStep()
        _newParetoFront = self._algorithm.runStep()
        return hypervolume(_newParetoFront)
```